

EXPECTED-SECURITY-COST OPTIMAL
POWER FLOW WITH SMALL-SIGNAL
STABILITY CONSTRAINTS

By

JOHN CONDREN

Bachelor of Science

Oklahoma State University

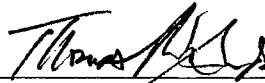
Stillwater, Oklahoma

1999

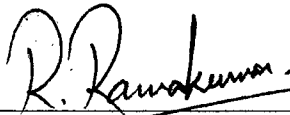
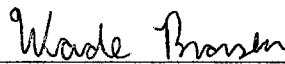
Submitted to the Faculty
of the Graduate College of
Oklahoma State University
in partial fulfillment
of the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
August, 2003

EXPECTED-SECURITY-COST OPTIMAL
POWER FLOW WITH SMALL-SIGNAL
STABILITY CONSTRAINTS

Thesis Approved:



Thesis Adviser



Dean of the Graduate College

Acknowledgements

I wish to thank my wife, Lydia, for her love, patience, understanding, and motivation. I greatly appreciate the support of my advisor Dr. Thomas Gedra, who has been influential in directing me toward the path of studying power systems. His knowledge and insight have helped me a great deal. I would like to thank my advisory committee, Dr. Rama Ramakumar, Dr. Hermann Burchard, and Dr. Wade Brorsen, for their help. Also, thanks to my parents, Gary and Ruth Condren for their love, support, and guidance.

This work was funded in part by the National Science Foundation under grant ECS-9501648 and by the Environmental Institute of Oklahoma State University.

Contents

1	Introduction	1
2	Loadflow	3
3	Economic dispatch	8
4	Optimal power flow	11
4.1	Cost, Benefit, and Social Welfare	11
4.2	Constraints	13
4.3	Information given by OPF solution	15
5	Power System Security	17
5.1	Security basics	17
5.2	Example of insecure operating point	18
5.3	Expected value of social welfare	22
5.4	Security costs	23
5.5	Coupling constraints	25
5.6	Expected-security-cost optimal power flow (ESCOPF)	26
6	Optimization problem solution methods	29
6.1	Optimization with no inequality constraints	29

CONTENTS

6.2	Active set methods	30
6.3	Primal-dual interior-point method	31
6.4	PDIP algorithm	37
6.5	Solution to OPF Problem	40
7	Synchronous machine model	42
7.1	Principles of operation	42
7.2	Basic model	43
7.3	Transformation and scaling	46
7.4	Linear magnetic circuit	49
7.5	Steady-state condition for single machine	53
7.6	Synchronous machine excitation control	56
7.7	Multi-machine power system model	63
8	Small-signal stability	69
8.1	Definition of small-signal stability	69
8.2	Zero eigenvalues	72
9	Eigenvalue Sensitivities	75
9.1	Assumptions	76
9.2	First-order eigenvalue sensitivities	77
9.3	Second-order eigenvalue sensitivities computed using all eigenvalues & eigenvectors	77
9.4	Second-order eigenvalue sensitivities computed using a single eigenvalue and eigenvector.	78
9.5	First order eigenvector sensitivities	78
9.5.1	Method I	78
9.5.2	Method II	79
10	OPF and ESCOPF with small-signal stability constraints	81
10.1	Adding small-signal stability constraints	81

CONTENTS

10.2 Test system	82
10.3 Implementation	84
10.4 Summary of SSSOPF Results	88
10.5 Summary of SSS ESCOPF Results	89
10.5.1 Using Arnoldi to speed up algorithm	92
11 Conclusions & future work	94
A Linearizing the differential and algebraic equations.	100
A.1 Linearizing the differential equations	100
A.2 Stator Algebraic Equation	103
A.3 Network Equation	104
A.3.1 Generator buses	104
A.3.2 Non-generator buses	106
B Forming the A_{sys} matrix	107
C Derivation of second order eigenvalue sensitivities computed from all eigenvalues	110
D WSCC 9-bus test system data	115
E Description of data file formats	117
E.1 Description of .cdf file format	117
E.2 Description of .gdf file format	118
E.3 Description of .ldf file format	118
E.4 Description of .sdf file format	118
F Small-signal stability OPF results ignoring security	121
G Small-signal stability ESCOPF Results	126

CONTENTS

H	Small-signal stability ESCOPF Matlab code	148
H.1	Executing the program	148
H.2	opf.m	149
H.3	createindex.m	150
H.4	variables.txt	157
H.5	subvariables.txt	158
H.6	ReadDF.m	159
H.7	getdata.m	164
H.8	index.m	167
H.9	initialize.m	173
H.10	pdip.m	178
H.11	sysmat.m	181
H.12	eq_eval.m	187
H.13	ineq_eval.m	188
H.14	eq_jacobian.m	189
H.15	ineq_jacobian.m	194
H.16	objective_jacobian.m	195
H.17	eq_hessian.m	196
H.18	ineq_hessian.m	203
H.19	objective_hessian.m	204
H.20	eig_jac_hess.m	205
H.21	asysderiv.m	207
H.22	getstate.m	244
H.23	eigderiv.m	249
H.24	update.m	255
H.25	sederiv.m	258

CONTENTS

H.26 se2deriv.m	259
H.27 se3deriv.m	260

List of Figures

2.1	Transmission line equivalent π circuit	3
5.1	Line properties for 5-bus case.	19
5.2	Insecure power system operating point for 5-bus case.	21
7.1	2-pole synchronous machine schematic	43
7.2	Excitation control block diagram.	57
7.3	Proportional/integral controller block diagram.	59
7.4	Stabilizing transformer equivalent circuit.	61
10.1	WSCC 9-bus system	83
10.2	Matlab function <code>opf.m</code>	85

List of Tables

5.1	Generator and load data for 5-bus case.	20
5.2	ESCOPF problem statement.	28
7.1	Basic synchronous machine equations.	48
7.2	Transformed and scaled synchronous machine equations.	50
7.3	Linear magnet circuit synchronous machine model	53
7.4	Excitation control model equations.	63
7.5	Synchronous machine scaled dynamic and algebraic equations for multi-machine model.	66
7.6	Multi-machine power system dynamic model	68
8.1	Description of power system variables.	71
10.1	Summary of SSSOPF results.	89
10.2	Summary of SSS ESCOPF results, $\mathbf{Re}(\lambda) \leq 0$	90
10.3	Summary of SSS ESCOPF results, $\mathbf{Re}(\lambda) \leq -0.15$	90
10.4	Summary of SSS ESCOPF results, $\mathbf{Re}(\lambda) \leq -0.20$	90
10.5	Summary of SSS ESCOPF results, $\mathbf{Re}(\lambda) \leq -0.25$	90
10.6	Summary of SSS ESCOPF results.	92
A.1	Size of matrices in linearized differential, algebraic, and network equations.	106

LIST OF TABLES

D.1	Transmission line and transformer data.	115
D.2	Bus voltage limits in per unit	115
D.3	Consumer benefit curve coefficients.	116
D.4	Generator cost curve coefficients and limits.	116
D.5	Generator and load security data for WSCC 9-bus case.	116
D.6	Synchronous machine & exciter data.	116
E.1	CDF file bus data section format.	117
E.2	CDF file branch data section format.	118
E.3	GDF data file format.	119
E.4	LDF data file format.	119
E.5	SDF data file format.	120
F.1	SSSOPF results with $\mathbf{Re}(\lambda_i) \leq 0$	122
F.2	SSSOPF results with $\mathbf{Re}(\lambda_i) \leq -0.2$	123
F.3	SSSOPF results with $\mathbf{Re}(\lambda_i) \leq -0.3$	124
F.4	SSSOPF results with $\mathbf{Re}(\lambda_i) \leq -0.4$	125
G.1	SSSESCOPF results with $\mathbf{Re}(\lambda_i) \leq 0$ (pre-contingency)	127
G.2	SSSESCOPF results with $\mathbf{Re}(\lambda_i) \leq 0$ (line 4-6 out)	128
G.3	SSSESCOPF results with $\mathbf{Re}(\lambda_i) \leq 0$ (line 4-5 out)	129
G.4	SSSESCOPF results with $\mathbf{Re}(\lambda_i) \leq 0$ (line 5-7 out)	130
G.5	SSSESCOPF results with $\mathbf{Re}(\lambda_i) \leq 0$ (line 6-9 out)	131
G.6	SSSESCOPF results with $\mathbf{Re}(\lambda_i) \leq 0$ (line 7-8 out)	132
G.7	SSSESCOPF results with $\mathbf{Re}(\lambda_i) \leq 0$ (line 9-8 out)	133
G.8	SSSESCOPF results with $\mathbf{Re}(\lambda_i) \leq -0.15$ pre-contingency	134
G.9	SSSESCOPF results with $\mathbf{Re}(\lambda_i) \leq -0.15$ (line 4-6 out)	135
G.10	SSSESCOPF results with $\mathbf{Re}(\lambda_i) \leq -0.15$ (line 4-5 out)	136

LIST OF TABLES

G.11 SSSESCOPF results with $\mathbf{Re}(\lambda_i) \leq -0.15$ (line 5-7 out) 137

G.12 SSSESCOPF results with $\mathbf{Re}(\lambda_i) \leq -0.15$ (line 6-9 out) 138

G.13 SSSESCOPF results with $\mathbf{Re}(\lambda_i) \leq -0.15$ (line 7-8 out) 139

G.14 SSSESCOPF results with $\mathbf{Re}(\lambda_i) \leq -0.15$ (line 9-8 out) 140

G.15 SSSESCOPF results with $\mathbf{Re}(\lambda_i) \leq -0.25$ pre-contingency 141

G.16 SSSESCOPF results with $\mathbf{Re}(\lambda_i) \leq -0.25$ (line 4-6 out) 142

G.17 SSSESCOPF results with $\mathbf{Re}(\lambda_i) \leq -0.25$ (line 4-5 out) 143

G.18 SSSESCOPF results with $\mathbf{Re}(\lambda_i) \leq -0.25$ (line 5-7 out) 144

G.19 SSSESCOPF results with $\mathbf{Re}(\lambda_i) \leq -0.25$ (line 6-9 out) 145

G.20 SSSESCOPF results with $\mathbf{Re}(\lambda_i) \leq -0.25$ (line 7-8 out) 146

G.21 SSSESCOPF results with $\mathbf{Re}(\lambda_i) \leq -0.25$ (line 9-8 out) 147

Chapter 1

Introduction

There is much to be checked when making sure an electric power system is operating in a proper manner. The most obvious is that there must be enough power generated to serve the load. Also, the voltage at the load must be kept at a desired level, and we can't push too much power down a transmission line or through a transformer.

In addition to these basic operating limits, we must consider what happens when some piece of the system suddenly becomes unavailable due to a fault. For example, if we lose a transmission line due to a short at some point along the line, how will the system react? The short-term analysis of such a large fault falls into the category of *transient stability* analysis.

If we assume that the system can survive the initial large fault, we must determine if the system can continue to serve the load after the faulted transmission line is removed from the system. We may consider adjusting the power output of generators in the system, and there it may be possible that some of the load has to be dropped so the system can survive. This type of analysis determines how *secure* the power system is.

A third measure of the health of the system is its ability to withstand tiny disturbances. If system load were to change slightly, or if there a tiny change in voltage a some location, what would happen? The primary area of concern is what happens to generator rotors throughout the system when these disturbances occur. Since all generators are interconnected via the power system transmission grid, a small disturbance at any point in the system can cause the rotor of one or more generators in the

CHAPTER 1. INTRODUCTION

system to accelerate or decelerate. Furthermore, groups of generators may accelerate and decelerate together. These types of events have occurred in Taiwan [1], the U.K. [2], and the western U.S. [3]. By performing a *small-signal stability* analysis on the system steady-state operating point, we can find out if these accelerations die out quickly or if they continue for a long time or if they continue to grow in magnitude.

Closely related to the concepts of operating limits, security, and stability analysis is the optimal economic dispatch of power generation. We would like to serve the load at the least possible cost while making sure the system is secure, stable, and doesn't violate any operating limits. Each generator has a different cost curve that describes the cost of producing a given amount of power. By adjusting how much each generator produces, we can find the cheapest possible way to serve the load. But the dispatch we choose must satisfy system constraints.

This is where the need to form a constrained optimization problem arises. Much research has been done in the area of power system constrained optimization. Various methods have been proposed to find a dispatch which minimizes some objective while obeying various constraints. One constraint that has yet to be added to the power system optimization problem is the small-signal stability constraint. The objective of this dissertation is to address such constraints.

We begin by reviewing basic concepts which describe power flows in the system. Economic concepts applied to the dispatch of power are then described followed by a presentation of the optimal power flow problem. Power system security is then discussed along with a method of including security constraints in the OPF. Next, we discuss small-signal stability and how it depends on the power system steady-state operating point.

After reviewing literature on power flow, optimization, security, and stability, we show the result of including small-signal stability constraints in an OPF problem applied to a 9-bus test case. The OPF is solved with and without security constraints so we can see the difference in the pre-contingency operating states.

Chapter 2

Loadflow

This chapter presents some basic power system concepts and applies them to the classic loadflow problem [4]. First, we note that we can compute the power flowing on a transmission line if we know the voltage at each end of the line and the transmission line properties. This assumes we model the transmission line using an equivalent π circuit as shown in Figure 2.1. The circuit consists of a series admittance $Y_{\text{series}_{ik}}$ between buses i and k , and it contains a shunt admittance $Y_{\text{shunt}_{ik}}$ at bus i and bus k . The voltage at bus i consists of a magnitude V_i and a phase angle θ_i which comprise a complex voltage phasor $\vec{V}_i = V_i e^{j\theta_i}$. The complex current flowing from bus i to bus k and from

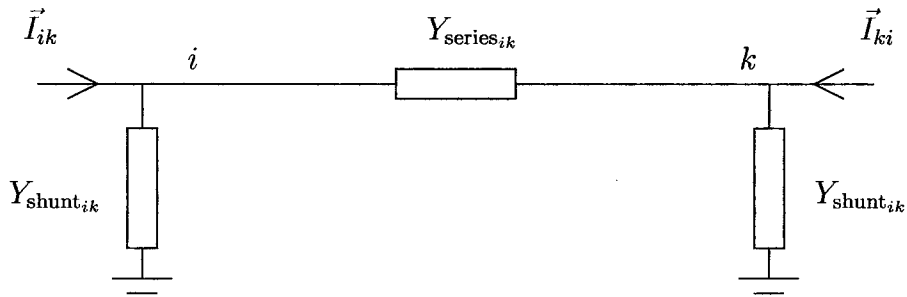


Figure 2.1: Transmission line equivalent π circuit

bus k to bus i are then

$$\vec{I}_{ik} = (\vec{V}_i - \vec{V}_k) Y_{\text{series}_{ik}} + \vec{V}_i Y_{\text{shunt}_{ik}} \quad (2.1)$$

$$\vec{I}_{ki} = (\vec{V}_k - \vec{V}_i) Y_{\text{series}_{ik}} + \vec{V}_k Y_{\text{shunt}_{ik}} \quad (2.2)$$

which can be used to compute the complex power flows

$$\vec{S}_{ik} = \vec{V}_i \vec{I}_{ik}^* \quad (2.3)$$

$$\vec{S}_{ki} = \vec{V}_k \vec{I}_{ki}^* \quad (2.4)$$

Bus i may have several buses connected to it, so we can compute the total complex power injected at bus i as

$$\vec{S}_i = \vec{V}_i \sum_{n=1}^{N_B} \vec{I}_{in}^* \quad (2.5)$$

$$= P_i(V, \theta) + j Q_i(V, \theta) \quad (2.6)$$

where N_B is the total number of buses in the system, V is the vector of bus voltage magnitudes, θ is the vector of bus voltage phase angles, and \vec{I}_{in} exists only if there is a transmission line connecting buses i and n . Now we have the real power, $P_i(V, \theta)$, and reactive power, $Q_i(V, \theta)$, injected at each bus as functions of voltage magnitudes, V , and voltage phase angles, θ .

Another way to compute the power injected into a power system bus requires knowledge of the amount of power generated and consumed at each bus. Suppose we have $P_{Gi} + j Q_{Gi}$ produced and $P_{Li} + j Q_{Li}$ consumed at bus i . Then the amount of power injected at the bus is simply

$$\vec{S}_i = P_{Gi} + j Q_{Gi} - [P_{Li} + j Q_{Li}] \quad (2.7)$$

We now have two ways to compute the power injected at bus i :

CHAPTER 2. LOADFLOW

1. Knowledge of V and θ

$$\vec{S}_i = P_i(V, \theta) + j Q_i(V, \theta)$$

2. Knowledge of $P_{Gi}, Q_{Gi}, P_{Li}, Q_{Li}$

$$\vec{S}_i = P_{Gi} + j Q_{Gi} - [P_{Li} + j Q_{Li}]$$

Equating the two and separating real and imaginary parts gives us

$$P_i(V, \theta) = P_{Gi} - P_{Li} \quad (2.8)$$

$$Q_i(V, \theta) = Q_{Gi} - Q_{Li} \quad (2.9)$$

These are the loadflow equations which define a steady-state power system operating point.

There are 6 sets of variables that we need to know before we can check the loadflow equations: V , θ , P_G , P_L , Q_G , Q_L . The loadflow problem assumes we are given values for some of these variables, and we have to solve for the unknown variables so that the loadflow equations are satisfied. In a loadflow problem, each bus is placed into a category depending on what values we know at the bus:

- *PV* bus - we know P_i and V_i
- *V θ* or “slack” bus - we know V_i and θ_i
- *PQ* or “load” bus - we know P_i and Q_i

We can use Newton-Raphson iteration to solve for the unknown variables. The first step is linearizing the loadflow equations by writing a first-order Taylor series about a “guessed” steady-state operating point

$$P_i(V, \theta) = P_i(V^k, \theta^k) + \frac{\partial P_i}{\partial V}(V - V^k) + \frac{\partial P_i}{\partial \theta}(\theta - \theta^k) \quad (2.10)$$

$$Q_i(V, \theta) = Q_i(V^k, \theta^k) + \frac{\partial Q_i}{\partial V}(V - V^k) + \frac{\partial Q_i}{\partial \theta}(\theta - \theta^k) \quad (2.11)$$

CHAPTER 2. LOADFLOW

where $P_i(V^k, \theta^k)$, $Q_i(V^k, \theta^k)$, V^k , and θ^k represent the steady-state operating point at iteration k about which we are linearizing and at which the Jacobian matrices in the expressions are evaluated. Let the values of the voltage magnitudes and phase angles at the solution to the loadflow equations be \hat{V} and $\hat{\theta}$. We write equations 2.10 and 2.11 in matrix notation, where we replace $\hat{V} - V^k$ with ΔV and $\hat{\theta} - \theta^k$ with $\Delta\theta$:

$$\begin{bmatrix} P_i(\hat{V}, \hat{\theta}) - P_i(V^k, \theta^k) \\ Q_i(\hat{V}, \hat{\theta}) - Q_i(V^k, \theta^k) \end{bmatrix} = \begin{bmatrix} \frac{\partial P_i}{\partial V} & \frac{\partial P_i}{\partial \theta} \\ \frac{\partial Q_i}{\partial V} & \frac{\partial Q_i}{\partial \theta} \end{bmatrix} \begin{bmatrix} \Delta V \\ \Delta\theta \end{bmatrix}. \quad (2.12)$$

Writing the above for each bus i where we know P_i and for each bus j where we know Q_j , we have

$$\begin{bmatrix} P(\hat{V}, \hat{\theta}) - P(V^k, \theta^k) \\ Q(\hat{V}, \hat{\theta}) - Q(V^k, \theta^k) \end{bmatrix} = \begin{bmatrix} \frac{\partial P}{\partial V} & \frac{\partial P}{\partial \theta} \\ \frac{\partial Q}{\partial V} & \frac{\partial Q}{\partial \theta} \end{bmatrix} \begin{bmatrix} \Delta V \\ \Delta\theta \end{bmatrix}. \quad (2.13)$$

We know $P(\hat{V}, \hat{\theta})$ and $Q(\hat{V}, \hat{\theta})$ and can compute $P(V^k, \theta^k)$, $Q(V^k, \theta^k)$, and the Jacobians. Also, we voltage magnitudes at some buses and phase angles at some buses. The columns of the Jacobian corresponding to these voltage magnitudes and phase angles can be eliminated. Therefore, we can solve the system of linear equations for ΔV and $\Delta\theta$. These vectors tell us where to move to find, hopefully, a steady-state operating point that results in values of $P_i(V^k, \theta^k)$ and $Q_j(V^k, \theta^k)$ that are closer to the actual values $P_i(\hat{V}, \hat{\theta})$ and $Q_j(\hat{V}, \hat{\theta})$. The update vectors are added to the current unknown values to obtain the new guess at iteration $k + 1$

$$V^{k+1} = V^k + \Delta V \quad (2.14)$$

$$\theta^{k+1} = \theta^k + \Delta\theta. \quad (2.15)$$

We continue in this fashion until either the update vector is very small or until the values $P(\hat{V}, \hat{\theta}) - P(V^k, \theta^k)$ and $Q(\hat{V}, \hat{\theta}) - Q(V^k, \theta^k)$ have very small norms.

The loadflow problem assumes known load values and known values for most real power genera-

CHAPTER 2. LOADFLOW

tion levels. It also assumes knowledge of some voltage magnitudes. In addition, we did not consider system operating limits when solving the problem. It could be that after solving the problem, some voltage magnitudes are too high, or some line flows are too large. If this is the case, then we would have to perform the difficult task of guessing how we could adjust some of the known values so that limits won't be violated. These are drawbacks to the loadflow problem. In the next section, we discuss economic dispatch, which is a way to determine optimal generation levels to serve a load.

Chapter 3

Economic dispatch

In the last chapter, we discussed ways to solve for a power system steady-state operating point given that we know some real power generation levels, some voltage magnitudes, and system loading levels. Now we discuss a way to determine the optimal power generation dispatch given the system loading level [4], [5].

We start by defining generator cost curves, where we assume cost of real power production for generator i of the form

$$C_i(P_{Gi}) = \gamma_i P_{Gi}^2 + \beta_i P_{Gi} + \alpha_i. \quad (3.1)$$

We have assumed a quadratic form, but cost curves may not be quadratic, or they may be piecewise quadratic. The total cost of generation is then

$$C_T = \sum_{i \in \mathcal{G}} C_i(P_{Gi}) \quad (3.2)$$

The goal of the basic optimal dispatch problem is to minimize the total cost of generation C_T subject

to the constraint that system load is the same as the total amount generated

$$\min_{P_G} \sum_{i \in \mathcal{G}} [\gamma_i P_{Gi}^2 + \beta_i P_{Gi} + \alpha_i] \quad (3.3)$$

subject to

$$\sum_{i \in \mathcal{G}} P_{Gi} = \sum_{i \in \mathcal{L}} P_{Li}. \quad (3.4)$$

To solve this problem, we write the Lagrangian function

$$\mathcal{L} = \sum_{i \in \mathcal{G}} [\gamma_i P_{Gi}^2 + \beta_i P_{Gi} + \alpha_i] \quad (3.5)$$

$$-z \left(\sum_{i \in \mathcal{G}} P_{Gi} - \sum_{i \in \mathcal{L}} P_{Li} \right) \quad (3.6)$$

The Lagrange multiplier z is usually denoted by the Greek symbol λ , but we use z here since λ will be used to describe something else later. After minimizing the Lagrangian, we know the optimal value of the Lagrange multiplier, which is often referred to as “system λ .” To minimize the Lagrangian function, take the partial derivative with respect to each variable and set each equal to zero

$$2\gamma_i P_{Gi} + \beta_i - z = 0, \quad i \in \mathcal{G} \quad (3.7)$$

$$\sum_{i \in \mathcal{G}} P_{Gi} - \sum_{i \in \mathcal{L}} P_{Li} = 0. \quad (3.8)$$

The total system load is fixed, so the unknowns are P_{Gi} and the Lagrange multiplier z . We can write each P_{Gi} as a function of the Lagrange multiplier z and then substitute the result into the power balance equation to obtain

$$\sum_{i \in \mathcal{G}} \left(\frac{z - \beta_i}{2\gamma_i} \right) - \sum_{i \in \mathcal{L}} P_{Li} = 0, \quad (3.9)$$

which can be solved for z . After z is known, we set each generator marginal cost equal to the

CHAPTER 3. ECONOMIC DISPATCH

Lagrange multiplier z and solve for P_{Gi} .

We may also consider real power generation limits when solving the economic dispatch problem. If the value of P_{Gk} for the generator at bus k violates its maximum, then we set $P_{Gk} = P_{Gk}^{\max}$. Since we now know P_{Gk} , we can lump it in with the known system load term and solve the economic dispatch problem for the remaining P_{Gi} .

The equality constraint included in the optimization problem does not consider real power losses in the systems, nor does it consider reactive power dispatch and reactive power losses. Methods exist to incorporate loss penalty factors into the problem so that real power losses are considered. However, simply knowing generation real power dispatch won't give us the solution to the loadflow equations. For doing economic dispatch and solving the loadflow in one step, we may solve an optimal power flow problem as presented in the next chapter.

Chapter 4

Optimal power flow

4.1 Cost, Benefit, and Social Welfare

The economic dispatch problem discussed in the previous section was missing the loadflow constraints. Adding these constraints creates an *optimal power flow* problem whose *objective function* is minimizing cost of power generation. Another possible objective function is maximizing social welfare.

To understand the concept of social welfare, we first state the two market participants:

- producers and
- consumers.

A producer is a supplier of some type of good. Of course, there is a cost to produce the good. Let's say we know the cost of producing the good as a function of the amount of the good that is produced. In our case, the producer is supplying electric power, so his cost is a function of the amount of power he produces. More specifically, we will say that his cost is a function of the amount of *real power* produced. For a given generator in our power system, the cost C of generation is a function of the amount of real power P_G generated. The total cost of real power generation is then

$$C_G = \sum_{i \in \mathcal{G}} C_i(P_{Gi}) \tag{4.1}$$

CHAPTER 4. OPTIMAL POWER FLOW

where \mathcal{G} is the set of indices of buses in the power system that have generators.

A consumer purchases a good because he receives some benefit from using the good. Let's say that this benefit that he receives can be measured in dollars and is a function of the amount of the good he buys. For a consumer of power, his benefit B in dollars is a function of the amount of real power P_L he consumes. The total benefit received from the consumption of real power by loads in the power system is

$$B_L = \sum_{i \in \mathcal{L}} B_i(P_{Li}) \quad (4.2)$$

where \mathcal{L} is the set of indices of buses in the power system that have loads.

Now we define the meaning of social welfare. It is the net benefit (benefit minus cost) received by society due to the production and consumption of the good. In the case of real power generation and consumption, the social welfare is

$$\begin{aligned} SW &= B_L - C_G \\ &= \sum_{i \in \mathcal{L}} B_i(P_{Li}) - \sum_{i \in \mathcal{G}} C_i(P_{Gi}) \end{aligned} \quad (4.3)$$

The opposite of the net benefit received by society due to the production and consumption of the good is the total cost to society due to the production and consumption of the good. This cost is the negative of social welfare, or

$$\begin{aligned} C &= -SW \\ &= \sum_{i \in \mathcal{G}} C_i(P_{Gi}) - \sum_{i \in \mathcal{L}} B_i(P_{Li}) \end{aligned} \quad (4.4)$$

Benefits can be viewed as negative costs. When this negative cost is added to the cost of generation, the total cost is obtained. This cost function is the function we want to minimize. It is an OPF problem *objective function*.

4.2 Constraints

The OPF is a constrained optimization problem. At the optimum, the loadflow equations must be satisfied, and system operating limits must not be violated. We have already discussed the loadflow equations

$$P_i(V, \theta) = P_{Gi} - P_{Li} \quad (4.5)$$

$$Q_i(V, \theta) = Q_{Gi} - Q_{Li}. \quad (4.6)$$

These are equality constraints that must be satisfied at the optimum.

The amount of reactive power consumed by the load, Q_{Li} , is not considered a variable since in this model. It will be assumed that each load has a fixed power factor where the power factor is defined as

$$\text{p.f.} = \frac{|P_L|}{\sqrt{P_L^2 + Q_L^2}}, \quad (4.7)$$

or the ratio of the real power consumed to the magnitude of the apparent power consumed. The power factor is *lagging* for positive reactive power consumed (inductive load) and *leading* for negative reactive power consumed (capacitive load). Assuming fixed power factor and an inductive load, Q_L is a linear function of the real power consumed, or

$$Q_L(P_L) = \frac{P_L}{\text{p.f.}} \sqrt{1 - \text{p.f.}^2}. \quad (4.8)$$

The system operating limits include maximum and minimum limits on the voltage magnitude at each bus and the magnitude of the apparent power flowing on each line. These can be written as

$$V_i^{\min} \leq V_i \leq V_i^{\max} \quad (4.9)$$

CHAPTER 4. OPTIMAL POWER FLOW

for the voltage magnitude V_i at bus i and

$$|S_{ik}(V, \theta)| \leq S_{ik\max} \quad (4.10)$$

where

$$S_{ik} = V_i e^{j\theta_i} [(V_i e^{-j\theta_i} - V_k e^{-j\theta_k}) Y_{ik}^* + V_i e^{-j\theta_i} Y_{\text{shunt}, ik}^*] \quad (4.11)$$

$$= V_i^2 (Y_{ik}^* + Y_{\text{shunt}, ik}^*) - V_i V_k e^{j(\theta_i - \theta_k)} Y_{ik}^* \quad (4.12)$$

for the magnitude of the apparent power flowing on the transmission line from system bus i to system bus k . Instead of constraining apparent power, we could have instead constrained current flowing on the line as in thermal limits, or we could have placed steady-state stability limits on the line. Also, in practice, we constrain $|S_{ik}(V, \theta)|^2$, which is the square of the magnitude of the apparent power flowing on the line because it is easier to compute derivatives of this constraint function as required by our OPF solution method.

There are also limits on the amount of real and reactive power produced by each generator. There may also be a limit on the amount real power consumed by each load. This gives us an additional set of inequality constraints.

$$\begin{aligned} P_{Gi}^{\min} &\leq P_{Gi} \leq P_{Gi}^{\max} \\ P_{Li}^{\min} &\leq P_{Li} \leq P_{Li}^{\max} \\ Q_{Gi}^{\min} &\leq Q_{Gi} \leq Q_{Gi}^{\max} \end{aligned} \quad (4.13)$$

We have now defined a simple OPF problem. It can be stated as follows.

$$\min_Y \left[\sum_{i \in \mathcal{G}} C_i(P_{Gi}) - \sum_{i \in \mathcal{L}} B_i(P_{Li}) \right] \quad (4.14)$$

subject to equality constraints

$$\begin{aligned} P_{Gi} - P_{Li} &= P_i(V, \theta), & i = 1 \dots N_B \\ Q_{Gi} - Q_{Li} &= Q_i(V, \theta), & i = 1 \dots N_B \end{aligned} \quad (4.15)$$

and inequality constraints

$$\begin{aligned} V_i^{\min} &\leq V_i \leq V_i^{\max}, & i = 1 \dots N_B \\ S_{ik}(V, \theta) \cdot S_{ik}^*(V, \theta) &\leq S_{ik_{\max}}^2, \\ P_{Gi}^{\min} &\leq P_{Gi} \leq P_{Gi}^{\max}, & i \in \mathcal{G} \\ P_{Li}^{\min} &\leq P_{Li} \leq P_{Li}^{\max}, & i \in \mathcal{L} \\ Q_{Gi}^{\min} &\leq Q_{Gi} \leq Q_{Gi}^{\max}, & i \in \mathcal{G} \end{aligned} \quad (4.16)$$

4.3 Information given by OPF solution

In the OPF problem, we are minimizing a cost function subject to a set of equality constraints and a set of inequality constraints. We want to find the vector of variables Y that solves the problem. This set of variables defines a steady-state operating point for our power system. This list of variables includes

- magnitude of voltage at each bus,
- phase angle of voltage at each bus,
- real and reactive power generated at each bus,
- real power consumed at each bus.

Another piece of information we gain by solving the OPF is the price of power at each bus in the system. This price is the Lagrange multiplier on the equality constraint corresponding to the

CHAPTER 4. OPTIMAL POWER FLOW

loadflow equations. Lagrange multipliers are discussed in the next chapter.

Chapter 5

Power System Security

5.1 Security basics

A secure power system is one that can serve load without violating system operating limits even after the occurrence of a credible contingency. A credible contingency could be a transmission line outage or the loss of a generator. We might also consider the simultaneous outage of two or more system components as credible contingencies. In this research, we only consider line outages.

One way to determine if a system is secure is to see if it can still provide load within *normal* operating limits after the occurrence of a contingency. Another way is to relax the operating limits slightly for the post-contingency condition so that we have emergency operating limits. Since the system will (hopefully) not operate in the post-contingency condition for too long, we can push the system to operate in a more stressful state.

Another item to consider is the possibility of post-contingency rescheduling. After a contingency, we may have the ability to do one or more of the following:

- adjust generator power output
- drop load.

By adjusting generation dispatch and/or dropping customer load, we can prevent operating limits from being violated. Of course these options come at a cost, but this cost is not infinite for generators

that we have the ability to adjust and for customers who have agreed to be subject to interruption. Assuming that we cannot adjust generation and consumption means that we have assigned infinite cost to these adjustments.

In this chapter, we first present an example of a system which is not secure. Then we consider an objective function for a modified optimal power flow problem that treats social welfare as a random variable. This new objective function will contain costs for maintaining a secure system and new constraints that limit the re-dispatch of power when a contingency occurs. Finally, we combine the new objective function and constraints to form an expected-security-cost optimal power flow (ESCOFF) problem.

5.2 Example of insecure operating point

We now know that it is possible to adjust generation levels and to drop load for some customers so we can survive a contingency. We also know that the system operating limits may be slightly relaxed after the contingency occurs. Our problem is determining how we should make adjustments to the optimal power dispatch to give us a secure system.

One way to choose the “best” operating point to account for system security is to form an optimal power flow problem. The goal of the basic optimal power flow problem is to maximize social welfare. For security analysis, we consider the possibility of line outages which are random events. At a given moment in time, these events have some probability of occurring. Therefore, at any steady state operating point we choose, there is some probability of operating in a state where the system has changed due to the occurrence of a contingency.

Suppose we were to simply maximize social welfare for the “normal” operating state where no contingencies have occurred. We will be discussing security of a 5-bus system whose transmission line properties are given in Figure 5.1. The property b_{ij} is the susceptance of the line, and the property π^k is the probability of operating in a state where the line has failed. Two line power flow limits are given: a normal limit and an emergency limit shown in parentheses. For the 5-bus case,

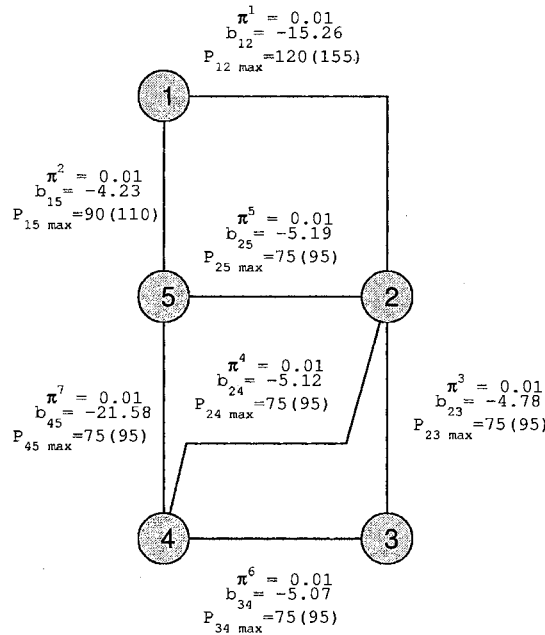


Figure 5.1: Line properties for 5-bus case.

we will use a linearized “DC” approximation to the power flow equations. The real power flowing from bus i to bus j is

$$P_{ij} = -b_{ij}(\theta_i - \theta_j), \quad (5.1)$$

where θ_i is the voltage phase angle at bus i . Reactive power flows and real power losses are neglected, and voltage magnitudes are approximated as 1 per unit. We sacrifice accuracy to obtain a simplified analysis which gives a close approximation to actual power flows in this example.

Generator data is given in Table 5.1 We assume a quadratic generator cost curve of the form

$$C_i(P_{Gi}) = \gamma_{Gi}P_{Gi}^2 + \beta_{Gi}P_{Gi} + \alpha_{Gi} \quad (5.2)$$

where P_{Gi} is the amount of real power generated by the machine at bus i . Similarly, consumer

Generator Data						Load Data					
Bus	1	2	3	4	5	Bus	1	2	3	4	5
γ_G	.000820	.000776	-	-	-	γ_L	-	-.2147	-.0531	-.0531	-.0930
β_G	11.0	12.0	-	-	-	β_L	-	21.43	23.00	22.48	22.11
$\Delta_{\max}^+ P_G$	50	35	-	-	-	β_I	-	100	100	100	100
$\Delta_{\max}^- P_G$	50	35	-	-	-						
$P_{G\max}$	250	150	-	-	-						
$P_{G\min}$	45	15	-	-	-						

Table 5.1: Generator and load data for 5-bus case.

benefit curves for a load P_{Li} at bus i is given by

$$B_i(P_{Li}) = \gamma_{Li} P_{Li}^2 + \beta_{Li} P_{Li}, \quad (5.3)$$

and the cost of interrupting the load at bus i is

$$C_{I_i}(P_{L_i}^0, P_{L_i}^k) = \beta_{I_i} [P_{L_i}^0 - P_{L_i}^k], \quad (5.4)$$

where $P_{L_i}^0$ is the pre-contingency load at bus i , and $P_{L_i}^k$ is the post-contingency load at bus i . From the Table 5.1, we see that it costs \$100/MW-hr to interrupt load at each of the buses. We have ignored the cost of adjusting generation levels, but we have included limits on how much each machine can change its real power generation. The upward limit on generator ramping is given by $\Delta_{\max}^+ P_G$, and the downward limit on generator ramping is given by $\Delta_{\max}^- P_G$. We can only change the real power output of generator 1 by 50 MW in the event of a contingency, whereas generator 2 may be changed by 35 MW in the event of a contingency. A generator has a limited time frame over which it must change its generation level to survive the contingency. Over this time frame, the generator has a physical limitation on how much it can change, leading to generator ramping limits.

Figure 5.2 shows the case in which social welfare is maximized while ignoring the possibility of contingencies. In other words, there is a 100% probability of operating in a state where no contingencies have occurred, or the probability of a contingency occurring is 0%. From the figure, we

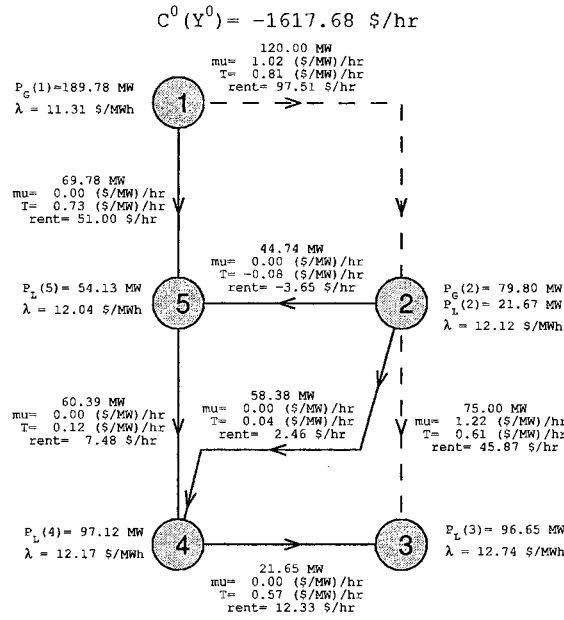


Figure 5.2: Insecure power system operating point for 5-bus case.

can see two lines are operating at maximum capacity as indicated by the dotted line. These are the lines of interest when evaluating contingencies.

For example, suppose we lose the line connecting buses 1 and 2. With no change in generation, the generator at bus 1 would still be producing 189.78 MW which would then be entirely directed to bus 5 leading to a violation of the emergency limit of 110 MW for line 1-5. Therefore, the generator at bus 1 **must** ramp down to prevent the line from exceeding its limits. More precisely, the generator must ramp down by 79.78 MW to keep the line from violating its limits. However, from the table we see that this generator may only ramp down by 50 MW in the event of a contingency. Therefore, the system as dispatched in Figure 5.2 is not secure since a contingency occurrence would lead to violation of a line flow constraint. To make the system secure, we would have to decrease the pre-contingency generation level to at least 160 MW, leaving room for the generator to ramp down by 50 MW to 110 MW, which does not violate the transmission line limit. Adjusting the pre-contingency generation level to make the system secure leads to a social welfare loss of \$40.61

in the pre-contingency operating state.

5.3 Expected value of social welfare

We now wish to consider contingencies in our optimal power flow formulation. The function we maximize should contain social welfare for the normal, pre-contingency operating state and for each of the post-contingency operating states. Note that the post-contingency social welfare function will depend on the pre-contingency operating state since part of our costs are for changing generation from the pre-contingency amount and dropping load that was being served in the pre-contingency state.

The issue to address is how we should form our OPF problem to account for the generator ramping and load interruption security costs. We choose a method as proposed in [6] where the objective function of the optimization problem is the *expected value* of social welfare. The term “expected value” is used in the probabilistic sense since social welfare is a random variable. Its optimal value at any time depends on the configuration of the power system, which will change when some kind of random outage occurs in the form of losing a transmission line or a generator. We refer to these outages as *contingencies*. In this research, we assume mutually exclusive contingency events such that the probability of simultaneous contingencies can be neglected. Expected value of social welfare can then be computed by multiplying the probability of operating in each possible system state by the social welfare obtained by operating in that state and then summing these products. If we denote social welfare for system state k as SW^k and the probability of operating in state k as π^k , the expected value of social welfare is

$$E[SW] = \sum_{k=0}^{N_{\text{cont}}} \pi^k SW^k. \quad (5.5)$$

State “0” is the pre-contingency state, and the remaining states are the post-contingency states.

Consider as an example the 5-bus system whose contingency information is shown in Figure 5.1.

For this system, we see contingencies associated with the loss of transmission lines. The probability of losing a transmission line is π^k . There are seven transmission lines, and we consider the loss of each one giving us seven post-contingency operating states. Line outages are considered mutually exclusive, so we can compute the probability of operating in the normal, pre-contingency operating state as

$$\pi^0 = 1 - \sum_{k=1}^7 \pi^k, \quad (5.6)$$

and the expected value of social welfare is

$$E[SW] = \sum_{k=0}^7 \pi^k SW^k. \quad (5.7)$$

5.4 Security costs

Social welfare as previously presented was the difference between consumer benefits and producer costs. For a power system, social welfare includes benefit from real power consumption and cost of real power generation. For the pre-contingency state, social welfare is computed using this standard formula. However, for the post-contingency states, we have additional costs corresponding to

- generator ramping
- load interruption.

Generator ramping cost depends on

- pre-contingency generation level
- post-contingency generation level

since it is a function of the difference between these two values. We choose simple ramping cost functions:

$$C_R^k(P_G^0, P_G^k) = \beta_R |P_G^k - P_G^0| \quad (5.8)$$

CHAPTER 5. POWER SYSTEM SECURITY

where β_{R+} and β_{R-} are the upward and downward ramping cost coefficient with units \$/MW-hr. We could also choose quadratic ramping cost functions which would depend on the square of the difference between pre-contingency and post-contingency generation. For the 5-bus example, we ignore ramping costs and include only customer load interruption costs which depend on

- pre-contingency consumption
- post-contingency consumption

since these costs are functions of the amount of load dropped when the contingency occurs. A simple load interruption cost function is

$$C_I^k(P_L^0, P_L^k) = \beta_I (P_L^0 - P_L^k) \quad (5.9)$$

where β_I is the load interruption cost coefficient with units \$/MW-hr.

To obtain a secure operating point, we may incur both generator ramping costs and load interruption costs when a contingency occurs. These costs are included in the social welfare function of the post-contingency state. Security costs couple the pre-contingency state and each of the post-contingency states since both generator ramping and load interruption depend on the state of the system before the contingency occurs. Therefore, when we write the post-contingency social welfare, we will have pre-contingency variables in the expression. Post-contingency social welfare includes four terms:

- generation cost
- customer benefit
- generator ramping cost
- customer load interruption cost.

The mathematical expression for social welfare corresponding to the k th contingency is

$$SW^k = \sum_{i \in \mathcal{L}} B_i(P_{Li}^k) - \sum_{i \in \mathcal{G}} C_i(P_{Gi}^k) - \sum_{i \in \mathcal{G}} C_{Ri}^k(P_{Gi}^0, P_{Gi}^k) - \sum_{i \in \mathcal{I}} C_{Li}^k(P_{Li}^0, P_{Li}^k) \quad (5.10)$$

where \mathcal{I} is the set of customers who have agreed to have their load subject to interruption.

5.5 Coupling constraints

In addition to security costs acting to couple the pre-contingency and post-contingency states, we have some constraints which couple pre-contingency and post-contingency states. Some of these constraints are on the amount we are allowed to change a generator's output from the pre-contingency level. Mathematically, these constraints are

$$P_{Gi}^k - P_{Gi}^0 \leq \Delta_{\max}^+ P_{Gi}, \quad i \in \mathcal{G} \quad (5.11)$$

$$P_{Gi}^0 - P_{Gi}^k \leq \Delta_{\max}^- P_{Gi}, \quad i \in \mathcal{G} \quad (5.12)$$

where $\Delta_{\max}^+ P_{Gi}$ is the limit corresponding to the amount we may increase real power output for the generator at bus i . We are also limited to decreasing output of the generator at bus i by at most an amount $\Delta_{\max}^- P_{Gi}$.

The rest of the coupling constraints limit us to only *dropping* customer load as opposed to having the ability to *increase* customer load after a contingency occurs. It would be more difficult for a customer to be asked to increase load since for an industrial customer, this would mean calling in more workers to increase production or by just turning on more equipment to add to the load. Thus we assume we are allowed to drop customer load but not increase it. Mathematically, these constraints are

$$P_{Li}^k \leq P_{Li}^0, \quad i \in \mathcal{I}. \quad (5.13)$$

Note that all constraints limit post-contingency variables and that they all contain their corresponding pre-contingency variables. Thus, each constraint is a function of both the pre-contingency operating state and the post-contingency operating state and therefore acts to couple those states.

5.6 Expected-security-cost optimal power flow (ESCOPF)

In this section, we form the optimization problem which has a solution that describes a power system operating point that maximizes the expected value of social welfare. Expression 5.5 gives the expected value of social welfare based on contingency probabilities and the resulting social welfare of the system during all possible post-contingency operating states. The expression for the value of social welfare during a post-contingency operating condition is given by 5.10, and social welfare for the pre-contingency operating condition is

$$SW^0 = \sum_{i \in \mathcal{L}} B_i(P_{Li}^0) - \sum_{i \in \mathcal{G}} C_i(P_{Gi}^0). \quad (5.14)$$

Let us assume there are N_{cont} mutually exclusive contingencies corresponding to loss of transmission lines and that we operate in a state associated with the k th contingency with probability π^k . Then the probability of operating in the pre-contingency state is

$$\pi^0 = 1 - \sum_{k=1}^{N_{\text{cont}}} \pi^k. \quad (5.15)$$

Substituting the social welfare expressions into 5.5, we have the expected value of social welfare:

$$\begin{aligned} E[SW] = \pi^0 & \left[\sum_{i \in \mathcal{L}} B_i(P_{Li}^0) - \sum_{i \in \mathcal{G}} C_i(P_{Gi}^0) \right] \\ & + \sum_{k=1}^{N_{\text{cont}}} \pi^k \left[\sum_{i \in \mathcal{L}} B_i(P_{Li}^k) - \sum_{i \in \mathcal{G}} C_i(P_{Gi}^k) \right. \\ & \quad \left. - \sum_{i \in \mathcal{G}} C_{Ri}^k(P_{Gi}^0, P_{Gi}^k) - \sum_{i \in \mathcal{I}} C_{Ii}^k(P_{Li}^0, P_{Li}^k) \right]. \quad (5.16) \end{aligned}$$

The objective function of our OPF formulation will be to maximize the expected value of social welfare function. However, in keeping with the standard optimization problem formulation of minimizing a function, we will treat the objective as minimizing the negative of the expected value of social welfare. The constraints for the ESCOPF problem are similar to the constraints for a non-secure OPF formulation; there are constraints on bus voltage magnitudes, generator power production, customer load consumption, and power flows on transmission lines. However, we now have N_{cont} sets of these inequality constraints corresponding to each post-contingency operating point. In addition, we add the coupling constraints: generator ramping and customer load interruption. Finally, we have equality constraints corresponding to the power flow equations for all states. The expected-security-cost optimal power flow problem state mathematically is given in Table 5.2.

In the table, we have some new notation so we can distinguish between pre-contingency and post-contingency variables. The set of buses that have generators is now \mathcal{G}^k with $k = 0$ denoting the set of generator buses for the pre-contingency state and $k = 1 \dots N_{\text{cont}}$ denoting the post-contingency states. Similarly, we denote the set of transmission lines for state k by \mathcal{T}^k .

$$\begin{aligned}
 \min_Y \pi^0 & \left[\sum_{i \in \mathcal{L}} B_i(P_{Li}^0) - \sum_{i \in \mathcal{G}} C_i(P_{Gi}^0) \right] \\
 & + \sum_{k=1}^{N_{\text{cont}}} \pi^k \left[\sum_{i \in \mathcal{L}} B_i(P_{Li}^k) - \sum_{i \in \mathcal{G}} C_i(P_{Gi}^k) \right. \\
 & \quad \left. - \sum_{i \in \mathcal{G}} C_{Ri}^k(P_{Gi}^0, P_{Gi}^k) - \sum_{i \in \mathcal{I}} C_{Li}^k(P_{Li}^0, P_{Li}^k) \right] \quad (5.17)
 \end{aligned}$$

subject to pre-contingency power flow equality constraints

$$\begin{aligned}
 P_{Gi}^0 - P_{Li}^0 &= P_i^0(V^0, \theta^0), \quad i = 1 \dots N_B \\
 Q_{Gi}^0 - Q_{Li}^0 &= Q_i^0(V^0, \theta^0), \quad i = 1 \dots N_B \quad (5.18)
 \end{aligned}$$

post-contingency power flow equality constraints

$$\begin{aligned}
 P_{Gi}^k - P_{Li}^k &= P_i^k(V^k, \theta^k), \quad i = 1 \dots N_B, \quad k = 1 \dots N_{\text{cont}} \\
 Q_{Gi}^k - Q_{Li}^k &= Q_i^k(V^k, \theta^k), \quad i = 1 \dots N_B, \quad k = 1 \dots N_{\text{cont}} \quad (5.19)
 \end{aligned}$$

pre-contingency & post-contingency inequality constraints

$$\begin{aligned}
 V_i^{(0)\text{min}} &\leq V_i^0 \leq V_i^{(0)\text{max}}, \quad i = 1 \dots N_B \\
 V_i^{(k)\text{min}} &\leq V_i^k \leq V_i^{(k)\text{max}}, \quad i = 1 \dots N_B, \quad k = 1 \dots N_{\text{cont}} \\
 |S_l^0(V^0, \theta^0)| &\leq S_l^{(0)\text{max}}, \quad l \in \mathcal{T}^0 \\
 |S_l^k(V^k, \theta^k)| &\leq S_l^{(k)\text{max}}, \quad l \in \mathcal{T}^k, \quad k = 1 \dots N_{\text{cont}} \\
 P_{Gi}^{(0)\text{min}} &\leq P_{Gi}^0 \leq P_{Gi}^{(0)\text{max}}, \quad i \in \mathcal{G}^0 \\
 P_{Gi}^{(k)\text{min}} &\leq P_{Gi}^k \leq P_{Gi}^{(k)\text{max}}, \quad i \in \mathcal{G}^k, \quad k = 1 \dots N_{\text{cont}} \\
 P_{Li}^{(0)\text{min}} &\leq P_{Li}^0 \leq P_{Li}^{(0)\text{max}}, \quad i \in \mathcal{L} \\
 P_{Li}^{(k)\text{min}} &\leq P_{Li}^k \leq P_{Li}^{(k)\text{max}}, \quad i \in \mathcal{L}, \quad k = 1 \dots N_{\text{cont}} \\
 Q_{Gi}^{(0)\text{min}} &\leq Q_{Gi}^0 \leq Q_{Gi}^{(0)\text{max}}, \quad i \in \mathcal{G}^0 \\
 Q_{Gi}^{(k)\text{min}} &\leq Q_{Gi}^k \leq Q_{Gi}^{(k)\text{max}}, \quad i \in \mathcal{G}^k, \quad k = 1 \dots N_{\text{cont}} \quad (5.20)
 \end{aligned}$$

coupling inequality constraints

$$P_{Gi}^k - P_{Gi}^0 \leq \Delta_{\text{max}}^+ P_{Gi}, \quad i \in \mathcal{G}^k \quad (5.21)$$

$$P_{Gi}^0 - P_{Gi}^k \leq \Delta_{\text{max}}^- P_{Gi}, \quad i \in \mathcal{G}^k \quad (5.22)$$

$$P_{Li}^k \leq P_{Li}^0, \quad i \in \mathcal{I}. \quad (5.23)$$

Table 5.2: ESCOPF problem statement.

Chapter 6

Optimization problem solution methods

6.1 Optimization with no inequality constraints

To solve a minimization problem that has equality constraints, we form a Lagrangian function.

Suppose we want to solve the problem

$$\min_Y f(Y) \tag{6.1}$$

subject to

$$h_i(Y) = 0, \quad i = 1 \dots N \tag{6.2}$$

where N is the number of equality constraints and Y is a vector of system variables. The Lagrangian function for this problem is

$$\mathcal{L} = f(Y) + \sum_{i=1}^N z_i h_i(Y). \tag{6.3}$$

When forming the Lagrangian function, a set of new variables was introduced. Each of these variables, z_i , is called a Lagrange multiplier. Now we want to minimize the Lagrangian function.

The first order conditions for optimality are

$$\nabla_Y f + \sum_{i=1}^N z_i \nabla_Y h_i = 0 \quad (6.4)$$

$$h_i(Y) = 0 \quad (6.5)$$

By solving these equations, we obtain the solution to the problem. The advantage to including the Lagrange multipliers is that a Lagrange multiplier indicates how much the objective function will improve if the equality constraint associated with the multiplier is relaxed by one unit. In economic terms, this amount represents the price of the quantity that is relaxed. In the OPF described in the previous sections, the Lagrange multiplier on the equality constraint on the real power injected at each bus due to Kirchoff's laws tells us how much the cost (in dollars per hour) will reduce if we relax the real power injection constraint by one megawatt. So the units on the Lagrange multiplier is $\frac{\$}{MW \cdot hr}$. This is the price of real power at that bus.

6.2 Active set methods

The proposed OPF problem has inequality constraints. To solve a problem with inequality constraints, i.e. $g_i(Y) \leq 0$, we can use an active set method in which certain constraints are assumed binding. Let the set of indices of binding inequality constraints be $\mathcal{N}_{\text{bind}}$. These constraints are then included in the problem as equality constraints:

$$\mathcal{L} = f(Y) + \sum_{i=1}^N z_i h_i(Y) + \sum_{i \in \mathcal{N}_{\text{bind}}} w_i g_i(Y). \quad (6.6)$$

If the solution algorithm converges to a point $\{Y^*, z^*, w^*\}$ we check the KKT conditions, which list the necessary conditions for optimality:

$$\nabla_Y f + J_h^T z^* + J_g^T w^* = 0, \quad (6.7)$$

$$h_i(Y^*) = 0, \quad i = 1 \dots N_1 \quad (6.8)$$

$$w_i^* g_i(Y^*) = 0, \quad i = 1 \dots N_2 \quad (6.9)$$

$$w_i^* \geq 0, \quad i = 1 \dots N_2 \quad (6.10)$$

$$g_i(Y^*) \leq 0, \quad i \in \mathcal{N}_{\text{bind}}, \quad (6.11)$$

where J_h and J_g are equality and inequality constraint Jacobians, respectively. If these conditions are satisfied, then $\{Y^*, z^*, w^*\}$ is a solution candidate. We use the word “candidate” because there may be other power system operating points which also satisfy the KKT conditions. We can attempt to find another solution candidate by choosing a different set of active inequality constraints, solving, and then checking the KKT conditions. After finding all candidates, which requires checking all possible combinations of binding inequality constraints, the solution to the problem is the one which evaluates to the smallest value of the objective function. In practice, we don’t check every possible combination since there are only a few which seem likely candidates. A modified active set approach is to drop and add constraints as the problem is being solved. This, however, may lead to a cycling behavior in which a constraint moves in and out of the set of presumably active constraints. Ad hoc heuristics are sometimes used to try to avoid this problem, causing the algorithm to get stuck.

6.3 Primal-dual interior-point method

To avoid having to guess which constraints are binding, we can use a primal-dual interior-point (PDIP) method [7]. The basic idea behind PDIP and a basic implementation of PDIP are presented here. Variations to the PDIP algorithm applied to the nonlinear OPF problem have been investigated

as summarized in [8]. The problem is stated with all inequality constraints included as

$$\min_Y f(Y) \quad (6.12)$$

subject to

$$h_i(Y) = 0, \quad i = 1 \dots N_1, \quad (6.13)$$

$$g_i(Y) \leq 0, \quad i = 1 \dots N_2, \quad (6.14)$$

To solve the problem, we first form the barrier function

$$B = f(Y) + \sum_{i=1}^{N_1} z_i h_i(Y) - \mu \sum_{i=1}^{N_2} \ln(-g_i(Y)). \quad (6.15)$$

Now our problem is to minimize the barrier function:

$$\min_Y B \quad (6.16)$$

We need to find the gradient of B with respect to Y :

$$\begin{aligned} \nabla_Y B &= \nabla_Y f(Y) + \sum_{i=1}^{N_1} z_i \nabla_Y h_i(Y) + \mu \sum_{i=1}^{N_2} \frac{1}{-g_i(Y)} \nabla_Y g_i(Y) \\ &= \nabla_Y f(Y) + \sum_{i=1}^{N_1} z_i \nabla_Y h_i(Y) + \mu \sum_{i=1}^{N_2} \frac{1}{s_i} \nabla_Y g_i(Y) \\ &= \nabla_Y f(Y) + \sum_{i=1}^{N_1} z_i \nabla_Y h_i(Y) + \sum_{i=1}^{N_2} w_i \nabla_Y g_i(Y) \end{aligned} \quad (6.17)$$

$$= \nabla_Y f + J_h^T z + J_g^T w \quad (6.18)$$

where

$$-g_i(Y) = s_i, \quad i = 1 \dots N_2 \quad (6.19)$$

$$w_i s_i = \mu, \quad i = 1 \dots N_2, \quad (6.20)$$

J_h^T is a matrix consisting of the gradients $\nabla_Y h_i(Y)$ as columns, and J_g^T is a matrix consisting of the gradients $\nabla_Y g_i(Y)$ as columns, or

$$J_h^T = [\nabla_Y h_1(Y) \quad \nabla_Y h_2(Y) \quad \dots \quad \nabla_Y h_{N_1}(Y)] \quad (6.21)$$

$$J_g^T = [\nabla_Y g_1(Y) \quad \nabla_Y g_2(Y) \quad \dots \quad \nabla_Y g_{N_2}(Y)]. \quad (6.22)$$

Also, z is a vector containing the elements z_i , and w is a vector containing the elements w_i . The scalar μ is called the *barrier parameter*. The set of equations we must solve is

$$\nabla_Y f + J_h^T z + J_g^T w = 0 \quad (6.23)$$

$$h_i(Y) = 0, \quad i = 1 \dots N_1 \quad (6.24)$$

$$g_i + s_i = 0, \quad i = 1 \dots N_2 \quad (6.25)$$

$$w_i s_i = \mu, \quad i = 1 \dots N_2. \quad (6.26)$$

These equations may also be written as

$$\nabla_Y f + J_h^T z + J_g^T w = 0 \quad (6.27)$$

$$h = 0 \quad (6.28)$$

$$g + s = 0 \quad (6.29)$$

$$W S e = \mu e \quad (6.30)$$

CHAPTER 6. OPTIMIZATION PROBLEM SOLUTION METHODS

where g , s , and h are column vectors containing the elements g_i, s_i , and h_i respectively. M is an $N_2 \times N_2$ diagonal matrix containing w_i on the diagonal, S is an $N_2 \times N_2$ diagonal matrix containing s_i on the diagonal, and e is a column vector whose N_2 elements are all ones.

These equations can be solved using the Newton iterative method. Our three sets of variables for which we must solve are Y , z , w , and s . The equation for a first order approximation of a Taylor series of a function, F , whose independent variables are Y, w, z , and s is

$$F(Y, z, w, s) = F(Y^0, z^0, w^0, Y^0) + \frac{\partial F}{\partial Y} \Delta Y + \frac{\partial F}{\partial z} \Delta z + \frac{\partial F}{\partial w} \Delta w + \frac{\partial F}{\partial s} \Delta s \quad (6.31)$$

We want to find the values of Y , z , w , and s where the expressions on the left side of the equations we want to solve evaluate to zero.

Taking the first order approximation to the Taylor series for each of the three expressions given in equations 6.27-6.30 and setting them equal to zero (the desired value for each) gives us

$$\begin{aligned} (\nabla_Y f + J_h^T z + J_g^T w) + (\nabla_Y^2 f + \sum_{i=1}^{N_1} z_i \nabla_Y^2 h_i + \sum_{i=1}^{N_2} w_i \nabla_Y^2 g_i) \Delta Y \\ + J_h^T \Delta z + J_g^T \Delta w = 0 \end{aligned} \quad (6.32)$$

$$h + J_h \Delta Y = 0 \quad (6.33)$$

$$(g + s) + J_g \Delta Y + I \Delta s = 0 \quad (6.34)$$

$$(WSe - \mu e) + S \Delta w + W \Delta s = 0 \quad (6.35)$$

The above equations can be written in matrix form as

$$\begin{bmatrix} \nabla_Y^2 f + \sum_{i=1}^{N_1} z_i \nabla_Y^2 h_i + \sum_{i=1}^{N_2} w_i \nabla_Y^2 g_i & J_h^T & J_g^T & 0 \\ & J_h & 0 & 0 & 0 \\ & J_g & 0 & 0 & I \\ & 0 & 0 & S & W \end{bmatrix} \begin{bmatrix} \Delta Y \\ \Delta z \\ \Delta w \\ \Delta s \end{bmatrix} = \begin{bmatrix} -f_Y - J_h^T z - J_g^T w \\ -h \\ -g - s \\ \mu e - WSe \end{bmatrix} \quad (6.36)$$

or

$$M\Delta v = \Delta y. \quad (6.37)$$

We initially set μ to some relatively large number, such as 10. Starting with an initial guess

$$v^0 = \begin{bmatrix} Y^0 \\ z^0 \\ w^0 \\ s^0 \end{bmatrix} \quad (6.38)$$

we calculate the M matrix and the Δy vector. If all the elements of Δy are sufficiently close to zero, we have found the solution (v^0). Otherwise we must solve the system of linear equations $M\Delta v = \Delta y$ for Δv : The original v is then updated:

$$v = v^0 + \alpha \kappa \Delta v \quad (6.39)$$

where the scaling coefficients α and κ will be discussed later.

An important thing to remember when updating v is the following:

$$w_i > 0 \tag{6.40}$$

$$s_i > 0 \tag{6.41}$$

Therefore, when calculating the updated w and s , we must make sure that each w_i and each s_i is still greater than zero when Δw_i and Δs_i is added to it. If adding Δw_i or Δs_i violates this condition, all Δw_i and all Δs_i must be scaled by some factor less than one before adding them.

Now that we have a new guess for v , the W matrix and Δy vector are calculated again. If the elements of the Δy vector are sufficiently close to zero, we have found the v which solves our problem. Otherwise we solve for Δv and update v again. This process is repeated until we find the v which makes Δy very close to zero.

After we solve for v , we reduce the barrier parameter μ by some factor which we will call ρ . For example, if $\rho = 0.4$, we would change μ by letting the new μ be 0.4 times the old μ . Using the value of v obtained using the old μ as an initial guess v^0 , we use the iterative procedure described above to solve for the value of v which makes Δy approximately zero for the new barrier parameter. The process of solving for v , decreasing μ , and then solving for v again is repeated until μ becomes a very small number, such as 10^{-4} . When μ gets this small, we have found the v that solves our problem.

When solving for v given a particular μ , it is not necessary to force Δy to be zero. What we really want to know is how close we are to the *central path*. The central path is defined by the solutions of v which make Δy evaluate to zero for every possible value of μ . One way of measuring how close we are to the central path is by checking to see how close each product $w_i s_i$ is to the barrier parameter μ . One proposed way of doing this is by first calculating the average value of $w_i s_i$, or

$$\sum_{i=1}^N \frac{w_i s_i}{N} \tag{6.42}$$

which evaluates to μ /when each product $w_i s_i$ is equal to μ . Since each w_i and s_i is greater than

zero, their average must be greater than or equal to μ and is equal to μ only when each product $w_i s_i = \mu$. Instead of checking to see if Δy is sufficiently close to zero, we check to see when the following is true:

$$\left\| WSe - \sum_{i=1}^N \frac{w_i s_i}{N} \right\|_2 \leq \tau \mu \quad (6.43)$$

where a typical value for the parameter τ is 0.5 and where $\| \cdot \|_2$ refers to the 2-norm. The 2-norm for a vector \mathbf{x} consisting of n elements is

$$\| \mathbf{x} \|_2 = \sqrt{\sum_{i=1}^n x_i^2}. \quad (6.44)$$

When this logical statement evaluates to true, we are sufficiently close to the central path, and we can reduce μ .

It should be emphasized that convergence to a solution when dealing with non-linear equations is uncertain when applying the primal-dual interior-point method. Staying close to the central path is our only hope of finding a solution, but the solution we find may not be the stationary point corresponding to the global solution. These are important issues that were not considered when writing this paper, and they are issues that must be further explored.

6.4 PDIP algorithm

We next list each step of the solution process.

1. Choose an initial operating point. This may be done by using system data to do an initial load flow. After doing the load flow, all bus voltage magnitudes and angles will be known. The apparent power generated and consumed at each bus will also be known after solving the loadflow.
2. Pick an initial value for the barrier parameter μ . The value should not be too small. I usually start with a value of 10.

CHAPTER 6. OPTIMIZATION PROBLEM SOLUTION METHODS

3. Calculate the eigenvalues and eigenvectors for the current state.
4. Form the J_h , $\nabla_Y^2 h_i$, J_g , $\nabla_Y^2 g_i$, $\nabla_Y f$, and $\nabla_Y^2 f$ matrices.
5. Form the W matrix.
6. Solve $W\Delta v = \Delta y$ for Δv .
7. Find $\alpha = \text{minimum}(\alpha_s, \alpha_w, 1)$. The scaling factor α is multiplied by Δv before updating v .

We must use the scaling factor so that s_i and w_i remain positive when they are updated. The scaling factor must satisfy the following inequality:

$$0 < \alpha \leq 1$$

The values of α_s and α_w can be calculated as follows:

$$\begin{aligned} s_i^{\text{new}} &= s_i^{\text{old}} + \alpha_s \Delta s_i > 0, & i = 1 \dots N_2 \\ w_i^{\text{new}} &= w_i^{\text{old}} + \alpha_w \Delta w_i > 0, & i = 1 \dots N_2 \end{aligned}$$

Since we want s_i to be greater than zero, we only need to look at the indices i where $\Delta s_i < 0$.

The value we choose for α must satisfy the following inequalities:

$$\begin{aligned} \alpha < \alpha_{s_i} &= -\frac{s_i^{\text{old}}}{\Delta s_i} & \forall i \mid \Delta s_i < 0 \\ \alpha < \alpha_{w_i} &= -\frac{w_i^{\text{old}}}{\Delta w_i} & \forall i \mid \Delta w_i < 0 \end{aligned}$$

We want to choose α such that it is less than or equal to one and also that it is less than the minimum of α_s and α_w . Also, we want to make sure each s_i^{new} and each w_i^{new} is at least slightly greater than zero, so we multiply alpha by another factor κ that is slightly less than

one when we update v . The scaling factor α is then calculated.

$$\begin{aligned} \alpha &= \text{minimum}(\alpha_{s_i}, \alpha_{w_j}, 1) \\ \forall i, j \mid \Delta s_i < 0 \text{ and } \Delta w_j < 0. \end{aligned} \tag{6.45}$$

8. Update v

$$v^{\text{new}} = v^{\text{old}} + \kappa\alpha\Delta v. \tag{6.46}$$

9. Check to see if we are sufficiently close to the central path using

$$\left\| WSe - \frac{1}{N} \sum_{i=1}^N w_i s_i \right\|_2 \leq \tau \mu. \tag{6.47}$$

If the expression evaluates to false, we go back to step 3. Otherwise continue to the next step.

10. If the barrier parameter μ is sufficiently close to zero, then we stop iterating and use the current state defined by v as the solution. Otherwise, we let $\mu_{\text{new}} = \rho \mu$ where ρ is a scaling factor such that $0 < \rho < 1$.

To summarize, one way to tune the OPF is by adjusting the following factors: μ_0 , κ , τ , and ρ from the following expressions

$$\begin{aligned} v^{\text{new}} &= v^{\text{old}} + \kappa\alpha\Delta v \\ \left\| WSe - \frac{1}{N} \sum_{i=1}^N w_i s_i \right\|_2 &\leq \tau \mu \\ \mu_{\text{new}} &= \rho \mu \end{aligned}$$

6.5 Solution to OPF Problem

The solution method just described can be applied to the OPF problem. We need to identify the variables and functions identified in the previous subsection. These functions and variables are

Y : the vector of independent variables

$f(Y)$: the function we wish to minimize

$h_i(Y)$: equality constraint function

$g_i(Y)$: inequality constraint function

Let Y be the vector of independent variables that give the steady-state operating point. This vector includes but is not limited to the following variables which describe the steady state operating point of the system:

V : the vector of bus voltage magnitudes

θ : the vector of bus voltage phase angles

P_G : the vector containing the amount of real power generated
at each generation bus

P_L : the vector containing the amount of real power consumed
at each load bus

Q_G : the vector containing the amount of reactive power generated
at each generation bus

There are equality constraints corresponding to the loadflow equations. The number of equality constraints, which we have previously denoted N_1 , is equal to $2N_B$ where N_B is the number of buses in the system. There are N_B constraints corresponding to the real power injected at each bus and N_B constraints corresponding to the reactive power injected at each bus. So the number of equality constraints is

$$N_1 = 2N_B \quad (6.48)$$

Now we will calculate the number of inequality constraints and make some clarifications with regard to some of the inequality constraint functions. There are $2N_B$ inequality constraints due to maximum and minimum bus voltage magnitude limits. We have $2N_G$ inequality constraints due to maximum and minimum real power generation limits and N_L inequality constraints due to minimum real power consumption constraints, where N_G is the number of generators in the system and N_L is the number of loads in the system. There are $2N_S$ inequality constraints due to the limit on the magnitude of the apparent power that can be sent down a transmission line, where N_S is the number of transmission lines in the system. There are $2N_S$ constraints because for a transmission line that connects buses i and j , there is a limit on the apparent power sent from bus i to bus j and a limit on the apparent power sent from bus j to bus i . This gives us 2 constraints for each line. Therefore, the total number of inequality constraints is

$$N_2 = 2N_B + 2N_G + N_L + 2N_S \quad (6.49)$$

Chapter 7

Synchronous machine model

7.1 Principles of operation

A synchronous generator provides electrical power to the system at a specified voltage. The basic components of a synchronous machine are the

- stator
- rotor.

The stator is the stationary part of the machine, and the rotor is the part that turns. Both the stator and rotor consist of windings, which are coils of wire wrapped around some ferromagnetic material. The endpoints of the coils are called terminals. The main winding on the rotor is called the field winding. By applying a DC voltage to the rotor field winding terminals, we set up a constant magnetic field. When the rotor turns, the stator winding sees a magnetic field that is changing. This changing magnetic field induces a sinusoidal voltage at the stator winding terminals. The frequency of this sinusoidal voltage depends on how fast the rotor is spinning. The convention in the U.S. is to have a sinusoidal voltage with a frequency of 60 Hz. Since the frequency of the voltage waveform is directly proportional to the shaft rotational speed, we know how fast the rotor must turn to produce a given electrical frequency. Also, the magnitude of the stator terminal sinusoidal voltage depends on the voltage applied to the rotor field winding.

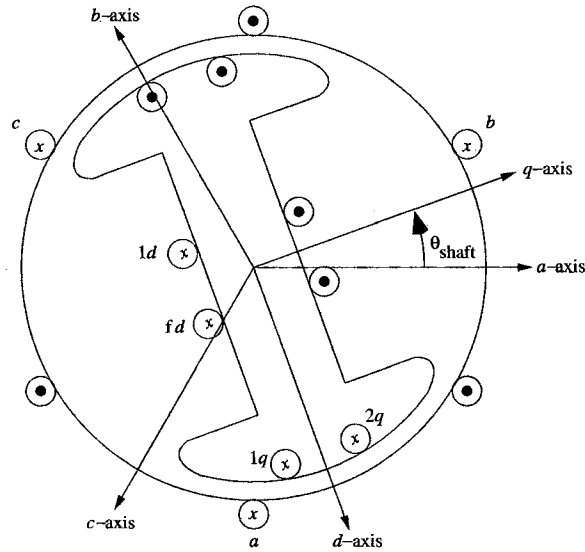


Figure 7.1: 2-pole synchronous machine schematic

7.2 Basic model

Rotor field and damper windings

The synchronous machine can be modeled as shown in figure 7.1 with four sets of windings on the salient pole rotor. The term “salient pole” refers to the dog-bone shape of the rotor as opposed to a round-shaped rotor found on other types of synchronous machines. The windings on the rotor include three sets of damper windings and a set of field windings. Recall that the purpose of the field windings is to produce the magnetic field that cuts the stator winding and results in an induced sinusoidal voltage at the synchronous machine terminals. Quantities related to field windings will be labeled with an “ fd ” subscript. When a DC voltage is applied to these windings, a DC current will flow resulting in a magnetic field pointing in the “d-axis” direction as indicated in figure 7.1. The d-axis direction always points along the rotor as shown in the figure.

The damper windings, as indicated by the name, are added to damp oscillations in synchronous machine dynamics. There is one set of damper windings in parallel with the field windings on the synchronous machine rotor. This set of windings will be labeled “ $1d$ ”. Magnetic fields created by

$1d$ damper winding point along the d-axis just as the fields created the field winding.

Another set of damper windings are wound perpendicular to the field and $1d$ damper windings. These loops of conducting material, called the $1q$ and $2q$ damper windings, produce magnetic fields perpendicular to the field and $1d$ damper windings. Their fields point in the direction of the “q-axis” as shown in figure 7.1. The q-axis always points in a direction perpendicular to the d-axis.

For a round-rotor machine, we can model the rotor circuits in the same way. However, in this case, the $1q$ and $2q$ rotor damper windings are just the metallic rotor itself.

Stator windings

In addition to the field and damper windings, there are loops of conducting material on the stator. There are at least three windings corresponding to three phase voltages and currents that are produced at the terminals of the windings. These three windings will be labeled “a”, “b”, and “c”. They are arranged 120 mechanical degrees apart for a 2-pole machine so that the resulting induced sinusoidal voltages at their terminals are 120 degrees out of phase.

Kirchhoff’s voltage law applied to windings

We can write expressions for each of the windings that indicate the basic Kirchhoff’s voltage law of electricity which says that the sum of the voltages in a conducting loop must add up to zero. There are three voltages that occur in the loop: voltage due to the changing magnetic flux linking the loop, a voltage drop across the electrical resistance of the loop, and the voltage applied to the loop terminals. The flux linking the loop is related to the direction of the magnetic field in relation to the loop and the path along which the magnetic field passes as it cuts the loop. These relationships will be discussed later. For now, we simply assume there is a flux cutting the loop and that this flux is changing with time leading to a voltage induced in the loop. For each loop then, the differential

equation we can write sums these three voltage terms:

$$-v_{\text{term}} + i_{\text{loop}}r_{\text{loop}} + \frac{d\lambda_{\text{loop}}}{dt} = 0. \quad (7.1)$$

The voltage at the winding terminals is v_{term} , and the current flowing through the loop is i_{loop} . Flux linking the loop is λ_{loop} , and the electrical resistance of the loop is given by r_{loop} . The signs are a result of the assumed polarity of the voltages.

Figure 7.1 shows a 2-pole machine, as indicated by only two fat ends of the salient pole rotor. In general, there may be more the 2 poles. A 4-pole rotor would have four fat ends and would be in the shape of a cross. When more poles are added to the rotor, we also add more sets of stator windings. For the 2-pole machine in the figure, there is one set of windings for each phase. In this configuration, the frequency ω of the sinusoidal voltage induced in the stator windings is the same as the angular speed ω_{shaft} of the rotor. For a 4-pole machine, there are two sets of windings for each phase, and these windings are connected such that the frequency of the voltage induced for that phase is twice the angular speed of the rotor. Thus, when there are more poles on the rotor, it takes a lower rotor speed to produce a 60 Hz electrical frequency. For a rotor that has P poles giving the electrical speed or frequency of the stator sinusoidal voltage is

$$\omega = \frac{P}{2} \omega_{\text{shaft}}. \quad (7.2)$$

Newton's laws

The final set of differential equations is related to Newton's laws applied to a rotating rigid body. Rotor speed will change if the mechanical torque applied to the rotor shaft is not balanced by the "electrical" torque plus losses. Another way to look at this rule is that the rotor accelerates if the mechanical power applied to the rotor is not the same as the electrical power output at the stator

terminals. In mathematical form, this rule is

$$J \frac{d\omega_{\text{shaft}}}{dt} = T_m - T_e - T_{fw} \quad (7.3)$$

$$J \frac{2}{P} \frac{d\omega}{dt} = T_m - T_e - T_{fw} \quad (7.4)$$

where J is the inertia constant of the rotor, P is the number of poles, T_m is the mechanical torque applied to the rotor, T_e is the “electrical” torque corresponding to magnetic field interaction, and T_{fw} is the friction and windage losses that must be overcome to turn the rotor. In the 7.4, we write the rotational acceleration equation in terms of electrical frequency as opposed to shaft speed. One more differential equation relates angular position to angular frequency. Just as when we integrate linear velocity to obtain linear position, we can integrate angular velocity to obtain angular position. Of course, there is a constant term corresponding to a reference angular position. We choose as a reference the position corresponding to the direction of the magnetic field created by the stator a -winding as indicated by the a -axis in the figure. In differential equation form, this relationship can be written as

$$\frac{d\theta_{\text{shaft}}}{dt} = \omega_{\text{shaft}} \quad (7.5)$$

$$\frac{d\theta_{\text{shaft}}}{dt} = \frac{2}{P} \omega \quad (7.6)$$

The equations given by Kirchoff’s, Faraday’s, and Newton’s laws are given in Table 7.1.

7.3 Transformation and scaling

At steady-state operation, the synchronous machine will have sinusoidal voltages and currents of the form $v(t) = V_s \cos(\omega_s t)$ where V_s is the amplitude of the sine wave and $\omega_s = 2\pi(60)$ is its frequency in radians per second corresponding to 60 Hz electrical frequency. We have three sets of stator terminal windings distributed 120 degrees from one another around the stator. At steady-state, we

have balanced three-phase sinusoidal voltages of the form

$$\begin{bmatrix} v_a(t) \\ v_b(t) \\ v_c(t) \end{bmatrix} = \begin{bmatrix} \sqrt{2}V_s \cos(\omega_s t) \\ \sqrt{2}V_s \cos(\omega_s t - \frac{2\pi}{3}) \\ \sqrt{2}V_s \cos(\omega_s t + \frac{2\pi}{3}) \end{bmatrix} \quad (7.7)$$

We use Park's transformation [9] to transform these sinusoidal steady-state voltages into constant states via multiplication by the matrix

$$\begin{bmatrix} v_d(t) \\ v_q(t) \\ v_0(t) \end{bmatrix} = T_{dqo} \begin{bmatrix} v_a(t) \\ v_b(t) \\ v_c(t) \end{bmatrix} \quad (7.8)$$

$$= \frac{2}{3} \begin{bmatrix} \sin \frac{P}{2} \theta_{\text{shaft}} & \sin (\frac{P}{2} \theta_{\text{shaft}} - \frac{2\pi}{3}) & \sin (\frac{P}{2} \theta_{\text{shaft}} + \frac{2\pi}{3}) \\ \cos \frac{P}{2} \theta_{\text{shaft}} & \cos (\frac{P}{2} \theta_{\text{shaft}} - \frac{2\pi}{3}) & \cos (\frac{P}{2} \theta_{\text{shaft}} + \frac{2\pi}{3}) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} v_a(t) \\ v_b(t) \\ v_c(t) \end{bmatrix} \quad (7.9)$$

The result is

$$\begin{bmatrix} v_d(t) \\ v_q(t) \\ v_0(t) \end{bmatrix} = \begin{bmatrix} V_s \sin(\theta - \omega_s t) \\ V_s \cos(\theta - \omega_s t) \\ 0 \end{bmatrix} \quad (7.10)$$

where $\theta = \frac{P}{2} \theta_{\text{shaft}}$ is the electrical rotor angle in radians. We say the resulting quantities are in the “ dqo ” reference frame. This means we view the quantities from a reference frame that rotates at synchronous speed rather than from a stationary reference frame. Stator voltages in the dqo reference frame are constant when the rotor is moving at synchronous speed. When the machine deviates from synchronous speed such as when some disturbance occurs in the system, these quantities will change. All stator quantities will be converted to the rotating reference frame including stator winding flux linkages and currents. We then have $[i_d i_q i_o]^T = T_{dqo} [i_a i_b i_c]^T$ and $[\lambda_d \lambda_q \lambda_o]^T = T_{dqo} [\lambda_a \lambda_b \lambda_c]^T$. To

$\frac{d\lambda_a}{dt} = -i_a r_s + v_a$	(7.11)
$\frac{d\lambda_b}{dt} = -i_b r_s + v_b$	(7.12)
$\frac{d\lambda_c}{dt} = -i_c r_s + v_c$	(7.13)
$\frac{d\lambda_{fd}}{dt} = -i_{fd} r_{fd} + v_{fd}$	(7.14)
$\frac{d\lambda_{1d}}{dt} = -i_{1d} r_{1d} + v_{1d}$	(7.15)
$\frac{d\lambda_{1q}}{dt} = -i_{1q} r_{1q} + v_{1q}$	(7.16)
$\frac{d\lambda_{2q}}{dt} = -i_{2q} r_{2q} + v_{2q}$	(7.17)
$\frac{d\theta_{\text{shaft}}}{dt} = \frac{2}{P} \omega$	(7.18)
$J \frac{2}{P} \frac{d\omega}{dt} = T_m - T_e - T_{fw}$	(7.19)

Table 7.1: Basic synchronous machine equations.

convert back to the stationary reference frame, we multiply the rotating reference frame quantities by the inverse of the Park's transformation matrix.

We need to take care of a few more details before scaling the equations to eliminate units. The electrical torque term in Newton's equation for rotational acceleration of a rigid body can be computed by performing a power and energy balance. It then becomes a function of flux linking the stator windings referred to the rotating d -axis and q -axis and a function of stator currents referred to the d -axis and q -axis:

$$T_e = - \left(\frac{3}{2} \right) \left(\frac{P}{2} \right) (\lambda_d i_q - \lambda_q i_d). \quad (7.20)$$

Also, we can define an angle that is constant for constant shaft speed as

$$\delta = \frac{P}{2} \theta_{\text{shaft}} - \omega_s t \quad (7.21)$$

$$= \theta - \omega_s t. \quad (7.22)$$

where ω_s is the rated synchronous speed in electrical radians per second. The angle δ is the angle that appeared in the dqo quantities of equation 7.10 after scaling sinusoidal functions via Park's transformation. From equation 7.6, we can see that the angle θ can be obtained by integrating the speed ω . Therefore, δ will only change when rotor speed is not the synchronous speed.

Next we scale the variables using the concept of per-unit [4]. In the per-unit system, we eliminate units making it easier to see if quantities are normal or if they are wildly varying from what they should be. For example, a normal voltage magnitude in the system might be 230 kV. If we see a magnitude of 250 kV, we would like to know by what percent this exceeds the normal value. By converting to per-unit, 230 kV becomes 1.0 per-unit and 250 kV becomes 1.087 per-unit. It's now easy to see that the voltage is 8.7 % above its nominal value. Each section of the system has a different voltage base, but the entire system has a single power base.

The scale factors that convert synchronous machine quantities to per-unit are given in [10]. In most cases, the scaled variable is written with an upper-case letter. However, flux linkages λ become fluxes ψ after scaling. Also, the inertia constant J is scaled to define a new constant H . The scaled and transformed set of equations are given in Table 7.2

7.4 Linear magnetic circuit

Recall that magnetic fields are set up due to current flowing in loops of conducting material. On the stator, we have windings with current flowing that set up magnetic fields. Windings for damping and for setting up the magnetic field to induce stator sinusoidal voltages are found on the rotor. Also, we said that the voltage induced in a winding depended on

- magnetic field intensity
- direction of magnetic field with respect to winding
- material in which the magnetic field exists.

$$\frac{1}{\omega_s} \frac{d\psi_d}{dt} = R_s I_d + \omega \psi_q + V_d \quad (7.23)$$

$$\frac{1}{\omega_s} \frac{d\psi_q}{dt} = R_s I_q - \omega \psi_d + V_q \quad (7.24)$$

$$\frac{1}{\omega_s} \frac{d\psi_o}{dt} = R_s I_o + V_o \quad (7.25)$$

$$\frac{1}{\omega_s} \frac{d\psi_{fd}}{dt} = -R_{fd} I_{fd} + V_{fd} \quad (7.26)$$

$$\frac{1}{\omega_s} \frac{d\psi_{1d}}{dt} = -R_{1d} I_{1d} + V_{1d} \quad (7.27)$$

$$\frac{1}{\omega_s} \frac{d\psi_{1q}}{dt} = -R_{1q} I_{1q} + V_{1q} \quad (7.28)$$

$$\frac{1}{\omega_s} \frac{d\psi_{2q}}{dt} = -R_{2q} I_{2q} + V_{2q} \quad (7.29)$$

$$\frac{d\delta}{dt} = \omega - \omega_s \quad (7.30)$$

$$\frac{2H}{\omega_s} \frac{d\omega}{dt} = T_M - (\psi_d I_q - \psi_q I_d) - T_{FW} \quad (7.31)$$

Table 7.2: Transformed and scaled synchronous machine equations.

Actually, all three of these determine the flux linking the winding, and it is this changing flux that results in induced voltage in the winding. Magnetic field intensity is related to current flowing in the windings. Since there is current flowing in stator windings and in rotor windings, there will be a magnetic field created by each of these windings. This magnetic field will point in a direction determined by the orientation of the winding which created the field. For the rotor windings, the direction will change as the rotor turns. Magnetic fields created by the stator will only alternate in sign; the field direction will change by 180° due to a sinusoidal current, *i.e.* a current which changes sign. The sign-changing nature of the sinusoidal stator currents is due to the rotating magnetic field winding on the rotor. So all magnetic fields are related to the spinning rotor. Also, as the rotor moves, the material through which the field passes will change (note the shape of the rotor). In light of this dependence of flux linking the windings on rotor movement and winding current, we assume this flux is a linear function of current with coefficients in the linear expression that depend on rotor movement or more specifically, the rotor angle θ . Each coefficient in the linear function is called an

inductance. The expression relating flux and current written in matrix notation is

$$\begin{bmatrix} \lambda_a(t) \\ \lambda_b(t) \\ \lambda_c(t) \end{bmatrix} = L_{ss}(\theta_{\text{shaft}}) \begin{bmatrix} i_a(t) \\ i_b(t) \\ i_c(t) \end{bmatrix} + L_{sr}(\theta_{\text{shaft}}) \begin{bmatrix} i_{fd}(t) \\ i_{1d}(t) \\ i_{1q}(t) \\ i_{2q}(t) \end{bmatrix} \quad (7.32)$$

where the matrices $L_{ss}(\theta_{\text{shaft}})$ and $L_{sr}(\theta_{\text{shaft}}) = L_{rs}^T(\theta_{\text{shaft}})$ have inductance elements which are functions of the rotor angle θ_{shaft} . We use Park's transformation to eliminate dependence on θ_{shaft} , and we scale to get per-unit quantities. The result is

$$\begin{bmatrix} \psi_d \\ \psi_{fd} \\ \psi_{1d} \\ \psi_q \\ \psi_{1q} \\ \psi_{2q} \\ \psi_o \end{bmatrix} = \begin{bmatrix} -X_d & X_{md} & X_{md} & 0 & 0 & 0 & 0 \\ -X_{md} & X_{fd} & X_{md} & 0 & 0 & 0 & 0 \\ -X_{md} & X_{md} & X_{1d} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -X_q & X_{mq} & X_{mq} & 0 \\ 0 & 0 & 0 & -X_{mq} & X_{1q} & X_{mq} & 0 \\ 0 & 0 & 0 & -X_{mq} & X_{mq} & X_{2q} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -X_{ls} \end{bmatrix} \begin{bmatrix} I_d \\ I_{fd} \\ I_{1d} \\ I_q \\ I_{1q} \\ I_{2q} \\ I_o \end{bmatrix} \quad (7.33)$$

In this form, we can see the relationship between d -axis rotor currents and stator flux for different windings involves the mutual d -axis reactance X_{md} . Also, the mutual reactance X_{mq} is present in the relationship between q -axis rotor currents and stator flux of different q -axis windings.

Our goal is to eliminate some variables from the differential equations of Table 7.2. Since we now have flux linkages in terms of currents, we will eliminate the rotor current variables I_{fd} , I_{1d} , I_{1q} , and I_{2q} . We will keep the rotor flux linkages as state variables, and after scaling, some of these flux variables along with the field voltage are converted to different variable names:

$$\psi_{fd} \Rightarrow E'_q, \quad V_{fd} \Rightarrow E_{fd}, \quad \psi_{1q} \Rightarrow E'_d. \quad (7.34)$$

CHAPTER 7. SYNCHRONOUS MACHINE MODEL

Therefore, some equations of 7.33 are used to eliminate current variables. After rearranging and lumping the reactances together, these equations are

$$I_{fd} = \frac{1}{X_{md}} [E'_q + (X_d - X'_d)(I_d - I_{1d})] \quad (7.35)$$

$$I_{1d} = \frac{X'_d - X''_d}{(X'_d - X_{ls})^2} [\psi_{1d} + (X'_d - X'_{ls})I_d - E'_q] \quad (7.36)$$

$$I_{1q} = \frac{1}{X_{mq}} [-E'_d + (X_q - X'_q)(I_q - I_{2q})] \quad (7.37)$$

$$I_{2q} = \frac{X'_q - X''_q}{(X'_q - X_{ls})^2} [\psi_{2q} + (X'_q - X'_{ls})I_q - E'_d] \quad (7.38)$$

where the new lumped d -axis transient and subtransient reactances, respectively, are X'_d and X''_d , and the new lumped q -axis transient and subtransient reactances, respectively, are X'_q and X''_q . The rest of the equations are kept as algebraic equations:

$$\psi_d = -X''_d I_d + \frac{(X''_d - X_{ls})}{(X'_d - X_{ls})} E'_q + \frac{(X'_d - X''_d)}{(X'_d - X_{ls})} \psi_{1d} \quad (7.39)$$

$$\psi_q = -X''_q I_q - \frac{(X''_q - X_{ls})}{(X'_q - X_{ls})} E'_d + \frac{(X'_q - X''_q)}{(X'_q - X_{ls})} \psi_{2q} \quad (7.40)$$

$$\psi_o = -X_{ls} I_o \quad (7.41)$$

Making substitutions into the differential equations to eliminate all rotor currents and adding the flux algebraic equations, we obtain the results shown in Table 7.3.

$\frac{1}{\omega_s} \frac{d\psi_d}{dt} = R_s I_d + \frac{\omega}{\omega_s} \psi_q + V_d$	(7.42)
$\frac{1}{\omega_s} \frac{d\psi_q}{dt} = R_s I_q - \frac{\omega}{\omega_s} \psi_d + V_q$	(7.43)
$\frac{1}{\omega_s} \frac{d\psi_o}{dt} = R_s I_o + V_o$	(7.44)
$T'_{do} \frac{dE'_q}{dt} = -E'_q - (X_d - X'_d) \left[I_d - \frac{X'_d - X''_d}{(X'_d - X_{ls})^2} (\psi_{1d} + (X'_d - X'_{ls}) I_d - E'_q) \right] + E_{fd}$	(7.45)
$T''_{do} \frac{d\psi_{1d}}{dt} = -\psi_{1d} + E'_q - (X'_d - X_{ls}) I_d$	(7.46)
$T'_{qo} \frac{dE'_d}{dt} = -E'_d + (X_q - X'_q) \left[I_q - \frac{X'_q - X''_q}{(X'_q - X_{ls})^2} (\psi_{2q} + (X'_q - X'_{ls}) I_q - E'_d) \right]$	(7.47)
$T''_{qo} \frac{d\psi_{2q}}{dt} = -\psi_{2q} - E'_d - (X'_q - X_{ls}) I_q$	(7.48)
$\frac{d\delta}{dt} = \omega - \omega_s$	(7.49)
$\frac{2H}{\omega_s} \frac{d\omega}{dt} = T_M - (\psi_d I_q - \psi_q I_d) - T_{FW}$	(7.50)
$\psi_d = -X''_d I_d + \frac{(X''_d - X_{ls})}{(X'_d - X_{ls})} E'_q + \frac{(X'_d - X''_d)}{(X'_d - X_{ls})} \psi_{1d}$	(7.51)
$\psi_q = -X''_q I_q - \frac{(X''_q - X_{ls})}{(X'_q - X_{ls})} E'_d + \frac{(X'_q - X''_q)}{(X'_q - X_{ls})} \psi_{2q}$	(7.52)
$\psi_o = -X_{ls} I_o$	(7.53)

Table 7.3: Linear magnet circuit synchronous machine model

7.5 Steady-state condition for single machine

Suppose we have a set of balanced sinusoidal voltages and currents at the generator terminals of the form

$$V_a = \sqrt{2} V_s \cos(\omega_s t + \theta_s) \quad (7.54)$$

$$V_b = \sqrt{2} V_s \cos\left(\omega_s t + \theta_s - \frac{2\pi}{3}\right) \quad (7.55)$$

$$V_c = \sqrt{2} V_s \cos\left(\omega_s t + \theta_s + \frac{2\pi}{3}\right) \quad (7.56)$$

$$I_a = \sqrt{2} I_s \cos(\omega_s t + \phi_s) \quad (7.57)$$

$$I_b = \sqrt{2} I_s \cos\left(\omega_s t + \phi_s - \frac{2\pi}{3}\right) \quad (7.58)$$

$$I_c = \sqrt{2} I_s \cos\left(\omega_s t + \phi_s + \frac{2\pi}{3}\right) \quad (7.59)$$

CHAPTER 7. SYNCHRONOUS MACHINE MODEL

When doing circuit analysis, we use phasors to compute power flows within the power system [4]. We assume balanced three-phase voltages and currents, which allows us to analyze the flows on a single phase rather than having to perform the analysis for all three phases. The single phasor that would be used for the set of three-phase balanced voltages is $V_s e^{j\theta_s}$, and the current phasor is $I_s e^{j\phi_s}$. We need to know the relationship between these phasors representing the voltage and current at the terminals of the generator and the transformed voltage and current variables V_d , V_q , I_d , and I_q . This relationship will be used to write a basic circuit equation for the synchronous machine in terms of V_d , V_q , I_d , and I_q that will be used to determine how the system behaves when the generator is connected to it.

Park's transformation can be used to transform the stator variables from a fixed reference frame to a rotating reference frame via multiplication by the matrix T_{dqo} . The result of this transformation is

$$V_d = V_s \sin(\delta - \theta_s) \quad (7.60)$$

$$V_q = V_s \cos(\delta - \theta_s) \quad (7.61)$$

$$I_d = I_s \sin(\delta - \phi_s) \quad (7.62)$$

$$I_q = I_s \cos(\delta - \phi_s). \quad (7.63)$$

These can be written in complex form as

$$(V_d + jV_q)e^{j(\delta - \pi/2)} = V_s e^{j\theta_s} \quad (7.64)$$

$$(I_d + jI_q)e^{j(\delta - \pi/2)} = I_s e^{j\phi_s} \quad (7.65)$$

At the generator terminals, we can write the voltage and current in terms of the variables from the rotating reference frame. This provides an important link between the set of differential equations and the set of algebraic equations that describe power flows within the power system at steady-state

conditions.

We can use these relationships to compute the steady-state values of the variables that appear in the differential and algebraic equations. Suppose we know the amount of real power generated by the machine, and we know its terminal voltage phasor. From these two pieces of information, it should be possible to compute steady-state values of all variables.

At steady-state, we must have constant speed ω and constant angle δ . There are no deviations in state variables when the system has settled to a steady-state condition. Therefore, we may set the left-hand side of the differential equations in Table 7.3 to zero and simplify yielding

$$V_d = -R_s I_d - \psi_q \quad (7.66)$$

$$V_q = -R_s I_q + \psi_d \quad (7.67)$$

$$0 = -E'_q - (X_d - X'_d)I_d + E_{fd} \quad (7.68)$$

$$0 = -\psi_{1d} + E'_q - (X'_d - X_{ls})I_d \quad (7.69)$$

$$0 = -E'_d + (X_q - X'_q)I_q \quad (7.70)$$

$$0 = -\psi_{2q} - E'_d - (X'_q - X_{ls})I_q \quad (7.71)$$

$$0 = T_M - (\psi_d I_q - \psi_q I_d) - T_{FW} \quad (7.72)$$

$$\psi_d = E'_q - X'_d I_d \quad (7.73)$$

$$\psi_q = -E'_d - X'_q I_q. \quad (7.74)$$

We can combine the first two and last two equations to obtain

$$V_d = -R_s I_d + E'_d + X'_q I_q \quad (7.75)$$

$$V_q = -R_s I_q + E'_q - X'_d I_d. \quad (7.76)$$

These two equations can be written in phasor form as

$$(V_d + jV_q)e^{j(\delta-\pi/2)} + (R_s + jX_q)(I_d + jI_q)e^{j(\delta-\pi/2)} = \bar{E} \quad (7.77)$$

where

$$\bar{E} = [(X_q - X_d)I_d + E_{fd}]e^{j\delta}. \quad (7.78)$$

Recall that $(V_d + jV_q)e^{j(\delta-\pi/2)} = V_s e^{j\theta_s}$ and $(I_d + jI_q)e^{j(\delta-\pi/2)} = I_s e^{j\phi_s}$ are the balanced sinusoidal terminal voltage and current phasors, and we assumed these were known. Actually, we assumed the terminal voltage phasor and the terminal complex power injection were known. From these, we can compute the complex terminal current phasor from the equation $V_s e^{j\theta_s} I_s e^{-j\phi_s} = P_{Gi} + jQ_{Gi}$. The trick is that we now can calculate the left side of 7.77, which means we know the magnitude and phase angle of the left side. On the left side, the phase angle is δ . Therefore, we know δ after computing the left side of 7.77. After finding δ , we can compute everything else using the above equations.

7.6 Synchronous machine excitation control

In our current set of synchronous machine equations, there is one input to the system that can be adjusted to give a desired output: the field voltage E_{fd} . In this section, we discuss methods for generating and controlling this field excitation voltage. The idea is that we input a reference value into our controller, and at steady-state, the synchronous machine excitation field voltage and terminal voltage magnitude will settle to a desired value. Also, when a disturbance occurs in the system that affects generator speed and other state variables, the terminal voltage magnitude should settle quickly to the desired value.

We choose the control model shown in Figure 7.2 which consists of three components

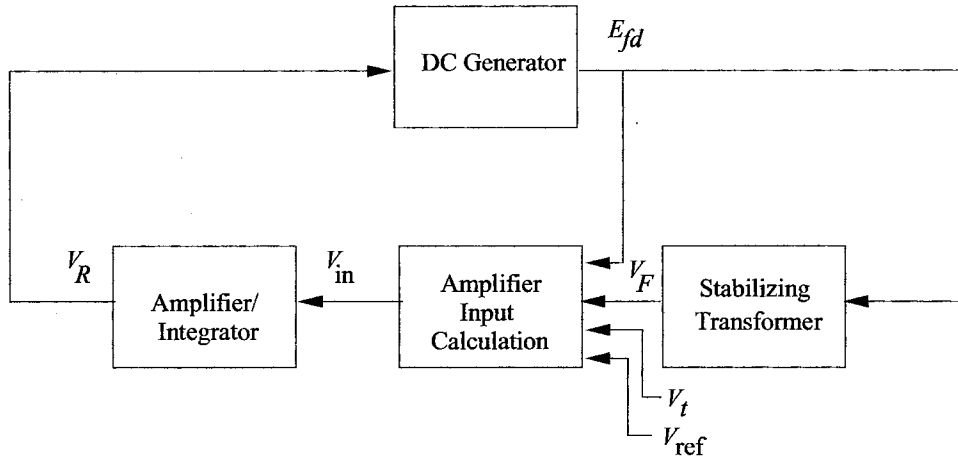


Figure 7.2: Excitation control block diagram.

1. DC generator
2. integrator
3. stabilizing transformer

This is the same controller found in [10] that is used in discussions of both steady-state and transient stability. We need to apply a DC voltage to the field winding of the synchronous machine. In this model, the DC voltage comes from a DC generator which also has a field voltage winding that requires a DC voltage. Therefore, by controlling the field voltage of the DC generator, we adjust the output voltage of the DC generator which is fed into the synchronous machine as its field winding voltage.

DC generator

In a DC machine, the field voltage is found on the stator. When the rotor spins, the constant magnetic field set up by stator field winding passes through a rotating winding on the rotor, inducing a sinusoidal voltage in the winding which is mechanically rectified via slip commutator and brushes. The result of the rectification is a DC voltage at the DC generator output voltage terminals. A separately-excited or self-excited DC generator may be used to produce the stator field voltage.

CHAPTER 7. SYNCHRONOUS MACHINE MODEL

Separately-excited machines have an external voltage source providing the field voltage, and self-excited machines use their own output voltage to supply the field voltage and the stator. In either case, the DC generator model yields a differential equation of the form

$$T_E \frac{dE_{fd}}{dt} = -(K_E + S_E(E_{fd}))E_{fd} + V_R \quad (7.79)$$

where $S_E(E_{fd})$ gives the saturation level as a function of the DC generator scaled terminal voltage E_{fd} , and V_R is the DC generator's scaled exciter field voltage. The saturation function gives information on how the magnetic field density and therefore the magnetic flux changes as current changes. If we wish to consider the nonlinear relationship between the two, then we must include this function. Otherwise, we approximate the relationship as linear.

From the differential equation, a steady-state field voltage V_R can be computed by noting that the voltage E_{fd} will not be changing under steady-state conditions. Therefore, the derivative term on the left side can be set to zero.

A proportional/integral (PI) controller block is a simple way to drive the DC generator output to a desired value. First, we compute the difference between the actual output voltage of the DC generator and a reference voltage. The integrator will then use this error to change the field voltage being applied to the generator in such a way that the error between actual and desired DC generator output voltage will gradually be driven to zero.

The value of the reference voltage that must be chosen can be found from the differential equation associated with the PI controller block. The desired steady-state synchronous machine field voltage E_{fd} must also be known. As discussed previously, this voltage can be computed if we know the steady-state terminal voltage and complex power output. Under steady-state conditions, the field voltage E_{fd} will not be changing, so the derivative of this voltage will be zero. Solving 7.79 for the required steady-state value of \hat{V}_R , we have $\hat{V}_R = (K_E + S_E(\hat{E}_{fd}))\hat{E}_{fd}$ where the "hat" notation is used to denote the evaluated steady-state values of the variables.

If we provide a voltage \hat{V}_R to the DC generator field voltage, the resulting DC generator output

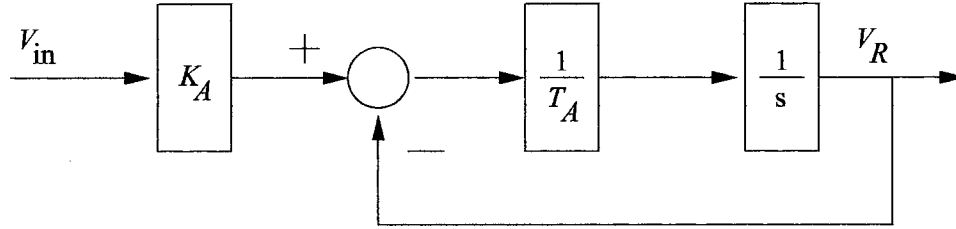


Figure 7.3: Proportional/integral controller block diagram.

voltage \hat{E}_{fd} will result in the desired synchronous machine terminal voltage magnitude under steady-state conditions. However, if we deviate from the steady-state condition, there is nothing at this point that will compensate for this deviation by adjusting the DC generator field voltage in such a way that we return to the desired operating point. The PI controller described in the next section can perform this function.

Proportional/integral controller

The job of the PI controller is to return the output of the controller to a desired setting when a disturbance occurs. To do this, the output of the controller is compared with a reference input. When there is a difference, the error is integrated to cause a change in the output that will drive it back to the desired value. A block diagram for this process is shown in Figure 7.3, and the differential equation corresponding to the diagram is

$$T_A \frac{dV_R}{dt} = -V_R + K_A V_{in}. \quad (7.80)$$

The voltage V_{in} is the input to the controller, and the constants T_A and K_A are chosen to get a good performance from the controller, *i.e.* a fast correction time with minimal overshoot. The output of the controller will be the DC generator field voltage V_R . An external input V_{ref} must also be applied to the system. This voltage input will allow us to set the desired steady-state output of the controller. For the case where there is no other feedback from the rest of the system, the

reference input will be fed directly into the controller so that $V_{in} = V_{ref}$.

When computing the value of the external input V_{ref} that will be applied to the controller, we must consider the desired steady state output of the controller, \hat{V}_R . Under steady-state conditions, V_R , which is the controller output, will not be changing, and therefore, its derivative with respect to time will be zero. Making this substitution into 7.80 and solving for V_{in} , we have $V_{in} = \hat{V}_R/K_A =$. The desired steady-state value of the DC generator field voltage \hat{V}_R was computed in the previous section and was shown to be dependent on the desired steady-state DC generator output voltage \hat{E}_{fd} . By applying the appropriate V_{in} to the PI controller, we are able to account for disturbances in the DC generator field voltage. However, for improved performance, we should also consider disturbances in the DC generator output voltage, which is used as the synchronous machine field voltage.

Stabilizing transformer

The output of the PI controller is fed to the DC generator field. This DC generator field voltage is fed back to the input of the controller to account for disturbances in the DC generator field voltage. We would also like to account for disturbances in the output of the DC generator, which is E_{fd} . By feeding this back to the controller input, we can eliminate deviations between the desired and actual E_{fd} . In addition to simply feeding back the DC generator output voltage, we also apply this voltage to a stabilizing transformer. The output voltage of the transformer V_F is also fed back to the controller input. The transformer enhances the stability of our control system by adding some additional dynamics. Its equivalent circuit model is shown in Figure 7.4. We can use the equivalent circuit model to write a differential equation

$$\frac{dV_F}{dt} = \frac{N_2}{N_1} L_{tm} \frac{1}{L_{t1} + L_{tm}} \left[\frac{dE_{fd}}{dt} - \frac{R_{t1}}{L_{tm}} \frac{N_1}{N_2} V_F \right] \quad (7.81)$$

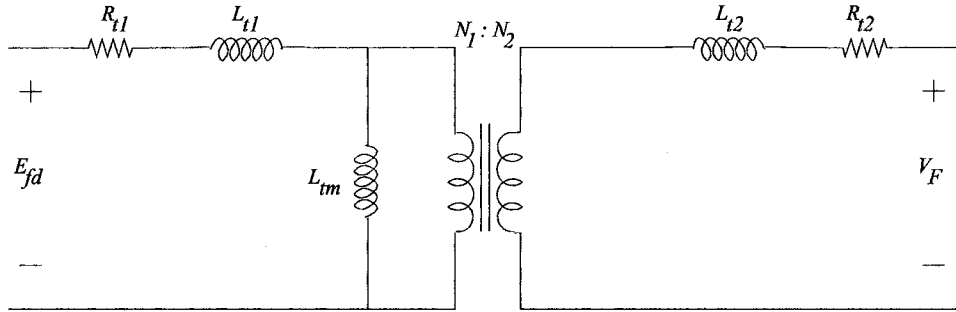


Figure 7.4: Stabilizing transformer equivalent circuit.

which reduces to

$$T_F \frac{dV_F}{dt} = -V_F + K_F \frac{dE_{fd}}{dt}. \quad (7.82)$$

Defining a new variable

$$R_f = \frac{K_F}{T_F} E_{fd} - V_F \quad (7.83)$$

allows simplification to

$$T_F \frac{dR_f}{dt} = -R_f + \frac{K_F}{T_F} E_{fd}. \quad (7.84)$$

In the case where we add the stabilizing transformer and decide to feed back both its output V_F and the DC generator output voltage E_{fd} , the input to the controller is computed as a weighted sum of these values and the reference voltage, or $V_{in} = V_{ref} - \frac{K_F}{T_F} E_{fd} - V_F$. For this case, the reference voltage would be computed by determining the desired steady-state output of the system. We will hold off computing the reference until the next section since there is one more feedback voltage that will be added to improve system performance.

Terminal voltage feedback & reference input

So far, we have added a feedback corresponding to DC generator output voltage and stabilizing transformer output voltage. The final feedback added will account for disturbances in the synchronous machine terminal voltage magnitude, which is really the voltage we are ultimately interested in controlling. By feeding this voltage back to the PI controller and including it in the input calculation, any deviations in synchronous machine terminal voltage magnitude will be driven to zero provided we choose the proper reference input voltage V_{ref} to the system.

The input to the controller V_{in} now includes

E_{fd} - DC generator output voltage

V_F - stabilizing transformer output voltage

V_t - generator per-unit terminal voltage

V_{ref} - reference voltage

and is computed by comparing the reference input to a weighted sum of the feedback voltages:

$$V_{\text{in}} = V_{\text{ref}} - V_t - \left(\frac{K_F}{T_F} E_{fd} - V_F \right) \quad (7.85)$$

$$= V_{\text{ref}} - V_t - R_f. \quad (7.86)$$

The only thing left to be computed is the value of the reference input V_{ref} that will result in the desired steady-state operating point. If we know the desired steady-state synchronous machine terminal voltage \hat{V}_t , then we can compute the desired synchronous machine field voltage \hat{E}_{fd} and the DC generator field voltage \hat{V}_R that is needed to produce this \hat{E}_{fd} . We saw in the section describing the PI controller that the input to the controller V_{in} needed to achieve the desired steady-state value of \hat{V}_R was \hat{V}_R/K_A . Also, at steady-state, the input to the stabilizing transformer is a DC voltage, so the output of the transformer would be $\hat{V}_F = 0$. Making these substitutions in 7.85, and solving

$$T_A \frac{dV_R}{dt} = -V_R + K_A R_f + \frac{K_A K_F}{T_F} E_{fd} + K_A (V_{\text{ref}} - V_t) \quad (7.88)$$

$$T_E \frac{dE_{fd}}{dt} = -(K_E + S_E(E_{fd})) E_{fd} + V_R \quad (7.89)$$

$$T_F \frac{dR_f}{dt} = -R_f + \frac{K_F}{T_F} E_{fd}. \quad (7.90)$$

Table 7.4: Excitation control model equations.

for V_{ref} , we have

$$V_{\text{ref}} = \frac{1}{K_A} \hat{V}_R + \hat{V}_t + \left(\frac{K_F}{T_F} \hat{E}_{fd} \right). \quad (7.87)$$

Control model summary

Our control model consists of three differential equations given in Table 7.4.

7.7 Multi-machine power system model

We wish to develop a model for an m machine power system that contains n buses. First we must scale the machine variables for each machine i by applying a transformation into a common reference frame. The common transformation matrix and its inverse are

$$T_{dqos} = \frac{2}{3} \begin{bmatrix} \cos \omega_s t & \cos(\omega_s t - \frac{2\pi}{3}) & \cos(\omega_s t + \frac{2\pi}{3}) \\ -\sin \omega_s t & -\sin(\omega_s t - \frac{2\pi}{3}) & -\sin(\omega_s t + \frac{2\pi}{3}) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \quad (7.91)$$

$$T_{dqos}^{-1} = \begin{bmatrix} \cos \omega_s t & -\sin \omega_s t & 1 \\ \cos(\omega_s t - \frac{2\pi}{3}) & -\sin(\omega_s t - \frac{2\pi}{3}) & 1 \\ \cos(\omega_s t + \frac{2\pi}{3}) & -\sin(\omega_s t + \frac{2\pi}{3}) & 1 \end{bmatrix}. \quad (7.92)$$

CHAPTER 7. SYNCHRONOUS MACHINE MODEL

We denote variables for machine i with subscript i . Recall that

$$\begin{bmatrix} V_{ai} \\ V_{bi} \\ V_{ci} \end{bmatrix} = \frac{1}{\sqrt{2}} T_{dqoi}^{-1} \begin{bmatrix} V_{di} \\ V_{qi} \\ V_{oi} \end{bmatrix} \quad (7.93)$$

where T_{dqoi} is the the transformation matrix for machine i and

$$\delta_i = \frac{P}{2} \theta_{\text{shaft}i} - \omega_s t. \quad (7.94)$$

We want to transform the variables for machine i to the common reference frame by applying the common transformation matrix

$$\begin{bmatrix} V_{Di} \\ V_{Qi} \\ V_{Oi} \end{bmatrix} = \frac{1}{\sqrt{2}} T_{dqos} \begin{bmatrix} V_{ai} \\ V_{bi} \\ V_{oi} \end{bmatrix} \quad (7.95)$$

$$= T_{dqos} T_{dqoi}^{-1} \begin{bmatrix} V_{di} \\ V_{qi} \\ V_{oi} \end{bmatrix} \quad (7.96)$$

$$= \begin{bmatrix} \sin \delta_i & \cos \delta_i & 0 \\ -\cos \delta_i & \sin \delta_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_{di} \\ V_{qi} \\ V_{oi} \end{bmatrix} \quad (7.97)$$

The transformation in complex notation is

$$(V_{Di} + jV_{Qi}) = (V_{di} + jV_{qi})e^{j(\delta_i - \frac{\pi}{2})} \quad (7.98)$$

$$(I_{Di} + jI_{Qi}) = (I_{di} + jI_{qi})e^{j(\delta_i - \frac{\pi}{2})} \quad (7.99)$$

$$(\psi_{Di} + j\psi_{Qi}) = (\psi_{di} + j\psi_{qi})e^{j(\delta_i - \frac{\pi}{2})} \quad (7.100)$$

CHAPTER 7. SYNCHRONOUS MACHINE MODEL

We introduce the following scaled speed and time constants

$$\omega_{ti} = T_{si}(\omega_i - \omega_s) \quad (7.101)$$

$$T_{si} = \sqrt{\frac{2H_i}{\omega_s}} \quad (7.102)$$

and assume H_i is large enough so that

$$\frac{1}{\omega_s} \ll T_{si}. \quad (7.103)$$

After scaling the machine variables using system-wide bases, we have the equations of Table 7.5. Assuming $\frac{1}{\omega_s}$ is sufficiently small, we can set the left side of 7.104-7.106 to zero. Ignoring the “zero” variables, writing in complex form, and substituting variables associated with individual machine reference frame, we have

$$\begin{aligned} 0 = & R_{si}(I_{di} + jI_{qi})e^{j(\delta_i - \pi/2)} - j(\psi_{di} + j\psi_{qi})e^{j(\delta_i - \pi/2)} \\ & + (V_{di} + jV_{qi})e^{j(\delta_i - \pi/2)} \end{aligned} \quad (7.116)$$

Using 7.113 and 7.114, we can eliminate the flux variables ψ_{di} and ψ_{qi} leaving us with

$$\begin{aligned} 0 = & (R_{si} + jX''_{di})(I_{di} + jI_{qi})e^{j(\delta_i - \pi/2)} + (V_{di} + jV_{qi})e^{j(\delta_i - \pi/2)} \\ & - \left[(X''_{qi} - X''_{di})I_{qi} + \frac{(X''_{qi} - X_{lsi})}{(X'_{qi} - X_{lsi})}E'_{di} - \frac{(X'_{qi} - X''_{qi})}{(X'_{qi} - X_{lsi})}\psi_{2qi} \right. \\ & \left. + j\frac{(X''_{di} - X_{lsi})}{(X'_{di} - X_{lsi})}E'_{qi} + j\frac{(X'_{di} - X''_{di})}{(X'_{di} - X_{lsi})}\psi_{1di} \right] e^{j(\delta_i - \pi/2)} \end{aligned} \quad (7.117)$$

$$\frac{1}{\omega_s} \frac{d\psi_{Di}}{dt} = R_s I_{Di} + \psi_{Qi} + V_{Di} \quad (7.104)$$

$$\frac{1}{\omega_s} \frac{d\psi_{Qi}}{dt} = R_s I_{Qi} - \psi_{Di} + V_{Qi} \quad (7.105)$$

$$\frac{1}{\omega_s} \frac{d\psi_{Oi}}{dt} = R_s I_{Oi} + V_{Oi} \quad (7.106)$$

$$T'_{doi} \frac{dE'_{qi}}{dt} = -E'_{qi} - (X_{di} - X'_{di}) \left[I_{di} - \frac{X'_{di} - X''_{di}}{(X'_{di} - X_{lsi})^2} (\psi_{1di} + (X'_{di} - X'_{lsi}) I_{di} - E'_{qi}) \right] + E_{fd} \quad (7.107)$$

$$T''_{doi} \frac{d\psi_{1di}}{dt} = -\psi_{1di} + E'_{qi} - (X'_{di} - X_{lsi}) I_{di} \quad (7.108)$$

$$T'_{qoi} \frac{dE'_{di}}{dt} = -E'_{di} + (X_{qi} - X'_{qi}) \left[I_{qi} - \frac{X'_{qi} - X''_{qi}}{(X'_{qi} - X_{lsi})^2} (\psi_{2qi} + (X'_{qi} - X'_{lsi}) I_{qi} - E'_{di}) \right] \quad (7.109)$$

$$T''_{qoi} \frac{d\psi_{2qi}}{dt} = -\psi_{2qi} - E'_{di} - (X'_{qi} - X_{lsi}) I_{qi} \quad (7.110)$$

$$T_{si} \frac{d\delta_i}{dt} = \omega_{ti} \quad (7.111)$$

$$T_{si} \frac{d\omega_{ti}}{dt} = T_{Mi} - (\psi_{di} I_{qi} - \psi_{qi} I_{di}) - T_{FWi} \quad (7.112)$$

$$\psi_{di} = -X''_{di} I_{di} + \frac{(X''_{di} - X_{lsi})}{(X'_{di} - X_{lsi})} E'_{qi} + \frac{(X'_{di} - X''_{di})}{(X'_{di} - X_{lsi})} \psi_{1di} \quad (7.113)$$

$$\psi_{qi} = -X''_{qi} I_{qi} - \frac{(X''_{qi} - X_{lsi})}{(X'_{qi} - X_{lsi})} E'_{di} + \frac{(X'_{qi} - X''_{qi})}{(X'_{qi} - X_{lsi})} \psi_{2qi} \quad (7.114)$$

$$\psi_{oi} = -X_{lsi} I_{oi} \quad (7.115)$$

Table 7.5: Synchronous machine scaled dynamic and algebraic equations for multi-machine model.

If T''_{doi} and T''_{qoi} are sufficiently small, we can set the left side of 7.108 and 7.110 to zero

$$0 = -\psi_{1di} + E'_{qi} - (X'_{di} - X_{lsi}) I_{di} \quad (7.118)$$

$$0 = -\psi_{2qi} - E'_{di} - (X'_{qi} - X_{lsi}) I_{qi} \quad (7.119)$$

CHAPTER 7. SYNCHRONOUS MACHINE MODEL

and eliminate ψ_{1di} and ψ_{2qi} from the remaining equations. Recall the generator terminal voltage and current flowing from the generator are

$$(V_{di} + jV_{qi})e^{j(\delta_i - \pi/2)} = V_i e^{j\theta_i} \quad (7.120)$$

$$(I_{di} + jI_{qi})e^{j(\delta_i - \pi/2)} = I_G e^{j\gamma_i}. \quad (7.121)$$

The damping torque term is approximated as being linearly proportional to the difference between the machine speed and the reference speed

$$T_{FW_i} = D_i(\omega_i - \omega_s). \quad (7.122)$$

Also, a new machine constant is defined as

$$M_i = \frac{2H_i}{\omega_s} \quad (7.123)$$

The resulting multi-machine dynamic model is given in Table 7.6.

Differential Equations

$$\frac{d\delta_i}{dt} = \omega_i - \omega_s \quad (7.124)$$

$$\frac{d\omega_i}{dt} = \frac{T_{Mi}}{M_i} - \frac{[E'_{qi} - X'_{di}I_{di}]I_{qi}}{M_i} - \frac{[E'_{di} + X'_{qi}I_{qi}]I_{di}}{M_i} - \frac{D_i(\omega_i - \omega_s)}{M_i} \quad (7.125)$$

$$\frac{dE'_{qi}}{dt} = -\frac{E'_{qi}}{T'_{doi}} - \frac{(X'_{di} - X'_{qi})I_{di}}{T'_{doi}} + \frac{E_{fdi}}{T'_{doi}} \quad (7.126)$$

$$\frac{dE'_{di}}{dt} = -\frac{E'_{di}}{T'_{qoi}} + \frac{(X'_{qi} - X'_{di})I_{qi}}{T'_{qoi}} \quad (7.127)$$

$$\frac{dE_{fdi}}{dt} = -\frac{K_{Ei} + S_E(E_{fdi})}{T_{Ei}} E_{fdi} + \frac{V_{Ri}}{T_{Ei}} \quad (7.128)$$

$$\frac{dV_{Ri}}{dt} = -\frac{V_{Ri}}{T_{Ai}} + \frac{K_{Ai}}{T_{Ai}} R_{fi} - \frac{K_{Ai}K_{Fi}}{T_{Ai}T_{Fi}} E_{fdi} + \frac{K_{Ai}}{T_{Ai}} (V_{refi} - V_i) \quad (7.129)$$

$$\frac{dR_{Fi}}{dt} = -\frac{R_{Fi}}{T_{Fi}} + \frac{K_{Fi}}{(T_{Fi})^2} E_{fdi} \quad (7.130)$$

for $i = 1, \dots, m$

Stator Algebraic Equations

$$0 = V_i e^{j\theta_i} + (R_{si} + jX'_{di})(I_{di} + jI_{qi})e^{j(\delta_i - \pi/2)} - [E'_{di} + (X'_{qi} - X'_{di})I_{qi} + jE'_{qi}]e^{j(\delta_i - \pi/2)} \quad (7.131)$$

$i = 1, \dots, m$

Generator Bus Network Equations

$$V_i e^{j\theta_i} (I_{di} - jI_{qi})e^{-j(\delta_i - \pi/2)} - P_{Li} - jQ_{Li} = S_i(V, \theta) \quad (7.132)$$

$i = 1, \dots, m$

Load Bus Network Equations

$$-P_{Li} - jQ_{Li} = S_i(V, \theta) \quad i = m + 1, \dots, n \quad (7.133)$$

Table 7.6: Multi-machine power system dynamic model

Chapter 8

Small-signal stability

8.1 Definition of small-signal stability

We would like to know if the power system exhibits small-signal stability at a given steady-state operating point. Thus, traditional small-signal stability analysis methods require knowledge of a steady-state power system operating point before any calculations can be done. In [10], it was assumed that a load-flow computation was done to determine the operating point. An important thing to note is that changes in the steady-state operating point will affect system stability.

To prove this, we first make a linear approximation of the nonlinear differential and algebraic equations of the power system by writing a first-order Taylor series about the steady-state operating point in question. We have a set of non-linear differential equations of the form

$$\dot{x} = f_d(x, Y, U) \tag{8.1}$$

where x is the set of system state variables, U is the input, and Y is the set of other system variables.

Writing the first order Taylor series, we have

$$\dot{x} = f_d(x_0, Y_0) + \frac{\partial f_d}{\partial x}(x - x_0) + \frac{\partial f_d}{\partial Y}(Y - Y_0) + \frac{\partial f_d}{\partial U}(U - U_0) \tag{8.2}$$

CHAPTER 8. SMALL-SIGNAL STABILITY

where $\frac{\partial f_d}{\partial x}$ and $\frac{\partial f_d}{\partial Y}$ are Jacobians of the differential equation vector and are evaluated at the **steady-state operating point**. The linearized differential equations may also be written as

$$\Delta \dot{x} = \dot{x} - f_d(x_0, Y_0) \quad (8.3)$$

$$= \frac{\partial f_d}{\partial x}(x - x_0) + \frac{\partial f_d}{\partial Y}(Y - Y_0) + \frac{\partial f_d}{\partial U}(U - U_0) \quad (8.4)$$

$$= \frac{\partial f_d}{\partial x} \Delta x + \frac{\partial f_d}{\partial Y} \Delta Y + \frac{\partial f_d}{\partial U} \Delta U \quad (8.5)$$

In addition to the differential equations, we have a set of nonlinear algebraic equations of the form

$$0 = f_a(x, Y) \quad (8.6)$$

which can also be linearized to obtain the following:

$$0 = 0 - f_a(x_0, Y_0) \quad (8.7)$$

$$= \frac{\partial f_a}{\partial x}(x - x_0) + \frac{\partial f_a}{\partial Y}(Y - Y_0) \quad (8.8)$$

$$= \frac{\partial f_a}{\partial x} \Delta x + \frac{\partial f_a}{\partial Y} \Delta Y \quad (8.9)$$

where $\frac{\partial f_a}{\partial x}$ and $\frac{\partial f_a}{\partial Y}$ are Jacobians of the nonlinear algebraic equation vector and are evaluated at the **steady-state operating point**. The input U is missing since the algebraic equations are not functions of the input in this case. However, in general the algebraic equations may be functions of the input.

Now we have a set of differential and algebraic equations written as functions of changes in the variables from their initial operating point, which is the **steady-state operating point**. Also, the Jacobians are functions of the **steady-state operating point**.

CHAPTER 8. SMALL-SIGNAL STABILITY

x	-	includes state variables of synchronous machines
I_g	-	includes direct-axis and quadrature-axis currents I_d, I_q of synchronous machines
V_g	-	includes voltage magnitudes and phase angles at generator buses
V_l	-	includes voltage magnitudes and phase angles at load buses
Y	-	includes $I_g, V_g,$ and V_l
U	-	system inputs such as mechanical torque

Table 8.1: Description of power system variables.

After linearizing the actual differential and algebraic equations from Chapter 7.7, we have

$$\Delta \dot{x} = A_1 \Delta x + B_1 \Delta I_g + B_2 \Delta V_g + E_1 \Delta U \quad (8.10)$$

$$0 = C_1 \Delta x + D_1 \Delta I_g + D_2 \Delta V_g \quad (8.11)$$

$$0 = C_2 \Delta x + D_3 \Delta I_g + D_4 \Delta V_g + D_5 \Delta V_l \quad (8.12)$$

$$0 = D_6 \Delta V_g + D_7 \Delta V_l \quad (8.13)$$

where the variable descriptions are given in Table 8.1, and the Jacobian matrices are given by A_1, B_1, \dots, D_7 . Equation 8.10 is the set of generator differential equations, and 8.11 is the set of stator algebraic equations. The network power flow equations are given by 8.12 for buses that have generators and by 8.13 for buses that do not have generators. The details of the linearization process are given in Appendix A. We can eliminate the non-state variables as shown in Appendix B to obtain a single set of differential equations

$$\Delta \dot{x} = A_{\text{sys}} \Delta x + E_1 \Delta U. \quad (8.14)$$

Neglecting changes in the input, we have a set of linear differential equations of the form

$$\Delta \dot{x} = A_{\text{sys}} \Delta x. \quad (8.15)$$

which has a solution of the form

$$x(t) = \psi_1 \mathbf{x}(0) e^{\lambda_1 t} \phi_1 + \psi_2 \mathbf{x}(0) e^{\lambda_2 t} \phi_2 + \dots + \psi_n \mathbf{x}(0) e^{\lambda_n t} \phi_n \quad (8.16)$$

where ψ_i is a right *eigenvector*, ϕ_i is a left *eigenvector*, and λ_i is an *eigenvalue* of the A_{sys} matrix, and we assume that we have a complete set of n eigenvalues. Eigenvalues in this paper were evaluated using a built-in Matlab function based on eigenvalue analysis methods given in [12] [13].

An eigenvalue may be complex, but we are interested in the real part because it determines whether the response associated with the eigenvalue decays exponentially or grows exponentially. A negative real part indicates a response that decays exponentially while a positive real part means the response grows exponentially. A stable system has a response which quickly dies out, which means the state variables move away from their steady-state values for a short time and then quickly return to steady-state. Therefore, we require the eigenvalues to have a **negative** real part in order for the system to be stable.

In addition to being stable, we desire a system that has a satisfactory *stability margin*, *i.e.* we don't want the eigenvalue real parts to be too close to the origin. If an eigenvalue has a real part which is negative but is only slightly less than zero, the system is stable. However, the deviations from steady-state will die out very slowly. Also, our A_{sys} matrix is not exactly correct, so the eigenvalue of the true A_{sys} matrix may in fact have a slightly positive real part. Therefore, we would like the eigenvalue real parts to be negative and to be located far enough away from the origin so that we have some stability margin.

8.2 Zero eigenvalues

Since we desire eigenvalues that have negative real parts, it is important to note that using our current framework, the A_{sys} matrix for the linearized power system dynamic model has two eigenvalues located at the origin. This, however, does not indicate a violation of stability margin limits. A zero

CHAPTER 8. SMALL-SIGNAL STABILITY

eigenvalue appears because our rotating system's reference frame is an arbitrary angle rotating at synchronous speed. The other zero eigenvalue is a result of the system having no way of returning system electrical speeds back to 60 Hz after the disturbance. Therefore, we need to first define angle and speed references. Our choice is the center of inertia (COI) reference frame. A new angle state variable and a new speed state variable are added by taking a weighted average of the angles and speeds of machines in the system.

$$\delta_{COI} = \frac{1}{M_T} \sum_{i=1}^m M_i \delta_i \quad (8.17)$$

$$\omega_{COI} = \frac{1}{M_T} \sum_{i=1}^m M_i \omega_i. \quad (8.18)$$

We now refer all angles and speeds to the center of inertia variables.

$$\hat{\delta}_i = \delta_i - \delta_{COI} \quad i = 1, \dots, m \quad (8.19)$$

$$\hat{\omega}_i = \omega_i - \omega_{COI} \quad i = 1, \dots, m \quad (8.20)$$

This leaves us with two more differential equations and two more state variables. Since these equations and variables are linear combinations of other equations and variables, we can eliminate the speed and angle differential equations and state variables for one of the machines. Assuming we choose to eliminate for the “first” machine, we are left with differential equations and state variables $(\delta_{COI}, \omega_{COI})$ and $(\hat{\delta}_i, \hat{\omega}_i)$ for $i = 2, \dots, m$. The differential equations and state variables corresponding to δ_{COI} and ω_{COI} are set aside since their dynamics indicate how the machines in our system all return to synchronous speed after a disturbance occurs. After making these changes, the order of the system is reduced, the two eigenvalues located at the origin are removed, and the rest of the eigenvalues are the same as before.

An equivalent technique is to make the speed reference and angle reference that of a particular machine in the system, which is the technique used to produce the results given in this report. The

CHAPTER 8. SMALL-SIGNAL STABILITY

“first” machine was chosen as reference, and its differential equations and state variables corresponding to angle and speed were eliminated. This technique eliminated the two zero eigenvalues while leaving the rest unchanged.

Chapter 9

Eigenvalue Sensitivities

We plan to place small-signal stability constraints into our optimal power flow formulation. As discussed in the previous chapter, small-signal stability can be determined by evaluating the eigenvalues of the system state matrix, which is unique for a given power system operating point. The primal-dual interior-point method and Newton's method used to solve the optimization problem require knowledge of how eigenvalues of the system state matrix change when the power system operating point changes because we are including inequality constraints on eigenvalues. These changes are computed by evaluating the partial derivatives of eigenvalues with respect to the variables describing the power system operating point. We will need to know both first-order and second-order partial derivatives, and both are discussed in this chapter. Eigenvalue derivatives will also be referred to as eigenvalue sensitivities.

In [14], eigenvalue sensitivities with respect to general system operating parameters were discussed. The sensitivities were used to determine how system loading levels affect stability. Reference [15] applies eigenvalue sensitivities to controller design. This chapter also discusses some additional eigenvalue and eigenvector sensitivity computation techniques not presented in [14] or [15].

9.1 Assumptions

Suppose the system state matrix A_{sys} has complete sets of n distinct eigenvalues $\{\lambda_1, \dots, \lambda_n\}$, right eigenvectors $\{\phi_1, \dots, \phi_n\}$, and left eigenvectors $\{\psi_1, \dots, \psi_n\}$, where right eigenvectors are column vectors and left eigenvectors are row vectors. The eigenvalues and eigenvectors satisfy

$$A_{\text{sys}} \phi_i = \lambda_i \phi_i \quad (9.1)$$

and

$$\psi_i A_{\text{sys}} = \lambda_i \psi_i. \quad (9.2)$$

We can form a matrix Φ containing the vectors $\{\phi_1, \dots, \phi_n\}$ as its columns, a matrix Ψ containing the vectors $\{\psi_1, \dots, \psi_n\}$ as its rows, and a matrix Λ which contains the complex scalars $\{\lambda_1, \dots, \lambda_n\}$ as its diagonal elements. Then in matrix notation, we have $A_{\text{sys}} \Phi = \Phi \Lambda$.

Suppose we have a right eigenvector ϕ_i corresponding to λ_i and a left eigenvector ψ_j corresponding to λ_j where $i \neq j$. Then the following holds true:

$$\psi_j \phi_i = 0. \quad (9.3)$$

Now suppose we have a right eigenvector ϕ_i and a left eigenvector ψ_i corresponding to the same eigenvalue λ_i . Then we have

$$\psi_i \phi_i = c_i \quad (9.4)$$

where $c_i \in \mathbb{C}$. Note that since scalar multiples of the eigenvectors are also eigenvectors, we can set $c_i = 1$ by normalizing 9.4, allowing us to simplify expressions that will appear later.

9.2 First-order eigenvalue sensitivities

Now we are ready to find $\frac{\partial \lambda_i}{\partial \epsilon}$ as shown in [15] where $\epsilon \in Y$. Note that A_{sys} is a function of the set of OPF decision variables Y . First, we take the partial derivative of both sides of 9.1 with respect to ϵ :

$$\frac{\partial A_{\text{sys}}}{\partial \epsilon} \phi_i + A_{\text{sys}} \frac{\partial \phi_i}{\partial \epsilon} = \frac{\partial \lambda_i}{\partial \epsilon} \phi_i + \lambda_i \frac{\partial \phi_i}{\partial \epsilon}. \quad (9.5)$$

Multiplying both sides of 9.5 on the left by ψ_i and simplifying results in

$$\frac{\partial \lambda_i}{\partial \epsilon} = \frac{\psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon} \phi_i}{\psi_i \phi_i}. \quad (9.6)$$

9.3 Second-order eigenvalue sensitivities computed using all eigenvalues & eigenvectors

In this section, we review a method proposed by [15] and [16] to compute the second-order eigenvalue sensitivities. This method requires that we know all the eigenvalues and eigenvectors even though we may only wish to calculate the second-order sensitivity for one eigenvalue.

Suppose we are trying to compute $\frac{\partial^2 \lambda_i}{\partial \epsilon_2 \partial \epsilon_1}$ for some $\epsilon_1, \epsilon_2 \in Y$. The second-order sensitivity, derived in Appendix C, is then

$$\frac{\partial^2 \lambda_i}{\partial \epsilon_2 \partial \epsilon_1} = \frac{1}{\psi_i \phi_i} \left[\psi_i \frac{\partial^2 A_{\text{sys}}}{\partial \epsilon_2 \partial \epsilon_1} \phi_i + \psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_1} \sum_{\substack{k=1 \\ k \neq i}}^n \beta_{ik} \phi_k + \psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_2} \sum_{\substack{k=1 \\ k \neq i}}^n \alpha_{ik} \phi_k \right] \quad (9.7)$$

where

$$\alpha_{ij} = \frac{\psi_j \frac{\partial A_{\text{sys}}}{\partial \epsilon_1} \phi_i}{\psi_j \phi_j (\lambda_i - \lambda_j)} \quad (9.8)$$

$$\beta_{ij} = \frac{\psi_j \frac{\partial A_{\text{sys}}}{\partial \epsilon_2} \phi_i}{\psi_j \phi_j (\lambda_i - \lambda_j)}. \quad (9.9)$$

Again, this formula requires knowledge of all eigenvalues and eigenvectors which is a very large problem to solve when the system state matrix is very large.

9.4 Second-order eigenvalue sensitivities computed using a single eigenvalue and eigenvector.

We would like to find the second-order eigenvalue sensitivities without having to know all the eigenvalues and eigenvectors. The second-order eigenvalue sensitivity can be written in terms of two first-order eigenvector sensitivities:

$$\frac{\partial^2 \lambda_i}{\partial \epsilon_2 \partial \epsilon_1} = \frac{1}{\psi_i \phi_i} \left\{ \psi_i \frac{\partial^2 A_{\text{sys}}}{\partial \epsilon_2 \partial \epsilon_1} \phi_i + \psi_i \left[\frac{\partial A_{\text{sys}}}{\partial \epsilon_1} - \frac{\partial \lambda_i}{\partial \epsilon_1} I \right] \frac{\partial \phi_i}{\partial \epsilon_2} + \psi_i \left[\frac{\partial A_{\text{sys}}}{\partial \epsilon_2} - \frac{\partial \lambda_i}{\partial \epsilon_2} I \right] \frac{\partial \phi_i}{\partial \epsilon_1} \right\}. \quad (9.10)$$

If we know the eigenvector sensitivities $\frac{\partial \phi_i}{\partial \epsilon_1}$ and $\frac{\partial \phi_i}{\partial \epsilon_2}$, then we can compute $\frac{\partial^2 \lambda_i}{\partial \epsilon_2 \partial \epsilon_1}$. In the next section, we review two ways to compute the eigenvector sensitivities.

9.5 First order eigenvector sensitivities

9.5.1 Method I

We start by rearranging 9.5 to give us

$$(A_{\text{sys}} - \lambda_i I) \frac{\partial \phi_i}{\partial \epsilon_1} = - \left(\frac{\partial A_{\text{sys}}}{\partial \epsilon_1} - \frac{\partial \lambda_i}{\partial \epsilon_1} I \right) \phi_i. \quad (9.11)$$

By definition of the eigenvalue, $(A_{\text{sys}} - \lambda_i I)$ is singular, so we can't simply multiply both sides of 9.11 by $(A_{\text{sys}} - \lambda_i I)^{-1}$ to solve for the eigenvector sensitivity. However, it can be shown that the term on the right side of 9.11 is indeed in the range of $(A_{\text{sys}} - \lambda_i I)$ [17] even though $(A_{\text{sys}} - \lambda_i I)$

is singular. We can use 9.11 and a normalization condition

$$\left\| \phi_i(\epsilon_1) \right\|^2 = \phi_i^*(\epsilon_1) \phi_i(\epsilon_1) = 1 \quad (9.12)$$

on the eigenvectors to solve for the real and imaginary parts of $\frac{\partial \phi_i}{\partial \epsilon_1}$. The system of equations to solve is

$$\begin{bmatrix} A_{\text{sys}} - \text{Re}(\lambda_i I) & \text{Im}(\lambda_i I) \\ -\text{Im}(\lambda_i I) & A_{\text{sys}} - \text{Re}(\lambda_i I) \\ \text{Re}(\phi_i^*) & \text{Im}(\phi_i^*) \end{bmatrix} \begin{bmatrix} \text{Re} \left(\frac{\partial \phi_i}{\partial \epsilon_1} \right) \\ \text{Im} \left(\frac{\partial \phi_i}{\partial \epsilon_1} \right) \end{bmatrix} = \begin{bmatrix} -\text{Re} \left(\left(\frac{\partial A_{\text{sys}}}{\partial \epsilon_1} - \frac{\partial \lambda_i}{\partial \epsilon_1} I \right) \phi_i \right) \\ -\text{Im} \left(\left(\frac{\partial A_{\text{sys}}}{\partial \epsilon_1} - \frac{\partial \lambda_i}{\partial \epsilon_1} I \right) \phi_i \right) \\ 0 \end{bmatrix}. \quad (9.13)$$

9.5.2 Method II

A second method for computing eigenvector sensitivities is proposed in [17]. The results of [17] will be given here with equations using our notation and assumptions. First, we can take the partial derivative with respect to ϵ_1 of both sides of 9.12 to obtain

$$\phi_i^* \frac{\partial \phi_i}{\partial \epsilon_1} + \left(\phi_i^* \frac{\partial \phi_i}{\partial \epsilon_1} \right)^* = 0. \quad (9.14)$$

From this equation, we deduce that the real part of $\phi_i^* \frac{\partial \phi_i}{\partial \epsilon_1}$ is zero since any complex number added to its conjugate is twice the real part of the number. Also, no matter what we choose for the imaginary part of $\phi_i^* \frac{\partial \phi_i}{\partial \epsilon_1}$, 9.14 will still hold true. Therefore, let's choose the imaginary part to be zero. If $\phi_i^* \frac{\partial \phi_i}{\partial \epsilon_1}$ is real and its imaginary part is zero, then 9.14 becomes

$$\phi_i^* \frac{\partial \phi_i}{\partial \epsilon_1} = 0. \quad (9.15)$$

Multiplying both sides of 9.15 on the left by ϕ_i , we have

$$\phi_i \phi_i^* \frac{\partial \phi_i}{\partial \epsilon_1} = 0. \quad (9.16)$$

CHAPTER 9. EIGENVALUE SENSITIVITIES

Adding $\phi_i \phi_i^* \frac{\partial \phi_i}{\partial \epsilon_1}$ to the left side of 9.11, we have

$$(A_{\text{sys}} - \lambda_i I + \phi_i \phi_i^*) \frac{\partial \phi_i}{\partial \epsilon_1} = - \left(\frac{\partial A_{\text{sys}}}{\partial \epsilon_1} - \frac{\partial \lambda_i}{\partial \epsilon_1} I \right) \phi_i. \quad (9.17)$$

Adding $\phi_i \phi_i^*$ to $(A_{\text{sys}} - \lambda_i I)$ is a rank-1 modification of $(A_{\text{sys}} - \lambda_i I)$. Since $\left(\frac{\partial A_{\text{sys}}}{\partial \epsilon_1} - \frac{\partial \lambda_i}{\partial \epsilon_1} I \right) \phi_i$ is in the range of $(A_{\text{sys}} - \lambda_i I + \phi_i \phi_i^*)$, then we need only solve the system of equations given by 9.17 to compute the eigenvector sensitivity. This method is different from the previous one in the way the normalization condition is handled.

As emphasized earlier, both methods need only knowledge of the i th eigenvalue and left eigenvector to compute the eigenvector sensitivity. Once we know the eigenvector sensitivity, it can be plugged into 9.10 to obtain the second-order eigenvalue sensitivity.

Chapter 10

OPF and ESCOPF with small-signal stability constraints

The previous sections defined small-signal stability the optimal power flow (OPF) problem, and the expected-security-cost optimal power flow (ESCOPF) problem. In this section, OPF and ESCOPF problems are described which include inequality constraints to ensure that the system exhibits small-signal stability. The software used to solve these problems is described, and a summary of the results is presented.

10.1 Adding small-signal stability constraints

Small-signal stability was discussed in Chapter 8. It was noted that each eigenvalue of the linearized system's state matrix must have a real part that is negative. Then if there are a total of N state variables in the system, the real part of each eigenvalue must be less than zero so that the system is not unstable, or

$$\mathbf{Re}(\lambda_i) < 0, \quad i = 1, \dots, N. \quad (10.1)$$

Another way to write the stability constraint is with some security margin $\lambda_{\max} < 0$,

$$\mathbf{Re}(\lambda_i) \leq \lambda_{\max}, \quad i = 1 \dots N. \quad (10.2)$$

This will ensure that oscillations die out as quickly as we desire.

We choose to constrain the eigenvalue real part, but we could also have chosen to constrain the damping factor, which is the ratio of the real part to the magnitude of the eigenvalue. Constraining the damping factor makes sure that oscillations die out quickly and that there are a limited number of cycles of oscillation before the oscillations die out.

To solve the problem, we use the PDIP algorithm. This algorithm requires knowledge of inequality constraint Jacobian and Hessian matrices. Since some inequality constraints are on eigenvalues, we need Jacobian and Hessian matrices for each eigenvalue. Therefore, we need first- and second-order partial derivatives of each eigenvalue with respect to the OPF decision variables. Expressions for these eigenvalue sensitivities are given in Chapter 9.

10.2 Test system

WSCC 9-bus system

The system analyzed is the WSCC 9-bus system shown in Figure 10.1. This system is used for stability analysis as described in [10] [18] [19] [20]. It consists of 3 generators, 3 loads, and 9 transmission lines. The system data including generator cost curves, consumer benefit curves, and security data is given in Appendix D.

Cost and Benefit Curves

Since the objective function is social welfare, each generator must have a cost curve, and each load must have a benefit curve. Coefficients were chosen so the system would be operating at an operating point similar to the one described on page 171 of [10]. This operating point was desired so that

CHAPTER 10. OPF AND ESCOPF WITH SMALL-SIGNAL STABILITY CONSTRAINTS

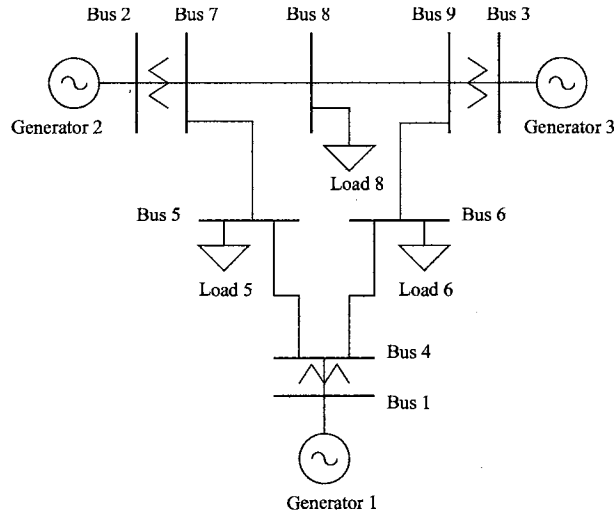


Figure 10.1: WSCC 9-bus system

stability analysis results would be similar to those obtained in the book. Generator cost curves were assumed to be of the form

$$C(P_{Gi}) = \gamma_{Gi} P_{Gi}^2 + \beta_{Gi} P_{Gi} + \alpha_{Gi}.$$

Consumer benefit curves were assumed to be of the form

$$B(P_{Li}) = \gamma_{Li} P_{Li}^2 + \beta_{Li} P_{Li} + \alpha_{Li}.$$

The cost curve coefficients were adjusted until the desired operating point was reached with real power prices in the \$10-\$15 per MW range.

For the ESCOPF, load interruption cost was modeled as a linearly proportional of the amount of power interrupted,

$$C_{Li}(P_{Li}^0, P_{Li}^k) = \beta_{Li} [P_{Li}^0 - P_{Li}^k], \quad (10.3)$$

Ramping constraints for each synchronous machine limit how much the machine can adjust its generation when a contingency occurs. Load interruption cost coefficients and generator ramping constraints are given in Appendix D.

10.3 Implementation

To solve the problem, several MatLab functions were written. These basic functions performed the following tasks:

- read system data from text files
- assign MatLab variable names to data
- create indices for decision variable, Lagrange multiplier, and slack variables
- initialize variables
- execute primal-dual interior-point (PDIP) algorithm

These functions are called in sequence from the `opf.m` function as shown in Figure 10.3. The first item on the list is reading system data, which is performed by the `ReadDF.m` function. System data is contained in plain ASCII text files:

- `.cdf` file
- `.gdf` file
- `.ldf` file
- `.sdf` file

where the “.” in each file name is preceded by the name of the case being studied. For example, `wsc9` would indicate we are analyzing the WSCC 9-bus system. The `*.cdf` file is system loadflow data in the IEEE Common Data Format [21], and the `*.gdf` and `*.ldf` file formats, which give generator and load cost curve and generation limit data, are first presented in [6]. We introduce the synchronous machine dynamic data file and give it a `.sdf` extension. Its format is described in Appendix E.

After reading these files, each piece of data that describes the system is given a MatLab variable name. For example, we must assign names to transmission line admittances, generator cost curve

CHAPTER 10. OPF AND ESCOPF WITH SMALL-SIGNAL STABILITY CONSTRAINTS

```
clear all
clear variables
format compact
%Enter name of file without cdf,gdf,or ldf extension
cdfname = input('Enter file name : ','s');
%cdfname = 'wsc9';
file_name.cdfname = strcat(cdfname, '.cdf');
file_name.ldfname = strcat(cdfname, '.ldf');
file_name.gdfname = strcat(cdfname, '.gdf');
file_name.sdfname = strcat(cdfname, '.sdf');

% Read data from files
raw_data=ReadDF(file_name);

% Store data to variables
data = getdata(raw_data);

% Create index
ind = index(data);

% Eigenvalue limit data
%data.eig_ind = 1:19;
%data.eig_ind = 1:ind.ineq.eig_real.end;
data.eig_ind = 1:10;
data.neig_const = length(data.eig_ind);
data.limits.eig_max = -0.25*ones(data.neig_const,1);

% Initialize variables
var = initialize(data,ind);

% Primal-dual interior-point algorithm
sol = pdip(var,data,ind);
```

Figure 10.2: Matlab function `opf.m`

coefficients, and bus voltage operating limits. The MatLab function written to assign variable names to data is `getdata.m`, and its code is listed in Appendix H.

In the OPF problem, there are decision variables, Lagrange multiplier variables, and slack variables. The set of decision variables was assigned the name Y , and it contained real and reactive power generation variables, bus voltage magnitude and phase angle variables, and any other variables that describe the power system steady-state operating point. We denoted the set of equality constraint Lagrange multipliers by z . Among the variables included in this vector are Lagrange multipliers on real and reactive power injected at system buses. Finally, we have Lagrange multipliers on inequality constraints, denoted by the vector w , and we have a set of slack variables denoted by s .

The primal-dual interior-point algorithm is an iterative algorithm. At iteration i , we obtain an updated vector

$$v^i = \begin{bmatrix} Y^i \\ z^i \\ w^i \\ s^i \end{bmatrix} \quad (10.4)$$

that hopefully is closer to the solution than the vector obtained in the previous iteration. To compute the update, we need to know Jacobian and Hessian matrix entries for the equality and inequality constraints. The Jacobian and Hessian matrix elements are functions of the variables in Y^i . Therefore, we need to know which elements of Y^i correspond to real power generation, reactive power generation, bus voltage magnitudes, etc. The function `index.m` creates a list of matrix indices so we can extract the desired variables from the updated vector v^i .

The `index.m` function does not need to be created manually. Another function, `createindex.m`, creates the function `index.m` by examining a text file, `variables.txt` that contains a list of decision variable, equality constraint, and inequality constraint names. Therefore, the file `variables.txt` is divided into three sections:

- Indices for variables Y
- Indices for equality constraints
- Indices for inequality constraints.

Each line of the file contains a variable description which will be used to name a vector that contains those variables. Next to the variable description is the number of variables in the vector by that name. The function `createindex.m` reads this list and uses the information to create the function `index.m`, which is executed from `opf.m` to create a Matlab structure of variable indices.

CHAPTER 10. OPF AND ESCOPF WITH SMALL-SIGNAL STABILITY CONSTRAINTS

An initial guess for each variable is required to start the algorithm. The function `initialize.m` performs this task. After initializing the variables, we may then run the primal-dual interior-point algorithm to solve the optimization problem. The function `pdip.m` executes the PDIP algorithm and contains calls to other functions:

- `eq_eval.m` - evaluate vector of equality constraints
- `ineq_eval.m` - evaluate vector of inequality constraints
- `objective_eval.m` - evaluate objective function
- `eq_jacobian.m` - evaluate equality constraint Jacobian matrix
- `ineq_jacobian.m` - evaluate inequality constraint Jacobian matrix
- `objective_jacobian.m` - evaluate objective function gradient vector
- `eq_hessian.m` - evaluate equality constraint Hessian matrices
- `ineq_hessian.m` - evaluate inequality constraint Hessian matrices
- `objective_hessian.m` - evaluate objective function Hessian matrix
- `update.m` - compute updated vector v^i .

These functions are executed during each Newton-Raphson iteration. In addition, during each iteration, a line is printed in the Matlab console indicating the progress of the solution algorithm.

The following information is printed:

- norm of update vector for iteration i , $\|\Delta v^i\|$
- current value of barrier parameter μ
- angle between current update vector Δv^i and previous update vector Δv^{i-1}
- norm of vector representing KKT conditions, $\|\Delta y^i\|$

- scaling parameter α for iteration i .

We display the norm of the current update vector since as we approach the solution, it should get smaller. Also, the vector representing the KKT conditions should have a norm of zero at the solution. The angle between update vectors can indicate how the step direction changes between Newton-Raphson iterations. This angle, σ , is computed from the formula

$$\cos \sigma = \frac{(\Delta v^{i-1})^* (\Delta v^i)}{\|\Delta v^{i-1}\| \|\Delta v^i\|} \quad (10.5)$$

which says that the cosine of the angle is the inner product of the two vectors divided by the product of their norms. If the angle between consecutive updates is small, then the movement is in a similar direction. But if the computed angle is close to 180° each time, then it is likely that we are stuck in a corner, and the algorithm needs to be restarted after making some changes. Finally, the scaling parameter α can also indicate if the algorithm is banging against the wall of an inequality constraint, as indicated by a very small α . Together, these indicators can tell us if the algorithm is stuck and not likely to converge. If it doesn't seem to be converging, then we can adjust some of the PDIP parameters including κ , μ_0 , τ , and ρ . We can also adjust the initial guess of the problem solution. In addition, there could be an error in the code that evaluates Jacobian and Hessian matrices. All of these must be checked, making troubleshooting difficult.

10.4 Summary of SSSOPF Results

The small-signal stability OPF problem was solved for four cases. For each case, a different stability margin was chosen. In the first case, we require all eigenvalues of the system state matrix to have negative real parts. At the operating point defined by the solution, no stability constraints were binding or active, so this is actually the case where eigenvalues are unconstrained. The right-most eigenvalue in this unconstrained case had a real part of -0.1873, which indicates a stable system.

For the remaining cases, we gradually increased the stability margin. The social welfare, and

Constraint $\text{Re}(\lambda)$	Right-most eigenvalue	Lagrange multiplier (\$/MW-hr)	SW (\$/hr)
0	$-0.1873 \pm j7.2083$	0.00	2319.90
-0.2	$-0.2000 \pm j8.4992$	0.66	2319.80
-0.3	$-0.3000 \pm j8.3586$	13.86	2307.40
-0.4	$-0.4000 \pm j7.8145$	17.78	2275.10

Table 10.1: Summary of SSSOPF results.

the Lagrange multiplier on the binding eigenvalue constraints are given in Table 10.1. As expected, social welfare decreases as the stability constraints are made more strict. The Lagrange multipliers on the constraints show how much social welfare would increase if we were allowed to relax the stability constraint slightly. Complete, detailed results for each case are given in Appendix F.

10.5 Summary of SSS ESCOPF Results

The small-signal stability ESCOPF problem was solved using a PDIP algorithm for four cases with each case having a different stability margin. In the first case, a basic stability margin constraint was placed on the eigenvalues in which they were limited to having a negative real part. None of the constraints were binding, but the eigenvalue with with the right-most real part in the pre-contingency operating point was closer to the imaginary axis than in the unconstrained small-signal stability OPF problem solution. For the ESCOPF, we have six post-contingency operating points to look at in addition to the pre-contingency operating point. When lines 4-6, 4-5, and 4-7 are removed, the critical eigenvalues lie even farther to the right than in the pre-contingency case. The most critical eigenvalue from all cases has a real part of -0.1266 and occurs when line 5-7 is removed from the system. Table 10.2 compares the critical eigenvalues and the real power dispatch of the pre-contingency and post-contingency states.

Note the ESCOPF pre-contingency operating state is different than the result of the OPF problem. Social welfare is not as high since we must account for the possibility of losing a line and the possible loss of load that may result.

After solving the unconstrained case, the stability margin was increased to -0.15. The stability

CHAPTER 10. OPF AND ESCOPF WITH SMALL-SIGNAL STABILITY CONSTRAINTS

State	Critical λ	P_{G1}	P_{G2}	P_{G3}	P_{L5}	P_{L6}	P_{L8}
pre-cont.	$-0.1784 \pm j9.5491$	66.47	161.47	82.81	117.92	83.86	104.73
line 4-6	$-0.1302 \pm j8.2630$	43.28	164.99	103.79	117.92	83.86	104.73
line 4-5	$-0.1448 \pm j7.5089$	30.52	148.32	104.24	89.72	83.86	104.73
line 5-7	$-0.1266 \pm j7.4864$	116.47	126.27	69.35	117.92	83.86	104.73
line 6-9	$-0.1888 \pm j8.0426$	115.23	147.88	47.81	117.92	83.86	104.73
line 7-8	$-0.2053 \pm j8.9385$	69.57	126.27	117.81	117.92	83.86	104.73
line 9-8	$-0.1703 \pm j9.0671$	68.57	181.47	60.89	117.92	83.86	104.73

Table 10.2: Summary of SSS ESCOPF results, $\text{Re}(\lambda) \leq 0$.

State	Critical λ	P_{G1}	P_{G2}	P_{G3}	P_{L5}	P_{L6}	P_{L8}
pre-cont.	$-0.1996 \pm j9.4823$	82.08	146.93	80.84	117.77	83.84	104.81
line 4-6	$-0.1508 \pm j8.1927$	58.22	159.66	93.71	117.77	83.84	104.81
line 4-5	$-0.1525 \pm j7.4783$	38.96	146.28	97.65	89.70	83.84	104.81
line 5-7	$-0.1504 \pm j7.3542$	132.08	111.93	67.10	117.77	83.84	104.81
line 6-9	$-0.1941 \pm j8.0361$	115.34	149.62	45.84	117.77	83.84	104.81
line 7-8	$-0.2426 \pm j8.8572$	84.77	111.93	115.84	117.77	83.84	104.81
line 9-8	$-0.1793 \pm j9.0672$	68.47	181.49	60.87	117.77	83.84	104.81

Table 10.3: Summary of SSS ESCOPF results, $\text{Re}(\lambda) \leq -0.15$.

State	Critical λ	P_{G1}	P_{G2}	P_{G3}	P_{L5}	P_{L6}	P_{L8}
pre-cont.	$-0.2395 \pm j9.3605$	101.85	128.17	79.15	117.55	83.86	104.90
line 4-6	$-0.2002 \pm j8.0186$	83.91	150.62	76.85	117.55	83.86	104.90
line 4-5	$-0.2007 \pm j7.2663$	80.75	135.91	66.02	89.34	83.86	104.90
line 5-7	$-0.2002 \pm j7.2127$	151.85	93.17	64.92	117.55	83.86	104.90
line 6-9	$-0.2040 \pm j8.0177$	120.91	145.34	44.15	117.55	83.86	104.90
line 7-8	$-0.2968 \pm j8.7305$	100.08	97.36	114.15	117.55	83.86	104.90
line 9-8	$-0.2000 \pm j9.9721$	90.35	161.99	57.66	117.55	83.86	104.90

Table 10.4: Summary of SSS ESCOPF results, $\text{Re}(\lambda) \leq -0.20$.

State	Critical λ	P_{G1}	P_{G2}	P_{G3}	P_{L5}	P_{L6}	P_{L8}
pre-cont.	$-0.2809 \pm j9.2405$	116.67	114.19	77.92	117.37	83.88	104.95
line 4-6	$-0.2503 \pm j7.7805$	103.72	145.36	62.66	117.37	83.88	104.95
line 4-5	$-0.2506 \pm j7.0335$	114.23	126.28	42.92	88.68	83.88	104.95
line 5-7	$-0.2503 \pm j7.0745$	166.67	79.19	63.58	117.37	83.88	104.95
line 6-9	$-0.2513 \pm j8.8616$	155.18	111.13	42.92	117.37	83.88	104.95
line 7-8	$-0.2963 \pm j8.7323$	101.13	97.45	112.92	117.37	83.88	104.95
line 9-8	$-0.2500 \pm j8.7992$	114.44	140.67	54.36	117.37	83.88	104.95

Table 10.5: Summary of SSS ESCOPF results, $\text{Re}(\lambda) \leq -0.25$.

CHAPTER 10. OPF AND ESCOPF WITH SMALL-SIGNAL STABILITY CONSTRAINTS

margin constraint was binding for the post-contingency states in which lines 4-6, 4-5, and 5-7 were taken out as shown in Table 10.3. As the stability margin was increased, to -0.20 and -0.25, four post-contingency states had binding stability inequality constraints. Also, the amount of power generated by the machine at bus 1 gradually went up, and the amount of power generated by machines at buses 2 and 3 went down. This means Generator 1 is more expensive, but using more power from it improves small-signal stability, which is probably because its inertia constant is much higher than the other two machines. Tables 10.4 and 10.5 list results when the stability margin is increased to -0.20 and -0.25.

Generator ramping limits play a role in each case. Bold face generation levels in the tables correspond to instances where generator ramping limit constraints were active. The first generator wants to ramp higher when line 5-7 is lost, but it can't because of the upward ramping limit of 50 MW. The second generator wants to ramp down by more than 35 MW for the line 5-7 and line 7-8 post-contingency states when the stability margin is set at -0.15. However, when the stability margin increases to -0.20, the downward ramping limit for Generator 2 is binding for only the line 5-7 constraint. The line 6-9 contingency and line 7-8 contingencies result in binding upward and downward generator ramping constraints.

Table 10.6 shows how the optimal expected value of social welfare obtained from solving the ESCOPF problem changes as the security margin is changed. As expected, the expected value of social welfare decreases as the security margin increases. By comparing social welfare values in Table 10.6 with those in Table 10.1, we can also see that the pre-contingency social welfare is lower for the unconstrained OPF case than for the unconstrained ESCOPF case. The same holds true for the case with a stability margin of -0.20. The reason for the decrease in the optimal pre-contingency social welfare is that we consider the costs associated with ramping and dropping load in the ESCOPF formulation. By operating at a slightly different pre-contingency operating point than the one found from solving the OPF, we get a higher expected value of social welfare.

The derivative of the expected value of social welfare with respect to the stability margin λ_{\max}

Constraint $\text{Re}(\lambda)$	Right-most eigenvalue	$E[SW]$ (\$/hr)	SW^0 (\$/hr)	$dE[SW]/d\lambda_{\max}$
0	$-0.1266 \pm j7.4864$	2314.59	2317.23	-
-0.15	$-0.1504 \pm j7.3542$	2313.45	2315.97	3.18
-0.20	$-0.2000 \pm j9.9721$	2307.96	2310.54	6.67
-0.25	$-0.2500 \pm j8.7992$	2300.80	2303.84	8.19

Table 10.6: Summary of SSS ESCOPF results.

is computed by summing the Lagrange multipliers on the binding stability constraints. The positive values shown in Table 10.6 indicate the expected value of social welfare increases when the stability constraint is relaxed.

10.5.1 Using Arnoldi to speed up algorithm

To speed up convergence, the program was also run placing constraints on only the ten most critical eigenvalues. This was accomplished using the `eigs.m` function of Matlab, which accepts arguments that specify how many eigenvalues to calculate and what type of eigenvalues to calculate. It can, for example, calculate the 10 right-most eigenvalues or the 10 eigenvalues with largest magnitude. The `eigs.m` function makes calls to Fortran routines written by students and faculty at the Department of Computational and Applied Mathematics (CAAM) at Rice University. They have written an entire collection of routines and given them the name ARPACK [22] which stands for “ARnoldi PACKAge.” The software in the package uses a variant of the Arnoldi process called the Implicitly Restarted Arnoldi Method (IRAM), and the package is suitable for large, sparse matrices. The A_{sys} matrix for which we desire to find the eigenvalues contains 341 elements of which around 107 are non-zero, so it is somewhat sparse.

Using ARPACK routines, we need only compute a few of the dominant eigenvalues, and we place inequality constraints on only those few eigenvalues. This has the effect of saving a significant amount of computation time. For the 9-bus test case, each Newton-Raphson iteration was taking 65 seconds on a system with an Athlon XP 1700+ processor. That time was shrunk to 45 seconds after eliminating constraints on the non-critical eigenvalues. For such a small system, eigenvalue

CHAPTER 10. OPF AND ESCOPF WITH SMALL-SIGNAL STABILITY CONSTRAINTS

computation time is a very insignificant amount of the total time needed for each iteration. The major time savings for the 9-bus case resulted from not having to calculate the extra eigenvalue and eigenvector sensitivities. However, for larger systems, the savings in eigenvalue computation time would be much more important.

Chapter 11

Conclusions & future work

The objective of this dissertation was to solve an optimization problem that includes constraints on power system security, small-signal stability, and basic operating limits. Our optimization problem objective function is the expected value of social welfare and includes costs on generator ramping and load interruption that may be necessary when a contingency occurs. Extra constraints on generator ramping and load interruption were included. These constraints limit how much a generator can adjust its real power output when a contingency occurs, and they represent the ability to drop load for interruptible customers but not to increase their load.

To include small-signal stability constraints, we linearize the system differential and algebraic equations about a power system operating point, which gives us a system state matrix that characterizes the set of linear differential equations. We place limits on the eigenvalues of the state matrix that force the real parts of the eigenvalues to be negative and to lie a minimum distance (the stability margin) from the imaginary axis of the complex plane.

The eigenvalues of the system state matrix change as the system operating point changes and are therefore a function of the variables that describe the power system operating point. Since we use the primal-dual interior-point algorithm to solve the optimization problem, we need the gradient and Hessian matrix for each eigenvalue so we can compute the Newton step at each iteration. This means we need first-order and second-order sensitivities of the eigenvalues with respect to the optimization problem decision variables.

CHAPTER 11. CONCLUSIONS & FUTURE WORK

To make the optimization problem process solution process more efficient, we can constrain only the critical eigenvalues. This leads to a solution in a smaller amount of time because the optimization problem has fewer constraints and is therefore smaller. Also, computing eigenvalues at each iteration is time consuming for large matrices. By constraining only a few eigenvalues, we need to calculate only a few, which can be done using an implicitly restarted arnoldi method.

A Matlab program was written to solve the small-signal stability constrained expected-security-cost optimal power flow problem. The program reads plain text files which contain system data. This gives the ability to solve the problem for any system by simply entering the data describing the system into the text files.

The small-signal stability OPF and ESCOPF was solved for a 9-bus, 3-machine test system. The solution to the problems changed when the stability constraints were tightened, and the eigenvalue real part was successfully constrained for each case. For the OPF problem, the value of social welfare decreased from \$2319.90/hr in the unconstrained case to \$2275.10/hr when the stability margin was increased to -0.40 . For the ESCOPF problem, the expected value of social welfare decreased from \$2317.23/hr in the unconstrained case to \$2303.84/hr when the stability margin was increased to -0.25 . Therefore, as expected, the optimal value of the objective function (social welfare for the OPF problem and expected value of social welfare for the ESCOPF problem) decreases as the stability margin is increased. These results show that it is possible to improve the system small-signal stability margin by including constraints on the eigenvalue real parts in an OPF or ESCOPF problem.

Although our algorithm successfully solves the small-signal stability ESCOPF problem, it converges somewhat slowly in that it takes many iterations to get close initially to the barrier path and then move down the path to the solution. Future work should involve improving the convergence properties of the algorithm.

The ESCOPF problem is very large and we make it even larger by adding stability constraints. Computing updates at every Newton iteration requires solving of a large set of linear algebraic

CHAPTER 11. CONCLUSIONS & FUTURE WORK

equations. Future studies should focus on finding ways to decouple the problem into pre-contingency and post-contingency subproblems which could be solved simultaneously using distributed computing resources.

Finally, we describe a method for computing the optimal operating point assuming we are given cost curve and benefit curve information for each system participant. From this information, we can determine the optimal generation levels, consumption levels, and the price of power at each node in the system that leads to these generation and consumption levels. Marginal values of increasing ramping rates, increasing stability margins, increasing line flow limits, and interrupting a “negative amount of load” are also obtained. Work is needed in the future on designing a market which would lead to the optimal operating point for which both security costs and stability constraints are taken into account.

Bibliography

- [1] C.-L. Chang, A.-S. Liu, and C.-T. Huang, "Oscillatory stability analysis using real-time measured data," *IEEE Transactions on Power Systems*, vol. 8, pp. 823–829, August 1993.
- [2] W. Fairney, A. Miles, T. Whitelegg, and N. Murray, "Low frequency oscillations on the inter-connected system between Scotland and England," *CIGRE*, September 1982.
- [3] R. Cressap and J. Hauer, "Emergence of a new swing mode in the western power system," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-100, pp. 1027–2043.
- [4] J. D. Glover and M. Sarma, *Power System Analysis and Design*. Boston: PWS, 1994.
- [5] T. W. Gedra, *Power Economics and Regulation*. To Be Published.
- [6] P. Damrongkulkamjorn, *Optimal Power Flow with Expected Security Costs*. PhD thesis, Oklahoma State University, 1999.
- [7] S. J. Wright, *Primal-Dual Interior-Point Methods*. Philadelphia: SIAM: Society for Industrial and Applied Mathematics, 1997.
- [8] J. Momoh, M. El-Hawary, and R. Adapa, "A review of selected optimal power flow literature to 1993. ii. newton, linear programming and interior point methods," *IEEE Transactions on Power Systems*, vol. 14, pp. 105–111, February 1999.
- [9] P. Krause, *Analysis of Electric Machinery*. New York: McGraw-Hill, 1986.

BIBLIOGRAPHY

- [10] P. W. Sauer and M. Pai, *Power System Dynamics and Stability*. Upper Saddle River, NJ: Prentice Hall, 1998.
- [11] P. Kundur, *Power System Stability and Control*. McGraw-Hill, 1994.
- [12] G. Golub and C. V. Loan, *Matrix Computations*. Baltimore: Johns Hopkins U. Press, 3rd ed., 1996.
- [13] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*. Philadelphia: SIAM, 1997.
- [14] K. Wang, C. Chung, C. Tse, and K. Tsang, "Multimachine eigenvalue sensitivities of power system parameters," *IEEE Transactions on Power Systems*, vol. 15, pp. 741–747, May 2000.
- [15] H. Z. El-Din and R. Alden, "Second order eigenvalue sensitivities applied to power system dynamics," *IEEE Transaction on Power Apparatus and Systems*, vol. PAS-96, pp. 1928–1936, November/December 1977.
- [16] D. Faddeev and V. Faddeeva, *Computational Methods of Linear Algebra*. San Francisco: Freeman, 1963.
- [17] M. Jankovic, "Exact n th derivatives of eigenvalues and eigenvectors," *Journal of Guidance, Control, and Dynamics*, vol. 17, pp. 136–144, January-February 1994.
- [18] P. Sauer, B. Lesieutre, and M. Pai, "Maximum loadability and voltage stability in power systems," *International Journal of Electrical Power and Energy Systems*, vol. 15, pp. 145–154, June 1993.
- [19] P. Anderson and A. Fouad, *Power System Control and Stability*. Ames, IA: Iowa State University Press, 1977.
- [20] M. Pal, "Voltage stability analysis needs," *IEE Proceedings of Part C*, vol. 140, pp. 279–286, July 1993.

BIBLIOGRAPHY

- [21] W. G. on a Common Format for the Exchange of Solved Load Flow Data, "Common data format for the exchange of solved load flow data," *IEEE Transaction on Power Apparatus and Systems*, vol. PAS-92, pp. 1916–1925, November/December 1973.
- [22] R. Lehoucq, D. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. Philadelphia: SIAM, 1998.

Appendix A

Linearizing the differential and algebraic equations.

A.1 Linearizing the differential equations

The next step is to linearize the differential equations. The differential equations are

$$\frac{d\delta_i}{dt} = \omega_i - \omega_s \quad (\text{A.1})$$

$$\frac{d\omega_i}{dt} = \frac{T_{Mi}}{M_i} - \frac{[E'_{qi} - X'_{di}I_{di}]I_{qi}}{M_i} - \frac{[E'_{di} + X'_{qi}I_{qi}]I_{di}}{M_i} - \frac{D_i(\omega_i - \omega_s)}{M_i} \quad (\text{A.2})$$

$$\frac{dE'_{qi}}{dt} = -\frac{E'_{qi}}{T'_{doi}} - \frac{(X'_{di} - X'_{qi})I_{di}}{T'_{doi}} + \frac{E_{fdi}}{T'_{doi}} \quad (\text{A.3})$$

$$\frac{dE'_{di}}{dt} = -\frac{E'_{di}}{T'_{qoi}} + \frac{(X'_{qi} - X'_{di})I_{qi}}{T'_{qoi}} \quad (\text{A.4})$$

$$\frac{dE_{fdi}}{dt} = -\frac{K_{Ei} + S_E(E_{fdi})}{T_{Ei}} E_{fdi} + \frac{V_{Ri}}{T_{Ei}} \quad (\text{A.5})$$

$$\frac{dV_{Ri}}{dt} = -\frac{V_{Ri}}{T_{Ai}} + \frac{K_{Ai}}{T_{Ai}} R_{fi} - \frac{K_{Ai}K_{Fi}}{T_{Ai}T_{Fi}} E_{fdi} + \frac{K_{Ai}}{T_{Ai}} (V_{\text{ref}_i} - V_i) \quad (\text{A.6})$$

$$\frac{dR_{Fi}}{dt} = -\frac{R_{Fi}}{T_{Fi}} + \frac{K_{Fi}}{(T_{Fi})^2} E_{fdi} \quad (\text{A.7})$$

for $i = 1, \dots, m$

APPENDIX A. LINEARIZING THE DIFFERENTIAL AND ALGEBRAIC EQUATIONS.

These equations can be linearized as follows:

$$\frac{d\Delta\delta_i}{dt} = \Delta\omega_i - \omega_s \quad (\text{A.8})$$

$$\begin{aligned} \frac{d\Delta\omega_i}{dt} &= \frac{1}{M_i} \Delta T_{Mi} - \frac{E'_{qio}}{M_i} \Delta I_{qi} - \frac{I_{qio}}{M_i} \Delta E'_{qi} + \frac{X'_{di} I_{dio}}{M_i} \Delta I_{qi} + \frac{X'_{di} I_{qio}}{M_i} \Delta I_{di} \\ &\quad - \frac{E'_{dio}}{M_i} \Delta I_{di} - \frac{I_{dio}}{M_i} \Delta E'_{di} - \frac{X'_{qi} I_{dio}}{M_i} \Delta I_{qi} - \frac{X'_{qi} I_{qio}}{M_i} \Delta I_{di} - \frac{D_i}{M_i} \Delta\omega_i \end{aligned} \quad (\text{A.9})$$

$$\frac{d\Delta E'_{qi}}{dt} = -\frac{1}{T'_{doi}} \Delta E'_{qi} - \frac{(X_{di} - X'_{di})}{T'_{doi}} \Delta I_{di} + \frac{1}{T'_{doi}} \Delta E_{fdi} \quad (\text{A.10})$$

$$\frac{d\Delta E'_{di}}{dt} = -\frac{1}{T'_{qoi}} \Delta E'_{di} + \frac{(X_{qi} - X'_{qi})}{T'_{qoi}} \Delta I_{qi} \quad (\text{A.11})$$

$$\frac{d\Delta E_{fdi}}{dt} = -\frac{K_{Ei}}{T_{Ei}} \Delta E_{fdi} - \frac{1}{T_{Ei}} \left(S_E(E_{fdio}) + E_{fdio} \frac{\partial S_E}{\partial E_{fdio}} \right) \Delta E_{fdi} + \frac{1}{T_{Ei}} \Delta V_{Ri} \quad (\text{A.12})$$

$$\begin{aligned} \frac{d\Delta V_{Ri}}{dt} &= -\frac{1}{T_{Ai}} \Delta V_{Ri} + \frac{K_{Ai}}{T_{Ai}} \Delta R_{fi} - \frac{K_{Ai} K_{Fi}}{T_{Ai} T_{Fi}} \Delta E_{fdi} + \frac{K_{Ai}}{T_{Ai}} \Delta V_{\text{ref}_i} \\ &\quad - \frac{K_{Ai}}{T_{Ai}} \Delta V_i \end{aligned} \quad (\text{A.13})$$

$$\frac{d\Delta R_{Fi}}{dt} = -\frac{1}{T_{Fi}} \Delta R_{Fi} + \frac{K_{Fi}}{(T_{Fi})^2} \Delta E_{fdi} \quad (\text{A.14})$$

for $i = 1, \dots, m$

APPENDIX A. LINEARIZING THE DIFFERENTIAL AND ALGEBRAIC EQUATIONS.

In matrix notation, the linearized differential equations can be written as

$$\begin{aligned}
 \begin{bmatrix} \Delta \dot{\delta}_i \\ \Delta \dot{\omega}_i \\ \Delta \dot{E}'_{qi} \\ \Delta \dot{E}'_{di} \\ \Delta \dot{E}'_{fdi} \\ \Delta \dot{V}_{Ri} \\ \Delta \dot{R}_{Fi} \end{bmatrix} &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{D_i}{M_i} & -\frac{I_{qio}}{M_i} & -\frac{I_{dio}}{M_i} & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{T'_{doi}} & 0 & \frac{1}{T'_{doi}} & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{T'_{qoi}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & f_{si}(E_{fdio}) & \frac{1}{T_{Ei}} & 0 \\ 0 & 0 & 0 & 0 & -\frac{K_{Ai}K_{Fi}}{T_{Ai}T_{Fi}} & -\frac{1}{T_{Ai}} & \frac{K_{Ai}}{T_{Ai}} \\ 0 & 0 & 0 & 0 & -\frac{K_{Fi}}{(T_{Fi})^2} & 0 & -\frac{1}{T_{Fi}} \end{bmatrix} \begin{bmatrix} \Delta \delta_i \\ \Delta \omega_i \\ \Delta E'_{qi} \\ \Delta E'_{di} \\ \Delta E_{fdi} \\ \Delta V_{Ri} \\ \Delta R_{Fi} \end{bmatrix} \\
 &+ \begin{bmatrix} 0 & 0 \\ \frac{I_{qio}(X'_{di}-X'_{qi})-E'_{dio}}{M_i} & \frac{I_{dio}(X'_{di}-X'_{qi})-E'_{qio}}{M_i} \\ -\frac{(X_{di}-X'_{di})}{T'_{doi}} & 0 \\ 0 & -\frac{(X_{qi}-X'_{qi})}{T'_{qoi}} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta I_{di} \\ \Delta I_{qi} \end{bmatrix} \\
 &+ \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -\frac{K_{Ai}}{T_{Ai}} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta V_i \\ \Delta \theta_i \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{1}{M_i} & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \frac{K_{Ai}}{T_{Ai}} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta T_{Mi} \\ \Delta V_{refi} \end{bmatrix} \tag{A.15}
 \end{aligned}$$

for $i = 1, \dots, m$

where $f_{si}(E_{fdio}) = -\frac{K_{Ei} + E_{fdio} \frac{\partial S_E}{\partial E_{fdi}} + S_E(E_{fdio})}{T_{Ei}}$.

APPENDIX A. LINEARIZING THE DIFFERENTIAL AND ALGEBRAIC EQUATIONS.

Denoting $\begin{bmatrix} \Delta I_{di} \\ \Delta I_{qi} \end{bmatrix} = \Delta I_{gi}$, $\begin{bmatrix} \Delta V_i \\ \Delta \theta_i \end{bmatrix} = \Delta V_{gi}$, $\begin{bmatrix} \Delta T_{Mi} \\ \Delta V_{refi} \end{bmatrix} = \Delta u_i$, A.15 can be written as

$$\Delta \dot{x}_i = A_{1i} \Delta x_i + B_{1i} \Delta I_{gi} + B_{2i} \Delta V_{gi} + E_{1i} \Delta u_i \text{ for } i = 1, \dots, m$$

For the m -machine system, A.16 can be expressed as

$$\Delta \dot{x} = A_1 \Delta x + B_1 \Delta I_g + B_2 \Delta V_g + E_1 \Delta U \quad (\text{A.16})$$

A.2 Stator Algebraic Equation

The static algebraic equation is

$$E'_{di} + jE'_{qi} - jV_i e^{j(\theta_i - \delta_i)} - (R_{si} + jX'_{di})I_{di} - (R_{si} + jX'_{qi})jI_{qi} = 0 \quad (\text{A.17})$$

$$\text{for } i = 1, \dots, m$$

It can be linearized as follows:

$$\begin{aligned} \Delta E'_{di} + j\Delta E'_{qi} - (R_{si} + jX'_{di})\Delta I_{di} - (jR_{si} - X'_{qi})\Delta I_{qi} - j e^{j(\theta_{io} - \delta_{io})} \Delta V_i \\ + V_{io} e^{j(\theta_{io} - \delta_{io})} \Delta \theta_i - V_{io} e^{j(\theta_{io} - \delta_{io})} \Delta \delta_i = 0 \end{aligned} \quad (\text{A.18})$$

APPENDIX A. LINEARIZING THE DIFFERENTIAL AND ALGEBRAIC EQUATIONS.

In matrix form, this is

$$\begin{aligned}
 & \begin{bmatrix} \operatorname{Re}(-V_{i0}e^{j(\theta_{i0}-\delta_{i0})}) & 0 & 0 & 1 & 0 & 0 & 0 \\ \operatorname{Im}(-V_{i0}e^{j(\theta_{i0}-\delta_{i0})}) & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\delta_i \\ \Delta\omega_i \\ \Delta E'_{qi} \\ \Delta E'_{di} \\ \Delta E_{fdi} \\ \Delta V_{Ri} \\ \Delta R_{Fi} \end{bmatrix} \\
 & + \begin{bmatrix} -R_{si} & X'_{qi} \\ -X'_{di} & -R_{si} \end{bmatrix} \begin{bmatrix} \Delta I_{di} \\ \Delta I_{qi} \end{bmatrix} \\
 & + \begin{bmatrix} \operatorname{Re}(-je^{j(\theta_{i0}-\delta_{i0})}) & \operatorname{Re}(V_{i0}e^{j(\theta_{i0}-\delta_{i0})}) \\ \operatorname{Im}(-je^{j(\theta_{i0}-\delta_{i0})}) & \operatorname{Im}(V_{i0}e^{j(\theta_{i0}-\delta_{i0})}) \end{bmatrix} \begin{bmatrix} \Delta V_i \\ \Delta\theta_i \end{bmatrix} \tag{A.19}
 \end{aligned}$$

We can rewrite A.19 as

$$0 = C_{1i}\Delta x_i + D_{1i}\Delta I_{gi} + D_{2i}\Delta V_{gi} \quad \text{for } i = 1, \dots, m \tag{A.20}$$

Finally, we may combine stator algebraic equations for all buses and write

$$0 = C_1\Delta x + D_1\Delta I_g + D_2\Delta V_g \tag{A.21}$$

A.3 Network Equation

A.3.1 Generator buses

The complex network equation relates real and reactive power injections at each bus to voltage magnitudes and phase angles at the system buses. First we will right the network equations for the

APPENDIX A. LINEARIZING THE DIFFERENTIAL AND ALGEBRAIC EQUATIONS.

generator buses:

$$jV_i e^{j(\theta_i - \delta_i)} (I_{di} - jI_{qi}) - P_{Li} - jQ_{Li} = S_i(V, \theta) \quad (\text{A.22})$$

for $i = 1, \dots, m$

It can be linearized as follows:

$$\begin{aligned} jV_i e^{j(\theta_i - \delta_i)} \Delta I_{di} + V_i e^{j(\theta_i - \delta_i)} \Delta I_{qi} + \left[jI_{di} e^{j(\theta_i - \delta_i)} + I_{qi} e^{j(\theta_i - \delta_i)} \right] \Delta V_i \\ + \left[-I_{di} V_i e^{j(\theta_i - \delta_i)} + jI_{qi} V_i e^{j(\theta_i - \delta_i)} \right] \Delta \theta_i \\ - \left[-I_{di} V_i e^{j(\theta_i - \delta_i)} + jI_{qi} V_i e^{j(\theta_i - \delta_i)} \right] \Delta \delta_i \\ = \frac{\partial S_i(V, \theta)}{\partial V} \Delta V + \frac{\partial S_i(V, \theta)}{\partial \theta} \Delta \theta \end{aligned}$$

We can write this in matrix notation as

$$0 = C_2 \Delta x + D_3 \Delta I_g + D_4 \Delta V_g + D_5 \Delta V_l \quad (\text{A.23})$$

where

$$\Delta V_l = \begin{bmatrix} \Delta V_{m+1} \\ \vdots \\ \Delta V_n \\ \Delta \theta_{m+1} \\ \vdots \\ \Delta \theta_n \end{bmatrix}$$

Because we break down each network equation into real and imaginary parts, C_2 , D_3 , D_4 , and D_5 have $2m$ rows. There are seven state variables, so C_2 will have $7m$ columns. ΔI_g has $2m$ columns, and so will D_3 . $D_4 \Delta V_g$ and therefore D_4 also have $2m$ columns. Finally, both ΔV_l and D_5 have $2(n - m)$ columns.

APPENDIX A. LINEARIZING THE DIFFERENTIAL AND ALGEBRAIC EQUATIONS.

matrix	number of rows	number of columns
A_1	$7m$	$7m$
B_1	$7m$	$2m$
B_2	$7m$	$2m$
E_1	$7m$	$2m$
C_1	$2m$	$7m$
D_1	$2m$	$2m$
D_2	$2m$	$2m$
C_2	$2m$	$7m$
D_3	$2m$	$2m$
D_4	$2m$	$2m$
D_5	$2m$	$2(n - m)$
D_6	$2(n - m)$	$2m$
D_7	$2(n - m)$	$2(n - m)$

Table A.1: Size of matrices in linearized differential, algebraic, and network equations.

A.3.2 Non-generator buses

The complex network equation for buses that do not have a generator is

$$-P_{Li} - jQ_{Li} = S_i(V, \theta) \quad (\text{A.24})$$

$$\text{for } i = m + 1, \dots, n$$

It can be linearized as follows:

$$0 = \frac{\partial S_i(V, \theta)}{\partial V} \Delta V + \frac{\partial S_i(V, \theta)}{\partial \theta} \Delta \theta$$

In matrix notation, we have

$$0 = D_6 \Delta V_g + D_7 \Delta V_i \quad (\text{A.25})$$

Each of these matrices has $2(n - m)$ rows because we break the network equation for the non-generator buses into real and imaginary parts. Table A.1 summarizes matrix sizes.

Appendix B

Forming the A_{sys} matrix

The system A_{sys} matrix is the matrix that relates state variables and inputs in the following way:

$$\Delta \dot{x} = A_{\text{sys}} \Delta x + E_1 \Delta U. \quad (\text{B.1})$$

We would like to compute the A_{sys} matrix. First, let's review the equations we have so far

$$\Delta \dot{x} = A_1 \Delta x + B_1 \Delta I_g + B_2 \Delta V_g + E_1 \Delta U \quad (\text{B.2})$$

$$0 = C_1 \Delta x + D_1 \Delta I_g + D_2 \Delta V_g \quad (\text{B.3})$$

$$0 = C_2 \Delta x + D_3 \Delta I_g + D_4 \Delta V_g + D_5 \Delta V_l \quad (\text{B.4})$$

$$0 = D_6 \Delta V_g + D_7 \Delta V_l \quad (\text{B.5})$$

These can be rearranged as follows:

$$\begin{bmatrix} \Delta \dot{x} \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} A_1 & B_1 & B_2 & 0 \\ C_1 & D_1 & D_2 & 0 \\ C_2 & D_3 & D_4 & D_5 \\ 0 & 0 & D_6 & D_7 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta I_g \\ \Delta V_g \\ \Delta V_l \end{bmatrix} + \begin{bmatrix} E_1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Delta U \quad (\text{B.6})$$

APPENDIX B. FORMING THE A_{SYS} MATRIX

We will first solve for ΔI_g in B.3. This gives us

$$\Delta I_g = -D_1^{-1}C_1\Delta x - D_1^{-1}D_2\Delta V_g. \quad (\text{B.7})$$

Substituting this into B.2 and B.4 gives us

$$\begin{bmatrix} \Delta \dot{x} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} A_1 - B_1D_1^{-1}C_1 & B_2 - B_1D_1^{-1}D_2 & 0 \\ C_2 - D_3D_1^{-1}C_1 & D_4 - D_3D_1^{-1}D_2 & D_5 \\ 0 & D_6 & D_7 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta V_g \\ \Delta V_l \end{bmatrix} + \begin{bmatrix} E_1 \\ 0 \\ 0 \end{bmatrix} \Delta U \quad (\text{B.8})$$

Let us extract the top and bottom parts of B.8 as follows:

$$\Delta \dot{x} = \begin{bmatrix} A_1 - B_1D_1^{-1}C_1 & B_2 - B_1D_1^{-1}D_2 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta V_g \\ \Delta V_l \end{bmatrix} + E_1\Delta U \quad (\text{B.9})$$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} C_2 - D_3D_1^{-1}C_1 \\ 0 \end{bmatrix} \Delta x + \begin{bmatrix} D_4 - D_3D_1^{-1}D_2 & D_5 \\ D_6 & D_7 \end{bmatrix} \begin{bmatrix} \Delta V_g \\ \Delta V_l \end{bmatrix}. \quad (\text{B.10})$$

Let us make the following substitutions:

$$K_3 = \begin{bmatrix} B_2 - B_1D_1^{-1}D_2 & 0 \end{bmatrix} \quad (\text{B.11})$$

$$K_4 = \begin{bmatrix} D_4 - D_3D_1^{-1}D_2 & D_5 \\ D_6 & D_7 \end{bmatrix} \quad (\text{B.12})$$

$$K_5 = \begin{bmatrix} C_2 - D_3D_1^{-1}C_1 \\ 0 \end{bmatrix}. \quad (\text{B.13})$$

APPENDIX B. FORMING THE A_{sys} MATRIX

Solving B.10 for $\begin{bmatrix} \Delta V_g \\ \Delta V_l \end{bmatrix}$ and substituting into B.9, we have

$$\Delta \dot{x} = \left[A_1 - B_1 D_1^{-1} C_1 - K_3 K_4^{-1} K_5 \right] \Delta x + E_1 \Delta U. \quad (B.14)$$

We now have an expression for the system A_{sys} matrix:

$$A_{\text{sys}} = A_1 - B_1 D_1^{-1} C_1 - K_3 K_4^{-1} K_5 \quad (B.15)$$

Appendix C

Derivation of second order eigenvalue sensitivities computed from all eigenvalues

Here, a more detailed derivation of the second-order sensitivity than the one found in [15] is presented using our notation. Multiplying both sides of 9.5 on the left by ψ_j where $i \neq j$ gives us

$$\psi_j \frac{\partial A_{\text{sys}}}{\partial \epsilon_1} \phi_i + \underbrace{\psi_j A_{\text{sys}}}_{\psi_j \lambda_j} \frac{\partial \phi_i}{\partial \epsilon_1} = \underbrace{\psi_j \phi_i}_0 \frac{\partial \lambda_i}{\partial \epsilon_1} + \psi_j \lambda_i \frac{\partial \phi_i}{\partial \epsilon_1}. \quad (\text{C.1})$$

After making the substitutions, we have

$$\psi_j \frac{\partial A_{\text{sys}}}{\partial \epsilon_1} \phi_i + \underbrace{\psi_j \lambda_j}_{\psi_j \lambda_j} \frac{\partial \phi_i}{\partial \epsilon_1} = \underbrace{\psi_j \lambda_i}_{\psi_j \lambda_i} \frac{\partial \phi_i}{\partial \epsilon_1}. \quad (\text{C.2})$$

Group like terms and factor:

$$\psi_j \frac{\partial A_{\text{sys}}}{\partial \epsilon_1} \phi_i = (\lambda_i - \lambda_j) \psi_j \frac{\partial \phi_i}{\partial \epsilon_1}, \quad (\text{C.3})$$

APPENDIX C. DERIVATION OF SECOND ORDER EIGENVALUE SENSITIVITIES COMPUTED FROM ALL EIGENVALUES

i.e.

$$\psi_j \frac{\partial \phi_i}{\partial \epsilon_1} = \frac{\psi_j \frac{\partial A_{\text{sys}}}{\partial \epsilon_1} \phi_i}{(\lambda_i - \lambda_j)}. \quad (\text{C.4})$$

Since the set of eigenvectors span \mathbb{C}^n , we can write the eigenvector sensitivity as a linear combination of the eigenvectors, i.e.

$$\frac{\partial \phi_i}{\partial \epsilon_1} = \sum_{k=1}^n \alpha_{ik} \phi_k. \quad (\text{C.5})$$

Then

$$\psi_j \frac{\partial \phi_i}{\partial \epsilon_1} = \psi_j \sum_{k=1}^n \alpha_{ik} \phi_k \quad (\text{C.6})$$

$$= \psi_j \alpha_{ij} \phi_j \quad (\text{C.7})$$

since $\psi_j \phi_k = 0$ for $j \neq k$. Substituting the right side of C.6 for the left side of C.4,

$$\psi_j \alpha_{ij} \phi_j = \frac{\psi_j \frac{\partial A_{\text{sys}}}{\partial \epsilon_1} \phi_i}{(\lambda_i - \lambda_j)}, \quad (\text{C.8})$$

i.e.

$$\alpha_{ij} = \frac{\psi_j \frac{\partial A_{\text{sys}}}{\partial \epsilon_1} \phi_i}{\psi_j \phi_j (\lambda_i - \lambda_j)}. \quad (\text{C.9})$$

The coefficient α_{ij} is not defined for $i = j$. We will leave α_{ii} as an unknown for now and show that it can be eliminated later.

Taking the partial derivative with respect to ϵ_2 of both sides of 9.5 and then multiplying on the left by ψ_i gives us

$$\begin{aligned} & \psi_i \frac{\partial^2 A_{\text{sys}}}{\partial \epsilon_2 \partial \epsilon_1} \phi_i + \psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_1} \frac{\partial \phi_i}{\partial \epsilon_2} + \psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_2} \frac{\partial \phi_i}{\partial \epsilon_1} + \underbrace{\psi_i A_{\text{sys}}}_{\psi_i \lambda_i} \frac{\partial^2 \phi_i}{\partial \epsilon_2 \partial \epsilon_1} \\ & = \psi_i \frac{\partial^2 \lambda_i}{\partial \epsilon_2 \partial \epsilon_1} \phi_i + \psi_i \frac{\partial \lambda_i}{\partial \epsilon_1} \frac{\partial \phi_i}{\partial \epsilon_2} + \psi_i \frac{\partial \lambda_i}{\partial \epsilon_2} \frac{\partial \phi_i}{\partial \epsilon_1} + \underbrace{\psi_i \lambda_i}_{\psi_i A_{\text{sys}}} \frac{\partial^2 \phi_i}{\partial \epsilon_2 \partial \epsilon_1} \end{aligned} \quad (\text{C.10})$$

APPENDIX C. DERIVATION OF SECOND ORDER EIGENVALUE SENSITIVITIES COMPUTED FROM ALL EIGENVALUES

Since $\psi_i A_{\text{sys}} = \psi_i \lambda_i$, we can reduce the equation to

$$\begin{aligned} \psi_i \frac{\partial^2 A_{\text{sys}}}{\partial \epsilon_2 \partial \epsilon_1} \phi_i + \psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_1} \underbrace{\frac{\partial \phi_i}{\partial \epsilon_2}}_{\sum_{k=1}^n \beta_{ik} \phi_k} + \psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_2} \underbrace{\frac{\partial \phi_i}{\partial \epsilon_1}}_{\sum_{k=1}^n \alpha_{ik} \phi_k} \\ = \psi_i \frac{\partial^2 \lambda_i}{\partial \epsilon_2 \partial \epsilon_1} \phi_i + \psi_i \frac{\partial \lambda_i}{\partial \epsilon_1} \underbrace{\frac{\partial \phi_i}{\partial \epsilon_2}}_{\sum_{k=1}^n \beta_{ik} \phi_k} + \psi_i \frac{\partial \lambda_i}{\partial \epsilon_2} \underbrace{\frac{\partial \phi_i}{\partial \epsilon_1}}_{\sum_{k=1}^n \alpha_{ik} \phi_k} \end{aligned} \quad (\text{C.11})$$

Making the substitutions

$$\frac{\partial \phi_i}{\partial \epsilon_1} = \sum_{k=1}^n \alpha_{ik} \phi_k, \quad (\text{C.12})$$

$$\frac{\partial \phi_i}{\partial \epsilon_2} = \sum_{k=1}^n \beta_{ik} \phi_k, \quad (\text{C.13})$$

where α_{ik} and β_{ik} for $i \neq k$ are computed using C.9, we have

$$\begin{aligned} \psi_i \frac{\partial^2 A_{\text{sys}}}{\partial \epsilon_2 \partial \epsilon_1} \phi_i + \psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_1} \sum_{k=1}^n \beta_{ik} \phi_k + \psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_2} \sum_{k=1}^n \alpha_{ik} \phi_k \\ = \frac{\partial^2 \lambda_i}{\partial \epsilon_2 \partial \epsilon_1} \psi_i \phi_i + \frac{\partial \lambda_i}{\partial \epsilon_1} \psi_i \sum_{k=1}^n \beta_{ik} \phi_k + \frac{\partial \lambda_i}{\partial \epsilon_2} \psi_i \sum_{k=1}^n \alpha_{ik} \phi_k. \end{aligned} \quad (\text{C.14})$$

The summation terms can be split so that the term in which $k = i$ is not included in the sum; it is

APPENDIX C. DERIVATION OF SECOND ORDER EIGENVALUE SENSITIVITIES COMPUTED FROM ALL EIGENVALUES

written separately, i.e.

$$\begin{aligned}
& \psi_i \frac{\partial^2 A_{\text{sys}}}{\partial \epsilon_2 \partial \epsilon_1} \phi_i + \psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_1} \sum_{\substack{k=1 \\ k \neq i}}^n \beta_{ik} \phi_k + \beta_{ii} \psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_1} \phi_i \\
& \quad + \psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_2} \sum_{\substack{k=1 \\ k \neq i}}^n \alpha_{ik} \phi_k + \alpha_{ii} \psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_2} \phi_i \\
& = \frac{\partial^2 \lambda_i}{\partial \epsilon_2 \partial \epsilon_1} \psi_i \phi_i + \underbrace{\frac{\partial \lambda_i}{\partial \epsilon_1} \psi_i \sum_{\substack{k=1 \\ k \neq i}}^n \beta_{ik} \phi_k + \beta_{ii} \frac{\partial \lambda_i}{\partial \epsilon_1} \psi_i \phi_i}_0 \\
& \quad + \underbrace{\frac{\partial \lambda_i}{\partial \epsilon_2} \psi_i \sum_{\substack{k=1 \\ k \neq i}}^n \alpha_{ik} \phi_k + \alpha_{ii} \frac{\partial \lambda_i}{\partial \epsilon_2} \psi_i \phi_i}_0. \quad (\text{C.15})
\end{aligned}$$

Since $\psi_i \phi_k = 0$ for $k \neq i$, we can cancel the bracketed terms, which results in

$$\begin{aligned}
& \psi_i \frac{\partial^2 A_{\text{sys}}}{\partial \epsilon_2 \partial \epsilon_1} \phi_i + \psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_1} \sum_{\substack{k=1 \\ k \neq i}}^n \beta_{ik} \phi_k + \beta_{ii} \underbrace{\psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_1} \phi_i}_{\frac{\partial \lambda_i}{\partial \epsilon_1} \psi_i \phi_i} \\
& \quad + \psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_2} \sum_{\substack{k=1 \\ k \neq i}}^n \alpha_{ik} \phi_k + \alpha_{ii} \psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_2} \phi_i \\
& = \frac{\partial^2 \lambda_i}{\partial \epsilon_2 \partial \epsilon_1} \psi_i \phi_i + \beta_{ii} \underbrace{\frac{\partial \lambda_i}{\partial \epsilon_1} \psi_i \phi_i}_{\psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_1} \phi_i} + \alpha_{ii} \frac{\partial \lambda_i}{\partial \epsilon_2} \psi_i \phi_i. \quad (\text{C.16})
\end{aligned}$$

Using the eigenvalue sensitivity equation

$$\frac{\partial \lambda_i}{\partial \epsilon_1} \psi_i \phi_i = \frac{\partial A_{\text{sys}}}{\partial \epsilon_1} \phi_i$$

APPENDIX C. DERIVATION OF SECOND ORDER EIGENVALUE SENSITIVITIES COMPUTED FROM ALL EIGENVALUES

we can eliminate the designated terms, leaving us with

$$\begin{aligned}
 & \psi_i \frac{\partial^2 A_{\text{sys}}}{\partial \epsilon_2 \partial \epsilon_1} \phi_i + \psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_1} \sum_{\substack{k=1 \\ k \neq i}}^n \beta_{ik} \phi_k \\
 & \quad + \psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_2} \sum_{\substack{k=1 \\ k \neq i}}^n \alpha_{ik} \phi_k + \alpha_{ii} \underbrace{\psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_2} \phi_i}_{\frac{\partial \lambda_i}{\partial \epsilon_2} \psi_i \phi_i} \\
 & \quad = \frac{\partial^2 \lambda_i}{\partial \epsilon_2 \partial \epsilon_1} \psi_i \phi_i + \alpha_{ii} \underbrace{\frac{\partial \lambda_i}{\partial \epsilon_2} \psi_i \phi_i}_{\psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_2} \phi_i}. \quad (\text{C.17})
 \end{aligned}$$

Once again, we can use the eigenvalue sensitivity equation

$$\frac{\partial \lambda_i}{\partial \epsilon_2} \psi_i \phi_i = \frac{\partial A_{\text{sys}}}{\partial \epsilon_2} \phi_i$$

to eliminate terms. The result is

$$\psi_i \frac{\partial^2 A_{\text{sys}}}{\partial \epsilon_2 \partial \epsilon_1} \phi_i + \psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_1} \sum_{\substack{k=1 \\ k \neq i}}^n \beta_{ik} \phi_k + \psi_i \frac{\partial A_{\text{sys}}}{\partial \epsilon_2} \sum_{\substack{k=1 \\ k \neq i}}^n \alpha_{ik} \phi_k = \frac{\partial^2 \lambda_i}{\partial \epsilon_2 \partial \epsilon_1} \psi_i \phi_i, \quad (\text{C.18})$$

which is equivalent to the second-order sensitivity equation.

Appendix D

WSCC 9-bus test system data

All properties are the same as those given on page 171 of [10] except for MVA ratings, which were not given in the book. Finally, the book contains no bus voltage limits, so the limits given in Table D.2 are used. Consumer benefit curve coefficients are given in Table D.3. Generator cost curve coefficients and real power generation limits are given in Table D.4. Table D.5 lists system security data including generator ramping limits and customer load interruption cost curve coefficients. Synchronous machine and exciter data is given in Table D.6.

Bus no.	Bus no.	R	X	Y	MVA Rating
4	6	0.0170	0.0920	0.158	325
4	5	0.0100	0.0850	0.176	390
5	7	0.0320	0.1610	0.306	375
6	9	0.0390	0.1700	0.358	375
7	8	0.0085	0.0720	0.149	375
9	8	0.1190	0.1008	0.209	375
1	4	0.0000	0.0576	0.000	450
2	7	0.0000	0.0625	0.000	320
2	7	0.0000	0.0586	0.000	335

Table D.1: Transmission line and transformer data.

Bus no.	1	2	3	4	5	6	7	8	9
V_{\min}	0.950	0.955	0.955	0.955	0.955	0.955	0.955	0.950	0.955
V_{\max}	1.04	1.045	1.045	1.09	1.09	1.09	1.09	1.09	1.09

Table D.2: Bus voltage limits in per unit

APPENDIX D. WSCC 9-BUS TEST SYSTEM DATA

	γ (\$/(MW) ²)	β (\$/MW)	α (\$)
P_{L5}	-0.1047	38.665	0.00
P_{L6}	-0.0231	16.844	0.00
P_{L8}	-0.0431	21.630	0.00

Table D.3: Consumer benefit curve coefficients.

	γ (\$/(MW) ²)	β (\$/MW)	α (\$)	P_{Gmin} (MW)	P_{Gmax} (MW)
P_{G1}	8.20e-4	12.712	0.00	21	250
P_{G2}	8.76e-4	12.001	0.00	25	271
P_{G3}	6.46e-4	12.290	0.00	25	285

Table D.4: Generator cost curve coefficients and limits.

Generator Data									
Bus	1	2	3	4	5	6	7	8	9
$\Delta_{max}^+ P_G$	50	35	35	-	-	-	-	-	-
$\Delta_{max}^- P_G$	50	35	35	-	-	-	-	-	-
P_{Gmax}	250	270	285	-	-	-	-	-	-
P_{Gmin}	25	25	35	-	-	-	-	-	-
Load Data									
Bus	1	2	3	4	5	6	7	8	9
β_I	-	-	-	-	100	100	-	100	-

Table D.5: Generator and load security data for WSCC 9-bus case.

	Machine 1	Machine 2	Machine 3
H (sec)	23.64	6.4	3.01
D (sec)	0.0	0.0	0.0
X_d	0.146	0.8958	1.3125
X'_d	0.0608	0.1198	0.1813
X_q	0.969	0.8645	1.2547
X'_q	0.969	0.1969	0.25
T'_{do}	8.96	6.0	5.89
T'_{go}	0.31	0.535	0.6
K_A	20	20	20
T_A (sec)	0.2	0.2	0.2
K_E	1.0	1.0	1.0
T_E (sec)	0.314	0.314	0.314
K_F	0.063	0.063	0.063
T_F (sec)	0.35	0.35	0.35
$S_{Ei}(E_{fdi}) = 0.0039e^{1.555E_{fdi}}$			

Table D.6: Synchronous machine & exciter data.

Appendix E

Description of data file formats

E.1 Description of .cdf file format

The IEEE common data format (cdf) is described in [21] and contains data needed to perform a load flow calculation. It contains two sections: a bus data section and a branch data section. The bus data section and branch data section formats are described in Tables E.1 and E.2.

Columns 1-4	:	Bus number
Columns 26	:	Bus type
	:	0 = Unregulated bus (load bus)
	:	1 = Hold MVar generation within voltage limits
	:	2 = Hold bus voltage within generator MVar limits
	:	3 = Hold bus voltage and angle (swing bus)
Columns 28-33	:	Bus voltage magnitude from solved loadflow to be used as initial guess
Columns 34-40	:	Bus voltage phase angle from solved loadflow to be used as initial guess
Columns 41-49	:	Load MW (real load result from solved loadflow case)
Columns 50-58	:	Load MVar (reactive load result from solved loadflow case)
Columns 59-67	:	Generated MW (real power generation from solved loadflow case to be used as initial guess)
Columns 68-75	:	Generated MVar (reactive power generation from solved loadflow case to be used as initial guess)
Columns 91-98	:	Maximum voltage for bus type 0
Columns 99-106	:	Minimum voltage for bus type 0
Columns 107-114	:	Bus capacitor
Columns 115-122	:	Bus reactor

Table E.1: CDF file bus data section format.

APPENDIX E. DESCRIPTION OF DATA FILE FORMATS

Columns 1-4	:	From bus
Columns 6-9	:	From bus
Columns 19	:	Line type
	:	1 = Fixed voltage ratio and/or fixed phase shifter
	:	2 = Fixed phase angle and variable voltage ratio with voltage control
	:	3 = Fixed phase angle and variable voltage ratio with MVar control
	:	4 = Fixed voltage ratio and variable phase shifter with MW control
Columns 20-29	:	Line resistance
Columns 30-39	:	Line reactance
Columns 41-49	:	Line charging capacitance
Columns 57-61	:	Normal line MVA rating
Columns 62-67	:	Emergency line MVA rating
Columns 77-82	:	Transformer tap ratio
Columns 84-90	:	Phase angle
Columns 91-97	:	Minimum voltage tap ratio or phase shifter angle
Columns 98-104	:	Maximum voltage tap ratio or phase shifter angle
Columns 134-139	:	Probability of line outage in percent (added to cdf file for ESCOPF)

Table E.2: CDF file branch data section format.

E.2 Description of .gdf file format

The generator data file has a *.gdf extension and was created to provide a location for generator cost curve data and generator operating voltage and power output operating limits. Its format is described in Table E.3.

E.3 Description of .ldf file format

The load data file has a *.ldf extension and was created to provide a location for consumer benefit curve data, interruption cost coefficients, and load power factor data. Its format is described in Table E.4.

E.4 Description of .sdf file format

The synchronous machine dynamic data file has a *.sdf extension. Each line of the file contains a record for a synchronous machine in the system which are assumed to have an IEEE Type I exciter. The first two lines of the file are used for identifying column numbers and the data associated with

APPENDIX E. DESCRIPTION OF DATA FILE FORMATS

Columns 1-4	: Generator bus number
Columns 6-9	: Generator name
Columns 47	: Adjustable real power generation?
	: 0 = Real power is not adjustable
	: 1 = Real power is adjustable
Columns 49	: Adjustable reactive power generation?
	: 0 = Reactive power is not adjustable
	: 1 = Reactive power is adjustable
Columns 51	: Availability
	: 0 = not available
	: 1 = available
Columns 53-59	: Generator α cost curve coefficient
Columns 61-66	: Generator β cost curve coefficient
Columns 68-75	: Generator γ cost curve coefficient
Columns 77-81	: Maximum voltage magnitude at generator bus
Columns 83-87	: Minimum voltage magnitude at generator bus
Columns 89-93	: Maximum MW upward ramping
Columns 95-99	: Minimum MW downward ramping
Columns 101-106	: Maximum MVar upward ramping
Columns 108-113	: Minimum MVar downward ramping
Columns 115-120	: Generator β_R ramping cost curve coefficient
Columns 122-129	: Generator γ_R ramping cost curve coefficient

Table E.3: GDF data file format.

Columns 1-4	: Load bus number
Columns 7-13	: Load bus name
Columns 15-18	: Load type
	: 1 = Residential
	: 2 = Industrial
	: 3 = Commercial
Columns 20-25	: Maximum MW load
Columns 27-32	: Minimum MW load
Columns 34-40	: Maximum MVar load
Columns 42-48	: Minimum MVar load
Columns 50	: Interruptible load
	: 0 = not interruptible
	: 1 = interruptible
Columns 52-59	: Maximum MW interruptible
Columns 61-68	: Customer benefit curve α_L
Columns 70-77	: Customer benefit curve β_L
Columns 79-87	: Customer benefit curve γ_L
Columns 89-96	: Interruption cost curve α_I
Columns 98-104	: Interruption cost curve β_I
Columns 106-113	: Interruption cost curve γ_I
Columns 115-120	: Load power factor

Table E.4: LDF data file format.

APPENDIX E. DESCRIPTION OF DATA FILE FORMATS

Columns 1-5	:	Bus number
Columns 7-15	:	Generator name
Columns 17-23	:	<i>d</i> -axis reactance
Columns 25-31	:	<i>d</i> -axis transient reactance
Columns 33-40	:	<i>d</i> -axis subtransient reactance
Columns 42-48	:	<i>q</i> -axis reactance
Columns 50-56	:	<i>q</i> -axis transient reactance
Columns 58-65	:	<i>q</i> -axis subtransient reactance
Columns 67-73	:	<i>q</i> -axis subtransient reactance
Columns 75-82	:	<i>d</i> -axis transient time constant
Columns 84-92	:	<i>d</i> -axis subtransient time constant
Columns 94-101	:	<i>q</i> -axis transient time constant
Columns 103-111	:	<i>q</i> -axis subtransient time constant
Columns 113-122	:	armature resistance
Columns 124-132	:	inertia constant
Columns 134-141	:	PI controller gain
Columns 143-150	:	PI controller gain
Columns 152-159	:	constant associated with DC generator
Columns 161-168	:	constant associated with DC generator
Columns 170-177	:	time constant associated with DC generator
Columns 179-186	:	gain constant associated with stabilizing transformer
Columns 188-195	:	time constant associated with stabilizing transformer
Columns 197-203	:	damping coefficient

Table E.5: SDF data file format.

those columns. Notation from [10] is used in the column description. Table E.5 describes the data format.

Appendix F

Small-signal stability OPF results ignoring security

The small-signal stability OPF problem was solved using a PDIP algorithm for four cases with each case having a different stability margin. The following tables list the operating point corresponding to the solution of each of the four cases. The most critical eigenvalue pair is in boldface in each table of eigenvalues.

APPENDIX F. SMALL-SIGNAL STABILITY OPF RESULTS IGNORING SECURITY

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MVAR-hr)
1	1.050∠-3.4497°	69.64	17.53	-	-	12.83	0.00
2	1.045∠ 5.3076°	164.70	7.09	-	-	12.29	0.00
3	1.045∠ 0.6869°	82.98	-10.85	-	-	12.40	0.00
4	1.041∠-5.5528°	-	-	-	-	12.83	0.01
5	1.015∠-7.2857°	-	-	122.68	49.06	12.94	0.09
6	1.033∠-6.9198°	-	-	84.33	28.10	12.94	0.03
7	1.045∠-0.0993°	-	-	-	-	12.29	0.04
8	1.035∠-3.1388°	-	-	106.12	37.13	12.46	0.06
9	1.052∠-1.8480°	-	-	-	-	12.40	0.02

(a) Bus voltages, power injections, and power prices.

Bus from (<i>i</i>)	4	4	5	6	7	9	1	2	3
Bus to (<i>j</i>)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	29.00	44.97	83.14	57.88	80.61	26.38	71.82	164.86	83.69

(b) Line flows.

-0.4315 ± j 0.4961
-0.4415 ± j 0.7372
-0.4491 ± j 1.1940
-3.2258
-3.3839
-5.0877
-0.1873 ± j 7.2083
-5.2217 ± j 7.8175
-5.3173 ± j 7.9199
-5.4907 ± j 7.9453
-0.7759 ± j 12.8953

(c) Eigenvalues.

Table F.1: SSSOPF results with $\text{Re}(\lambda_i) \leq 0$

APPENDIX F. SMALL-SIGNAL STABILITY OPF RESULTS IGNORING SECURITY

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MW-hr)
1	1.050∠-3.3211°	71.30	17.37	-	-	12.83	0.00
2	1.045∠5.0144°	165.58	6.89	-	-	12.29	0.00
3	1.045∠0.4341°	82.46	-10.90	-	-	12.40	0.00
4	1.041∠-5.6849°	-	-	-	-	12.82	0.01
5	1.015∠-7.4590°	-	-	122.70	49.07	12.94	0.09
6	1.033∠-7.0900°	-	-	84.38	28.12	12.93	0.03
7	1.046∠-0.3552°	-	-	-	-	12.29	0.04
8	1.035∠-3.3864°	-	-	106.12	37.13	12.46	0.06
9	1.052∠-2.0848°	-	-	-	-	12.40	0.02

(a) Bus voltages, power injections, and power prices.

Bus from (<i>i</i>)	4	4	5	6	7	9	1	2	3
Bus to (<i>j</i>)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	29.77	45.72	82.32	57.26	80.40	26.57	73.39	163.72	83.17

(b) Line flows.

-0.4317 ± j 0.4960
-0.4419 ± j 0.7368
-0.4501 ± j 1.1926
-3.2258
-3.3725
-5.0771
-0.2001 ± j 8.4992
-5.2201 ± j 7.8161
-5.3137 ± j 7.9201
-5.4838 ± j 7.9469
-0.7777 ± j 12.9430

(c) Eigenvalues.

Table F.2: SSSOPF results with $\text{Re}(\lambda_i) \leq -0.2$

APPENDIX F. SMALL-SIGNAL STABILITY OPF RESULTS IGNORING SECURITY

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MW-hr)
1	1.030 \angle -6.2644 $^\circ$	120.91	6.83	-	-	12.79	0.01
2	1.045 \angle -4.4087 $^\circ$	127.65	7.385	-	-	12.29	-0.04
3	1.045 \angle -7.5125 $^\circ$	69.05	-6.00	-	-	12.40	-0.01
4	1.028 \angle -10.035 $^\circ$	-	-	-	-	12.72	0.03
5	1.007 \angle -13.195 $^\circ$	-	-	123.08	49.22	12.85	0.09
6	1.023 \angle -12.640 $^\circ$	-	-	85.74	28.58	12.87	0.04
7	1.043 \angle -8.6060 $^\circ$	-	-	-	-	12.34	-0.00
8	1.032 \angle -11.347 $^\circ$	-	-	105.89	37.05	12.49	0.02
9	1.049 \angle -9.6279 $^\circ$	-	-	-	-	12.42	-0.01

(a) Bus voltages, power injections, and power prices.

Bus from (i)	4	4	5	6	7	9	1	2	3
Bus to (j)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	52.66	70.16	59.69	43.00	72.57	33.97	121.10	127.89	69.31

(b) Line flows.

-0.4581 \pm j 0.5168
-0.4534 \pm j 0.7389
-0.4668 \pm j 1.1640
-3.2026
-3.2258
-4.6266
-0.3000 \pm j 8.3586
-5.2052 \pm j 7.7976
-5.3852 \pm j 7.9039
-5.3135 \pm j 7.9364
-0.9785 \pm j 12.7279

(c) Eigenvalues.

Table F.3: SSSOPF results with $\text{Re}(\lambda_i) \leq -0.3$

APPENDIX F. SMALL-SIGNAL STABILITY OPF RESULTS IGNORING SECURITY

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MW-hr)
1	$0.958\angle-9.6717^\circ$	160.97	-3.42	-	-	12.74	0.03
2	$1.040\angle-13.978^\circ$	105.20	39.12	-	-	12.27	-0.02
3	$1.000\angle-16.520^\circ$	52.48	-4.67	-	-	12.39	-0.01
4	$0.965\angle-15.423^\circ$	-	-	-	-	12.61	0.09
5	$0.955\angle-20.176^\circ$	-	-	123.26	49.30	12.79	0.16
6	$0.964\angle-19.517^\circ$	-	-	86.25	28.74	12.83	0.09
7	$1.018\angle-17.541^\circ$	-	-	-	-	12.34	-0.02
8	$0.998\angle-20.311^\circ$	-	-	105.75	37.00	12.51	0.01
9	$1.004\angle-18.275^\circ$	-	-	-	-	12.44	-0.01

(a) Bus voltages, power injections, and power prices.

Bus from (i)	4	4	5	6	7	9	1	2	3
Bus to (j)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	72.40	90.49	55.39	39.03	72.15	36.50	161.00	112.23	52.69

(b) Line flows.

-0.4994 ± j 0.5533
-0.4841 ± j 0.7734
-0.4859 ± j 1.1812
-3.0289
-3.2258
-4.1265
-0.4000 ± j 7.8145
-5.1814 ± j 7.7899
-5.2574 ± j 7.9017
-5.3841 ± j 7.9154
-1.1863 ± j 12.0563

(c) Eigenvalues.

Table F.4: SSSOPF results with $\text{Re}(\lambda_i) \leq -0.4$

Appendix G

Small-signal stability ESCOPF

Results

The small-signal stability ESCOPF problem was solved using a PDIP algorithm for four cases with each case having a different stability margin. The steady-state operating point is given for the pre-contingency state and for each post-contingency state.

APPENDIX G. SMALL-SIGNAL STABILITY ESCOPF RESULTS

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MVAR-hr)
1	1.050∠-3.2987°	66.47	15.48	-	-	12.05	0.00
2	1.045∠ 5.3814°	161.27	5.24	-	-	11.55	0.00
3	1.045∠ 0.9320°	82.81	-11.77	-	-	11.65	0.00
4	1.042∠-5.3038°	-	-	-	-	12.05	0.01
5	1.017∠-6.9086°	-	-	117.92	47.16	12.15	0.08
6	1.034∠-6.6523°	-	-	83.86	27.95	12.15	0.03
7	1.046∠-0.0923°	-	-	-	-	11.56	0.03
8	1.036∠-2.8884°	-	-	104.73	36.64	11.71	0.05
9	1.053∠-1.5964°	-	-	-	-	11.65	0.02

(a) Bus voltages, power injections, and power prices.

Bus from (<i>i</i>)	4	4	5	6	7	9	1	2	3
Bus to (<i>j</i>)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	28.69	41.61	81.18	57.67	79.20	26.29	68.25	161.35	83.64

(b) Line flows.

-0.4266 ± j 0.4948
-0.4419 ± j 0.7372
-0.4596 ± j 1.1971
-3.2258
-3.4160
-5.2837
-5.2209 ± j 7.8139
-5.3113 ± j 7.9206
-0.1784 ± j 9.5491
-5.4657 ± j 7.9457
-0.6696 ± j 14.6558

(c) Eigenvalues.

Table G.1: SSSESCOPF results with $\text{Re}(\lambda_i) \leq 0$ (pre-contingency)

APPENDIX G. SMALL-SIGNAL STABILITY ESCOPF RESULTS

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MVAR-hr)
1	1.050∠-2.1453°	43.28	17.64	-	-	12.78	0.00
2	1.045∠ 6.6822°	164.99	10.51	-	-	12.29	0.00
3	1.045∠ 1.9688°	103.79	-11.77	-	-	12.42	0.00
4	1.041∠-3.4527°	-	-	-	-	12.78	0.01
5	1.016∠-5.3062°	-	-	117.92	47.16	12.89	0.08
6	0.980∠-9.0281°	-	-	83.86	27.95	13.35	0.32
7	1.043∠ 1.2555°	-	-	-	-	12.29	0.04
8	1.029∠-2.0840°	-	-	104.73	36.64	12.48	0.07
9	1.042∠-1.2348°	-	-	-	-	12.43	0.05

(a) Bus voltages, power injections, and power prices.

Bus from (i)	4	4	5	6	7	9	1	2	3
Bus to (j)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	0.00	46.32	76.19	88.40	88.49	17.04	46.74	165.32	104.20

(b) Line flows.

-0.4543 ± j 0.5606
-0.4532 ± j 0.8725
-0.4290 ± j 1.3227
-2.9261
-3.2258
-5.2078
-0.1302 ± j 8.2630
-5.2244 ± j 7.8344
-5.4271 ± j 7.8391
-5.5149 ± j 7.9731
-0.5882 ± j 13.9116

(c) Eigenvalues.

Table G.2: SSSESCOPF results with $\text{Re}(\lambda_i) \leq 0$ (line 4-6 out)

APPENDIX G. SMALL-SIGNAL STABILITY ESCOPF RESULTS

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MVAR-hr)
1	1.050∠-1.5366°	30.52	-00.17	-	-	12.76	0.00
2	1.045∠ 5.6271°	148.32	27.94	-	-	12.26	0.00
3	1.045∠ 4.2616°	104.24	-7.61	-	-	12.42	0.00
4	1.050∠-2.4498°	-	-	-	-	12.78	1.40
5	0.955∠-7.3181°	-	-	89.72	35.88	63.36	141.31
6	1.039∠-3.8528°	-	-	83.86	27.95	13.45	3.68
7	1.032∠ 0.6967°	-	-	-	-	15.07	32.54
8	1.027∠-1.4585°	-	-	104.73	36.64	16.13	22.77
9	1.051∠ 1.0731°	-	-	-	-	12.84	7.51

(a) Bus voltages, power injections, and power prices.

Bus from (<i>i</i>)	4	4	5	6	7	9	1	2	3
Bus to (<i>j</i>)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	30.52	0.00	96.63	54.87	55.90	50.46	30.52	150.92	104.20

(b) Line flows.

-0.4343 ± j 0.5167
-0.4571 ± j 0.8948
-0.4347 ± j 1.3106
-2.9261
-3.2258
-5.2655
-0.1448 ± j 7.5089
-5.2185 ± j 7.8338
-5.4139 ± j 7.8424
-5.4564 ± j 7.9557
-0.5730 ± j 14.8184

(c) Eigenvalues.

Table G.3: SSSESCOPF results with $\text{Re}(\lambda_i) \leq 0$ (line 4-5 out)

APPENDIX G. SMALL-SIGNAL STABILITY ESCOPF RESULTS

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MVAR-hr)
1	1.050∠-5.5882°	116.47	52.61	-	-	13.26	0.00
2	1.045∠ 9.4083°	126.27	9.38	-	-	12.06	0.00
3	1.045∠ 1.3552°	69.65	- 2.62	-	-	12.38	0.00
4	1.023∠-9.1685°	-	-	-	-	13.26	0.03
5	0.972∠-14.7269°	-	-	117.92	47.16	13.61	0.19
6	1.016∠-8.9291°	-	-	83.86	27.95	13.25	0.07
7	1.042∠ 5.2526°	-	-	-	-	12.97	0.03
8	1.030∠ 0.4307°	-	-	104.73	36.64	12.32	0.07
9	1.047∠-0.7821°	-	-	-	-	12.38	0.03

(a) Bus voltages, power injections, and power prices.

Bus from (<i>i</i>)	4	4	5	6	7	9	1	2	3
Bus to (<i>j</i>)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	3.14	127.22	0.00	87.65	126.27	22.23	127.80	126.61	69.70

(b) Line flows.

-0.4552 ± j 0.5268
-0.4608 ± j 0.8787
-0.4703 ± j 1.1917
-3.0093
-3.2258
-4.7301
-0.1266 ± j 7.4864
-5.2099 ± j 7.8050
-5.2884 ± j 7.8720
-5.3449 ± j 7.9366
-0.8628 ± j 13.9993

(c) Eigenvalues.

Table G.4: SSSESCOPF results with $\text{Re}(\lambda_i) \leq 0$ (line 5-7 out)

APPENDIX G. SMALL-SIGNAL STABILITY ESCOPF RESULTS

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MVAR-hr)
1	1.050 \angle -5.5783 $^\circ$	115.23	42.00	-	-	12.90	0.00
2	1.045 \angle 2.2675 $^\circ$	147.88	10.94	-	-	12.26	0.00
3	1.045 \angle -0.7886 $^\circ$	47.81	00.47	-	-	12.24	0.00
4	1.029 \angle -9.1005 $^\circ$	-	-	-	-	12.90	0.03
5	1.006 \angle -10.3795 $^\circ$	-	-	117.92	47.16	12.98	0.11
6	0.993 \angle -13.2377 $^\circ$	-	-	83.86	27.95	12.29	0.16
7	1.042 \angle -2.6006 $^\circ$	-	-	-	-	12.26	0.03
8	1.030 \angle -4.7562 $^\circ$	-	-	104.73	36.64	12.38	0.05
9	1.045 \angle -2.2586 $^\circ$	-	-	-	-	12.24	0.01

(a) Bus voltages, power injections, and power prices.

Bus from (i)	4	4	5	6	7	9	1	2	3
Bus to (j)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	87.18	33.80	89.21	0.00	57.55	47.82	122.64	148.29	47.81

(b) Line flows.

-0.4796 \pm j 0.5738
-0.4751 \pm j 0.8542
-0.4819 \pm j 1.1781
-2.7172
-3.2258
-4.6422
-0.1888 \pm j 8.0426
-5.2057 \pm j 7.7999
-5.2384 \pm j 7.8693
-5.4305 \pm j 7.9395
-0.9469 \pm j 13.0237

(c) Eigenvalues.

Table G.5: SSSESCOPF results with $\text{Re}(\lambda_i) \leq 0$ (line 6-9 out)

APPENDIX G. SMALL-SIGNAL STABILITY ESCOPF RESULTS

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MVAR-hr)
1	1.050∠-3.4293°	69.57	23.21	-	-	12.83	0.00
2	1.045∠ 9.8879°	126.27	3.34	-	-	11.82	0.00
3	1.045∠-4.4899°	117.81	19.10	-	-	13.02	0.00
4	1.038∠-5.5365°	-	-	-	-	12.82	0.01
5	1.009∠-5.1734°	-	-	117.92	47.16	12.82	0.08
6	1.026∠-9.1376°	-	-	83.86	27.95	13.14	0.03
7	1.046∠ 5.7467°	-	-	-	-	11.82	0.05
8	0.992∠-13.8631°	-	-	104.73	36.64	13.37	0.14
9	1.036∠-8.1445°	-	-	-	-	13.03	0.02

(a) Bus voltages, power injections, and power prices.

Bus from (i)	4	4	5	6	7	9	1	2	3
Bus to (j)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	73.38	27.16	121.60	25.06	0.00	109.54	73.34	126.31	119.35

(b) Line flows.

-0.4525 ± j 0.7442
-0.4817 ± j 0.8006
-0.4677 ± j 1.2267
-3.2258
-3.4099
-4.1484
-0.2053 ± j 8.9385
-5.2239 ± j 7.8185
-5.3267 ± j 7.8989
-5.5989 ± j 7.9154
-0.3419 ± j 11.7840

(c) Eigenvalues.

Table G.6: SSSESCOPF results with $\text{Re}(\lambda_i) \leq 0$ (line 7-8 out)

APPENDIX G. SMALL-SIGNAL STABILITY ESCOPF RESULTS

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MVAR-hr)
1	1.050∠-3.3885°	68.57	20.39	-	-	12.82	0.00
2	1.045∠ 5.4074°	181.47	35.22	-	-	12.32	0.00
3	1.045∠ 0.6411°	60.89	-15.24	-	-	12.37	0.00
4	1.040∠-5.4625°	-	-	-	-	12.82	0.01
5	1.010∠-7.3408°	-	-	117.92	47.16	12.94	0.08
6	1.032∠-6.6225°	-	-	83.86	27.95	12.93	0.03
7	1.030∠-0.6431°	-	-	-	-	12.32	0.05
8	0.997∠-4.7163°	-	-	104.73	36.64	12.56	0.14
9	1.054∠-1.2152°	-	-	-	-	12.37	0.02

(a) Bus voltages, power injections, and power prices.

Bus from (i)	4	4	5	6	7	9	1	2	3
Bus to (j)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	24.77	49.43	74.52	61.68	109.89	0.00	71.54	184.85	62.77

(b) Line flows.

-0.4578 ± j 0.6729
-0.4760 ± j 0.8090
-0.4736 ± j 1.1984
-3.2258
-3.2786
-4.0694
-0.1703 ± j 9.0671
-5.2009 ± j 7.8180
-5.2480 ± j 7.8977
-5.6925 ± j 7.9204
-0.4971 ± j 11.5589

(c) Eigenvalues.

Table G.7: SSSESCOPF results with $\text{Re}(\lambda_i) \leq 0$ (line 9-8 out)

APPENDIX G. SMALL-SIGNAL STABILITY ESCOPF RESULTS

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MVAR-hr)
1	1.050∠-4.0771°	82.08	13.79	-	-	11.96	0.00
2	1.045∠ 2.3469°	146.93	2.98	-	-	11.57	0.00
3	1.045∠-1.2764°	80.84	-12.34	-	-	11.65	0.00
4	1.043∠-6.5505°	-	-	-	-	11.97	0.01
5	1.019∠-8.5893°	-	-	117.77	47.16	12.08	0.08
6	1.035∠-8.2094°	-	-	83.84	27.94	12.09	0.03
7	1.047∠-2.4680°	-	-	-	-	11.57	0.03
8	1.036∠-5.2756°	-	-	104.81	36.67	11.72	0.05
9	1.053∠-3.7441°	-	-	-	-	11.65	0.01

(a) Bus voltages, power injections, and power prices.

Bus from (i)	4	4	5	6	7	9	1	2	3
Bus to (j)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	35.01	49.77	72.29	52.16	74.79	30.68	83.23	146.96	81.78

(b) Line flows.

-0.4259 ± j 0.4936
-0.4415 ± j 0.7333
-0.4619 ± j 1.1848
-3.2258
-3.4063
-5.2176
-5.2180 ± j 7.8048
-5.3046 ± j 7.9158
-0.1996 ± j 9.4823
-5.4056 ± j 7.9432
-0.6903 ± j 14.5911

(c) Eigenvalues.

Table G.8: SSSESCOPF results with $\text{Re}(\lambda_i) \leq -0.15$ pre-contingency

APPENDIX G. SMALL-SIGNAL STABILITY ESCOPF RESULTS

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MVAR-hr)
1	1.050∠-2.8887°	58.22	15.95	-	-	12.81	0.00
2	1.045∠ 3.2394°	159.66	9.73	-	-	12.28	0.00
3	1.045∠-2.5515°	93.71	9.23	-	-	12.42	0.00
4	1.042∠-4.6455°	-	-	-	-	12.78	0.01
5	1.018∠-7.1927°	-	-	117.77	47.10	12.87	0.08
6	0.980∠-13.2403°	-	-	83.84	27.94	13.36	0.32
7	1.044∠-2.0107°	-	-	-	-	12.31	0.04
8	1.029∠-5.7453°	-	-	104.81	36.67	12.49	0.07
9	1.041∠-5.4445°	-	-	-	-	12.44	0.05

(a) Bus voltages, power injections, and power prices.

Bus from (i)	4	4	5	6	7	9	1	2	3
Bus to (j)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	0.00	59.89	62.41	88.37	98.74	6.98	60.36	159.95	94.16

(b) Line flows.

-0.4551 ± j 0.5618
-0.4543 ± j 0.8716
-0.4368 ± j 1.3102
-2.9063
-3.2258
-5.1183
-0.1508 ± j 8.1927
-5.2262 ± j 7.8317
-5.3742 ± j 7.8333
-5.4810 ± j 7.9639
-0.6454 ± j 13.7376

(c) Eigenvalues.

Table G.9: SSESOCOPF results with $\text{Re}(\lambda_i) \leq -0.15$ (line 4-6 out)

APPENDIX G. SMALL-SIGNAL STABILITY ESCOPF RESULTS

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MVAR-hr)
1	1.050∠-1.9636°	38.96	1.45	-	-	12.78	0.00
2	1.045∠ 3.4738°	146.28	27.80	-	-	12.27	0.00
3	1.045∠ 2.1677°	97.65	- 7.85	-	-	12.42	0.00
4	1.051∠-3.1287°	-	-	-	-	12.75	1.40
5	0.955∠-9.4023°	-	-	89.70	35.87	63.37	141.31
6	1.006∠-4.9508°	-	-	83.84	27.94	13.38	3.68
7	1.042∠-1.3889°	-	-	-	-	15.08	32.54
8	1.030∠-3.4648°	-	-	104.81	36.67	16.12	22.77
9	1.046∠-0.8193°	-	-	-	-	12.83	7.51

(a) Bus voltages, power injections, and power prices.

Bus from (i)	4	4	5	6	7	9	1	2	3
Bus to (j)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	39.02	0.00	96.61	47.31	53.88	52.50	38.98	148.90	97.96

(b) Line flows.

-0.4346 ± j 0.5153
-0.4572 ± j 0.8947
-0.4384 ± j 1.3036
-2.9630
-2.9630
-5.1989
-0.1525 ± j 7.4783
-5.2184 ± j 7.8319
-5.3929 ± j 7.8503
-5.4351 ± j 7.9417
-0.6073 ± j 14.7192

(c) Eigenvalues.

Table G.10: SSSESCOPF results with $\text{Re}(\lambda_i) \leq -0.15$ (line 4-5 out)

APPENDIX G. SMALL-SIGNAL STABILITY ESCOPF RESULTS

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MVAR-hr)
1	1.030∠-6.6013°	132.08	46.07	-	-	24.04	0.03
2	1.045∠ 3.4578°	111.93	8.23	-	-	8.08	0.02
3	1.045∠-2.8639°	67.10	0.22	-	-	12.45	0.04
4	1.007∠-10.8062°	-	-	-	-	22.27	0.08
5	0.955∠-16.5460°	-	-	117.77	47.10	12.98	0.80
6	1.006∠-11.4731°	-	-	83.84	27.94	19.44	0.46
7	1.042∠-0.2249°	-	-	-	-	9.09	0.44
8	1.030∠-4.4902°	-	-	104.81	36.67	10.85	0.59
9	1.046∠-4.9263°	-	-	-	-	12.91	0.51

(a) Bus voltages, power injections, and power prices.

Bus from (<i>i</i>)	4	4	5	6	7	9	1	2	3
Bus to (<i>j</i>)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	15.53	127.47	0.00	74.30	111.93	8.43	139.89	112.23	67.10

(b) Line flows.

-0.4650 ± j 0.5351
-0.4650 ± j 0.8786
-0.4766 ± j 1.1788
-2.9885
-3.2258
-4.5771
-0.1504 ± j 7.3542
-5.2030 ± j 7.7964
-5.2683 ± j 7.8611
-5.3269 ± j 7.9400
-0.9207 ± j 13.8258

(c) Eigenvalues.

Table G.11: SSSESCOPF results with $\text{Re}(\lambda_i) \leq -0.15$ (line 5-7 out)

APPENDIX G. SMALL-SIGNAL STABILITY ESCOPF RESULTS

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MVAR-hr)
1	1.050∠-5.5783°	115.34	41.90	-	-	12.90	0.00
2	1.045∠ 2.2811°	149.62	11.07	-	-	12.26	0.00
3	1.045∠-1.0784°	45.84	00.50	-	-	12.26	0.00
4	1.029∠-9.1097°	-	-	-	-	12.90	0.03
5	1.006∠-10.3955°	-	-	117.77	47.10	12.98	0.11
6	0.993∠-13.2452°	-	-	83.84	27.94	12.29	0.16
7	1.042∠-2.6443°	-	-	-	-	12.26	0.03
8	1.030∠-4.8791°	-	-	104.81	36.67	12.38	0.05
9	1.045∠-2.4879°	-	-	-	-	12.26	0.01

(a) Bus voltages, power injections, and power prices.

Bus from (<i>i</i>)	4	4	5	6	7	9	1	2	3
Bus to (<i>j</i>)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	87.15	33.88	88.92	0.00	59.59	45.84	122.71	150.03	45.84

(b) Line flows.

-0.4820 ± j 0.5768
-0.4769 ± j 0.8555
-0.4830 ± j 1.1790
-2.6767
-3.2258
-4.6303
-0.1941 ± j 8.0361
-5.2044 ± j 7.7993
-5.2365 ± j 7.8684
-5.4371 ± j 7.9400
-0.9640 ± j 12.9902

(c) Eigenvalues.

Table G.12: SSSESCOPF results with $\text{Re}(\lambda_i) \leq -0.15$ (line 6-9 out)

APPENDIX G. SMALL-SIGNAL STABILITY ESCOPF RESULTS

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MVAR-hr)
1	1.050∠-4.1859°	84.77	20.50	-	-	12.85	0.00
2	1.045∠ 6.2642°	111.93	0.05	-	-	11.99	0.00
3	1.045∠-6.0358°	115.84	18.62	-	-	13.08	0.00
4	1.040∠-6.7492°	-	-	-	-	12.85	0.01
5	1.023∠-7.0190°	-	-	117.77	47.10	12.88	0.08
6	1.028∠-10.4379°	-	-	83.84	27.94	13.17	0.03
7	1.047∠ 2.5990°	-	-	-	-	12.00	0.05
8	0.992∠-15.3496°	-	-	104.81	36.67	13.43	0.14
9	1.037∠-9.6287°	-	-	-	-	13.08	0.02

(a) Bus voltages, power injections, and power prices.

Bus from (i)	4	4	5	6	7	9	1	2	3
Bus to (j)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	75.40	24.69	108.45	24.07	0.00	109.63	87.21	119.30	117.33

(b) Line flows.

-0.4530 ± j 0.7418
-0.4840 ± j 0.7960
-0.4703 ± j 1.2147
-3.2258
-3.3734
-4.0913
-0.2426 ± j 8.8572
-5.2185 ± j 7.8124
-5.2920 ± j 7.8937
-5.5782 ± j 7.9130
-0.3514 ± j 11.7665

(c) Eigenvalues.

Table G.13: SSSESCOPF results with $\text{Re}(\lambda_i) \leq -0.15$ (line 7-8 out)

APPENDIX G. SMALL-SIGNAL STABILITY ESCOPF RESULTS

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MVAR-hr)
1	1.050∠-3.3833°	68.47	20.33	-	-	12.82	0.00
2	1.045∠ 5.4168°	181.49	35.22	-	-	12.32	0.00
3	1.045∠ 0.6479°	60.87	-15.25	-	-	12.37	0.00
4	1.040∠-5.4539°	-	-	-	-	12.82	0.01
5	1.010∠-7.3274°	-	-	117.77	47.10	12.94	0.08
6	1.032∠-6.6136°	-	-	83.84	27.94	12.93	0.03
7	1.030∠-0.6346°	-	-	-	-	12.32	0.05
8	0.997∠-4.7110°	-	-	104.81	36.67	12.56	0.14
9	1.054∠-1.2078°	-	-	-	-	12.37	0.02

(a) Bus voltages, power injections, and power prices.

Bus from (i)	4	4	5	6	7	9	1	2	3
Bus to (j)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	24.76	49.32	74.47	61.66	109.98	0.00	71.42	184.88	62.75

(b) Line flows.

-0.4578 ± j 0.6728
-0.4760 ± j 0.8090
-0.4736 ± j 1.1983
-3.2258
-3.2783
-4.0693
-0.1703 ± j 9.0672
-5.2009 ± j 7.8180
-5.2480 ± j 7.8976
-5.6927 ± j 7.9204
-0.4972 ± j 11.5588

(c) Eigenvalues.

Table G.14: SSSESCOPF results with $\text{Re}(\lambda_i) \leq -0.15$ (line 9-8 out)

APPENDIX G. SMALL-SIGNAL STABILITY ESCOPF RESULTS

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MVAR-hr)
1	1.050∠-5.7940°	116.68	12.12	-	-	11.80	0.00
2	1.045∠-4.3964°	114.18	-0.37	-	-	11.62	0.00
3	1.045∠-6.0434°	77.92	-12.72	-	-	11.65	0.00
4	1.045∠-9.3045°	-	-	-	-	11.80	0.02
5	1.023∠-12.3128°	-	-	117.37	46.94	11.96	0.07
6	1.037∠-11.5319°	-	-	83.88	27.95	12.97	0.03
7	1.047∠-8.1345°	-	-	-	-	11.62	0.02
8	1.037∠-10.5297°	-	-	104.95	36.72	11.74	0.04
9	1.053∠-8.4216°	-	-	-	-	11.65	0.01

(a) Bus voltages, power injections, and power prices.

Bus from (i)	4	4	5	6	7	9	1	2	3
Bus to (j)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	48.65	69.57	53.28	41.17	64.13	41.37	117.31	114.18	78.95

(b) Line flows.

-0.4289 ± j 0.4944
-0.4459 ± j 0.7279
-0.4686 ± j 1.1652
-3.2258
-3.3430
-5.0190
-5.2054 ± j 7.7944
-5.2848 ± j 7.8999
-0.2809 ± j 9.2405
-5.3287 ± j 7.9482
-0.7462 ± j 14.4073

(c) Eigenvalues.

Table G.15: SSSESCOPF results with $\text{Re}(\lambda_i) \leq -0.25$ pre-contingency

APPENDIX G. SMALL-SIGNAL STABILITY ESCOPF RESULTS

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MVAR-hr)
1	1.039∠-5.2580°	103.74	13.41	-	-	12.88	0.00
2	1.045∠-7.3552°	145.36	14.91	-	-	12.25	0.02
3	1.045∠-16.6447°	62.65	12.97	-	-	12.44	0.02
4	1.033∠-8.4495°	-	-	-	-	12.76	0.09
5	1.013∠-13.1920°	-	-	117.37	46.94	12.83	0.07
6	0.976∠-26.4288°	-	-	83.88	27.95	13.44	0.29
7	1.040∠-12.1516°	-	-	-	-	12.34	0.04
8	1.024∠-17.1581°	-	-	104.95	36.72	12.54	0.07
9	1.038∠-18.5836°	-	-	-	-	12.52	0.03

(a) Bus voltages, power injections, and power prices.

Bus from (i)	4	4	5	6	7	9	1	2	3
Bus to (j)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	0.00	104.01	33.18	88.41	130.64	25.00	104.60	146.12	63.97

(b) Line flows.

-0.4784 ± j 0.5840
-0.4648 ± j 0.8780
-0.4630 ± j 1.2960
-2.7410
-3.2258
-4.7130
-0.2504 ± j 7.7804
-5.2370 ± j 7.7903
-5.2519 ± j 7.8465
-5.4306 ± j 7.9447
-0.8811 ± j 13.0110

(c) Eigenvalues.

Table G.16: SSSESCOPF results with $\text{Re}(\lambda_i) \leq -0.25$ (line 4-6 out)

APPENDIX G. SMALL-SIGNAL STABILITY ESCOPF RESULTS

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MVAR-hr)
1	1.050∠-1.9636°	38.96	1.45	-	-	12.78	0.00
2	1.045∠ 3.4738°	146.28	27.80	-	-	12.27	0.00
3	1.045∠ 2.1677°	97.65	- 7.85	-	-	12.42	0.00
4	1.053∠-3.1287°	-	-	-	-	12.75	1.40
5	0.955∠-9.4023°	-	-	89.70	35.87	63.37	141.31
6	1.038∠-4.9508°	-	-	83.84	27.94	13.38	3.68
7	1.031∠-1.3889°	-	-	-	-	15.08	32.54
8	1.024∠-3.4648°	-	-	104.81	36.67	16.12	22.77
9	1.046∠-0.8193°	-	-	-	-	12.83	7.51

(a) Bus voltages, power injections, and power prices.

Bus from (i)	4	4	5	6	7	9	1	2	3
Bus to (j)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	39.02	0.00	96.61	47.31	53.88	52.50	38.98	148.90	97.96

(b) Line flows.

-0.4346 ± j 0.5153
-0.4572 ± j 0.8947
-0.4384 ± j 1.3036
-2.9630
-2.9630
-5.1989
-0.1525 ± j 7.4783
-5.2184 ± j 7.8319
-5.3929 ± j 7.8503
-5.4351 ± j 7.9417
-0.6073 ± j 14.7192

(c) Eigenvalues.

Table G.17: SSSESCOPF results with $\text{Re}(\lambda_i) \leq -0.25$ (line 4-5 out)

APPENDIX G. SMALL-SIGNAL STABILITY ESCOPF RESULTS

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MVAR-hr)
1	1.030∠-6.6013°	132.08	46.07	-	-	24.04	0.03
2	1.045∠ 3.4578°	111.93	8.23	-	-	8.08	0.02
3	1.045∠-2.8639°	67.10	0.22	-	-	12.45	0.04
4	1.007∠-10.8062°	-	-	-	-	22.27	0.08
5	0.955∠-16.5460°	-	-	117.77	47.10	12.98	0.80
6	1.006∠-11.4731°	-	-	83.84	27.94	19.44	0.46
7	1.042∠-0.2249°	-	-	-	-	9.09	0.44
8	1.030∠-4.4902°	-	-	104.81	36.67	10.85	0.59
9	1.046∠-4.9263°	-	-	-	-	12.91	0.51

(a) Bus voltages, power injections, and power prices.

Bus from (<i>i</i>)	4	4	5	6	7	9	1	2	3
Bus to (<i>j</i>)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	15.53	127.47	0.00	74.30	111.93	8.43	139.89	112.23	67.10

(b) Line flows.

-0.4650 ± j 0.5351
-0.4650 ± j 0.8786
-0.4766 ± j 1.1788
-2.9885
-3.2258
-4.5771
-0.1504 ± j 7.3542
-5.2030 ± j 7.7964
-5.2683 ± j 7.8611
-5.3269 ± j 7.9400
-0.9207 ± j 13.8258

(c) Eigenvalues.

Table G.18: SSSESCOPF results with $\text{Re}(\lambda_i) \leq -0.25$ (line 5-7 out)

APPENDIX G. SMALL-SIGNAL STABILITY ESCOPF RESULTS

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MVAR-hr)
1	1.050∠-5.5783°	115.34	41.90	-	-	12.90	0.00
2	1.045∠ 2.2811°	149.62	11.07	-	-	12.26	0.00
3	1.045∠-1.0784°	45.84	00.50	-	-	12.26	0.00
4	1.029∠-9.1097°	-	-	-	-	12.90	0.03
5	1.006∠-10.3955°	-	-	117.77	47.10	12.98	0.11
6	0.993∠-13.2452°	-	-	83.84	27.94	12.29	0.16
7	1.042∠-2.6443°	-	-	-	-	12.26	0.03
8	1.030∠-4.8791°	-	-	104.81	36.67	12.38	0.05
9	1.045∠-2.4879°	-	-	-	-	12.26	0.01

(a) Bus voltages, power injections, and power prices.

Bus from (i)	4	4	5	6	7	9	1	2	3
Bus to (j)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	87.15	33.88	88.92	0.00	59.59	45.84	122.71	150.03	45.84

(b) Line flows.

-0.4820 ± j 0.5768
-0.4769 ± j 0.8555
-0.4830 ± j 1.1790
-2.6767
-3.2258
-4.6303
-0.1941 ± j 8.0361
-5.2044 ± j 7.7993
-5.2365 ± j 7.8684
-5.4371 ± j 7.9400
-0.9640 ± j 12.9902

(c) Eigenvalues.

Table G.19: SSSESCOPF results with $\text{Re}(\lambda_i) \leq -0.25$ (line 6-9 out)

APPENDIX G. SMALL-SIGNAL STABILITY ESCOPF RESULTS

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MVAR-hr)
1	1.050∠-4.1859°	84.77	20.50	-	-	12.85	0.00
2	1.045∠ 6.2642°	111.93	0.05	-	-	11.99	0.00
3	1.045∠-6.0358°	115.84	18.62	-	-	13.08	0.00
4	1.040∠-6.7492°	-	-	-	-	12.85	0.01
5	1.023∠-7.0190°	-	-	117.77	47.10	12.88	0.08
6	1.028∠-10.4379°	-	-	83.84	27.94	13.17	0.03
7	1.047∠ 2.5990°	-	-	-	-	12.00	0.05
8	0.992∠-15.3496°	-	-	104.81	36.67	13.43	0.14
9	1.037∠-9.6287°	-	-	-	-	13.08	0.02

(a) Bus voltages, power injections, and power prices.

Bus from (<i>i</i>)	4	4	5	6	7	9	1	2	3
Bus to (<i>j</i>)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	75.40	24.69	108.45	24.07	0.00	109.63	87.21	119.30	117.33

(b) Line flows.

-0.4530 ± j 0.7418
-0.4840 ± j 0.7960
-0.4703 ± j 1.2147
-3.2258
-3.3734
-4.0913
-0.2426 ± j 8.8572
-5.2185 ± j 7.8124
-5.2920 ± j 7.8937
-5.5782 ± j 7.9130
-0.3514 ± j 11.7665

(c) Eigenvalues.

Table G.20: SSSESCOPF results with $\text{Re}(\lambda_i) \leq -0.25$ (line 7-8 out)

APPENDIX G. SMALL-SIGNAL STABILITY ESCOPF RESULTS

Bus No.	Voltage (pu)	P_G (MW)	Q_G (MVAR)	P_L (MW)	Q_L (MVAR)	z_P (\$/MW-hr)	z_Q (\$/MVAR-hr)
1	1.050∠-3.3833°	68.47	20.33	-	-	12.82	0.00
2	1.045∠ 5.4168°	181.49	35.22	-	-	12.32	0.00
3	1.045∠ 0.6479°	60.87	-15.25	-	-	12.37	0.00
4	1.040∠-5.4539°	-	-	-	-	12.82	0.01
5	1.010∠-7.3274°	-	-	117.77	47.10	12.94	0.08
6	1.032∠-6.6136°	-	-	83.84	27.94	12.93	0.03
7	1.030∠-0.6346°	-	-	-	-	12.32	0.05
8	0.997∠-4.7110°	-	-	104.81	36.67	12.56	0.14
9	1.054∠-1.2078°	-	-	-	-	12.37	0.02

(a) Bus voltages, power injections, and power prices.

Bus from (<i>i</i>)	4	4	5	6	7	9	1	2	3
Bus to (<i>j</i>)	6	5	7	9	8	8	4	7	9
S_{ij} (MVA)	24.76	49.32	74.47	61.66	109.98	0.00	71.42	184.88	62.75

(b) Line flows.

-0.4578 ± j 0.6728
-0.4760 ± j 0.8090
-0.4736 ± j 1.1983
-3.2258
-3.2783
-4.0693
-0.1703 ± j 9.0672
-5.2009 ± j 7.8180
-5.2480 ± j 7.8976
-5.6927 ± j 7.9204
-0.4972 ± j 11.5588

(c) Eigenvalues.

Table G.21: SSSESCOPF results with $\text{Re}(\lambda_i) \leq -0.25$ (line 9-8 out)

Appendix H

Small-signal stability ESCOPF

Matlab code

H.1 Executing the program

The small-signal stability ESCOPF code can be found in the following location:

- directory - `/u/power/sss/condrej/escopf/arnoldi`
- server - `power.ecen.okstate.edu`

which is the power systems research group Linux server at Oklahoma State University. To execute the program, add the directory to the MatLab path and type `opf` at the MatLab prompt. The user then enters the name of the set of data files to be used. In this case the name to enter is `wsc9` since all the data files start with this name. To modify the eigenvalue constraint limits, the file `opf.m` must be modified on the line that defines the variable `data.limits.eig_max`, which is a vector containing the maximum eigenvalue real parts.

To add constraints to the problem, the variable to be constrained must be added to the file `variables.txt` and `subvariables.txt`. The `initialize.m` and `update.m` functions must be updated because of the Lagrange multipliers on the constraints, and the corresponding Jacobian, Hessian, and constraint evaluation files must also be updated.

H.2 opf.m

```
clear all
clear variables
format compact
%Enter name of file without cdf,gdf,or ldf extension
cdfname = input('Enter file name : ','s');
%cdfname = 'wsc9';
file_name.cdfname = strcat(cdfname, '.cdf');
file_name.ldfname = strcat(cdfname, '.ldf');
file_name.gdfname = strcat(cdfname, '.gdf');
file_name.sdfname = strcat(cdfname, '.sdf');

% Read data from files
raw_data=ReadDF(file_name);

% Store data to variables
data = getdata(raw_data);

% Create index
ind = index(data);

% Eigenvalue limit data
%data.eig_ind = 1:19;
%data.eig_ind = 1:ind.ineq.eig_real.end;
data.eig_ind = 1:10;
data.neig_const = length(data.eig_ind);
data.limits.eig_max = -0.25*ones(data.neig_const,1);

% Initialize variables
var = initialize(data,ind);

% Primal-dual interior-point algorithm
sol = pdip(var,data,ind);
```

H.3 createindex.m

```

%-----
% Opening file to write to
%-----
fid1 = fopen('index.m','w');
fprintf(fid1,'function aind=index(x)\n');
fprintf(fid1,'\n');

%-----
% Opening file to write from
%-----
fid2 = fopen('variables.txt');

%-----
% Getting comment lines from variables.m and printing to index.m
%-----
tline1 = fgetl(fid2);
fprintf(fid1,'%s\n',tline1);
tline2 = fgetl(fid2);
fprintf(fid1,'%s\n',tline2);
tline3 = fgetl(fid2);
fprintf(fid1,'%s\n',tline3);

%-----
% Getting each variable from Y variables
%-----
clear var1;
clear var2;
clear Yvar;
i=0;
tline = fgetl(fid2);
[var1, var2] = strtok(tline);
var2 = strtok(var2);

nY=0;
while (~isspace(var1))
    nY=nY+1;
    Yvar{nY,1} = var1;
    Yvar{nY,2} = var2;
    tline = fgetl(fid2);
    [var1, var2] = strtok(tline);
    var2 = strtok(var2);
end

%-----
% Printing Y variables to file index.m
%-----
fprintf(fid1,'aind.pre.%s.start = 1;\n',Yvar{1,1});
fprintf(fid1,'aind.pre.%s.end = %s;\n',Yvar{1,1},Yvar{1,2});

for k=2:nY
    tline = Yvar{k,1};
    fprintf(fid1,'aind.pre.%s.start = aind.pre.%s.end+1;\n',Yvar{k,1},Yvar{(k-1),1});
    fprintf(fid1,'aind.pre.%s.end = aind.pre.%s.end+%s;\n',Yvar{k,1},Yvar{(k-1),1},Yvar{k,2});
end
fprintf(fid1,'\n');

for k=1:nY
    tline = Yvar{k,1};
    fprintf(fid1,'aind.pre.%s.ind = aind.pre.%s.start',Yvar{k,1},Yvar{k,1});
    fprintf(fid1,': aind.pre.%s.end;\n',Yvar{k,1});
end

fprintf(fid1,'\n');

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

fprintf(fid1,'aind.pre.Y_len = aind.pre.%s.end;\n',Yvar{nY,1});
fprintf(fid1,'\n');
Ylenm = Yvar{nY,1};

%-----
% Getting comment lines from variables.m and printing to index.m
%-----
tline1 = fgetl(fid2);
fprintf(fid1,'%s\n',tline1);
tline2 = fgetl(fid2);
fprintf(fid1,'%s\n',tline2);
tline3 = fgetl(fid2);
fprintf(fid1,'%s\n',tline3);

%-----
% Getting each variable from equality constraints
%-----
clear var1;
clear var2;
clear Eq;
tline = fgetl(fid2);
[var1, var2] = strtok(tline);
var2 = strtok(var2);

neq=0;
while (~isspace(var1))
    neq = neq+1;
    Eq{neq,1} = var1;
    Eq{neq,2} = var2;
    tline = fgetl(fid2);
    [var1, var2] = strtok(tline);
    var2 = strtok(var2);
end

%-----
% Printing equality constraints to file index.m
%-----
fprintf(fid1,'aind.pre.eq.%s.start = 1;\n',Eq{1,1});
fprintf(fid1,'aind.pre.eq.%s.end = %s;\n',Eq{1,1},Eq{1,2});

for k=2:neq
    fprintf(fid1,'aind.pre.eq.%s.start = aind.pre.eq.%s.end+1;\n',Eq{k,1},Eq{(k-1),1});
    fprintf(fid1,'aind.pre.eq.%s.end = aind.pre.eq.%s.end+%s;\n',Eq{k,1},Eq{(k-1),1},Eq{k,2});
end
fprintf(fid1,'\n');

for k=1:neq
    tline = Eq{k,1};
    fprintf(fid1,'aind.pre.eq.%s.ind = aind.pre.eq.%s.start',Eq{k,1},Eq{k,1});
    fprintf(fid1,': aind.pre.eq.%s.end;\n',Eq{k,1});
end

fprintf(fid1,'\n');
fprintf(fid1,'aind.pre.neq = aind.pre.eq.%s.ind;\n', Eq{neq,1});
fprintf(fid1,'\n');

%-----
% Getting comment lines from variables.m and printing to index.m
%-----
tline1 = fgetl(fid2);
fprintf(fid1,'%s\n',tline1);
tline2 = fgetl(fid2);
fprintf(fid1,'%s\n',tline2);
tline3 = fgetl(fid2);
fprintf(fid1,'%s\n',tline3);

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

%-----
% Getting each variable from inequality constraints
%-----
clear var1;
clear var2;
clear Ineq;
tline = fgetl(fid2);
[var1, var2] = strtok(tline);
var2 = strtok(var2);

nineq=0;
%while (~isspace(var1))
while 1
    nineq=nineq+1;
    Ineq[nineq,1] = var1;
    Ineq[nineq,2] = var2;
    tline = fgetl(fid2);
    if ~isstr(tline), break, end
    [var1, var2] = strtok(tline);
    var2 = strtok(var2);
end

%-----
% Printing equality constraints to file index.m
%-----
fprintf(fid1,'aind.pre.ineq.%s.start = 1;\n',Ineq{1,1});
fprintf(fid1,'aind.pre.ineq.%s.end = %s;\n',Ineq{1,1},Ineq{1,2});

for k=2:nineq
    fprintf(fid1,'aind.pre.ineq.%s.start = aind.pre.ineq.%s.end+1;\n',Ineq{k,1},Ineq{(k-1),1});
    fprintf(fid1,'aind.pre.ineq.%s.end',Ineq{k,1});
    fprintf(fid1,'= aind.pre.ineq.%s.end+%s;\n',Ineq{k,1});
end
fprintf(fid1,'\n');

for k=1:nineq
    fprintf(fid1,'aind.pre.ineq.%s.ind ',Ineq{k,1});
    fprintf(fid1,'= aind.pre.ineq.%s.start : aind.pre.ineq.%s.end;\n',Ineq{k,1});
end

fprintf(fid1,'\n');
fprintf(fid1,'aind.pre.nineq = aind.pre.ineq.%s.end;\n',Ineq{nineq,1});
fprintf(fid1,'\n');

%-----
% Closing file to write from (variables.txt)
%-----
fclose(fid2);

%*****
%*****
%-----
% Sub-problems
%-----
%*****
%*****

%-----
% Opening file to write from
%-----
fid2 = fopen('subvariables.txt');

%-----
% Getting comment lines from variables.m and printing to index.m

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

%-----
tline1 = fgetl(fid2);
fprintf(fid1,'%s\n',tline1);
tline2 = fgetl(fid2);
fprintf(fid1,'%s\n',tline2);
tline3 = fgetl(fid2);
fprintf(fid1,'%s\n',tline3);

%-----
% Getting each variable from Y variables
%-----
clear var1;
clear var2;
clear Yvar;
i=0;
tline = fgetl(fid2);
[var1, var2] = strtok(tline);
var2 = strtok(var2);

nY=0;
while (~isspace(var1))
    nY=nY+1;
    Yvar{nY,1} = var1;
    Yvar{nY,2} = var2;
    tline = fgetl(fid2);
    [var1, var2] = strtok(tline);
    var2 = strtok(var2);
end

%-----
% Printing Y variables to file index.m
%-----
fprintf(fid1,'aind.post.%s.start = 1;\n',Yvar{1,1});
fprintf(fid1,'aind.post.%s.end = %s;\n',Yvar{1,1},Yvar{1,2});

for k=2:nY
    tline = Yvar{k,1};
    fprintf(fid1,'aind.post.%s.start = aind.post.%s.end+1;\n',Yvar{k,1},Yvar{(k-1),1});
    fprintf(fid1,'aind.post.%s.end = aind.post.%s.end+%s;\n',Yvar{k,1},Yvar{(k-1),1},Yvar{k,2});
end
fprintf(fid1,'\n');

for k=1:nY
    tline = Yvar{k,1};
    fprintf(fid1,'aind.post.%s.ind',Yvar{k,1});
    fprintf(fid1,'= [ aind.post.%s.start : aind.post.%s.end ] +aind.pre.Y_len;\n',Yvar{k,1},Yvar{k,1});
end%for

fprintf(fid1,'\n');
fprintf(fid1,'aind.post.Y_len = aind.post.%s.end;\n',Yvar{nY,1});
fprintf(fid1,'aind.Y_len = x.ncont*aind.post.Y_len + aind.pre.Y_len;');
fprintf(fid1,'\n');
Ylens = Yvar{nY,1};
fprintf(fid1,'\n');
fprintf(fid1,'for k=2:x.ncont\n');
for k=1:nY
    fprintf(fid1,'    aind.post.%s.ind(k,:)',Yvar{k,1});
    fprintf(fid1,'= aind.post.%s.ind(1,:)+(k-1)*aind.post.Y_len;\n',Yvar{k,1});
end%for
fprintf(fid1,'end\n');

fprintf(fid1,'aind.pre.eq.lagrange.ind');
fprintf(fid1,'= aind.Y_len+1 : (aind.Y_len + aind.pre.neq);\n');

%-----

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

% Getting comment lines from variables.m and printing to index.m
%-----
tline1 = fgetl(fid2);
fprintf(fid1,'%s\n',tline1);
tline2 = fgetl(fid2);
fprintf(fid1,'%s\n',tline2);
tline3 = fgetl(fid2);
fprintf(fid1,'%s\n',tline3);

%-----
% Getting each variable from equality constraints
%-----
clear var1;
clear var2;
clear Eq;
tline = fgetl(fid2);
[var1, var2] = strtok(tline);
var2 = strtok(var2);

neq=0;
while (~isspace(var1))
    neq = neq+1;
    Eq{neq,1} = var1;
    Eq{neq,2} = var2;
    tline = fgetl(fid2);
    [var1, var2] = strtok(tline);
    var2 = strtok(var2);
end

%-----
% Printing equality constraints to file index.m
%-----
fprintf(fid1,'aind.post.eq.%s.start = 1;\n',Eq{1,1});
fprintf(fid1,'aind.post.eq.%s.end = %s;\n',Eq{1,1},Eq{1,2});

for k=2:neq
    fprintf(fid1,'aind.post.eq.%s.start = aind.post.eq.%s.end+1;\n',Eq{k,1},Eq{(k-1),1});
    fprintf(fid1,'aind.post.eq.%s.end = aind.post.eq.%s.end+%s;\n',Eq{k,1},Eq{(k-1),1},Eq{k,2});
end
fprintf(fid1,'\n');

for k=1:neq
    tline = Eq{k,1};
    fprintf(fid1,'aind.post.eq.%s.ind',Eq{k,1});
    fprintf(fid1,'= [ aind.post.eq.%s.start : aind.post.eq.%s.end ];\n',Eq{k,1},Eq{k,1});
end

fprintf(fid1,'\n');
fprintf(fid1,'aind.post.neq = aind.post.eq.%s.end;\n', Eq{neq,1});
fprintf(fid1,'aind.neq = aind.pre.neq + x.ncont*aind.post.eq.%s.end;\n', Eq{neq,1});
fprintf(fid1,'\n');

fprintf(fid1,'aind.post.eq.lagrange.ind'
fprintf(fid1,'= (aind.Y_len+aind.pre.neq+1) : (aind.Y_len+aind.pre.neq+aind.post.neq);\n');
fprintf(fid1,'\n');

fprintf(fid1,'for k=2:x.ncont\n');
fprintf(fid1,'    aind.post.eq.lagrange.ind(k,:)');
fprintf(fid1,'= aind.post.eq.lagrange.ind(1,:)+(k-1)*aind.post.neq;\n');
fprintf(fid1,'end\n');

%-----
% Getting comment lines from variables.m and printing to index.m
%-----

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

tline1 = fgetl(fid2);
fprintf(fid1,'%s\n',tline1);
tline2 = fgetl(fid2);
fprintf(fid1,'%s\n',tline2);
tline3 = fgetl(fid2);
fprintf(fid1,'%s\n',tline3);

%-----
% Getting each variable from inequality constraints
%-----
clear var1;
clear var2;
clear Ineq;
tline = fgetl(fid2);
[var1, var2] = strtok(tline);
var2 = strtok(var2);

nineq=0;
%while (~isspace(var1))
while 1
    nineq=nineq+1;
    Ineq[nineq,1] = var1;
    Ineq[nineq,2] = var2;
    tline = fgetl(fid2);
    if ~isstr(tline), break, end
    [var1, var2] = strtok(tline);
    var2 = strtok(var2);
end

%-----
% Printing equality constraints to file index.m
%-----
fprintf(fid1,'aind.post.ineq.%s.start = 1;\n',Ineq{1,1});
fprintf(fid1,'aind.post.ineq.%s.end = %s;\n',Ineq{1,1},Ineq{1,2});

for k=2:nineq
    fprintf(fid1,'aind.post.ineq.%s.start = aind.post.ineq.%s.end+1;\n',Ineq{k,1},Ineq{(k-1),1});
    fprintf(fid1,'aind.post.ineq.%s.end = aind.post.ineq.%s.end+%s;\n',Ineq{k,1},Ineq{(k-1),1},Ineq{k,2});
end
fprintf(fid1,'\n');

for k=1:nineq
    fprintf(fid1,'aind.post.ineq.%s.ind',Ineq{k,1});
    fprintf(fid1,'= [ aind.post.ineq.%s.start : aind.post.ineq.%s.end];\n',Ineq{k,1},Ineq{k,1});
end
fprintf(fid1,'\n');

fprintf(fid1,'aind.post.nineq = aind.post.ineq.%s.end;\n',Ineq{nineq,1});
fprintf(fid1,'aind.nineq = aind.pre.nineq + x.ncont*aind.post.nineq;\n');
fprintf(fid1,'aind.pre.ineq.lagrange.ind'
fprintf(fid1,'= (aind.Y_len+aind.neq+1) : (aind.Y_len+aind.neq+aind.pre.nineq);\n');
fprintf(fid1,'aind.post.ineq.lagrange.ind'
fprintf(fid1,'= (aind.Y_len+aind.neq+aind.pre.nineq+1)'
fprintf(fid1,': (aind.Y_len+aind.neq+aind.pre.nineq+aind.post.nineq);\n');
fprintf(fid1,'aind.ineq.lagrange.ind'
fprintf(fid1,'= (aind.Y_len+aind.neq+1)'
fprintf(fid1,': (aind.Y_len+aind.neq+aind.pre.nineq+x.ncont*aind.post.nineq);\n');
fprintf(fid1,'aind.ineq.lagrange.ind'
fprintf(fid1,'= (aind.Y_len+aind.neq+1)'
fprintf(fid1,': (aind.Y_len+aind.neq+aind.nineq);\n');
fprintf(fid1,'aind.ineq.slack.ind'
fprintf(fid1,'= (aind.Y_len+aind.neq+aind.nineq+1)'
fprintf(fid1,': (aind.Y_len+aind.neq+2*aind.nineq);\n');
fprintf(fid1,'\n');

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```
fprintf(fid1,'for k=2:x.ncont\n');  
fprintf(fid1,'  aind.post.ineq.lagrange.ind(k,:)')  
fprintf(fid1,'= aind.post.ineq.lagrange.ind(1,:)+(k-1)*aind.post.nineq;\n');  
fprintf(fid1,'end\n');
```

```
%-----  
% Closing file to write to (index.m)  
%-----  
fclose(fid1);
```


H.4 variables.txt

```

%-----
% Indices for variables Y.
%-----
Pg      x.nPgen
Pl      x.nload
Qg      x.nQgen
V       x.nbuses
theta  x.nbuses
Pij     x.nlines
Qij     x.nlines
Ig      x.nPgen
gamma  x.nPgen
Adelt  x.nPgen
delta  x.nPgen
Vd      x.nPgen
Vq      x.nPgen
Id      x.nPgen
Iq      x.nPgen
Edp     x.nPgen
Eqp     x.nPgen
Efd     x.nPgen
Sij     x.nlines

%-----
% Indices for equality constraints
%-----
P       x.nbuses
Q       x.nbuses
Pij     x.nlines
Qij     x.nlines
Ig_real x.nPgen
Ig_imag x.nPgen
delta_real x.nPgen
delta_imag x.nPgen
Vdq_real x.nPgen
Vdq_imag x.nPgen
Idq_real x.nPgen
Idq_imag x.nPgen
Edp     x.nPgen
Eqp     x.nPgen
Efd     x.nPgen
Sij     x.nlines
slack  1

%-----
% Indices for inequality constraints
%-----
Pgmax   x.nPgen
Pgmin   x.nPgen
Vmax    x.nbuses
Vmin    x.nbuses
Qgmax   x.nQgen
Qgmin   x.nQgen
Igmin   x.nPgen
Adeltmin x.nPgen
eig_real 10
Plmin   x.nload
Sijmax  x.nlines

```

H.5 subvariables.txt

```

%-----
% Indices for variables Y.
%-----
Pg      x.nPgen
Pl      x.nload
Qg      x.nQgen
V       x.nbuses
theta  x.nbuses
Pij     x.nlines-1
Qij     x.nlines-1
Ig      x.nPgen
gamma  x.nPgen
Adelt  x.nPgen
delta  x.nPgen
Vd      x.nPgen
Vq      x.nPgen
Id      x.nPgen
Iq      x.nPgen
Edp     x.nPgen
Eqp     x.nPgen
Efd     x.nPgen
Sij     x.nlines-1

%-----
% Indices for equality constraints
%-----
P       x.nbuses
Q       x.nbuses
Pij     x.nlines-1
Qij     x.nlines-1
Ig_real x.nPgen
Ig_imag x.nPgen
delta_real x.nPgen
delta_imag x.nPgen
Vdq_real x.nPgen
Vdq_imag x.nPgen
Idq_real x.nPgen
Idq_imag x.nPgen
Edp     x.nPgen
Eqp     x.nPgen
Efd     x.nPgen
Sij     x.nlines-1
slack  1

%-----
% Indices for inequality constraints
%-----
Pgmax   x.nPgen
Pgmin   x.nPgen
Vmax    x.nbuses
Vmin    x.nbuses
Qgmax   x.nQgen
Qgmin   x.nQgen
Igmin   x.nPgen
Adeltmin x.nPgen
eig_real 10
Plmin   x.nload
Sijmax  x.nlines-1
Pgrampup x.nPgen
Pgrampdwn x.nPgen
Plmax   x.nload

```

H.6 ReadDF.m

```

function a=ReadDF(x)

%-----
%Read cdf file
% x.cdffile="filename.cdf"
% x.gdffile="filename.gdf"
% x.ldffile="filename.ldf"
%-----

%-----
%Find bus data section
%-----
message='FILE NOT FOUND';
%global b nl nb to fr Pmax busdata;
a.cdffid = fopen(x.cdffile,'r');
cdffid = fopen(x.cdffile,'r');
if cdffid == -1
% fprintf('%s\n',message);
error(message);
end%if
%s = fscanf(cdffid,'%s',2);
s=fgetl(cdffid);
a.Sbase=str2num(s(32:37));
tofind = 'bus data follows';
while 1
s = fgetl(cdffid);
pos = findstr(lower(s),tofind);
if isempty(pos)==0
break
end%if
if s == -1
fprintf(1,'EOF reached with no bus data found.\n')
break
end%if
end%while

%-----
%Find NBus_claimed
%-----
pos2 = findstr('items',lower(s));
if isempty(pos2), pos2=length(s)+1; end%if
NBus_claimed = str2num(s(pos+length(tofind):pos2-1));

%-----
%Get Bus Data
%-----
k=0;
while 1
s = fgetl(cdffid);
if isempty(findstr(s,'-999'))==0
break
end%if
k=k+1;
s(5:23)='';
busdata(k,:) = str2num(s);
end%while
busdata;
if k ~= NBus_claimed
fprintf(1,'Bus data does not match number of buses claimed.\n')
end%if
busdata;
a.busdata=busdata;

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

%-----
%Find NBranch_claimed
%-----
tofind='branch data follows';
while 1
    s = fgetl(cdfid);
    pos = findstr(lower(s),tofind);
    if isempty(pos)==0
        break
    end%if
    if s == -1
        fprintf(1,'EOF reached with no branch data found.\n')
        break
    end%if
end%while
pos2 = findstr('items',lower(s));
if isempty(pos2), pos2=length(s)+1; end%if
NBranch_claimed = str2num(s(pos+length(tofind):pos2-1));

%-----
%Get Branch Data
%-----
%You can have up to 51 line segments connecting two buses
%and up to 500 buses.
k=0;
bendx = sparse(500,50);
bendy = sparse(500,50);
bend = sparse(500,100);
%branchdata = sparse(500,120);
while 1
    s = fgetl(cdfid);
    if isempty(findstr(s,'-999'))==0
        break
    end%if
    k=k+1;
    %Start line points after column 145
    if length(s) > 145
        sbend = s(146:length(s));
        tbend = str2num(sbend);
        nbend(k) = length(tbend);
        if isempty(tbend) == 0
            bend(k,1:nbend(k)) = tbend;
            s=s(1:145);
        else nbend(k) = 0;
        end%if
    else
        nbend(k) = 0;
    end%if
    k2= 1:2:100 ;
    bendx = bend(:,k2);
    k2= 2:2:100 ;
    bendy = bend(:,k2) ;

    %branchdata(k,:) = [str2num(s) sparse(1,120-length(str2num(s)))];
    branchdata(k,:) = str2num(s) ;
end%while
nbend = nbend./2;
a.nbend = nbend./2;
nbend_old = nbend;
a.nbend_old = nbend;
branchdata;
a.branchdata=branchdata;
if k ~= NBranch_claimed
    fprintf(1,'Branch data does not match number of branches claimed.\n')

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

end%if

%-----
% Get Figure position data
%-----
tofind = 'figure and position data';
while 1
    s = fgetl(cdffid);
    pos = findstr(lower(s),tofind);
    if isempty(pos)==0
        break;
    end%if
    if s == -1
        fprintf(1,'EOF reached with no figure position and size data found.\n');
        break
    end%if
end%while

tofind = 'benefit';
%tofind = 'left';
k=0;
while 1
    s = fgetl(cdffid);
    s;
    isempty(findstr(lower(s),tofind));
    if isempty(findstr(lower(s),tofind)) == 0
        break
    end%if
    k=k+1;
    fig_data(k) = str2num( s(9:16) );
end%while
a.fig_data=fig_data;

tofind = 'benefit';
%while 1
% s = fgetl(cdffid);
% pos = findstr(lower(s),'-999');
% if isempty(pos)==0
%     break;
% end%if
% if s == -1
%     fprintf(1,'EOF reached with no Benefit position data found.\n');
%     break
% end%if
%end%while

%s = fgetl(cdffid);
benpos = zeros(1,4);
benpos = str2num( s(9:25) );
benpos(3: 4) = [230 45];
a.benpos=benpos;

fclose(cdffid);

%-----
%Read gdf file
%-----
global gdfdata
%Start reading data at line 3
gdffid = fopen(x.gdffile,'r');
s = fgetl(gdffid);
s = fgetl(gdffid);
k = 0;
while 1
    s = fgetl(gdffid);

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

    if s == -1
        break
    end%if
    k = k+1;
    s(7:15) = '';
    gdfdata(k,:) = str2num(s);
end%while
gdfdata;
a.gdfdata=gdfdata;

%-----
%Read ldf file
%-----
%Start reading data at line 3
%-----
ldffid = fopen(x.ldffile,'r');
s = fgetl(ldffid);
s = fgetl(ldffid);
k = 0;
while 1
    s = fgetl(ldffid);
    if s == -1
        break
    end%if
    k = k+1;
%   tofind = 'inf';
%   pos = findstr(lower(s),tofind);
%   s(7:60) = '';
    s(7:19) = '';
    s(33:60) = '';
    if isempty(find(s=='','))==0
        fprintf('NO COMMAS ALLOWED IN DATA, COMMA ON LINE %i\n',k);
    end%if
    ldfdata(k,:) = str2num(s);
end%while
a.ldfdata=ldfdata;

fclose(gdffid);
fclose(ldffid);

%-----
% Read sdf file
%-----
%Start reading data at line 3
sdffid = fopen(x.sdffile,'r');
s = fgetl(sdffid);
s = fgetl(sdffid);
k = 0;
while 1
    sp=feof(sdffid);
    if sp == 1
        break
    end%if
    s = fgetl(sdffid);
    %jec 11/8/2001
%   if s == -1
%       break
%   end%if
    %jec 11/8/2001
    k = k+1;
    s(7:15) = '';
    sdfdata(k,:) = str2num(s);
end%while

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```
sdfdata;  
a.sdfdata=sdfdata;
```

```
fclose(sdfid);
```

H.7 getdata.m

```

function o=getdata(x)
% Extract data from cdfdata, ldfdata, gdfdata, busdata, and branchdata
% matrices.

o.fr = x.branchdata(:,1);
o.to = x.branchdata(:,2);
o.nbuses = length(x.busdata(:,1));
o.nlines = length(o.fr);
o.Sbase = x.Sbase;

%*****
%gdf and ldf data
%*****
gen_ind = x.gdfdata(:,1);
o.nPgen = length(find(x.gdfdata(:,6)==1));
o.nQgen = length(find(x.gdfdata(:,7)==1));
o.Pg_ind = zeros(o.nbuses,1);
o.Pg_ind(gen_ind(find(x.gdfdata(:,6)==1))) = ones(o.nPgen,1);
o.Qg_ind = zeros(o.nbuses,1);
o.Qg_ind(gen_ind(find(x.gdfdata(:,7)==1))) = ones(o.nQgen,1);
o.Pgenbuses = gen_ind(find(x.gdfdata(:,6)==1));
o.Qgenbuses = gen_ind(find(x.gdfdata(:,7)==1));

load_ind = x.ldfdata(:,1);
o.nload = length(load_ind);
o.Pl_ind = zeros(o.nbuses,1);
o.Pl_ind(load_ind) = ones(o.nload,1);
o.Ploadbuses = load_ind;

%*****
%Get alpha,beta,gamma
%*****
o.gen.alpha = zeros(o.nbuses,1);
o.gen.beta = zeros(o.nbuses,1);
o.gen.gamma = zeros(o.nbuses,1);
o.gen.alpha(gen_ind) = x.gdfdata(:,9);
o.gen.beta(gen_ind) = x.gdfdata(:,10)*o.Sbase;
o.gen.gamma(gen_ind) = x.gdfdata(:,11)*o.Sbase^2;

o.gen.beta_ramp = zeros(o.nbuses,1);
o.gen.gamma_ramp = zeros(o.nbuses,1);
o.gen.beta_ramp(gen_ind) = x.gdfdata(:,18)*o.Sbase;
o.gen.gamma_ramp(gen_ind) = x.gdfdata(:,19)*o.Sbase^2;

o.load.alpha = zeros(o.nbuses,1);
o.load.beta = zeros(o.nbuses,1);
o.load.gamma = zeros(o.nbuses,1);
o.load.alpha(load_ind) = x.ldfdata(:,4);
o.load.beta(load_ind) = x.ldfdata(:,5)*o.Sbase;
o.load.gamma(load_ind) = x.ldfdata(:,6)*o.Sbase^2;

o.load.alpha_int = zeros(o.nbuses,1);
o.load.beta_int = zeros(o.nbuses,1);
o.load.gamma_int = zeros(o.nbuses,1);
o.load.alpha_int(load_ind) = x.ldfdata(:,7);
o.load.beta_int(load_ind) = x.ldfdata(:,8)*o.Sbase;
o.load.gamma_int(load_ind) = x.ldfdata(:,9)*o.Sbase^2;

%*****
%Line admittances

```


APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

%*****
R = x.branchdata(:,7);
X = x.branchdata(:,8);
Z = R+j*X;

o.Yvec = 1./Z;
o.Bvec = x.branchdata(:,9);

line_ind_fr = sparse(o.fr,1:o.nlines,ones(o.nlines,1),o.nbuses,o.nlines);
line_ind_to = sparse(o.to,1:o.nlines,ones(o.nlines,1),o.nbuses,o.nlines);
Ybusd = line_ind_fr*o.Yvec + line_ind_fr*j*o.Bvec/2 ...
        + line_ind_to*o.Yvec+line_ind_to*j*o.Bvec/2;
o.pre.Ybus = sparse([[1:o.nbuses]'; o.fr; o.to], ...
                    [[1:o.nbuses]'; o.to; o.fr], ...
                    [Ybusd; -o.Yvec; -o.Yvec;],o.nbuses,o.nbuses);

%*****
%Get line capacities, line text positions, and line probabilities
%*****
o.limits.Sijmax = x.branchdata(:,12);
o.limits.Sijmax_emerg = x.branchdata(:,11);
o.line.pi_k = .01*x.branchdata(:,22);
o.line.pi_0 = 1-sum(o.line.pi_k);
o.line.pi_k = o.line.pi_k(find(o.line.pi_k~=0));
o.cont_ind = find(o.line.pi_k>0);
o.ncont = length(o.cont_ind);

for k=1:o.ncont
nlines = o.nlines-1;
Yvec = o.Yvec;
Bvec = o.Bvec;
fr = o.fr;
to = o.to;
Yvec(k) = [];
Bvec(k) = [];
fr(k) = [];
to(k) = [];
line_ind_fr = sparse(fr,1:nlines,ones(nlines,1),o.nbuses,nlines);
line_ind_to = sparse(to,1:nlines,ones(nlines,1),o.nbuses,nlines);
Ybusd = line_ind_fr*Yvec + line_ind_fr*j*Bvec/2 ...
        + line_ind_to*Yvec+line_ind_to*j*Bvec/2;
o.post.Ybus{k} = sparse([[1:o.nbuses]'; fr; to], ...
                        [[1:o.nbuses]'; to; fr], ...
                        [Ybusd; -Yvec; -Yvec;],o.nbuses,o.nbuses);
end%for

%*****
%Get generator and load limits
%*****
o.limits.Pgmax = x.gdfdata(find(x.gdfdata(:,6)==1),2)/o.Sbase;
o.limits.Pgmin = x.gdfdata(find(x.gdfdata(:,6)==1),3)/o.Sbase;
o.limits.Qgmax = x.gdfdata(find(x.gdfdata(:,7)==1),4)/o.Sbase;
o.limits.Qgmin = x.gdfdata(find(x.gdfdata(:,7)==1),5)/o.Sbase;
o.limits.Pgrampupmax = x.gdfdata(find(x.gdfdata(:,6)==1),14)/o.Sbase;
o.limits.Pgrampdownmax = x.gdfdata(find(x.gdfdata(:,6)==1),15)/o.Sbase;
o.limits.Plmin = x.ldfdata(:,3)/o.Sbase;

o.bus_type = x.busdata(:,2);

o.slackbus = find(o.bus_type==3);

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

%*****
% Voltage limits
%*****
o.limits.volt_max = x.busdata(:,11);
o.limits.volt_max(gen_ind) = x.gdfdata(:,12);
o.limits.volt_min = x.busdata(:,12);
o.limits.volt_min(gen_ind) = x.gdfdata(:,13);
o.limits.volt_desired = x.busdata(:,10);
%o.limits.Pgrampupmax = x.gdfdata(:,14);
%o.limits.Pgrampdownmax = x.gdfdata(:,15);

%*****
% Initial guess for theta
%*****
o.theta_init = x.busdata(:,4)*pi/180;
o.Pg_init = x.busdata(:,7);
o.Pg_init = o.Pg_init(o.Pgenbuses)/o.Sbase;
o.Pl_init = x.busdata(:,5);
o.Pl_init = o.Pl_init(o.Ploadbuses)/o.Sbase;
o.Qg_init = x.busdata(:,8);
o.Qg_init = o.Qg_init(o.Qgenbuses)/o.Sbase;
o.Ql_init = x.busdata(:,6);
o.Ql_init = o.Ql_init(o.Ploadbuses)/o.Sbase;
o.V_init = x.busdata(:,3);

%*****
% Load power factor
%*****
o.pf = x.ldfdata(:,10);
o.kpf = sqrt(1./(o.pf.^2)-ones(length(o.pf),1));%.*sign(o.Ql_init);

%*****
% Get Machine Parameters
%*****
o.machine.Xd = x.sdfdata(:,2);
o.machine.Xdp = x.sdfdata(:,3);
o.machine.Xdpp = x.sdfdata(:,4);
o.machine.Xq = x.sdfdata(:,5);
o.machine.Xqp = x.sdfdata(:,6);
o.machine.Xqpp = x.sdfdata(:,7);
o.machine.XL = x.sdfdata(:,8);
o.machine.Tdop = x.sdfdata(:,9);
o.machine.Tdopp = x.sdfdata(:,10);
o.machine.Tqop = x.sdfdata(:,11);
o.machine.Tqopp = x.sdfdata(:,12);
o.machine.Ra = x.sdfdata(:,13);
o.machine.H = x.sdfdata(:,14);
o.machine.KD = x.sdfdata(:,15);
o.machine.KA = x.sdfdata(:,16);
o.machine.TA = x.sdfdata(:,17);
o.machine.KE = x.sdfdata(:,18);
o.machine.TE = x.sdfdata(:,19);
o.machine.KF = x.sdfdata(:,20);
o.machine.TF = x.sdfdata(:,21);
o.machine.D = x.sdfdata(:,22);

```

H.8 index.m

```

function aind=index(x)

%-----
% Indices for variables Y.
%-----
aind.pre.Pg.start = 1;
aind.pre.Pg.end = x.nPgen;
aind.pre.Pl.start = aind.pre.Pg.end+1;
aind.pre.Pl.end = aind.pre.Pg.end+x.nload;
aind.pre.Qg.start = aind.pre.Pl.end+1;
aind.pre.Qg.end = aind.pre.Pl.end+x.nQgen;
aind.pre.V.start = aind.pre.Qg.end+1;
aind.pre.V.end = aind.pre.Qg.end+x.nbuses;
aind.pre.theta.start = aind.pre.V.end+1;
aind.pre.theta.end = aind.pre.V.end+x.nbuses;
aind.pre.Pij.start = aind.pre.theta.end+1;
aind.pre.Pij.end = aind.pre.theta.end+x.nlines;
aind.pre.Qij.start = aind.pre.Pij.end+1;
aind.pre.Qij.end = aind.pre.Pij.end+x.nlines;
aind.pre.Ig.start = aind.pre.Qij.end+1;
aind.pre.Ig.end = aind.pre.Qij.end+x.nPgen;
aind.pre.gamma.start = aind.pre.Ig.end+1;
aind.pre.gamma.end = aind.pre.Ig.end+x.nPgen;
aind.pre.Adelt.start = aind.pre.gamma.end+1;
aind.pre.Adelt.end = aind.pre.gamma.end+x.nPgen;
aind.pre.delta.start = aind.pre.Adelt.end+1;
aind.pre.delta.end = aind.pre.Adelt.end+x.nPgen;
aind.pre.Vd.start = aind.pre.delta.end+1;
aind.pre.Vd.end = aind.pre.delta.end+x.nPgen;
aind.pre.Vq.start = aind.pre.Vd.end+1;
aind.pre.Vq.end = aind.pre.Vd.end+x.nPgen;
aind.pre.Id.start = aind.pre.Vq.end+1;
aind.pre.Id.end = aind.pre.Vq.end+x.nPgen;
aind.pre.Iq.start = aind.pre.Id.end+1;
aind.pre.Iq.end = aind.pre.Id.end+x.nPgen;
aind.pre.Edp.start = aind.pre.Iq.end+1;
aind.pre.Edp.end = aind.pre.Iq.end+x.nPgen;
aind.pre.Eqp.start = aind.pre.Edp.end+1;
aind.pre.Eqp.end = aind.pre.Edp.end+x.nPgen;
aind.pre.Efd.start = aind.pre.Eqp.end+1;
aind.pre.Efd.end = aind.pre.Eqp.end+x.nPgen;
aind.pre.Sij.start = aind.pre.Efd.end+1;
aind.pre.Sij.end = aind.pre.Efd.end+x.nlines;

aind.pre.Pg.ind = aind.pre.Pg.start : aind.pre.Pg.end;
aind.pre.Pl.ind = aind.pre.Pl.start : aind.pre.Pl.end;
aind.pre.Qg.ind = aind.pre.Qg.start : aind.pre.Qg.end;
aind.pre.V.ind = aind.pre.V.start : aind.pre.V.end;
aind.pre.theta.ind = aind.pre.theta.start : aind.pre.theta.end;
aind.pre.Pij.ind = aind.pre.Pij.start : aind.pre.Pij.end;
aind.pre.Qij.ind = aind.pre.Qij.start : aind.pre.Qij.end;
aind.pre.Ig.ind = aind.pre.Ig.start : aind.pre.Ig.end;
aind.pre.gamma.ind = aind.pre.gamma.start : aind.pre.gamma.end;
aind.pre.Adelt.ind = aind.pre.Adelt.start : aind.pre.Adelt.end;
aind.pre.delta.ind = aind.pre.delta.start : aind.pre.delta.end;
aind.pre.Vd.ind = aind.pre.Vd.start : aind.pre.Vd.end;
aind.pre.Vq.ind = aind.pre.Vq.start : aind.pre.Vq.end;
aind.pre.Id.ind = aind.pre.Id.start : aind.pre.Id.end;
aind.pre.Iq.ind = aind.pre.Iq.start : aind.pre.Iq.end;
aind.pre.Edp.ind = aind.pre.Edp.start : aind.pre.Edp.end;
aind.pre.Eqp.ind = aind.pre.Eqp.start : aind.pre.Eqp.end;
aind.pre.Efd.ind = aind.pre.Efd.start : aind.pre.Efd.end;
aind.pre.Sij.ind = aind.pre.Sij.start : aind.pre.Sij.end;

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

aind.pre.Y_len = aind.pre.Sij.end;

%-----
% Indices for equality constraints
%-----
aind.pre.eq.P.start = 1;
aind.pre.eq.P.end = x.nbuses;
aind.pre.eq.Q.start = aind.pre.eq.P.end+1;
aind.pre.eq.Q.end = aind.pre.eq.P.end+x.nbuses;
aind.pre.eq.Pij.start = aind.pre.eq.Q.end+1;
aind.pre.eq.Pij.end = aind.pre.eq.Q.end+x.nlines;
aind.pre.eq.Qij.start = aind.pre.eq.Pij.end+1;
aind.pre.eq.Qij.end = aind.pre.eq.Pij.end+x.nlines;
aind.pre.eq.Ig_real.start = aind.pre.eq.Qij.end+1;
aind.pre.eq.Ig_real.end = aind.pre.eq.Qij.end+x.nPgen;
aind.pre.eq.Ig_imag.start = aind.pre.eq.Ig_real.end+1;
aind.pre.eq.Ig_imag.end = aind.pre.eq.Ig_real.end+x.nPgen;
aind.pre.eq.delta_real.start = aind.pre.eq.Ig_imag.end+1;
aind.pre.eq.delta_real.end = aind.pre.eq.Ig_imag.end+x.nPgen;
aind.pre.eq.delta_imag.start = aind.pre.eq.delta_real.end+1;
aind.pre.eq.delta_imag.end = aind.pre.eq.delta_real.end+x.nPgen;
aind.pre.eq.Vdq_real.start = aind.pre.eq.delta_imag.end+1;
aind.pre.eq.Vdq_real.end = aind.pre.eq.delta_imag.end+x.nPgen;
aind.pre.eq.Vdq_imag.start = aind.pre.eq.Vdq_real.end+1;
aind.pre.eq.Vdq_imag.end = aind.pre.eq.Vdq_real.end+x.nPgen;
aind.pre.eq.Idq_real.start = aind.pre.eq.Vdq_imag.end+1;
aind.pre.eq.Idq_real.end = aind.pre.eq.Vdq_imag.end+x.nPgen;
aind.pre.eq.Idq_imag.start = aind.pre.eq.Idq_real.end+1;
aind.pre.eq.Idq_imag.end = aind.pre.eq.Idq_real.end+x.nPgen;
aind.pre.eq.Edp.start = aind.pre.eq.Idq_imag.end+1;
aind.pre.eq.Edp.end = aind.pre.eq.Idq_imag.end+x.nPgen;
aind.pre.eq.Eqp.start = aind.pre.eq.Edp.end+1;
aind.pre.eq.Eqp.end = aind.pre.eq.Edp.end+x.nPgen;
aind.pre.eq.Efd.start = aind.pre.eq.Eqp.end+1;
aind.pre.eq.Efd.end = aind.pre.eq.Eqp.end+x.nPgen;
aind.pre.eq.Sij.start = aind.pre.eq.Efd.end+1;
aind.pre.eq.Sij.end = aind.pre.eq.Efd.end+x.nlines;
aind.pre.eq.slack.start = aind.pre.eq.Sij.end+1;
aind.pre.eq.slack.end = aind.pre.eq.Sij.end+1;

aind.pre.eq.P.ind = aind.pre.eq.P.start : aind.pre.eq.P.end;
aind.pre.eq.Q.ind = aind.pre.eq.Q.start : aind.pre.eq.Q.end;
aind.pre.eq.Pij.ind = aind.pre.eq.Pij.start : aind.pre.eq.Pij.end;
aind.pre.eq.Qij.ind = aind.pre.eq.Qij.start : aind.pre.eq.Qij.end;
aind.pre.eq.Ig_real.ind = aind.pre.eq.Ig_real.start : aind.pre.eq.Ig_real.end;
aind.pre.eq.Ig_imag.ind = aind.pre.eq.Ig_imag.start : aind.pre.eq.Ig_imag.end;
aind.pre.eq.delta_real.ind = aind.pre.eq.delta_real.start : aind.pre.eq.delta_real.end;
aind.pre.eq.delta_imag.ind = aind.pre.eq.delta_imag.start : aind.pre.eq.delta_imag.end;
aind.pre.eq.Vdq_real.ind = aind.pre.eq.Vdq_real.start : aind.pre.eq.Vdq_real.end;
aind.pre.eq.Vdq_imag.ind = aind.pre.eq.Vdq_imag.start : aind.pre.eq.Vdq_imag.end;
aind.pre.eq.Idq_real.ind = aind.pre.eq.Idq_real.start : aind.pre.eq.Idq_real.end;
aind.pre.eq.Idq_imag.ind = aind.pre.eq.Idq_imag.start : aind.pre.eq.Idq_imag.end;
aind.pre.eq.Edp.ind = aind.pre.eq.Edp.start : aind.pre.eq.Edp.end;
aind.pre.eq.Eqp.ind = aind.pre.eq.Eqp.start : aind.pre.eq.Eqp.end;
aind.pre.eq.Efd.ind = aind.pre.eq.Efd.start : aind.pre.eq.Efd.end;
aind.pre.eq.Sij.ind = aind.pre.eq.Sij.start : aind.pre.eq.Sij.end;
aind.pre.eq.slack.ind = aind.pre.eq.slack.start : aind.pre.eq.slack.end;

aind.pre.neq = aind.pre.eq.slack.ind;

%-----
% Indices for inequality constraints
%-----
aind.pre.ineq.Pgmax.start = 1;

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

aind.pre.ineq.Pgmax.end = x.nPgen;
aind.pre.ineq.Pgmin.start = aind.pre.ineq.Pgmax.end+1;
aind.pre.ineq.Pgmin.end = aind.pre.ineq.Pgmax.end+x.nPgen;
aind.pre.ineq.Vmax.start = aind.pre.ineq.Pgmin.end+1;
aind.pre.ineq.Vmax.end = aind.pre.ineq.Pgmin.end+x.nbuses;
aind.pre.ineq.Vmin.start = aind.pre.ineq.Vmax.end+1;
aind.pre.ineq.Vmin.end = aind.pre.ineq.Vmax.end+x.nbuses;
aind.pre.ineq.Qgmax.start = aind.pre.ineq.Vmin.end+1;
aind.pre.ineq.Qgmax.end = aind.pre.ineq.Vmin.end+x.nQgen;
aind.pre.ineq.Qgmin.start = aind.pre.ineq.Qgmax.end+1;
aind.pre.ineq.Qgmin.end = aind.pre.ineq.Qgmax.end+x.nQgen;
aind.pre.ineq.Igmin.start = aind.pre.ineq.Qgmin.end+1;
aind.pre.ineq.Igmin.end = aind.pre.ineq.Qgmin.end+x.nPgen;
aind.pre.ineq.Adeltmin.start = aind.pre.ineq.Igmin.end+1;
aind.pre.ineq.Adeltmin.end = aind.pre.ineq.Igmin.end+x.nPgen;
aind.pre.ineq.eig_real.start = aind.pre.ineq.Adeltmin.end+1;
aind.pre.ineq.eig_real.end = aind.pre.ineq.Adeltmin.end+10;
aind.pre.ineq.Plmin.start = aind.pre.ineq.eig_real.end+1;
aind.pre.ineq.Plmin.end = aind.pre.ineq.eig_real.end+x.nload;
aind.pre.ineq.Sijmax.start = aind.pre.ineq.Plmin.end+1;
aind.pre.ineq.Sijmax.end = aind.pre.ineq.Plmin.end+x.nlines;

aind.pre.ineq.Pgmax.ind = aind.pre.ineq.Pgmax.start : aind.pre.ineq.Pgmax.end;
aind.pre.ineq.Pgmin.ind = aind.pre.ineq.Pgmin.start : aind.pre.ineq.Pgmin.end;
aind.pre.ineq.Vmax.ind = aind.pre.ineq.Vmax.start : aind.pre.ineq.Vmax.end;
aind.pre.ineq.Vmin.ind = aind.pre.ineq.Vmin.start : aind.pre.ineq.Vmin.end;
aind.pre.ineq.Qgmax.ind = aind.pre.ineq.Qgmax.start : aind.pre.ineq.Qgmax.end;
aind.pre.ineq.Qgmin.ind = aind.pre.ineq.Qgmin.start : aind.pre.ineq.Qgmin.end;
aind.pre.ineq.Igmin.ind = aind.pre.ineq.Igmin.start : aind.pre.ineq.Igmin.end;
aind.pre.ineq.Adeltmin.ind = aind.pre.ineq.Adeltmin.start : aind.pre.ineq.Adeltmin.end;
aind.pre.ineq.eig_real.ind = aind.pre.ineq.eig_real.start : aind.pre.ineq.eig_real.end;
aind.pre.ineq.Plmin.ind = aind.pre.ineq.Plmin.start : aind.pre.ineq.Plmin.end;
aind.pre.ineq.Sijmax.ind = aind.pre.ineq.Sijmax.start : aind.pre.ineq.Sijmax.end;

aind.pre.nineq = aind.pre.ineq.Sijmax.end;

%-----
% Indices for variables Y.
%-----
aind.post.Pg.start = 1;
aind.post.Pg.end = x.nPgen;
aind.post.Pl.start = aind.post.Pg.end+1;
aind.post.Pl.end = aind.post.Pg.end+x.nload;
aind.post.Qg.start = aind.post.Pl.end+1;
aind.post.Qg.end = aind.post.Pl.end+x.nQgen;
aind.post.V.start = aind.post.Qg.end+1;
aind.post.V.end = aind.post.Qg.end+x.nbuses;
aind.post.theta.start = aind.post.V.end+1;
aind.post.theta.end = aind.post.V.end+x.nbuses;
aind.post.Pij.start = aind.post.theta.end+1;
aind.post.Pij.end = aind.post.theta.end+x.nlines-1;
aind.post.Qij.start = aind.post.Pij.end+1;
aind.post.Qij.end = aind.post.Pij.end+x.nlines-1;
aind.post.Ig.start = aind.post.Qij.end+1;
aind.post.Ig.end = aind.post.Qij.end+x.nPgen;
aind.post.gamma.start = aind.post.Ig.end+1;
aind.post.gamma.end = aind.post.Ig.end+x.nPgen;
aind.post.Adelt.start = aind.post.gamma.end+1;
aind.post.Adelt.end = aind.post.gamma.end+x.nPgen;
aind.post.delta.start = aind.post.Adelt.end+1;
aind.post.delta.end = aind.post.Adelt.end+x.nPgen;
aind.post.Vd.start = aind.post.delta.end+1;
aind.post.Vd.end = aind.post.delta.end+x.nPgen;
aind.post.Vq.start = aind.post.Vd.end+1;
aind.post.Vq.end = aind.post.Vd.end+x.nPgen;

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

aind.post.Id.start = aind.post.Vq.end+1;
aind.post.Id.end = aind.post.Vq.end+x.nPgen;
aind.post.Iq.start = aind.post.Id.end+1;
aind.post.Iq.end = aind.post.Id.end+x.nPgen;
aind.post.Edp.start = aind.post.Iq.end+1;
aind.post.Edp.end = aind.post.Iq.end+x.nPgen;
aind.post.Eqp.start = aind.post.Edp.end+1;
aind.post.Eqp.end = aind.post.Edp.end+x.nPgen;
aind.post.Efd.start = aind.post.Eqp.end+1;
aind.post.Efd.end = aind.post.Eqp.end+x.nPgen;
aind.post.Sij.start = aind.post.Efd.end+1;
aind.post.Sij.end = aind.post.Efd.end+x.nlines-1;

aind.post.Pg.ind = [ aind.post.Pg.start : aind.post.Pg.end ] +aind.pre.Y_len;
aind.post.Pl.ind = [ aind.post.Pl.start : aind.post.Pl.end ] +aind.pre.Y_len;
aind.post.Qg.ind = [ aind.post.Qg.start : aind.post.Qg.end ] +aind.pre.Y_len;
aind.post.V.ind = [ aind.post.V.start : aind.post.V.end ] +aind.pre.Y_len;
aind.post.theta.ind = [ aind.post.theta.start : aind.post.theta.end ] +aind.pre.Y_len;
aind.post.Pij.ind = [ aind.post.Pij.start : aind.post.Pij.end ] +aind.pre.Y_len;
aind.post.Qij.ind = [ aind.post.Qij.start : aind.post.Qij.end ] +aind.pre.Y_len;
aind.post.Ig.ind = [ aind.post.Ig.start : aind.post.Ig.end ] +aind.pre.Y_len;
aind.post.gamma.ind = [ aind.post.gamma.start : aind.post.gamma.end ] +aind.pre.Y_len;
aind.post.Adelt.ind = [ aind.post.Adelt.start : aind.post.Adelt.end ] +aind.pre.Y_len;
aind.post.delta.ind = [ aind.post.delta.start : aind.post.delta.end ] +aind.pre.Y_len;
aind.post.Vd.ind = [ aind.post.Vd.start : aind.post.Vd.end ] +aind.pre.Y_len;
aind.post.Vq.ind = [ aind.post.Vq.start : aind.post.Vq.end ] +aind.pre.Y_len;
aind.post.Id.ind = [ aind.post.Id.start : aind.post.Id.end ] +aind.pre.Y_len;
aind.post.Iq.ind = [ aind.post.Iq.start : aind.post.Iq.end ] +aind.pre.Y_len;
aind.post.Edp.ind = [ aind.post.Edp.start : aind.post.Edp.end ] +aind.pre.Y_len;
aind.post.Eqp.ind = [ aind.post.Eqp.start : aind.post.Eqp.end ] +aind.pre.Y_len;
aind.post.Efd.ind = [ aind.post.Efd.start : aind.post.Efd.end ] +aind.pre.Y_len;
aind.post.Sij.ind = [ aind.post.Sij.start : aind.post.Sij.end ] +aind.pre.Y_len;

aind.post.Y_len = aind.post.Sij.end;
aind.Y_len = x.ncont*aind.post.Y_len + aind.pre.Y_len;

for k=2:x.ncont
    aind.post.Pg.ind(k,:) = aind.post.Pg.ind(1,:)+(k-1)*aind.post.Y_len;
    aind.post.Pl.ind(k,:) = aind.post.Pl.ind(1,:)+(k-1)*aind.post.Y_len;
    aind.post.Qg.ind(k,:) = aind.post.Qg.ind(1,:)+(k-1)*aind.post.Y_len;
    aind.post.V.ind(k,:) = aind.post.V.ind(1,:)+(k-1)*aind.post.Y_len;
    aind.post.theta.ind(k,:) = aind.post.theta.ind(1,:)+(k-1)*aind.post.Y_len;
    aind.post.Pij.ind(k,:) = aind.post.Pij.ind(1,:)+(k-1)*aind.post.Y_len;
    aind.post.Qij.ind(k,:) = aind.post.Qij.ind(1,:)+(k-1)*aind.post.Y_len;
    aind.post.Ig.ind(k,:) = aind.post.Ig.ind(1,:)+(k-1)*aind.post.Y_len;
    aind.post.gamma.ind(k,:) = aind.post.gamma.ind(1,:)+(k-1)*aind.post.Y_len;
    aind.post.Adelt.ind(k,:) = aind.post.Adelt.ind(1,:)+(k-1)*aind.post.Y_len;
    aind.post.delta.ind(k,:) = aind.post.delta.ind(1,:)+(k-1)*aind.post.Y_len;
    aind.post.Vd.ind(k,:) = aind.post.Vd.ind(1,:)+(k-1)*aind.post.Y_len;
    aind.post.Vq.ind(k,:) = aind.post.Vq.ind(1,:)+(k-1)*aind.post.Y_len;
    aind.post.Id.ind(k,:) = aind.post.Id.ind(1,:)+(k-1)*aind.post.Y_len;
    aind.post.Iq.ind(k,:) = aind.post.Iq.ind(1,:)+(k-1)*aind.post.Y_len;
    aind.post.Edp.ind(k,:) = aind.post.Edp.ind(1,:)+(k-1)*aind.post.Y_len;
    aind.post.Eqp.ind(k,:) = aind.post.Eqp.ind(1,:)+(k-1)*aind.post.Y_len;
    aind.post.Efd.ind(k,:) = aind.post.Efd.ind(1,:)+(k-1)*aind.post.Y_len;
    aind.post.Sij.ind(k,:) = aind.post.Sij.ind(1,:)+(k-1)*aind.post.Y_len;
end
aind.pre.eq.lagrange.ind = aind.Y_len+1 : (aind.Y_len + aind.pre.neq);
%-----
% Indices for equality constraints
%-----
aind.post.eq.P.start = 1;
aind.post.eq.P.end = x.nbuses;
aind.post.eq.Q.start = aind.post.eq.P.end+1;
aind.post.eq.Q.end = aind.post.eq.P.end+x.nbuses;

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

aind.post.eq.Pij.start = aind.post.eq.Q.end+1;
aind.post.eq.Pij.end = aind.post.eq.Q.end+x.nlines-1;
aind.post.eq.Qij.start = aind.post.eq.Pij.end+1;
aind.post.eq.Qij.end = aind.post.eq.Pij.end+x.nlines-1;
aind.post.eq.Ig_real.start = aind.post.eq.Qij.end+1;
aind.post.eq.Ig_real.end = aind.post.eq.Qij.end+x.nPgen;
aind.post.eq.Ig_imag.start = aind.post.eq.Ig_real.end+1;
aind.post.eq.Ig_imag.end = aind.post.eq.Ig_real.end+x.nPgen;
aind.post.eq.delta_real.start = aind.post.eq.Ig_imag.end+1;
aind.post.eq.delta_real.end = aind.post.eq.Ig_imag.end+x.nPgen;
aind.post.eq.delta_imag.start = aind.post.eq.delta_real.end+1;
aind.post.eq.delta_imag.end = aind.post.eq.delta_real.end+x.nPgen;
aind.post.eq.Vdq_real.start = aind.post.eq.delta_imag.end+1;
aind.post.eq.Vdq_real.end = aind.post.eq.delta_imag.end+x.nPgen;
aind.post.eq.Vdq_imag.start = aind.post.eq.Vdq_real.end+1;
aind.post.eq.Vdq_imag.end = aind.post.eq.Vdq_real.end+x.nPgen;
aind.post.eq.Idq_real.start = aind.post.eq.Vdq_imag.end+1;
aind.post.eq.Idq_real.end = aind.post.eq.Vdq_imag.end+x.nPgen;
aind.post.eq.Idq_imag.start = aind.post.eq.Idq_real.end+1;
aind.post.eq.Idq_imag.end = aind.post.eq.Idq_real.end+x.nPgen;
aind.post.eq.Edp.start = aind.post.eq.Idq_imag.end+1;
aind.post.eq.Edp.end = aind.post.eq.Idq_imag.end+x.nPgen;
aind.post.eq.Eqp.start = aind.post.eq.Edp.end+1;
aind.post.eq.Eqp.end = aind.post.eq.Edp.end+x.nPgen;
aind.post.eq.Efd.start = aind.post.eq.Eqp.end+1;
aind.post.eq.Efd.end = aind.post.eq.Eqp.end+x.nPgen;
aind.post.eq.Sij.start = aind.post.eq.Efd.end+1;
aind.post.eq.Sij.end = aind.post.eq.Efd.end+x.nlines-1;
aind.post.eq.slack.start = aind.post.eq.Sij.end+1;
aind.post.eq.slack.end = aind.post.eq.Sij.end+1;

aind.post.eq.P.ind = [ aind.post.eq.P.start : aind.post.eq.P.end ];
aind.post.eq.Q.ind = [ aind.post.eq.Q.start : aind.post.eq.Q.end ];
aind.post.eq.Pij.ind = [ aind.post.eq.Pij.start : aind.post.eq.Pij.end ];
aind.post.eq.Qij.ind = [ aind.post.eq.Qij.start : aind.post.eq.Qij.end ];
aind.post.eq.Ig_real.ind = [ aind.post.eq.Ig_real.start : aind.post.eq.Ig_real.end ];
aind.post.eq.Ig_imag.ind = [ aind.post.eq.Ig_imag.start : aind.post.eq.Ig_imag.end ];
aind.post.eq.delta_real.ind = [ aind.post.eq.delta_real.start : aind.post.eq.delta_real.end ];
aind.post.eq.delta_imag.ind = [ aind.post.eq.delta_imag.start : aind.post.eq.delta_imag.end ];
aind.post.eq.Vdq_real.ind = [ aind.post.eq.Vdq_real.start : aind.post.eq.Vdq_real.end ];
aind.post.eq.Vdq_imag.ind = [ aind.post.eq.Vdq_imag.start : aind.post.eq.Vdq_imag.end ];
aind.post.eq.Idq_real.ind = [ aind.post.eq.Idq_real.start : aind.post.eq.Idq_real.end ];
aind.post.eq.Idq_imag.ind = [ aind.post.eq.Idq_imag.start : aind.post.eq.Idq_imag.end ];
aind.post.eq.Edp.ind = [ aind.post.eq.Edp.start : aind.post.eq.Edp.end ];
aind.post.eq.Eqp.ind = [ aind.post.eq.Eqp.start : aind.post.eq.Eqp.end ];
aind.post.eq.Efd.ind = [ aind.post.eq.Efd.start : aind.post.eq.Efd.end ];
aind.post.eq.Sij.ind = [ aind.post.eq.Sij.start : aind.post.eq.Sij.end ];
aind.post.eq.slack.ind = [ aind.post.eq.slack.start : aind.post.eq.slack.end ];

aind.post.neq = aind.post.eq.slack.end;
aind.neq = aind.pre.neq + x.ncont*aind.post.eq.slack.end;

aind.post.eq.lagrange.ind = (aind.Y_len+aind.pre.neq+1) : (aind.Y_len+aind.pre.neq+aind.post.neq);

for k=2:x.ncont
    aind.post.eq.lagrange.ind(k,:) = aind.post.eq.lagrange.ind(1,:)+(k-1)*aind.post.neq;
end
%-----
% Indices for inequality constraints
%-----
aind.post.ineq.Pgmax.start = 1;
aind.post.ineq.Pgmax.end = x.nPgen;
aind.post.ineq.Pgmin.start = aind.post.ineq.Pgmax.end+1;
aind.post.ineq.Pgmin.end = aind.post.ineq.Pgmax.end+x.nPgen;
aind.post.ineq.Vmax.start = aind.post.ineq.Pgmin.end+1;

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

aind.post.ineq.Vmax.end = aind.post.ineq.Pgmin.end+x.nbuses;
aind.post.ineq.Vmin.start = aind.post.ineq.Vmax.end+1;
aind.post.ineq.Vmin.end = aind.post.ineq.Vmax.end+x.nbuses;
aind.post.ineq.Qgmax.start = aind.post.ineq.Vmin.end+1;
aind.post.ineq.Qgmax.end = aind.post.ineq.Vmin.end+x.nQgen;
aind.post.ineq.Qgmin.start = aind.post.ineq.Qgmax.end+1;
aind.post.ineq.Qgmin.end = aind.post.ineq.Qgmax.end+x.nQgen;
aind.post.ineq.Igmin.start = aind.post.ineq.Qgmin.end+1;
aind.post.ineq.Igmin.end = aind.post.ineq.Qgmin.end+x.nPgen;
aind.post.ineq.Adeltmin.start = aind.post.ineq.Igmin.end+1;
aind.post.ineq.Adeltmin.end = aind.post.ineq.Igmin.end+x.nPgen;
aind.post.ineq.eig_real.start = aind.post.ineq.Adeltmin.end+1;
aind.post.ineq.eig_real.end = aind.post.ineq.Adeltmin.end+10;
aind.post.ineq.Plmin.start = aind.post.ineq.eig_real.end+1;
aind.post.ineq.Plmin.end = aind.post.ineq.eig_real.end+x.nload;
aind.post.ineq.Sijmax.start = aind.post.ineq.Plmin.end+1;
aind.post.ineq.Sijmax.end = aind.post.ineq.Plmin.end+x.nlines-1;
aind.post.ineq.Pgrampup.start = aind.post.ineq.Sijmax.end+1;
aind.post.ineq.Pgrampup.end = aind.post.ineq.Sijmax.end+x.nPgen;
aind.post.ineq.Pgrampdown.start = aind.post.ineq.Pgrampup.end+1;
aind.post.ineq.Pgrampdown.end = aind.post.ineq.Pgrampup.end+x.nPgen;
aind.post.ineq.Plmax.start = aind.post.ineq.Pgrampdown.end+1;
aind.post.ineq.Plmax.end = aind.post.ineq.Pgrampdown.end+x.nload;

aind.post.ineq.Pgmax.ind = [ aind.post.ineq.Pgmax.start : aind.post.ineq.Pgmax.end];
aind.post.ineq.Pgmin.ind = [ aind.post.ineq.Pgmin.start : aind.post.ineq.Pgmin.end];
aind.post.ineq.Vmax.ind = [ aind.post.ineq.Vmax.start : aind.post.ineq.Vmax.end];
aind.post.ineq.Vmin.ind = [ aind.post.ineq.Vmin.start : aind.post.ineq.Vmin.end];
aind.post.ineq.Qgmax.ind = [ aind.post.ineq.Qgmax.start : aind.post.ineq.Qgmax.end];
aind.post.ineq.Qgmin.ind = [ aind.post.ineq.Qgmin.start : aind.post.ineq.Qgmin.end];
aind.post.ineq.Igmin.ind = [ aind.post.ineq.Igmin.start : aind.post.ineq.Igmin.end];
aind.post.ineq.Adeltmin.ind = [ aind.post.ineq.Adeltmin.start : aind.post.ineq.Adeltmin.end];
aind.post.ineq.eig_real.ind = [ aind.post.ineq.eig_real.start : aind.post.ineq.eig_real.end];
aind.post.ineq.Plmin.ind = [ aind.post.ineq.Plmin.start : aind.post.ineq.Plmin.end];
aind.post.ineq.Sijmax.ind = [ aind.post.ineq.Sijmax.start : aind.post.ineq.Sijmax.end];
aind.post.ineq.Pgrampup.ind = [ aind.post.ineq.Pgrampup.start : aind.post.ineq.Pgrampup.end];
aind.post.ineq.Pgrampdown.ind = [ aind.post.ineq.Pgrampdown.start : aind.post.ineq.Pgrampdown.end];
aind.post.ineq.Plmax.ind = [ aind.post.ineq.Plmax.start : aind.post.ineq.Plmax.end];

aind.post.nineq = aind.post.ineq.Plmax.end;
aind.nineq = aind.pre.nineq + x.ncont*aind.post.nineq;
aind.pre.ineq.lagrange.ind = (aind.Y_len+aind.neq+1) : ...
    (aind.Y_len+aind.neq+aind.pre.nineq);
aind.post.ineq.lagrange.ind = (aind.Y_len+aind.neq+aind.pre.nineq+1) : ...
    (aind.Y_len+aind.neq+aind.pre.nineq+aind.post.nineq);
aind.ineq.lagrange.ind = (aind.Y_len+aind.neq+1) : ...
    (aind.Y_len+aind.neq+aind.pre.nineq+x.ncont*aind.post.nineq);
aind.ineq.lagrange.ind = (aind.Y_len+aind.neq+1) : ...
    (aind.Y_len+aind.neq+aind.nineq);
aind.ineq.slack.ind = (aind.Y_len+aind.neq+aind.nineq+1) : ...
    (aind.Y_len+aind.neq+2*aind.nineq);

for k=2:x.ncont
    aind.post.ineq.lagrange.ind(k,:) = ...
        aind.post.ineq.lagrange.ind(1,:)+(k-1)*aind.post.nineq;
end

```


H.9 initialize.m

```

function avar=initialize(data,index)

%*****
% Initialize V
%*****
avar.pre.V = data.V_init;
avar.allvariables(index.pre.V.ind) = avar.pre.V;

%*****
% Initialize Pg, Pl
%*****
%genbuses = find(.gen.numgl>0);
%loadbuses = find(o.load.numgl>0);
avar.pre.Pg = data.Pg_init;
avar.pre.Qg = data.Qg_init;
avar.pre.Pl = data.Pl_init;
avar.pre.Ig = data.Pg_init./data.V_init(data.Pgenbuses);
avar.pre.gamma = angle(avar.pre.Pg-j*avar.pre.Qg);

avar.allvariables(index.pre.Pg.ind) = avar.pre.Pg;
avar.allvariables(index.pre.Pl.ind) = avar.pre.Pl;
avar.allvariables(index.pre.Qg.ind) = avar.pre.Qg;
avar.allvariables(index.pre.Ig.ind) = avar.pre.Ig;
avar.allvariables(index.pre.gamma.ind) = avar.pre.gamma;

%*****
% Initialize lambda.P, lambda.Q,
%*****
%avar.pre.eq.lagrange.P = 1e-3*.2*(1:data.nbuses)';
%avar.pre.eq.lagrange.Q = .5*(1:data.nbuses)';
%avar.pre.eq.lagrange.P = -120*.5*(ones(1,data.nbuses))';
%avar.pre.eq.lagrange.Q = .5*(ones(1,data.nbuses))';
%avar.pre.eq.lagrange.Pij= .5*(1:data.nlines)';
%avar.pre.eq.lagrange.Qij= .5*(1:data.nlines)';
%avar.pre.eq.lagrange.Sij= .5*(1:data.nlines)';
%avar.pre.eq.lagrange.Ig_real= -10*.5*(1:data.nPgen)';
%avar.pre.eq.lagrange.Ig_imag= 10*.5*(1:data.nPgen)';
%avar.pre.eq.lagrange.delta_real= ones(data.nPgen,1);
%avar.pre.eq.lagrange.delta_imag= ones(data.nPgen,1);
%avar.pre.eq.lagrange.Idq_real= ones(data.nPgen,1);
%avar.pre.eq.lagrange.Idq_imag= ones(data.nPgen,1);
%avar.pre.eq.lagrange.Vdq_real= ones(data.nPgen,1);
%avar.pre.eq.lagrange.Vdq_imag= ones(data.nPgen,1);
%avar.pre.eq.lagrange.Edp= ones(data.nPgen,1);
%avar.pre.eq.lagrange.Eqp= ones(data.nPgen,1);
%avar.pre.eq.lagrange.Efd= ones(data.nPgen,1);
%avar.pre.eq.lagrange.slack= .5;
avar.allvariables(index.pre.eq.lagrange.ind(index.pre.eq.P.ind)) = avar.pre.eq.lagrange.P;
avar.allvariables(index.pre.eq.lagrange.ind(index.pre.eq.Q.ind)) = avar.pre.eq.lagrange.Q;
avar.allvariables(index.pre.eq.lagrange.ind(index.pre.eq.Pij.ind)) = ...
    avar.pre.eq.lagrange.Pij;
avar.allvariables(index.pre.eq.lagrange.ind(index.pre.eq.Qij.ind)) = ...
    avar.pre.eq.lagrange.Qij;
avar.allvariables(index.pre.eq.lagrange.ind(index.pre.eq.Sij.ind)) = ...
    avar.pre.eq.lagrange.Sij;
avar.allvariables(index.pre.eq.lagrange.ind(index.pre.eq.Ig_real.ind)) = ...
    avar.pre.eq.lagrange.Ig_real;
avar.allvariables(index.pre.eq.lagrange.ind(index.pre.eq.Ig_imag.ind)) = ...
    avar.pre.eq.lagrange.Ig_imag;
avar.allvariables(index.pre.eq.lagrange.ind(index.pre.eq.delta_real.ind)) = ...
    avar.pre.eq.lagrange.delta_real;
avar.allvariables(index.pre.eq.lagrange.ind(index.pre.eq.delta_imag.ind)) = ...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

                                avar.pre.eq.lagrange.delta_imag;
avar.allvariables(index.pre.eq.lagrange.ind(index.pre.eq.Idq_real.ind)) = ...
                                avar.pre.eq.lagrange.Idq_real;
avar.allvariables(index.pre.eq.lagrange.ind(index.pre.eq.Idq_imag.ind)) = ...
                                avar.pre.eq.lagrange.Idq_imag;
avar.allvariables(index.pre.eq.lagrange.ind(index.pre.eq.Vdq_real.ind)) = ...
                                avar.pre.eq.lagrange.Vdq_real;
avar.allvariables(index.pre.eq.lagrange.ind(index.pre.eq.Vdq_imag.ind)) = ...
                                avar.pre.eq.lagrange.Vdq_imag;
avar.allvariables(index.pre.eq.lagrange.ind(index.pre.eq.Edp.ind)) = ...
                                avar.pre.eq.lagrange.Edp;
avar.allvariables(index.pre.eq.lagrange.ind(index.pre.eq.Eqp.ind)) = ...
                                avar.pre.eq.lagrange.Eqp;
avar.allvariables(index.pre.eq.lagrange.ind(index.pre.eq.Efd.ind)) = ...
                                avar.pre.eq.lagrange.Efd;
avar.allvariables(index.pre.eq.lagrange.ind(index.pre.eq.slack.ind)) = ...
                                avar.pre.eq.lagrange.slack;
%avar.pre.eq.lagrange.total = avar.allvariables(index.pre.eq.lagrange.ind)';
%avar.eq.lagrange.total = avar.allvariables(index.pre.eq.lagrange.ind)';

%*****
% Initialize theta
%*****
avar.pre.theta = data.theta_init;
avar.pre.Adelt = ...
    abs(avar.pre.V(data.Pgenbuses).*exp(j*avar.pre.theta(data.Pgenbuses)) ...
        +j*data.machine.Xq.*avar.pre.Ig.*exp(j*avar.pre.gamma));
avar.pre.delta = ...
    angle(avar.pre.V(data.Pgenbuses).*exp(j*avar.pre.theta(data.Pgenbuses)) ...
        +j*data.machine.Xq.*avar.pre.Ig.*exp(j*avar.pre.gamma));
avar.pre.Id = real(j*avar.pre.Ig ...
    .*exp(j*avar.pre.gamma).*exp(-j*avar.pre.delta));
avar.pre.Iq = imag(j*avar.pre.Ig ...
    .*exp(j*avar.pre.gamma).*exp(-j*avar.pre.delta));
avar.pre.Vd = real(j*avar.pre.V(data.Pgenbuses) ...
    .*exp(j*avar.pre.theta(data.Pgenbuses)).*exp(-j*avar.pre.delta));
avar.pre.Vq = imag(j*avar.pre.V(data.Pgenbuses) ...
    .*exp(j*avar.pre.theta(data.Pgenbuses)).*exp(-j*avar.pre.delta));
avar.pre.Edp = avar.pre.Vd + ...
    data.machine.Ra.*avar.pre.Id - data.machine.Xqp.*avar.pre.Iq;
avar.pre.Eqp = avar.pre.Vq + ...
    data.machine.Ra.*avar.pre.Iq + data.machine.Xdp.*avar.pre.Id;
avar.pre.Efd = avar.pre.Eqp + (data.machine.Xd-data.machine.Xdp).*avar.pre.Id;
avar.allvariables(index.pre.theta.ind) = avar.pre.theta;
avar.allvariables(index.pre.Adelt.ind) = avar.pre.Adelt;
avar.allvariables(index.pre.delta.ind) = avar.pre.delta;
avar.allvariables(index.pre.Id.ind) = avar.pre.Id;
avar.allvariables(index.pre.Iq.ind) = avar.pre.Iq;
avar.allvariables(index.pre.Vd.ind) = avar.pre.Vd;
avar.allvariables(index.pre.Vq.ind) = avar.pre.Vq;
avar.allvariables(index.pre.Edp.ind) = avar.pre.Edp;
avar.allvariables(index.pre.Eqp.ind) = avar.pre.Eqp;
avar.allvariables(index.pre.Efd.ind) = avar.pre.Efd;

%*****
% Initialize inequality multipliers(mu) and slacks(s)
%*****
avar.ineq.lagrange = ones(index.nineq,1);
avar.ineq.slack = 1.2*ones(index.nineq,1);
avar.ineq.g = zeros(index.nineq,1);

avar.allvariables(index.ineq.lagrange.ind) = avar.ineq.lagrange;
avar.allvariables(index.ineq.slack.ind) = avar.ineq.slack;

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

avar.ineq.Slack = sparse(diag(avar.ineq.slack));
avar.ineq.Lagrange = sparse(diag(avar.ineq.lagrange));
%avar.ineq.g(:) = -10;

%*****
% Calculate line flows
%*****
avar.pre.Sij = zeros(data.nlines,1);
avar.pre.Pij = zeros(data.nlines,1)+0.02;
avar.pre.Qij = zeros(data.nlines,1)+0.02;
avar.pre.Sij = zeros(data.nlines,1)+0.02;
avar.pre.Iij = zeros(data.nlines,1)+0.01;

avar.allvariables(index.pre.Pij.ind) = avar.pre.Pij;
avar.allvariables(index.pre.Qij.ind) = avar.pre.Qij;
avar.allvariables(index.pre.Sij.ind) = avar.pre.Sij;
%avar.pre.allvariables(index.pre.Pij.ind) = avar.pre.Pij;
%avar.pre.allvariables(index.pre.Qij.ind) = avar.pre.Qij;
%avar.pre.allvariables(index.pre.Sij.ind) = avar.pre.Sij;
%*****
%*****
% POST CONTINGENCY
%*****
%*****
%*****

%*****
% Initialize V
%*****
avar.post.V = data.V_init*ones(1,data.ncont);
avar.post.V = avar.post.V./avar.post.V;
avar.allvariables(index.post.V.ind) = avar.post.V;

%*****
% Initialize Pg, Pl
%*****
%genbuses = find(.gen.numgl>0);
%loadbuses = find(o.load.numgl>0);
avar.post.Pg = data.Pg_init*ones(1,data.ncont);
avar.post.Qg = data.Qg_init*ones(1,data.ncont);
avar.post.Pl = data.Pl_init*ones(1,data.ncont);
avar.post.Ig = data.Pg_init./data.V_init(data.Pgenbuses)*ones(1,data.ncont);
avar.post.gamma = angle(avar.post.Pg-j*avar.post.Qg);

avar.allvariables(index.post.Pg.ind) = avar.post.Pg;
avar.allvariables(index.post.Pl.ind) = avar.post.Pl;
avar.allvariables(index.post.Qg.ind) = avar.post.Qg;
avar.allvariables(index.post.Ig.ind) = avar.post.Ig;
avar.allvariables(index.post.gamma.ind) = avar.post.gamma;

%*****
% Initialize lambda.P, lambda.Q,
%*****
%avar.post.eq.lagrange.P = .2*(1:data.nbuses)''*ones(1,data.ncont);
%avar.post.eq.lagrange.Q = .5*(1:data.nbuses)''*ones(1,data.ncont);
avar.post.eq.lagrange.P = -.01*120*.5*(ones(1,data.nbuses)')*ones(1,data.ncont);
avar.post.eq.lagrange.Q = -.01*.5*(ones(1,data.nbuses)')*ones(1,data.ncont);
avar.post.eq.lagrange.Pij= 0*.5*(1:data.nlines-1)''*ones(1,data.ncont);
avar.post.eq.lagrange.Qij= 0*.5*(1:data.nlines-1)''*ones(1,data.ncont);
avar.post.eq.lagrange.Sij= 0*.5*(1:data.nlines-1)''*ones(1,data.ncont);
avar.post.eq.lagrange.Ig_real= -.01*10*.5*(1:data.nPgen)''*ones(1,data.ncont);
avar.post.eq.lagrange.Ig_imag= .01*10*.5*(1:data.nPgen)''*ones(1,data.ncont);

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

avar.post.eq.lagrange.delta_real= ones(data.nPgen,1)*ones(1,data.ncont);
avar.post.eq.lagrange.delta_imag= ones(data.nPgen,1)*ones(1,data.ncont);
avar.post.eq.lagrange.Idq_real= 0*ones(data.nPgen,1)*ones(1,data.ncont);
avar.post.eq.lagrange.Idq_imag= 0*ones(data.nPgen,1)*ones(1,data.ncont);
avar.post.eq.lagrange.Vdq_real= 0*ones(data.nPgen,1)*ones(1,data.ncont);
avar.post.eq.lagrange.Vdq_imag= 0*ones(data.nPgen,1)*ones(1,data.ncont);
avar.post.eq.lagrange.Edp= 0*ones(data.nPgen,1)*ones(1,data.ncont);
avar.post.eq.lagrange.Eqp= 0*ones(data.nPgen,1)*ones(1,data.ncont);
avar.post.eq.lagrange.Efd= 0*ones(data.nPgen,1)*ones(1,data.ncont);
avar.post.eq.lagrange.slack= .5*ones(1,data.ncont);

for k=1:data.ncont
post_lagrange_ind = index.post.eq.lagrange.ind(k,:);
avar.allvariables(post_lagrange_ind(index.post.eq.P.ind)) = ...
    avar.post.eq.lagrange.P(:,k);
avar.allvariables(post_lagrange_ind(index.post.eq.Q.ind)) = ...
    avar.post.eq.lagrange.Q(:,k);
avar.allvariables(post_lagrange_ind(index.post.eq.Pij.ind)) = ...
    avar.post.eq.lagrange.Pij(:,k);
avar.allvariables(post_lagrange_ind(index.post.eq.Qij.ind)) = ...
    avar.post.eq.lagrange.Qij(:,k);
avar.allvariables(post_lagrange_ind(index.post.eq.Sij.ind)) = ...
    avar.post.eq.lagrange.Sij(:,k);
avar.allvariables(post_lagrange_ind(index.post.eq.Ig_real.ind)) = ...
    avar.post.eq.lagrange.Ig_real(:,k);
avar.allvariables(post_lagrange_ind(index.post.eq.Ig_imag.ind)) = ...
    avar.post.eq.lagrange.Ig_imag(:,k);
avar.allvariables(post_lagrange_ind(index.post.eq.delta_real.ind)) = ...
    avar.post.eq.lagrange.delta_real(:,k);
avar.allvariables(post_lagrange_ind(index.post.eq.delta_imag.ind)) = ...
    avar.post.eq.lagrange.delta_imag(:,k);
avar.allvariables(post_lagrange_ind(index.post.eq.Idq_real.ind)) = ...
    avar.post.eq.lagrange.Idq_real(:,k);
avar.allvariables(post_lagrange_ind(index.post.eq.Idq_imag.ind)) = ...
    avar.post.eq.lagrange.Idq_imag(:,k);
avar.allvariables(post_lagrange_ind(index.post.eq.Vdq_real.ind)) = ...
    avar.post.eq.lagrange.Vdq_real(:,k);
avar.allvariables(post_lagrange_ind(index.post.eq.Vdq_imag.ind)) = ...
    avar.post.eq.lagrange.Vdq_imag(:,k);
avar.allvariables(post_lagrange_ind(index.post.eq.Edp.ind)) = ...
    avar.post.eq.lagrange.Edp(:,k);
avar.allvariables(post_lagrange_ind(index.post.eq.Eqp.ind)) = ...
    avar.post.eq.lagrange.Eqp(:,k);
avar.allvariables(post_lagrange_ind(index.post.eq.Efd.ind)) = ...
    avar.post.eq.lagrange.Efd(:,k);
avar.allvariables(post_lagrange_ind(index.post.eq.slack.ind))= ...
    avar.post.eq.lagrange.slack(:,k);
avar.eq.lagrange.total = [avar.eq.lagrange.total; avar.allvariables(index.post.eq.lagrange.ind(k,:))'];
end%for

%*****
% Initialize theta
%*****
avar.post.theta = data.theta_init*ones(1,data.ncont);
avar.post.Adelt = ...
    (abs(avar.pre.V(data.Pgenbuses).*exp(j*avar.pre.theta(data.Pgenbuses)) ...
        +j*data.machine.Xq.*avar.pre.Ig.*exp(j*avar.pre.gamma))) ...
    *ones(1,data.ncont);
avar.post.delta = ...
    (angle(avar.pre.V(data.Pgenbuses).*exp(j*avar.pre.theta(data.Pgenbuses)) ...
        +j*data.machine.Xq.*avar.pre.Ig.*exp(j*avar.pre.gamma))) ...
    *ones(1,data.ncont);
avar.post.Id = real(j*avar.pre.Ig ...
    .*exp(j*avar.pre.gamma).*exp(-j*avar.pre.delta))*ones(1,data.ncont);
avar.post.Iq = imag(j*avar.pre.Ig ...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

        .*exp(j*avar.pre.gamma).*exp(-j*avar.pre.delta))*ones(1,data.ncont);
avar.post.Vd = real(j*avar.pre.V(data.Pgenbuses) ...
        .*exp(j*avar.pre.theta(data.Pgenbuses)).*exp(-j*avar.pre.delta)) ...
        *ones(1,data.ncont);
avar.post.Vq = imag(j*avar.pre.V(data.Pgenbuses) ...
        .*exp(j*avar.pre.theta(data.Pgenbuses)).*exp(-j*avar.pre.delta)) ...
        *ones(1,data.ncont);
avar.post.Edp = (avar.pre.Vd + ...
        data.machine.Ra.*avar.pre.Id - data.machine.Xqp.*avar.pre.Iq)*ones(1,data.ncont);
avar.post.Eqp = (avar.pre.Vq + ...
        data.machine.Ra.*avar.pre.Iq + data.machine.Xdp.*avar.pre.Id)*ones(1,data.ncont);
avar.post.Efd = (avar.pre.Eqp + (data.machine.Xd-data.machine.Xdp).*avar.pre.Id)*ones(1,data.ncont);

avar.allvariables(index.post.theta.ind) = avar.post.theta;
avar.allvariables(index.post.Adelt.ind) = avar.post.Adelt;
avar.allvariables(index.post.delta.ind) = avar.post.delta;
avar.allvariables(index.post.Id.ind) = avar.post.Id;
avar.allvariables(index.post.Iq.ind) = avar.post.Iq;
avar.allvariables(index.post.Vd.ind) = avar.post.Vd;
avar.allvariables(index.post.Vq.ind) = avar.post.Vq;
avar.allvariables(index.post.Edp.ind) = avar.post.Edp;
avar.allvariables(index.post.Eqp.ind) = avar.post.Eqp;
avar.allvariables(index.post.Efd.ind) = avar.post.Efd;

%*****
% Initialize inequality multipliers(mu) and slacks(s)
%*****
avar.ineq.lagrange = ones(index.nineq,1);
avar.ineq.slack = 1.2*ones(index.nineq,1);
avar.ineq.g = zeros(index.nineq,1);

avar.allvariables(index.ineq.lagrange.ind) = avar.ineq.lagrange;
avar.allvariables(index.ineq.slack.ind) = avar.ineq.slack;

avar.ineq.Slack = sparse(diag(avar.ineq.slack));
avar.ineq.Lagrange = sparse(diag(avar.ineq.lagrange));
%avar.ineq.g(:) = -10;

%*****
% Calculate line flows
%*****
avar.post.Pij = zeros(data.nlines-1,1)*ones(1,data.ncont);
avar.post.Qij = zeros(data.nlines-1,1)*ones(1,data.ncont);
avar.post.Sij = zeros(data.nlines-1,1)*ones(1,data.ncont);
avar.post.Iij = zeros(data.nlines-1,1)*ones(1,data.ncont);

%avar.post.allvariables(index.post.Pij.ind) = avar.post.Pij;
%avar.post.allvariables(index.post.Qij.ind) = avar.post.Qij;
%avar.post.allvariables(index.post.Sij.ind) = avar.post.Sij;
avar.allvariables(index.post.Pij.ind) = avar.post.Pij;
avar.allvariables(index.post.Qij.ind) = avar.post.Qij;
avar.allvariables(index.post.Sij.ind) = avar.post.Sij;
%avar.post.allvariables = avar.post.allvariables';
%avar.post.allvariables = avar.post.allvariables';
avar.allvariables = avar.allvariables';

```

H.10 pdip.m

```

function soln = pdip(var,data,ind)

%*****
% Initialize barrier parameter nu and closeness parameter chi
%*****
nu = 20;
chi = 0.02;
%Sigma = 0.45;
%nufact = 0.60;
Sigma = 0.75;
nufact = 0.60;
dy=15;

%*****
% zero and identity portions of W matrix
%*****
zero14 = sparse(ind.Y_len,ind.nineq);
zero32 = sparse(ind.nineq,ind.neq);
zero33 = sparse(ind.nineq,ind.nineq);
zero22 = sparse(ind.neq,ind.neq);
zero23 = zero32';
zero24 = zero23;
zero41 = zero14';
zero42 = zero32;
I34 = speye(ind.nineq,ind.nineq);
%*****
%var.ineq.lagrange = 1./var.ineq.slack*nu;
%var.ineq.Lagrange = diag(var.ineq.lagrange);
%var.allvariables(ind.ineq.lagrange,ind) = var.ineq.lagrange;
%*****
% Solve
%*****
oldx=ones(ind.Y_len+ind.neq+2*ind.nineq,1);
xangle=0;
dv=0;
fid=fopen('iterations.txt','w');
alpha_min=1;
total_iter = 0;
iteration=0;
while nu > 1e-08
    first=1;
    while first>0 | (norm(var.ineq.Lagrange*var.ineq.slack ...
        -var.ineq.lagrange'*var.ineq.slack/ind.nineq)>chi*nu ) ...
        | (norm(dy)>1e-3 & nu<1e-08)
        %
        if nu<0.5
            Sigma = 0.98;
            nu=1e-9;
        end%if
        total_iter = total_iter+1;
        iteration=iteration+1;
        fprintf('iter=%3i normdv= %.3e nu=%.3e',total_iter,norm(dv),nu)
        fprintf('angle= %4.1f normdy=%.3e alph=%1.1f\n',xangle,norm(dy),alpha_min)
        fprintf(fid,'iter=%3i normdv= %.3e nu=%.3e',total_iter,norm(dv),nu);
        fprintf(fid,'angle= %5.1f normdy=%.3e alph=%1.1f\n',xangle,norm(dy),alpha_min);
        first = 0;
        %*****
        var.pre.eigen = sysmat(var,data,ind,0);
        var.pre.eigenvalues = diag(var.pre.eigen.Lambda);
    %
    var.pre.eigenvalues
    for k=1:data.ncont
        var.post.eigen{k} = sysmat(var,data,ind,k);
        var.post.eigenvalues(:,k) = diag(var.post.eigen{k}.Lambda);
    end%for

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

%*****
h = eq_eval(data,var,ind);
g = ineq_eval(data,var,ind);
%*****
deriv.J_h = eq_jacobian(data,var,ind);
deriv.J_g = ineq_jacobian(data,var,ind);
deriv.J_f = objective_jacobian(data,var,ind);
%*****
deriv.H_h = eq_hessian(data,var,ind);
deriv.H_g = ineq_hessian(data,var,ind);
deriv.H_f = objective_hessian(data,var,ind);
%*****
[deriv.J_g, deriv.H_g] = eig_jac_hess(data,ind,var,deriv.J_g,deriv.H_g);
%*****
M=[deriv.H_h+deriv.H_f+deriv.H_g  deriv.J_h'  deriv.J_g'      zero14
  deriv.J_h                    zero22  zero23      zero24
  deriv.J_g                    zero32  zero33      I34
  zero41                       zero42  var.ineq.Slack  var.ineq.Lagrange      ];
dy=[-deriv.J_f-deriv.J_h'*var.eq.lagrange.total-deriv.J_g'*var.ineq.lagrange
    -h
    -g-var.ineq.slack
    nu*ones(ind.nineq,1)-var.ineq.lagrange.*var.ineq.slack ];
dv = M\dy;
%*****
dw = dv(ind.ineq.lagrange.ind);
ds = dv(ind.ineq.slack.ind);
%*****
ind_s_neg=find(ds<0);
alpha_s=min([1; -var.ineq.slack(ind_s_neg)./ds(ind_s_neg)]);
ind_w=find(dw<0);
alpha_w=min([1; -var.ineq.lagrange(ind_w)./dw(ind_w)]);
alpha_min=min(alpha_s,alpha_w);
alpha = alpha_min;
%   alpha_s = alpha;
%   alpha_w = alpha;
%*****
%   var.allvariables = var.allvariables + alpha*Sigma*dv;
var.allvariables(1:ind.Y_len) = var.allvariables(1:ind.Y_len) ...
    + alpha_s*Sigma*dv(1:ind.Y_len);
var.allvariables(ind.pre.eq.lagrange.ind) = ...
    var.allvariables(ind.pre.eq.lagrange.ind) ...
    + alpha_s*Sigma*dv(ind.pre.eq.lagrange.ind);
var.allvariables(ind.post.eq.lagrange.ind) = ...
    var.allvariables(ind.post.eq.lagrange.ind) ...
    + alpha_s*Sigma*dv(ind.post.eq.lagrange.ind);
var.allvariables(ind.ineq.lagrange.ind) = ...
    var.allvariables(ind.ineq.lagrange.ind) ...
    + alpha_w*Sigma*dv(ind.ineq.lagrange.ind);
var.allvariables(ind.ineq.slack.ind) = ...
    var.allvariables(ind.ineq.slack.ind) ...
    + alpha_s*Sigma*dv(ind.ineq.slack.ind);
var = update(data,var,ind);
%*****
xangle = 180/pi*acos(alpha_s*dv.'*oldx/norm(alpha_s*dv)/norm(oldx));
oldx = alpha_s*dv;
%*****
%   Cost = var.Pg'.^2*data.gen.gamma(data.Pgenbuses)+ ...
%   var.Pg'*data.gen.beta(data.Pgenbuses)+sum(data.gen.alpha)- ...
%   var.Pl'.^2*data.load.gamma(data.Ploadbuses)- ...
%   var.Pl'*data.load.beta(data.Ploadbuses)-sum(data.load.alpha);
%   SW = -Cost;

end%while
nu = nufact*nu;

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```
end%while
%*****
var.pre.eigen = sysmat(var,data,ind,0);
var.pre.eigenvalues = diag(var.pre.eigen.Lambda);
for k=1:data.ncont
var.post.eigen{k} = sysmat(var,data,ind,k);
var.post.eigenvalues(:,k) = diag(var.post.eigen{k}.Lambda);
end%for
%*****
var.SW = objective_eval(var,data,ind);
keyboard
```


H.11 sysmat.m

```

function x=sysmat(var,data,ind,k)

current = getstate_before_eigen(data,ind,var,k);
f=60;
ws=1;
H = data.machine.H;
KD = data.machine.KD;
KA = data.machine.KA;
TA = data.machine.TA;
KE = data.machine.KE;
TE = data.machine.TE;
KF = data.machine.KF;
TF = data.machine.TF;
Xd = data.machine.Xd;
Xdp = data.machine.Xdp;
Xq = data.machine.Xq;
Xqp = data.machine.Xqp;
D = data.machine.D;
XL = data.machine.XL;
Tdop = data.machine.Tdop;
Tqop = data.machine.Tqop;
Ra = data.machine.Ra;
%Lad = Ld-LL;
%Lads = Lad;
%Laq = Lq-LL;
%Laqs = Laq;
%Lfd = ( Lad.*(a.machine.Xdp-LL) ) ./ ( Lad - a.machine.Xdp + LL );
%Ladsp = 1./(1./Lads+1./Lfd);
%Rfd = ( Lad + Lfd ) ./ ( 2*pi*f*a.machine.Tdop );
ngen = data.nPgen;
gen_bus = data.Pgenbuses;
not_gen_bus = 1:data.nbuses;
not_gen_bus(gen_bus) = [];
Ybus = abs(current.data.Ybus);
alpha = angle(current.data.Ybus);

%-----
% Determine inital operating point
%-----
Vp = current.var.V.*exp(j*current.var.theta);
Ip = current.data.Ybus * Vp;
Sp = Vp.*conj(Ip);
theta = angle(Vp);
I = abs(Ip);
phi = angle(Ip);
S = Vp .* conj(Ip);
P = real(S);
Q = imag(S);
Sgen = current.var.Pg+j*current.var.Qg;
Etp = Vp(data.Pgenbuses);
Et = current.var.V(data.Pgenbuses);

%-----
% Calculate Generator Currents
%-----
%IG = conj(Sgen./Etp);
IG = current.var.Ig.*exp(j*current.var.gamma);

%-----
% Calculate deltas
%-----
delta = current.var.delta;

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

%-----
% Calculate Id, Iq, Vd, Vq
%-----
Id = current.var.Id;
Iq = current.var.Iq;
Vd = current.var.Vd;
Vq = current.var.Vq;
Ig = zeros(data.nbuses,1);
Ig(data.Pgenbuses) = Id+j*Iq;

%-----
% Calculate Edp,Eqp,Efd,VR,Rf,Vref,TM
%-----
Edp = current.var.Edp;
Eqp = current.var.Eqp;
Efd = current.var.Efd;
%VR = current.var.VR;
VR = (KE+sefun(current.var.Efd)).*current.var.Efd;
dS_dEfd = 1.555*0.0039*exp(1.555*Efd);
SE = 0.0039*exp(1.555*Efd);
fsi = (KE+Efd.*dS_dEfd+SE)./TE;
%Rf = current.var.Rf;
Rf = (KF./TF).*current.var.Efd;
Vref = Etp+VR./KA;
TM = Edp.*Id+Eqp.*Iq+(Xqp-Xdp).*Id.*Iq;
M = 2*H/ws;
M = 2*H/2/pi/60;
%Tdop = Tdop./60;
%Tqop = Tqop./60;

%-----
% Calculate Edp,Eqp,Efd,VR,Rf,Vref,TM
%-----
nmachines = data.nPgen;
B2 = zeros(7*nmachines,2*data.nPgen);
for k=1:nmachines
A1k = zeros(7,7);
A1k(1,2) = 1;
A1k(2,2) = -D(k)/M(k);
A1k(2,3) = -Iq(k)/M(k);
A1k(2,4) = -Id(k)/M(k);
A1k(3,3) = -1/Tdop(k);
A1k(3,5) = 1/Tdop(k);
A1k(4,4) = -1/Tqop(k);
A1k(5,5) = -fsi(k);
A1k(5,6) = 1/TE(k);
A1k(6,5) = -KA(k)*KF(k)/TA(k)/TF(k);
A1k(6,6) = -1/TA(k);
A1k(6,7) = KA(k)/TA(k);
A1k(7,5) = KF(k)/TF(k)^2;
A1k(7,7) = -1/TF(k);

B1k = zeros(7,2);
B1k(2,1) = (Iq(k)*(Xdp(k)-Xqp(k))-Edp(k))/M(k);
B1k(2,2) = (Id(k)*(Xdp(k)-Xqp(k))-Eqp(k))/M(k);
B1k(3,1) = -(Xd(k)-Xdp(k))/Tdop(k);
B1k(4,2) = (Xq(k)-Xqp(k))/Tqop(k);

%B2k = zeros(7,2);
%B2k(6,2) = -KA(k)/TA(k);
B2k = zeros(7,1);
B2k(6,1) = -KA(k)/TA(k);

C1k = zeros(2,7);

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

C1k(1,1) = -abs(Etp(k))*cos(delta(k)-theta(k));
C1k(2,1) = abs(Etp(k))*sin(delta(k)-theta(k));
C1k(1,4) = 1;
C1k(2,3) = 1;

D1k = [-Ra(k) Xqp(k); -Xdp(k) -Ra(k)];

%D2k = [ Et(k)*cos(delta(k)-theta(k))  -sin(delta(k)-theta(k))
%      -Et(k)*sin(delta(k)-theta(k))  -cos(delta(k)-theta(k)) ];
D2kV = [ -sin(delta(k)-theta(k))
         -cos(delta(k)-theta(k)) ];
D2ktheta = [ Et(k)*cos(delta(k)-theta(k))
            -Et(k)*sin(delta(k)-theta(k)) ];

ind_start = (k-1)*7+1;
ind_end = k*7;
%A1((n-1)*7+1:7*n,(n-1)*7+1:7*n) = A1k;
A1(ind_start:ind_end,ind_start:ind_end) = A1k;

B1((k-1)*7+1:7*k,(k-1)*2+1:2*k) = B1k;
%B2((k-1)*7+1:7*k,(k-1)*2+1:2*k) = B2k;
B2((k-1)*7+1:7*k,k) = B2k;

C1((k-1)*2+1:2*k,(k-1)*7+1:7*k) = C1k;

D1((k-1)*2+1:2*k,(k-1)*2+1:2*k) = D1k;
%D2((k-1)*2+1:2*k,(k-1)*2+1:2*k) = D2k;
D2((k-1)*2+1:2*k,k) = D2kV;
D2((k-1)*2+1:2*k,k+data.nPgen) = D2ktheta;
end%for
A1(9,[3 4]) = [Iq(1)/M(1) Id(1)/M(1) ];
A1(16,[3 4]) = [Iq(1)/M(1) Id(1)/M(1) ];
B1(9,[1 2]) = -[(Iq(1)*(Xdp(1)-Xqp(1))-Edp(1))/M(1) (Id(1)*(Xdp(1)-Xqp(1))-Eqp(1))/M(1)];
B1(16,[1 2]) = -[(Iq(1)*(Xdp(1)-Xqp(1))-Edp(1))/M(1) (Id(1)*(Xdp(1)-Xqp(1))-Eqp(1))/M(1)];
A1(1,:) = [];
A1(:,1) = [];
A1(1,:) = [];
A1(:,1) = [];
B1(1,:) = [];
B1(1,:) = [];
B2(1,:) = [];
B2(1,:) = [];
C1(:,1) = [];
C1(:,1) = [];

D4 = zeros(2*length(Et),2*length(Et));

% Sload = var.P1+j*o.kpf.*var.P1;
%
% Y_L = conj(Sload./(a.op.V).^2);
% Y_L = Y_L-Y_L;

load_bus = 1:data.nbuses;
load_bus(gen_bus) = [];
load_bus_ind = find(load_bus);

D5 = zeros(2*length(Et),2*length(load_bus));
D6 = zeros(2*length(load_bus),2*length(gen_bus));
D7 = zeros(2*length(load_bus),2*length(load_bus));

x.A1 = A1;
x.B1 = B1;
x.B2 = B2;

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

x.C1 = C1;
x.D1 = D1;
x.D2 = D2;

load_bus_V = load_bus_ind*2;
load_bus_theta = load_bus_ind*2-1;

gen_bus_V = gen_bus*2;
gen_bus_theta = gen_bus*2-1;

line_ind_fr = zeros(data.nbuses,data.nlines);
line_ind_to = zeros(data.nbuses,data.nlines);
for k=1:data.nlines
    line_ind_fr(data.fr(k),k) = 1;
    line_ind_to(data.to(k),k) = 1;
end%for

ejTheta = exp(j*theta);
ejDelta = zeros(data.nbuses,1);
ejDelta(gen_bus) = exp(j*(delta-pi/2));
ejdelta = zeros(data.nbuses,1);
ejdelta(gen_bus) = exp(j*(delta));

J_S_V = diag(Vp)*conj(current.data.Ybus)*conj(diag(ejTheta))+ ...
    diag(ejTheta.*conj(Ip));
J_S_theta = -j*diag(Vp)*conj(current.data.Ybus)*conj(diag(Vp))+diag(j*Sp);

dnetS_dV = diag(ejTheta.*conj(Ig.*ejDelta)) ...
    - J_S_V;
% - diag(2*a.op.V.*conj(Y_L)) ...
% - J_S_V(gen_bus,:);
dnetS_dtheta = diag(j*Vp.*conj(Ig.*ejDelta)) ...
    - J_S_theta;
% - J_S_theta(gen_bus,gen_bus);

imag_ind = gen_bus_V;
real_ind = gen_bus_theta;
real_ind2 = 1:ngen;
imag_ind2 = (1:ngen)+ngen;
load_bus2 = 1:length(load_bus);
load_bus3 = (1:length(load_bus))+length(load_bus);

% D4(real_ind,gen_bus_V) = real(dnetS_dV(gen_bus,gen_bus));
% D4(imag_ind,gen_bus_V) = imag(dnetS_dV(gen_bus,gen_bus));
% D4(real_ind,gen_bus_theta) = real(dnetS_dtheta(gen_bus,gen_bus));
% D4(imag_ind,gen_bus_theta) = imag(dnetS_dtheta(gen_bus,gen_bus));
%
% D5(real_ind,load_bus_V) = real(dnetS_dV(gen_bus,load_bus));
% D5(imag_ind,load_bus_V) = imag(dnetS_dV(gen_bus,load_bus));
% D5(real_ind,load_bus_theta) = real(dnetS_dtheta(gen_bus,load_bus));
% D5(imag_ind,load_bus_theta) = imag(dnetS_dtheta(gen_bus,load_bus));

D4(real_ind2,1:ngen) = real(dnetS_dV(gen_bus,gen_bus));
D4(imag_ind2,1:ngen) = imag(dnetS_dV(gen_bus,gen_bus));
D4(real_ind2,(1:ngen)+ngen) = real(dnetS_dtheta(gen_bus,gen_bus));
D4(imag_ind2,(1:ngen)+ngen) = imag(dnetS_dtheta(gen_bus,gen_bus));

D5(real_ind2,load_bus2) = real(dnetS_dV(gen_bus,load_bus));
D5(imag_ind2,load_bus2) = imag(dnetS_dV(gen_bus,load_bus));
D5(real_ind2,load_bus3) =real(dnetS_dtheta(gen_bus,load_bus));
D5(imag_ind2,load_bus3) =imag(dnetS_dtheta(gen_bus,load_bus));

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

imag_ind = load_bus_V;
real_ind = load_bus_theta;

% D6(real_ind,gen_bus_V) = real(dnetS_dV(load_bus,gen_bus));
% D6(imag_ind,gen_bus_V) = imag(dnetS_dV(load_bus,gen_bus));
% D6(real_ind,gen_bus_theta) = real(dnetS_dtheta(load_bus,gen_bus));
% D6(imag_ind,gen_bus_theta) = imag(dnetS_dtheta(load_bus,gen_bus));
%
% D7(real_ind,load_bus_V) = real(dnetS_dV(load_bus,load_bus));
% D7(imag_ind,load_bus_V) = imag(dnetS_dV(load_bus,load_bus));
% D7(real_ind,load_bus_theta) = real(dnetS_dtheta(load_bus,load_bus));
% D7(imag_ind,load_bus_theta) = imag(dnetS_dtheta(load_bus,load_bus));

real_ind2 = 1:length(load_bus);
imag_ind2 = (1:length(load_bus))+length(load_bus);
D6(real_ind2,1:data.nPgen) = real(dnetS_dV(load_bus,gen_bus));
D6(imag_ind2,1:data.nPgen) = imag(dnetS_dV(load_bus,gen_bus));
D6(real_ind2,(1:data.nPgen)+data.nPgen) = real(dnetS_dtheta(load_bus,gen_bus));
D6(imag_ind2,(1:data.nPgen)+data.nPgen) = imag(dnetS_dtheta(load_bus,gen_bus));

D7(real_ind2,load_bus2) = real(dnetS_dV(load_bus,load_bus));
D7(imag_ind2,load_bus2) = imag(dnetS_dV(load_bus,load_bus));
D7(real_ind2,load_bus3) = real(dnetS_dtheta(load_bus,load_bus));
D7(imag_ind2,load_bus3) = imag(dnetS_dtheta(load_bus,load_bus));
x.D4 = D4;
x.D5 = D5;
x.D6 = D6;
x.D7 = D7;

C2 = zeros(2*nmachines,7*nmachines);
D3 = zeros(2*nmachines,2*nmachines);
delta_ind = 1:7:(nmachines-1)*7+1;
imag_ind = 2*gen_bus;
real_ind = 2*gen_bus-1;
real_ind2 = 1:nngen;
imag_ind2 = real_ind2+nngen;
%C2(real_ind,delta_ind) = diag(real(Vp(gen_bus).*conj(j*Ig(gen_bus).*ejDelta(gen_bus))));
C2(real_ind2,delta_ind) = diag(real(Vp(gen_bus).*conj(Ig(gen_bus).*ejdelta(gen_bus))));
%C2(imag_ind,delta_ind) = diag(imag(Vp(gen_bus).*conj(j*Ig(gen_bus).*ejDelta(gen_bus))));
C2(imag_ind2,delta_ind) = diag(imag(Vp(gen_bus).*conj(Ig(gen_bus).*ejdelta(gen_bus))));
%D3(real_ind,real_ind) = diag(real(Vp(gen_bus).*conj(ejDelta(gen_bus))));
%D3(real_ind,real_ind) = diag(real(Vp(gen_bus).*conj(ejDelta(gen_bus))));
%D3(real_ind,imag_ind) = diag(real(Vp(gen_bus).*conj(j*ejDelta(gen_bus))));
%D3(imag_ind,real_ind) = diag(imag(Vp(gen_bus).*conj(ejDelta(gen_bus))));
%D3(imag_ind,imag_ind) = diag(imag(Vp(gen_bus).*conj(j*ejDelta(gen_bus))));
D3(real_ind2,real_ind) = diag(real(j*Vp(gen_bus).*conj(ejdelta(gen_bus))));
D3(real_ind2,imag_ind) = diag(real(Vp(gen_bus).*conj(ejdelta(gen_bus))));
D3(imag_ind2,real_ind) = diag(imag(j*Vp(gen_bus).*conj(ejdelta(gen_bus))));
D3(imag_ind2,imag_ind) = diag(imag(Vp(gen_bus).*conj(ejdelta(gen_bus))));

C2(:,1) = [];
C2(:,1) = [];

K1 = D4-D3*(D1\D2);
K2 = C2-D3*(D1\C1);

K1 = D4-D3/D1*D2;
K2 = C2-D3/D1*C1;

%Asys = A1-B1*(D1\C1)+(B2-B1*(D1\D2))*((D5*(D7\D6)-K1)\K2);
%Asys = A1-B1*inv(D1)*C1-(B2-B1*inv(D1)*D2)*(inv(D4-D3*inv(D1)*D2-D5*inv(D7)*D6)*(C2-D3*inv(D1)*C1));
%Asys = A1-[B1 B2]*inv([D1 D2; D3 D4-D5*inv(D7)*D6])*[C1;C2];

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% x.Asys = A1-[B1 B2]*([D1 D2; D3 D4-D5*(D7\D6)]\[C1;C2]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[rowB2,q] = size(B2);
[q,columnD5] = size(D5);
[rowD6,q] = size(D6);
[q,columnC2] = size(C2);
zerok3 = zeros(rowB2,columnD5);
zerok5 = zeros(rowD6,columnC2);
K3 = [B2-B1/D1*D2 zerok3];
K4 = [K1      D5
      D6      D7];
K5 = [K2; zerok5];
x.K3 = K3;
x.K4 = K4;
x.K5 = K5;
x.D3 = D3;
x.zerok5 = zerok5;
x.zerok3 = zerok3;
x.zeroD5 = sparse(zeros(size(x.D5)));
x.zeroD6 = sparse(zeros(size(x.D6)));
x.zeroD7 = sparse(zeros(size(x.D7)));
x.Asys = A1-B1/D1*C1-K3/K4*K5;
[x.phi,x.Lambda] = eig(x.Asys);
[x.phi,x.Lambda] = eigs(x.Asys,data.neig_const,'lr');
opts.disp=data.neig_const;
opts.disp=0;
[x.phi,x.Lambda] = eigs(x.Asys,data.neig_const,'lr',opts);
lambda=diag(x.Lambda);
lambda2 = abs(lambda);
[lambda2,Ind] = sort(lambda2);
lambda = lambda(Ind);
x.phi = x.phi(:,Ind);
%lambdarealind = find(abs(imag(lambda))<1e-3);
%lambdaimagind = find(abs(imag(lambda))>1e-3);
%lambdareal = lambda(lambdarealind);
%phireal = x.phi(:,lambdarealind);
%lambdarealabs = abs(lambdareal);
%[nothing,rInd] = sort(lambdarealabs);
%lambda = [lambdareal(rInd); lambda(lambdaimagind)];
x.Lambda = diag(lambda);
x.values = lambda;
%x.phi = [phireal(:,rInd) x.phi(:,lambdaimagind)];
%o.eig = lambda;

```

H.12 eq_eval.m

```

function h=eq_eval(data,var,index)

h = zeros(index.neq,1);
for k=0:data.ncont
current = getstate(data,index,var,k);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Bus power injection
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Vp = current.var.V .* exp(j*current.var.theta);
Ip = current.data.Ybus*Vp;
S = Vp.*conj(Ip);
Peq = full(sparse(data.Pgenbuses,1,current.var.Pg,data.nbuses,1)) ...
      -full(sparse(data.Ploadbuses,1,current.var.Pl,data.nbuses,1)) ...
      -real(S);
Qeq = full(sparse(data.Qgenbuses,1,current.var.Qg,data.nbuses,1)) ...
      -full(sparse(data.Ploadbuses,1,data.kpf.*current.var.Pl,data.nbuses,1)) ...
      -imag(S);
Ig = current.var.Ig.*exp(j*current.var.gamma);
Adelt = current.var.Adelt.*exp(j*current.var.delta);
ejdelta = exp(j*current.var.delta);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Line flows
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Sij = Vp(current.data.fr).*conj(current.var.Iij);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Equality constraint vector
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
h(current.ind.eq.P.ind) = Peq;
h(current.ind.eq.Q.ind) = Qeq;
h(current.ind.eq.Pij.ind) = current.var.Pij - real(Sij);
h(current.ind.eq.Qij.ind) = current.var.Qij - imag(Sij);
h(current.ind.eq.Sij.ind) = current.var.Sij ...
      - abs(current.var.Pij+j*current.var.Qij).^2;
h(current.ind.eq.Ig_real.ind) = real(Vp(data.Pgenbuses).*conj(Ig)) ...
      -current.var.Pg;
h(current.ind.eq.Ig_imag.ind) = imag(Vp(data.Pgenbuses).*conj(Ig)) ...
      -current.var.Qg;
h(current.ind.eq.delta_real.ind) = real(Vp(data.Pgenbuses) ...
      +(data.machine.Ra+j*data.machine.Xq).*Ig - Adelt);
h(current.ind.eq.delta_imag.ind) = imag(Vp(data.Pgenbuses) ...
      +(data.machine.Ra+j*data.machine.Xq).*Ig - Adelt);
h(current.ind.eq.Idq_real.ind) = current.var.Id-real(j*Ig.*conj(ejdelta));
h(current.ind.eq.Idq_imag.ind) = current.var.Iq-imag(j*Ig.*conj(ejdelta));
h(current.ind.eq.Vdq_real.ind) = ...
      current.var.Vd-real(j*Vp(data.Pgenbuses).*conj(ejdelta));
h(current.ind.eq.Vdq_imag.ind) = ...
      current.var.Vq-imag(j*Vp(data.Pgenbuses).*conj(ejdelta));
h(current.ind.eq.Edp.ind) = -current.var.Edp+current.var.Vd ...
      +data.machine.Ra.*current.var.Id ...
      +data.machine.Xqp.*current.var.Iq;
h(current.ind.eq.Eqp.ind) = -current.var.Eqp+current.var.Vq ...
      +data.machine.Ra.*current.var.Iq ...
      +data.machine.Xdp.*current.var.Id;
h(current.ind.eq.Efd.ind) = current.var.Eqp ...
      +(data.machine.Xd-data.machine.Xdp).*current.var.Id ...
      -current.var.Efd;
h(current.ind.eq.slack.ind)= current.var.theta(data.slackbus);
end%for

```

H.13 ineq_eval.m

```

function g = ineq_eval(data,var,index)

g = zeros(data.ncont,1);
for k=0:data.ncont
current = getstate(data,index,var,k);
g(current.ind.ineq.Pgmax.ind) = current.var.Pg-current.data.limits.Pgmax;
g(current.ind.ineq.Pgmin.ind) = -current.var.Pg+current.data.limits.Pgmin;
g(current.ind.ineq.Plmin.ind) = -current.var.Pl+current.data.limits.Plmin;
g(current.ind.ineq.Qgmax.ind) = current.var.Qg-current.data.limits.Qgmax;
g(current.ind.ineq.Qgmin.ind) = -current.var.Qg+current.data.limits.Qgmin;
g(current.ind.ineq.Vmax.ind)= current.var.V-current.data.limits.volt_max;
g(current.ind.ineq.Vmin.ind)= -current.var.V+current.data.limits.volt_min;
g(current.ind.ineq.Igmin.ind) = -current.var.Ig;
g(current.ind.ineq.Adeltmin.ind) = -current.var.Adelt;
g(current.ind.ineq.Sijmax.ind)= current.var.Sij-current.data.limits.Sijmax.^2./data.Sbase^2;
g(current.ind.ineq.eig_real.ind) = real(current.var.eigenvalues(data.eig_ind))-data.limits.eig_max;
if k>0
    g(current.ind.ineq.Pgrampup.ind) = current.var.Pg-var.pre.Pg ...
        -data.limits.Pgrampupmax;
    g(current.ind.ineq.Pgrampdwn.ind) = -current.var.Pg+var.pre.Pg ...
        -data.limits.Pgrampdwnmax;
    g(current.ind.ineq.Plmax.ind) = current.var.Pl-var.pre.Pl;
end%if
end%for

```


H.14 eq_jacobian.m

```

function J_h=eq_jacobian(data,var,index)

J_h = full(sparse(index.neq,index.Y_len));

for k=0:data.ncont
current = getstate(data,index,var,k);

ejTheta = exp(j*current.var.theta);
ejgamma = exp(j*current.var.gamma);
ejdelta = exp(j*current.var.delta);
Vp = current.var.V.*ejTheta;
Ip = current.data.Ybus*Vp;
Sp = Vp.*conj(Ip);
Ig = current.var.Ig.*exp(j*current.var.gamma);
Adelt = current.var.Adelt.*exp(j*current.var.delta);

line_ind_fr = ...
    sparse(current.data.fr,1:current.data.nlines, ...
        ones(current.data.nlines,1),current.data.nbuses,current.data.nlines);
line_ind_to = ...
    sparse(current.data.to,1:current.data.nlines, ...
        ones(current.data.nlines,1),current.data.nbuses,current.data.nlines);

grad_Svec_V =line_ind_fr*diag(ejTheta(current.data.fr))*diag(conj(current.var.Iij))+ ...
    line_ind_fr*diag(conj(current.data.Yvec+ ...
        j*current.data.Bvec/2))*diag(current.var.V(current.data.fr))- ...
    line_ind_to*diag(Vp(current.data.fr))*conj(diag(ejTheta(current.data.to))*diag(current.data.Yvec));
grad_Ivec_V = ...
    line_ind_fr*diag(current.data.Yvec+j*current.data.Bvec/2)*diag(ejTheta(current.data.fr)) - ...
    line_ind_to*diag(ejTheta(current.data.to))*diag(current.data.Yvec);
grad_Svec_theta = j*line_ind_fr*diag(Vp(current.data.fr))*diag(conj(current.var.Iij))- ...
    j*line_ind_fr*diag(conj(current.data.Yvec+ ...
        j*current.data.Bvec/2))*diag(current.var.V(current.data.fr).^2)+ ...
    j*line_ind_to*diag(Vp(current.data.fr))*diag(conj(Vp(current.data.to)))* ...
        conj(diag(current.data.Yvec));
%        conj(diag(ejTheta(current.data.to))*diag(current.data.Yvec));
grad_Ivec_theta = ...
    j*line_ind_fr*diag(Vp(current.data.fr))*diag(current.data.Yvec+j*current.data.Bvec/2) - ...
    j*line_ind_to*diag(Vp(current.data.to))*diag(current.data.Yvec);

grad_Svec = [grad_Svec_V; grad_Svec_theta];

grad_Svec_real = real(grad_Svec);
grad_Svec_imag = imag(grad_Svec);

J_S_V = diag(Vp)*conj(current.data.Ybus)*conj(diag(ejTheta))+ ...
    diag(ejTheta.*conj(Ip));
J_S_theta = -j*diag(Vp)*conj(current.data.Ybus)*conj(diag(Vp))+diag(j*Sp);
J_Y = [real(J_S_V) real(J_S_theta);
    imag(J_S_V) imag(J_S_theta)];

%*****
% Ig Jacobian
%*****
dh.dIg_real.dPg = -eye(current.data.nPgen);
dh.dIg_imag.dQg = -eye(current.data.nPgen);
dh.dIg.dV = diag(ejTheta(current.data.Pgenbuses).*conj(Ig));
dh.dIg.dtheta = diag(j*Vp(current.data.Pgenbuses).*conj(Ig));
dh.dIg.dIg = diag(Vp(current.data.Pgenbuses).*ejTheta(current.data.Pgenbuses).*conj(ejgamma));
dh.dIg.dgamma = diag(-j*Vp(current.data.Pgenbuses).*conj(Ig));

%*****

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

% delta Jacobian
%*****
dh.ddelta.dV = diag(ejTheta(current.data.Pgenbuses));
dh.ddelta.dtheta = diag(j*Vp(current.data.Pgenbuses));
dh.ddelta.dIg = diag((data.machine.Ra+j*data.machine.Xq).*ejgamma);
dh.ddelta.dgamma = diag(j*(data.machine.Ra+j*data.machine.Xq).*Ig);
dh.ddelta.dAdelt = diag(-ejdelta);
dh.ddelta.ddelta = diag(-j*Adelt);

%*****
% Idq Jacobian
%*****
dh.dIdq.dIg = diag(-j*ejgamma.*conj(ejdelta));
dh.dIdq.dgamma = diag(Ig.*conj(ejdelta));
dh.dIdq.ddelta = diag(-Ig.*conj(ejdelta));
dh.dIdq.dId = eye(data.nPgen);
dh.dIdq.dIq = eye(data.nPgen);

%*****
% Vdq Jacobian
%*****
dh.dVdq.dV = diag(-j*ejTheta(data.Pgenbuses).*conj(ejdelta));
dh.dVdq.dtheta = diag(Vp(data.Pgenbuses).*conj(ejdelta));
dh.dVdq.ddelta = diag(-Vp(data.Pgenbuses).*conj(ejdelta));
dh.dVdq.dVd = eye(data.nPgen);
dh.dVdq.dVq = eye(data.nPgen);

%*****
% Edp Jacobian
%*****
dh.dEdp.dId = diag(data.machine.Ra);
dh.dEdp.dIq = diag(-data.machine.Xqp);
dh.dEdp.dVd = eye(data.nPgen);
dh.dEdp.dEdp = -eye(data.nPgen);

%*****
% Eqp Jacobian
%*****
dh.dEqp.dIq = diag(data.machine.Ra);
dh.dEqp.dId = diag(data.machine.Xdp);
dh.dEqp.dVq = eye(data.nPgen);
dh.dEqp.dEqp = -eye(data.nPgen);

%*****
% Efd Jacobian
%*****
dh.dEfd.dId = diag(data.machine.Xd-data.machine.Xdp);
dh.dEfd.dEqp = eye(data.nPgen);
dh.dEfd.dEfd = -eye(data.nPgen);

%-----
% Form J_h
%-----
J_h([current.ind.eq.P.ind current.ind.eq.Q.ind], [current.ind.V.ind current.ind.theta.ind]) = ...
    -J_Y;
J_h(current.ind.eq.Pij.ind, [current.ind.V.ind current.ind.theta.ind]) = -grad_Svec_real.';
J_h(current.ind.eq.Qij.ind, [current.ind.V.ind current.ind.theta.ind]) = -grad_Svec_imag.';
J_h(current.ind.eq.P.ind, current.ind.Pg.ind) = ...
    sparse(current.data.Pgenbuses, 1:current.data.nPgen, ones(current.data.nPgen, 1), ...
        current.data.nbuses, current.data.nPgen);
J_h(current.ind.eq.P.ind, current.ind.Pl.ind) = ...
    -sparse(current.data.Ploadbuses, 1:current.data.nload, ones(current.data.nload, 1), ...
        current.data.nbuses, current.data.nload);
J_h(current.ind.eq.Q.ind, current.ind.Qg.ind) = ...
    sparse(current.data.Qgenbuses, 1:current.data.nQgen, ones(current.data.nQgen, 1), ...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

        current.data.nbuses, current.data.nQgen);
J_h(current.ind.eq.Q.ind, current.ind.Pl.ind) = ...
    -sparse(current.data.Ploadbuses, ...
        1:current.data.nload, current.data.kpf.*ones(current.data.nload,1), ...
        current.data.nbuses, current.data.nload);
J_h(current.ind.eq.Pij.ind, current.ind.Pij.ind) = eye(current.data.nlines);
J_h(current.ind.eq.Sij.ind, current.ind.Pij.ind) = -diag(2*current.var.Pij);
J_h(current.ind.eq.Qij.ind, current.ind.Qij.ind) = eye(current.data.nlines);
J_h(current.ind.eq.Sij.ind, current.ind.Qij.ind) = -diag(2*current.var.Qij);
J_h(current.ind.eq.Sij.ind, current.ind.Sij.ind) = eye(current.data.nlines);
J_h(current.ind.eq.slack.ind, current.ind.theta.ind(current.data.slackbus)) = 1;

%*****
% Ig Jacobian
%*****
J_h(current.ind.eq.Ig_real.ind, current.ind.Pg.ind) = ...
    dh.dIg_real.dPg;
J_h(current.ind.eq.Ig_imag.ind, current.ind.Qg.ind) = ...
    dh.dIg_imag.dQg;
J_h(current.ind.eq.Ig_real.ind, current.ind.V.ind(data.Pgenbuses)) = ...
    real(dh.dIg.dV);
J_h(current.ind.eq.Ig_imag.ind, current.ind.V.ind(data.Pgenbuses)) = ...
    imag(dh.dIg.dV);
J_h(current.ind.eq.Ig_real.ind, current.ind.theta.ind(data.Pgenbuses)) = ...
    real(dh.dIg.dtheta);
J_h(current.ind.eq.Ig_imag.ind, current.ind.theta.ind(data.Pgenbuses)) = ...
    imag(dh.dIg.dtheta);
J_h(current.ind.eq.Ig_real.ind, current.ind.Ig.ind) = ...
    real(dh.dIg.dIg);
J_h(current.ind.eq.Ig_imag.ind, current.ind.Ig.ind) = ...
    imag(dh.dIg.dIg);
J_h(current.ind.eq.Ig_real.ind, current.ind.gamma.ind) = ...
    real(dh.dIg.dgamma);
J_h(current.ind.eq.Ig_imag.ind, current.ind.gamma.ind) = ...
    imag(dh.dIg.dgamma);

%*****
% delta Jacobian
%*****
J_h(current.ind.eq.delta_real.ind, current.ind.V.ind(data.Pgenbuses)) = ...
    real(dh.ddelta.dV);
J_h(current.ind.eq.delta_imag.ind, current.ind.V.ind(data.Pgenbuses)) = ...
    imag(dh.ddelta.dV);
J_h(current.ind.eq.delta_real.ind, current.ind.theta.ind(data.Pgenbuses)) = ...
    real(dh.ddelta.dtheta);
J_h(current.ind.eq.delta_imag.ind, current.ind.theta.ind(data.Pgenbuses)) = ...
    imag(dh.ddelta.dtheta);
J_h(current.ind.eq.delta_real.ind, current.ind.Ig.ind) = ...
    real(dh.ddelta.dIg);
J_h(current.ind.eq.delta_imag.ind, current.ind.Ig.ind) = ...
    imag(dh.ddelta.dIg);
J_h(current.ind.eq.delta_real.ind, current.ind.gamma.ind) = ...
    real(dh.ddelta.dgamma);
J_h(current.ind.eq.delta_imag.ind, current.ind.gamma.ind) = ...
    imag(dh.ddelta.dgamma);
J_h(current.ind.eq.delta_real.ind, current.ind.Adelt.ind) = ...
    real(dh.ddelta.dAdelt);
J_h(current.ind.eq.delta_imag.ind, current.ind.Adelt.ind) = ...
    imag(dh.ddelta.dAdelt);
J_h(current.ind.eq.delta_real.ind, current.ind.delta.ind) = ...
    real(dh.ddelta.ddelta);
J_h(current.ind.eq.delta_imag.ind, current.ind.delta.ind) = ...
    imag(dh.ddelta.ddelta);

%*****

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

% Idq Jacobian
%*****
J_h(current.ind.eq.Idq_real.ind,current.ind.Ig.ind) = ...
    real(dh.dIdq.dIg);
J_h(current.ind.eq.Idq_imag.ind,current.ind.Ig.ind) = ...
    imag(dh.dIdq.dIg);
J_h(current.ind.eq.Idq_real.ind,current.ind.gamma.ind) = ...
    real(dh.dIdq.dgamma);
J_h(current.ind.eq.Idq_imag.ind,current.ind.gamma.ind) = ...
    imag(dh.dIdq.dgamma);
J_h(current.ind.eq.Idq_real.ind,current.ind.Id.ind) = ...
    (dh.dIdq.dId);
J_h(current.ind.eq.Idq_imag.ind,current.ind.Iq.ind) = ...
    (dh.dIdq.dIq);
J_h(current.ind.eq.Idq_real.ind,current.ind.delta.ind) = ...
    real(dh.dIdq.ddelta);
J_h(current.ind.eq.Idq_imag.ind,current.ind.delta.ind) = ...
    imag(dh.dIdq.ddelta);

%*****
% Vdq Jacobian
%*****
J_h(current.ind.eq.Vdq_real.ind,current.ind.Vd.ind) = ...
    (dh.dVdq.dVd);
J_h(current.ind.eq.Vdq_imag.ind,current.ind.Vq.ind) = ...
    (dh.dVdq.dVq);
J_h(current.ind.eq.Vdq_real.ind,current.ind.V.ind(data.Pgenbuses)) = ...
    real(dh.dVdq.dV);
J_h(current.ind.eq.Vdq_imag.ind,current.ind.V.ind(data.Pgenbuses)) = ...
    imag(dh.dVdq.dV);
J_h(current.ind.eq.Vdq_real.ind,current.ind.theta.ind(data.Pgenbuses)) = ...
    real(dh.dVdq.dtheta);
J_h(current.ind.eq.Vdq_imag.ind,current.ind.theta.ind(data.Pgenbuses)) = ...
    imag(dh.dVdq.dtheta);
J_h(current.ind.eq.Vdq_real.ind,current.ind.delta.ind) = ...
    real(dh.dVdq.ddelta);
J_h(current.ind.eq.Vdq_imag.ind,current.ind.delta.ind) = ...
    imag(dh.dVdq.ddelta);

%*****
% Edp Jacobian
%*****
J_h(current.ind.eq.Edp.ind,current.ind.Vd.ind) = ...
    (dh.dEdp.dVd);
J_h(current.ind.eq.Edp.ind,current.ind.Edp.ind) = ...
    (dh.dEdp.dEdp);
J_h(current.ind.eq.Edp.ind,current.ind.Id.ind(data.Pgenbuses)) = ...
    (dh.dEdp.dId);
J_h(current.ind.eq.Edp.ind,current.ind.Iq.ind(data.Pgenbuses)) = ...
    (dh.dEdp.dIq);

%*****
% Eqp Jacobian
%*****
J_h(current.ind.eq.Eqp.ind,current.ind.Vd.ind) = ...
    (dh.dEqp.dVd);
J_h(current.ind.eq.Eqp.ind,current.ind.Eqp.ind) = ...
    (dh.dEqp.dEqp);
J_h(current.ind.eq.Eqp.ind,current.ind.Id.ind(data.Pgenbuses)) = ...
    (dh.dEqp.dId);
J_h(current.ind.eq.Eqp.ind,current.ind.Iq.ind(data.Pgenbuses)) = ...
    (dh.dEqp.dIq);

%*****

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```
% Efd Jacobian
%*****
J_h(current.ind.eq.Efd.ind,current.ind.Id.ind(data.Pgenbuses)) = ...
    (dh.dEfd.dId);
J_h(current.ind.eq.Efd.ind,current.ind.Eqp.ind) = ...
    (dh.dEfd.dEqp);
J_h(current.ind.eq.Efd.ind,current.ind.Efd.ind) = ...
    (dh.dEfd.dEfd);

end%for
```

H.15 ineq_jacobian.m

```

function J_g=ineq_jacobian(data,var,index)

%-----
% Form J_g
%-----
J_g=zeros(index.nineq,index.Y_len);

for k=0:data.ncont
current = getstate(data,index,var,k);

J_g( current.ind.ineq.Pgmax.ind, current.ind.Pg.ind) = eye(current.data.nPgen);
J_g( current.ind.ineq.Pgmin.ind, current.ind.Pg.ind) = -eye(current.data.nPgen);
J_g( current.ind.ineq.Igmin.ind, current.ind.Ig.ind) = -eye(current.data.nPgen);
J_g( current.ind.ineq.Adeltmin.ind, current.ind.Adelt.ind) = -eye(current.data.nPgen);
J_g( current.ind.ineq.Plmin.ind, current.ind.Pl.ind) = -eye(current.data.nload);
J_g( current.ind.ineq.Vmax.ind, current.ind.V.ind) = eye(current.data.nbuses);
J_g( current.ind.ineq.Vmin.ind, current.ind.V.ind) = -eye(current.data.nbuses);

J_g( current.ind.ineq.Qgmax.ind, current.ind.Qg.ind) = eye(current.data.nQgen);
J_g( current.ind.ineq.Qgmin.ind, current.ind.Qg.ind) = -eye(current.data.nQgen);
J_g( current.ind.ineq.Sijmax.ind, current.ind.Sij.ind) = eye(current.data.nlines);

if k>0
J_g( current.ind.ineq.Pgrampup.ind, index.pre.Pg.ind) = -eye(current.data.nPgen);
J_g( current.ind.ineq.Pgrampup.ind, current.ind.Pg.ind) = eye(current.data.nPgen);
J_g( current.ind.ineq.Pgrampdown.ind, index.pre.Pg.ind) = eye(current.data.nPgen);
J_g( current.ind.ineq.Pgrampdown.ind, current.ind.Pg.ind) = -eye(current.data.nPgen);
J_g( current.ind.ineq.Plmax.ind, index.pre.Pl.ind) = -eye(current.data.nload);
J_g( current.ind.ineq.Plmax.ind, current.ind.Pl.ind) = eye(current.data.nload);
end%if

end%for

```

H.16 objective_jacobian.m

```

function J_f = objective_jacobian(data,var,ind)

J_f = zeros(ind.Y_len,1);

J_f(ind.pre.Pg.ind) = ...
    data.line.pi_0*(diag(2*data.gen.gamma(data.Pgenbuses))*var.pre.Pg ...
    +data.gen.beta(data.Pgenbuses));
J_f(ind.pre.Pl.ind) = ...
    -data.line.pi_0*(2*diag(data.load.gamma(data.Ploadbuses))*var.pre.Pl ...
    +data.load.beta(data.Ploadbuses)) ...
    +data.load.beta_int(data.Ploadbuses)*ones(1,data.ncont)*data.line.pi_k;

for k=1:data.ncont
current = getstate(data,ind,var,k);

J_f(current.ind.Pg.ind) = ...
    data.line.pi_k(k)*(diag(2*data.gen.gamma(data.Pgenbuses))*current.var.Pg ...
    +data.gen.beta(data.Pgenbuses));
J_f(current.ind.Pl.ind) = ...
    -data.line.pi_k(k)*(2*diag(data.load.gamma(data.Ploadbuses))*current.var.Pl ...
    +data.load.beta(data.Ploadbuses) ...
    +data.load.beta_int(data.Ploadbuses));

end%for

```

H.17 eq_hessian.m

```

function H_h = eq_hessian(data,var,index)

H_h = zeros(index.Y_len,index.Y_len);

for k=0:data.ncont
current = getstate(data,index,var,k);

%-----
% Compute \sum_{k=1}^n (z_k H_{hk})
% where z_k is equality constraint lagrange multiplier on kth constraint
% and H_{hk} is Hessian matrix for kth equality constraint.
%-----

ejTheta = exp(j*current.var.theta);
ejgamma = exp(j*current.var.gamma);
ejdelta = exp(j*current.var.delta);
Vp = current.var.V .* ejTheta;
Ip = current.data.Ybus * Vp;
Sp = Vp.*conj(Ip);
Sij = Vp(current.data.fr).*conj(current.var.Iij);
Ig = current.var.Ig.*exp(j*current.var.gamma);
Adelt = current.var.Adelt.*exp(j*current.var.delta);

%-----
% Form Hessian for each equality constraint (Pi and Qi constraints).
% Then multiply each by lambda_i and add them together.
% The result is h_S.
%-----

line_ind_fr = sparse(current.data.fr,1:current.data.nlines, ...
    ones(current.data.nlines,1),current.data.nbuses,current.data.nlines);
line_ind_to = sparse(current.data.to,1:current.data.nlines, ...
    ones(current.data.nlines,1),current.data.nbuses,current.data.nlines);
Yvec_nc = conj(-current.data.Yvec);
ejTheta_c = conj(ejTheta);
Ybus_diag = line_ind_fr*current.data.Yvec + line_ind_fr*j*current.data.Bvec/2 ...
    + line_ind_to*current.data.Yvec+line_ind_to*j*current.data.Bvec/2;

h_S_VV = sparse([[1:current.data.nbuses]'; current.data.fr; ...
    current.data.to], [[1:current.data.nbuses]'; current.data.to; current.data.fr], ...
    [ 2*real(conj(Ybus_diag)).*current.var.eq.lagrange.P ...
    + 2*imag(conj(Ybus_diag)).*current.var.eq.lagrange.Q;
    real(ejTheta(current.data.fr).*Yvec_nc.*ejTheta_c(current.data.to)) ...
    .*current.var.eq.lagrange.P(current.data.fr) ...
    + real(ejTheta(current.data.to).*Yvec_nc.*ejTheta_c(current.data.fr)) ...
    .*current.var.eq.lagrange.P(current.data.to) ...
    + imag(ejTheta(current.data.fr).*Yvec_nc.*ejTheta_c(current.data.to)) ...
    .*current.var.eq.lagrange.Q(current.data.fr) ...
    + imag(ejTheta(current.data.to).*Yvec_nc.*ejTheta_c(current.data.fr)) ...
    .*current.var.eq.lagrange.Q(current.data.to); ...
    real(ejTheta(current.data.fr).*Yvec_nc.*ejTheta_c(current.data.to)) ...
    .*current.var.eq.lagrange.P(current.data.fr) ...
    + real(ejTheta(current.data.to).*Yvec_nc.*ejTheta_c(current.data.fr)) ...
    .*current.var.eq.lagrange.P(current.data.to) ...
    + imag(ejTheta(current.data.fr).*Yvec_nc.*ejTheta_c(current.data.to)) ...
    .*current.var.eq.lagrange.Q(current.data.fr) ...
    + imag(ejTheta(current.data.to).*Yvec_nc.*ejTheta_c(current.data.fr)) ...
    .*current.var.eq.lagrange.Q(current.data.to)], ...
    current.data.nbuses,current.data.nbuses);

h_S_tt_diag=real(-Sp+Vp.*conj(Vp.*Ybus_diag)).*current.var.eq.lagrange.P ...

```


APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

+ imag(-Sp*Vp.*conj(Vp.*Ybus_diag)).*current.var.eq.lagrange.Q ...
-real(conj(Vp).*(line_ind_fr*(Vp(current.data.to).*Yvec_nc ...
.*current.var.eq.lagrange.P(current.data.to)))) ...
-real(conj(Vp).*(line_ind_to*(Vp(current.data.fr).*Yvec_nc ...
.*current.var.eq.lagrange.P(current.data.fr)))) ...
-imag(conj(Vp).*(line_ind_fr*(Vp(current.data.to).*Yvec_nc ...
.*current.var.eq.lagrange.Q(current.data.to)))) ...
-imag(conj(Vp).*(line_ind_to*(Vp(current.data.fr).*Yvec_nc ...
.*current.var.eq.lagrange.Q(current.data.fr)))));

h_S_tt = sparse([1:current.data.nbuses]'; current.data.fr; ...
current.data.to], [1:current.data.nbuses]'; current.data.to; current.data.fr], ...
[ h_S_tt_diag;
real(Vp(current.data.fr).*Yvec_nc.*conj(Vp(current.data.to))) ...
.*current.var.eq.lagrange.P(current.data.fr) ...
+ real(Vp(current.data.to).*Yvec_nc.*conj(Vp(current.data.fr))) ...
.*current.var.eq.lagrange.P(current.data.to) ...
+ imag(Vp(current.data.fr).*Yvec_nc.*conj(Vp(current.data.to))) ...
.*current.var.eq.lagrange.Q(current.data.fr) ...
+ imag(Vp(current.data.to).*Yvec_nc.*conj(Vp(current.data.fr))) ...
.*current.var.eq.lagrange.Q(current.data.to); ...
real(Vp(current.data.fr).*Yvec_nc.*conj(Vp(current.data.to))) ...
.*current.var.eq.lagrange.P(current.data.fr) ...
+ real(Vp(current.data.to).*Yvec_nc.*conj(Vp(current.data.fr))) ...
.*current.var.eq.lagrange.P(current.data.to) ...
+ imag(Vp(current.data.fr).*Yvec_nc.*conj(Vp(current.data.to))) ...
.*current.var.eq.lagrange.Q(current.data.fr) ...
+ imag(Vp(current.data.to).*Yvec_nc.*conj(Vp(current.data.fr))) ...
.*current.var.eq.lagrange.Q(current.data.to)], ...
current.data.nbuses,current.data.nbuses) );

h_S_Vt_diag= ...
real(j*ejTheta.*conj(Ip)-j*current.var.V.*conj(Ybus_diag)) ...
.*current.var.eq.lagrange.P ...
+ imag(j*ejTheta.*conj(Ip)-j*current.var.V.*conj(Ybus_diag)) ...
.*current.var.eq.lagrange.Q ...
-real(j*ejTheta_c.*(line_ind_fr*(Vp(current.data.to).*Yvec_nc ...
.*current.var.eq.lagrange.P(current.data.to)))) ...
-real(j*ejTheta_c.*(line_ind_to*(Vp(current.data.fr).*Yvec_nc ...
.*current.var.eq.lagrange.P(current.data.fr)))) ...
-imag(j*ejTheta_c.*(line_ind_fr*(Vp(current.data.to).*Yvec_nc ...
.*current.var.eq.lagrange.Q(current.data.to)))) ...
-imag(j*ejTheta_c.*(line_ind_to*(Vp(current.data.fr).*Yvec_nc ...
.*current.var.eq.lagrange.Q(current.data.fr)))));

h_S_Vt = full(sparse([1:current.data.nbuses]'; current.data.fr; ...
current.data.to], [1:current.data.nbuses]'; current.data.to; current.data.fr], ...
[ h_S_Vt_diag;
real(-j*ejTheta(current.data.fr).*Yvec_nc.*conj(Vp(current.data.to))) ...
.*current.var.eq.lagrange.P(current.data.fr) ...
+ real(j*Vp(current.data.to).*Yvec_nc.*ejTheta_c(current.data.fr)) ...
.*current.var.eq.lagrange.P(current.data.to) ...
+ imag(-j*ejTheta(current.data.fr).*Yvec_nc.*conj(Vp(current.data.to))) ...
.*current.var.eq.lagrange.Q(current.data.fr) ...
+ imag(j*Vp(current.data.to).*Yvec_nc.*ejTheta_c(current.data.fr)) ...
.*current.var.eq.lagrange.Q(current.data.to); ...
real(-j*ejTheta(current.data.to).*Yvec_nc.*conj(Vp(current.data.fr))) ...
.*current.var.eq.lagrange.P(current.data.to) ...
+ real(j*Vp(current.data.fr).*Yvec_nc.*ejTheta_c(current.data.to)) ...
.*current.var.eq.lagrange.P(current.data.fr) ...
+ imag(-j*ejTheta(current.data.to).*Yvec_nc.*conj(Vp(current.data.fr))) ...
.*current.var.eq.lagrange.Q(current.data.to) ...
+ imag(j*Vp(current.data.fr).*Yvec_nc.*ejTheta_c(current.data.to)) ...
.*current.var.eq.lagrange.Q(current.data.fr)], ...
current.data.nbuses,current.data.nbuses) );

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

h_S = full([h_S_VV h_S_Vt; h_S_Vt' h_S_tt]);

H_h([current.ind.V.ind current.ind.theta.ind],[current.ind.V.ind current.ind.theta.ind]) = ...
    H_h([current.ind.V.ind current.ind.theta.ind],[current.ind.V.ind current.ind.theta.ind]) - ...
    h_S;
end%for
%*****
% In this section, we form the hessian for each complex
% Sij=(Vi-Vj)*Yij+Vi*Yshunt.
% These Hessians are stacked on top of each other to form
% the matrix hess_Sij.
% hess_Sij = [HessSij(1); HessSij(2); HessSij(3); ... HessSij(nlines)]
% Then we form [lambda.Pij(k)*eye(nbuses) ... lambda.Pij(nlines)*eye]
% which is multiplied by real(hess_Sij).
% This give sum_k(lambda.Pij(k)*HessSij(k));
%*****
for k=0:data.ncont
current = getstate(data,index,var,k);
dSvec_dVi = ejTheta(current.data.fr).*conj(current.var.Iij) ...
    + current.var.V(current.data.fr).*current.data.Yvec+j*current.data.Bvec;
dSvec_dVk = -Vp(current.data.fr).*conj(ejTheta(current.data.to).*current.data.Yvec);
dSvec_dthetai = j*Vp(current.data.fr).*conj(current.var.Iij) - ...
    j*current.var.V(current.data.fr).^2.*conj(current.data.Yvec+j*current.data.Bvec/2);
dSvec_dthetak = j*Vp(current.data.fr).*conj(Vp(current.data.to).*current.data.Yvec);

dSvec_dViVk = -ejTheta(current.data.fr).*conj(ejTheta(current.data.to).*current.data.Yvec);
dSvec_dViVi = 2*conj(current.data.Yvec+j*current.data.Bvec/2);
dSvec_dthetai_thetak = -Vp(current.data.fr).*conj(Vp(current.data.to).*current.data.Yvec);
dSvec_dthetak_thetak = Vp(current.data.fr).*conj(Vp(current.data.to).*current.data.Yvec);
dSvec_dthetai_thetai = -Vp(current.data.fr).*conj(current.var.Iij)+ ...
    current.var.V(current.data.fr).^2.*conj(current.data.Yvec+j*current.data.Bvec/2);
dSvec_dVi_dthetak = j*ejTheta(current.data.fr).*conj(Vp(current.data.to).*current.data.Yvec);
dSvec_dVk_dthetai = -j*Vp(current.data.fr).*conj(ejTheta(current.data.to).*current.data.Yvec);
dSvec_dVi_dthetai = j*ejTheta(current.data.fr).*conj(current.var.Iij)- ...
    j*current.var.V(current.data.fr).*conj(current.data.Yvec+j*current.data.Bvec/2);
dSvec_dVk_dthetak = j*Vp(current.data.fr).*conj(ejTheta(current.data.to).*current.data.Yvec);

%hess_Sij = zeros(2*current.data.nbuses*current.data.nlines,2*current.data.nbuses);

mod_V = 0:2*current.data.nbuses:2*current.data.nbuses*current.data.nlines-1;
mod_theta = current.data.nbuses:2*current.data.nbuses*current.data.nlines-1;
fr_V = current.data.fr+mod_V';
fr_theta = current.data.fr+mod_theta';
to_V = current.data.to+mod_V';
to_theta = current.data.to+mod_theta';

ii = [fr_V; to_V; fr_V; fr_theta; to_theta; fr_theta; to_theta; ...
    fr_V; to_V; fr_V; to_V; to_theta; fr_theta; fr_theta; to_theta];
jj = [current.data.to; current.data.fr; current.data.fr; ...
    current.data.to+current.data.nbuses; current.data.fr+current.data.nbuses; ...
    current.data.fr+current.data.nbuses; ...
    current.data.to+current.data.nbuses; ...
    current.data.to+current.data.nbuses; current.data.fr+current.data.nbuses; ...
    current.data.fr+current.data.nbuses; current.data.to+current.data.nbuses; ...
    current.data.fr; current.data.to; current.data.fr; current.data.to];
hess_Sij = sparse(ii,jj,[dSvec_dViVk; dSvec_dViVk; dSvec_dViVi; ...
    dSvec_dthetai_thetak; dSvec_dthetai_thetak; ...
    dSvec_dthetai_thetai; dSvec_dthetak_thetak; ...
    dSvec_dVi_dthetak; dSvec_dVk_dthetai; ...
    dSvec_dVi_dthetai; dSvec_dVk_dthetak; ...
    dSvec_dVi_dthetak; dSvec_dVk_dthetai; ...
    dSvec_dVi_dthetai; dSvec_dVk_dthetak] ...
,2*current.data.nbuses*current.data.nlines,2*current.data.nbuses);

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

%-----
% Now we multiply each Hessian by eq.lagrange.(P or Q)ij and add
% the Hessians together together.
%-----
evalue_eye= repmat(eye(2*current.data.nbuses),1,current.data.nlines);
realevalue_eye=evalue_eye;
imagevalue_eye=evalue_eye;
[ii,jj] = find(ones(2*current.data.nbuses,current.data.nlines));
realevalue_eye(find(evalue_eye)) = current.var.eq.lagrange.Pij(jj);
imagevalue_eye(find(evalue_eye)) = current.var.eq.lagrange.Qij(jj);
rhess=real(hess_Sij.)*realevalue_eye.';
ihess=imag(hess_Sij.)*imagevalue_eye.';
H_h([current.ind.V.ind current.ind.theta.ind],[current.ind.V.ind current.ind.theta.ind]) = ...
    H_h([current.ind.V.ind current.ind.theta.ind],[current.ind.V.ind current.ind.theta.ind]) - ...
    rhess;
H_h([current.ind.V.ind current.ind.theta.ind],[current.ind.V.ind current.ind.theta.ind]) = ...
    H_h([current.ind.V.ind current.ind.theta.ind],[current.ind.V.ind current.ind.theta.ind]) - ...
    ihess;

H_h(current.ind.Pij.ind,current.ind.Pij.ind) = -2*diag(current.var.eq.lagrange.Sij);
H_h(current.ind.Qij.ind,current.ind.Qij.ind) = -2*diag(current.var.eq.lagrange.Sij);

%*****
% Hessians for Ig_real and Ig_imag
%*****
dIg_dtheta_dV = diag(j*eJTheta(current.data.Pgenbuses).*conj(Ig));
dIg_dIg_dV = diag(eJTheta(current.data.Pgenbuses).*conj(ejgamma));
dIg_dgamma_dV = diag(-j*eJTheta(current.data.Pgenbuses).*conj(Ig));
dIg_dtheta_dtheta = diag(-Vp(current.data.Pgenbuses).*conj(Ig));
dIg_dIg_dtheta = diag(j*Vp(current.data.Pgenbuses).*conj(Ig));
dIg_dgamma_dtheta = diag(Vp(current.data.Pgenbuses).*conj(Ig));
dIg_dgamma_dIg = diag(-j*Vp(current.data.Pgenbuses).*conj(ejgamma));
dIg_dgamma_dgamma = diag(-Vp(current.data.Pgenbuses).*conj(Ig));

hess_Ig = zeros(index.Y_len,index.Y_len);
hess_Ig(current.ind.theta.ind(current.data.Pgenbuses), ...
    current.ind.V.ind(current.data.Pgenbuses)) = ...
    hess_Ig(current.ind.theta.ind(current.data.Pgenbuses), ...
    current.ind.V.ind(current.data.Pgenbuses)) + ...
    diag(current.var.eq.lagrange.Ig_real).*real(dIg_dtheta_dV);
hess_Ig(current.ind.theta.ind(current.data.Pgenbuses), ...
    current.ind.V.ind(current.data.Pgenbuses)) = ...
    hess_Ig(current.ind.theta.ind(current.data.Pgenbuses), ...
    current.ind.V.ind(current.data.Pgenbuses)) + ...
    diag(current.var.eq.lagrange.Ig_imag).*imag(dIg_dtheta_dV);
hess_Ig(current.ind.Ig.ind, current.ind.V.ind(current.data.Pgenbuses)) = ...
    hess_Ig(current.ind.Ig.ind, current.ind.V.ind(current.data.Pgenbuses)) + ...
    diag(current.var.eq.lagrange.Ig_real).*real(dIg_dIg_dV);
hess_Ig(current.ind.Ig.ind, current.ind.V.ind(current.data.Pgenbuses)) + ...
    diag(current.var.eq.lagrange.Ig_imag).*imag(dIg_dIg_dV);
hess_Ig(current.ind.gamma.ind, current.ind.V.ind(current.data.Pgenbuses)) = ...
    hess_Ig(current.ind.gamma.ind, current.ind.V.ind(current.data.Pgenbuses)) + ...
    diag(current.var.eq.lagrange.Ig_real).*real(dIg_dgamma_dV);
hess_Ig(current.ind.gamma.ind, current.ind.V.ind(current.data.Pgenbuses)) = ...
    hess_Ig(current.ind.gamma.ind, current.ind.V.ind(current.data.Pgenbuses)) + ...
    diag(current.var.eq.lagrange.Ig_imag).*imag(dIg_dgamma_dV);
hess_Ig(current.ind.gamma.ind, current.ind.V.ind(current.data.Pgenbuses)) = ...
    hess_Ig(current.ind.gamma.ind, current.ind.V.ind(current.data.Pgenbuses)) + ...
    diag(current.var.eq.lagrange.Ig_imag).*imag(dIg_dgamma_dV);
hess_Ig(current.ind.theta.ind(current.data.Pgenbuses), ...
    current.ind.theta.ind(current.data.Pgenbuses)) = ...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

hess_Ig(current.ind.theta.ind(current.data.Pgenbuses), ...
        current.ind.theta.ind(current.data.Pgenbuses)) + ...
        diag(current.var.eq.lagrange.Ig_real).*real(dIg_dtheta_dtheta);
hess_Ig(current.ind.theta.ind(current.data.Pgenbuses), ...
        current.ind.theta.ind(current.data.Pgenbuses)) = ...
hess_Ig(current.ind.theta.ind(current.data.Pgenbuses), ...
        current.ind.theta.ind(current.data.Pgenbuses)) + ...
        diag(current.var.eq.lagrange.Ig_imag).*imag(dIg_dtheta_dtheta);
hess_Ig(current.ind.Ig.ind, current.ind.theta.ind(current.data.Pgenbuses)) = ...
        hess_Ig(current.ind.Ig.ind, current.ind.theta.ind(current.data.Pgenbuses)) + ...
        diag(current.var.eq.lagrange.Ig_real).*real(dIg_dIg_dtheta);
hess_Ig(current.ind.Ig.ind, current.ind.theta.ind(current.data.Pgenbuses)) = ...
        hess_Ig(current.ind.Ig.ind, current.ind.theta.ind(current.data.Pgenbuses)) + ...
        diag(current.var.eq.lagrange.Ig_imag).*imag(dIg_dIg_dtheta);
hess_Ig(current.ind.gamma.ind, current.ind.theta.ind(current.data.Pgenbuses)) = ...
        hess_Ig(current.ind.gamma.ind, current.ind.theta.ind(current.data.Pgenbuses)) + ...
        diag(current.var.eq.lagrange.Ig_real).*real(dIg_dgamma_dtheta);
hess_Ig(current.ind.gamma.ind, current.ind.theta.ind(current.data.Pgenbuses)) = ...
        hess_Ig(current.ind.gamma.ind, current.ind.theta.ind(current.data.Pgenbuses)) + ...
        diag(current.var.eq.lagrange.Ig_imag).*imag(dIg_dgamma_dtheta);
hess_Ig(current.ind.gamma.ind, current.ind.Ig.ind) = ...
        hess_Ig(current.ind.gamma.ind, current.ind.Ig.ind) + ...
        diag(current.var.eq.lagrange.Ig_real).*real(dIg_dgamma_dIg);
hess_Ig(current.ind.gamma.ind, current.ind.Ig.ind) = ...
        hess_Ig(current.ind.gamma.ind, current.ind.Ig.ind) + ...
        diag(current.var.eq.lagrange.Ig_imag).*imag(dIg_dgamma_dIg);
hess_Ig(current.ind.gamma.ind, current.ind.gamma.ind) = ...
        hess_Ig(current.ind.gamma.ind, current.ind.gamma.ind) + ...
        diag(current.var.eq.lagrange.Ig_real).*real(dIg_dgamma_dgamma);
hess_Ig(current.ind.gamma.ind, current.ind.gamma.ind) = ...
        hess_Ig(current.ind.gamma.ind, current.ind.gamma.ind) + ...
        diag(current.var.eq.lagrange.Ig_imag).*imag(dIg_dgamma_dgamma);

hess_Ig = hess_Ig + hess_Ig';

H_h = H_h + hess_Ig;

%*****
% Hessians for delta_real and delta_imag
%*****
ddelta_dtheta_dV = diag(j*e*jTheta(current.data.Pgenbuses));
ddelta_dtheta_dtheta = diag(-Vp(current.data.Pgenbuses));
ddelta_ddelta_dAdelt = diag(-j*e*jdelta);
ddelta_ddelta_ddelta = diag(Adelt);
ddelta_dgamma_dIg = diag(j*(data.machine.Ra+j*data.machine.Xq).*ejgamma);
ddelta_dgamma_dgamma = diag(-(data.machine.Ra+j*data.machine.Xq).*Ig);

hess_delta = zeros(index.Y_len,index.Y_len);
hess_delta(current.ind.theta.ind(current.data.Pgenbuses), ...
        current.ind.V.ind(current.data.Pgenbuses)) = ...
        hess_delta(current.ind.theta.ind(current.data.Pgenbuses), ...
        current.ind.V.ind(current.data.Pgenbuses)) + ...
        diag(current.var.eq.lagrange.delta_real).*real(ddelta_dtheta_dV);
hess_delta(current.ind.theta.ind(current.data.Pgenbuses), ...
        current.ind.V.ind(current.data.Pgenbuses)) = ...
        hess_delta(current.ind.theta.ind(current.data.Pgenbuses), ...
        current.ind.V.ind(current.data.Pgenbuses)) + ...
        diag(current.var.eq.lagrange.delta_imag).*imag(ddelta_dtheta_dV);
hess_delta(current.ind.theta.ind(current.data.Pgenbuses), ...
        current.ind.theta.ind(current.data.Pgenbuses)) = ...
        hess_delta(current.ind.theta.ind(current.data.Pgenbuses), ...
        current.ind.theta.ind(current.data.Pgenbuses)) + ...
        diag(current.var.eq.lagrange.delta_real).*real(ddelta_dtheta_dtheta);
hess_delta(current.ind.theta.ind(current.data.Pgenbuses), ...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

        current.ind.theta.ind(current.data.Pgenbuses)) = ...
        hess_delta(current.ind.theta.ind(current.data.Pgenbuses), ...
        current.ind.theta.ind(current.data.Pgenbuses)) + ...
        diag(current.var.eq.lagrange.delta_imag).*imag(ddelta_dtheta_dtheta);
hess_delta(current.ind.delta.ind, current.ind.Adelt.ind) = ...
        hess_delta(current.ind.delta.ind, current.ind.Adelt.ind) + ...
        diag(current.var.eq.lagrange.delta_real).*real(ddelta_ddelta_dAdelt);
hess_delta(current.ind.delta.ind, current.ind.Adelt.ind) = ...
        hess_delta(current.ind.delta.ind, current.ind.Adelt.ind) + ...
        diag(current.var.eq.lagrange.delta_imag).*imag(ddelta_ddelta_dAdelt);
hess_delta(current.ind.delta.ind, current.ind.delta.ind) = ...
        hess_delta(current.ind.delta.ind, current.ind.delta.ind) + ...
        diag(current.var.eq.lagrange.delta_real).*real(ddelta_ddelta_ddelta);
hess_delta(current.ind.delta.ind, current.ind.delta.ind) = ...
        hess_delta(current.ind.delta.ind, current.ind.delta.ind) + ...
        diag(current.var.eq.lagrange.delta_imag).*imag(ddelta_ddelta_ddelta);
hess_delta(current.ind.gamma.ind, current.ind.Ig.ind) = ...
        hess_delta(current.ind.gamma.ind, current.ind.Ig.ind) + ...
        diag(current.var.eq.lagrange.delta_real).*real(ddelta_dgamma_dIg);
hess_delta(current.ind.gamma.ind, current.ind.Ig.ind) = ...
        hess_delta(current.ind.gamma.ind, current.ind.Ig.ind) + ...
        diag(current.var.eq.lagrange.delta_imag).*imag(ddelta_dgamma_dIg);
hess_delta(current.ind.gamma.ind, current.ind.gamma.ind) = ...
        hess_delta(current.ind.gamma.ind, current.ind.gamma.ind) + ...
        diag(current.var.eq.lagrange.delta_real).*real(ddelta_dgamma_dgamma);
hess_delta(current.ind.gamma.ind, current.ind.gamma.ind) = ...
        hess_delta(current.ind.gamma.ind, current.ind.gamma.ind) + ...
        diag(current.var.eq.lagrange.delta_imag).*imag(ddelta_dgamma_dgamma);

hess_delta = hess_delta + hess_delta';

H_h = H_h + hess_delta;

%*****
% Hessians for Idq_real and Idq_imag
%*****
dIdq_dgamma_dIg = diag(ejgamma.*conj(ejdelta));
dIdq_dgamma_dgamma = diag(j*Ig.*conj(ejdelta));
dIdq_ddelta_dIg = diag(-ejgamma.*conj(ejdelta));
dIdq_ddelta_ddelta = diag(j*Ig.*conj(ejdelta));
dIdq_ddelta_dgamma = diag(-j*Ig.*conj(ejdelta));

hess_Idq = zeros(index.Y_len,index.Y_len);
hess_Idq(current.ind.gamma.ind, current.ind.Ig.ind) = ...
        hess_Idq(current.ind.gamma.ind, current.ind.Ig.ind) + ...
        diag(current.var.eq.lagrange.Idq_real).*real(dIdq_dgamma_dIg);
hess_Idq(current.ind.gamma.ind, current.ind.Ig.ind) = ...
        hess_Idq(current.ind.gamma.ind, current.ind.Ig.ind) + ...
        diag(current.var.eq.lagrange.Idq_imag).*imag(dIdq_dgamma_dIg);
hess_Idq(current.ind.gamma.ind, current.ind.gamma.ind) = ...
        hess_Idq(current.ind.gamma.ind, current.ind.gamma.ind) + ...
        diag(current.var.eq.lagrange.Idq_real).*real(dIdq_dgamma_dgamma);
hess_Idq(current.ind.gamma.ind, current.ind.gamma.ind) = ...
        hess_Idq(current.ind.gamma.ind, current.ind.gamma.ind) + ...
        diag(current.var.eq.lagrange.Idq_imag).*imag(dIdq_dgamma_dgamma);
hess_Idq(current.ind.delta.ind, current.ind.Ig.ind) = ...
        hess_Idq(current.ind.delta.ind, current.ind.Ig.ind) + ...
        diag(current.var.eq.lagrange.Idq_real).*real(dIdq_ddelta_dIg);
hess_Idq(current.ind.delta.ind, current.ind.Ig.ind) = ...
        hess_Idq(current.ind.delta.ind, current.ind.Ig.ind) + ...
        diag(current.var.eq.lagrange.Idq_imag).*imag(dIdq_ddelta_dIg);
hess_Idq(current.ind.delta.ind, current.ind.delta.ind) = ...
        hess_Idq(current.ind.delta.ind, current.ind.delta.ind) + ...
        diag(current.var.eq.lagrange.Idq_real).*real(dIdq_ddelta_ddelta);
hess_Idq(current.ind.delta.ind, current.ind.delta.ind) = ...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

    hess_Idq(current.ind.delta.ind, current.ind.delta.ind) + ...
        diag(current.var.eq.lagrange.Idq_imag).*imag(dIdq_ddelta_ddelta);
    hess_Idq(current.ind.delta.ind, current.ind.gamma.ind) = ...
        hess_Idq(current.ind.delta.ind, current.ind.gamma.ind) + ...
        diag(current.var.eq.lagrange.Idq_real).*real(dIdq_ddelta_dgamma);
    hess_Idq(current.ind.delta.ind, current.ind.gamma.ind) = ...
        hess_Idq(current.ind.delta.ind, current.ind.gamma.ind) + ...
        diag(current.var.eq.lagrange.Idq_imag).*imag(dIdq_ddelta_dgamma);

    hess_Idq = hess_Idq + hess_Idq';

    H_h = H_h + hess_Idq;

%*****
% Hessians for Vdq_real and Vdq_imag
%*****
dVdq_dtheta_dV = diag(ejTheta(data.Pgenbuses).*conj(ejdelta));
dVdq_dtheta_dtheta = diag(j*Vp(data.Pgenbuses).*conj(ejdelta));
dVdq_ddelta_dV = diag(-ejTheta(data.Pgenbuses).*conj(ejdelta));
dVdq_ddelta_ddelta = diag(j*Vp(data.Pgenbuses).*conj(ejdelta));
dVdq_ddelta_dtheta = diag(-j*Vp(data.Pgenbuses).*conj(ejdelta));

    hess_Vdq = zeros(index.Y_len,index.Y_len);
    hess_Vdq(current.ind.theta.ind(data.Pgenbuses), current.ind.V.ind(data.Pgenbuses)) = ...
        hess_Vdq(current.ind.theta.ind(data.Pgenbuses), current.ind.V.ind(data.Pgenbuses)) + ...
        diag(current.var.eq.lagrange.Vdq_real).*real(dVdq_dtheta_dV);
    hess_Vdq(current.ind.theta.ind(data.Pgenbuses), current.ind.V.ind(data.Pgenbuses)) = ...
        hess_Vdq(current.ind.theta.ind(data.Pgenbuses), current.ind.V.ind(data.Pgenbuses)) + ...
        diag(current.var.eq.lagrange.Vdq_imag).*imag(dVdq_dtheta_dV);
    hess_Vdq(current.ind.theta.ind(data.Pgenbuses), current.ind.theta.ind(data.Pgenbuses)) = ...
        hess_Vdq(current.ind.theta.ind(data.Pgenbuses), current.ind.theta.ind(data.Pgenbuses)) + ...
        diag(current.var.eq.lagrange.Vdq_real).*real(dVdq_dtheta_dtheta);
    hess_Vdq(current.ind.theta.ind(data.Pgenbuses), current.ind.theta.ind(data.Pgenbuses)) = ...
        hess_Vdq(current.ind.theta.ind(data.Pgenbuses), current.ind.theta.ind(data.Pgenbuses)) + ...
        diag(current.var.eq.lagrange.Vdq_imag).*imag(dVdq_dtheta_dtheta);
    hess_Vdq(current.ind.delta.ind, current.ind.V.ind(data.Pgenbuses)) = ...
        hess_Vdq(current.ind.delta.ind, current.ind.V.ind(data.Pgenbuses)) + ...
        diag(current.var.eq.lagrange.Vdq_real).*real(dVdq_ddelta_dV);
    hess_Vdq(current.ind.delta.ind, current.ind.V.ind(data.Pgenbuses)) = ...
        hess_Vdq(current.ind.delta.ind, current.ind.V.ind(data.Pgenbuses)) + ...
        diag(current.var.eq.lagrange.Vdq_imag).*imag(dVdq_ddelta_dV);
    hess_Vdq(current.ind.delta.ind, current.ind.delta.ind) = ...
        hess_Vdq(current.ind.delta.ind, current.ind.delta.ind) + ...
        diag(current.var.eq.lagrange.Vdq_real).*real(dVdq_ddelta_ddelta);
    hess_Vdq(current.ind.delta.ind, current.ind.delta.ind) = ...
        hess_Vdq(current.ind.delta.ind, current.ind.delta.ind) + ...
        diag(current.var.eq.lagrange.Vdq_imag).*imag(dVdq_ddelta_ddelta);
    hess_Vdq(current.ind.delta.ind, current.ind.theta.ind(data.Pgenbuses)) = ...
        hess_Vdq(current.ind.delta.ind, current.ind.theta.ind(data.Pgenbuses)) + ...
        diag(current.var.eq.lagrange.Vdq_real).*real(dVdq_ddelta_dtheta);
    hess_Vdq(current.ind.delta.ind, current.ind.theta.ind(data.Pgenbuses)) = ...
        hess_Vdq(current.ind.delta.ind, current.ind.theta.ind(data.Pgenbuses)) + ...
        diag(current.var.eq.lagrange.Vdq_imag).*imag(dVdq_ddelta_dtheta);

    hess_Vdq = hess_Vdq + hess_Vdq';

    H_h = H_h + hess_Vdq;

end%for

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

H.18 `ineq_hessian.m`

```
function H_g = ineq_hessian(data,var,index)
```

```
H_g = zeros(index.Y_len,index.Y_len);
```

H.19 objective_hessian.m

```
function H_f = objective_hessian(data,var,index)

H_f = zeros(index.Y_len,index.Y_len);

H_f(index.pre.Pg.ind,index.pre.Pg.ind) = ...
    data.line.pi_0*diag(2*data.gen.gamma(data.Pgenbuses));
H_f(index.pre.Pl.ind,index.pre.Pl.ind) = ...
    -data.line.pi_0*diag(2*data.load.gamma(data.Ploadbuses));
for k=1:data.ncont
current = getstate(data,index,var,k);

H_f(current.ind.Pg.ind,current.ind.Pg.ind) = data.line.pi_k(k)*diag(2*data.gen.gamma(data.Pgenbuses));
H_f(current.ind.Pl.ind,current.ind.Pl.ind) = -data.line.pi_k(k)*diag(2*data.load.gamma(data.Ploadbuses));

end%for
```


H.20 eig_jac_hess.m

```

function [J_g, H_g] = eig_jac_hess(data,ind,var,J_g2,H_g2)

J_g = J_g2;
H_g = H_g2;
%eigen = sysmat(var,data,ind);
for cont=0:data.ncont
current = getstate(data,ind,var,cont);
asysderiv2 = asysderiv(current.data,current.var,current.ind);
for m=1:data.neig_const;
k=data.eig_ind(m);
eigderivatives = eigderiv(current.data,asysderiv2,k,current.var.eigen);
J_g(current.ind.ineq.eig_real.ind(m),current.ind.V.ind) = ...
    real(eigderivatives.deig.dV);
J_g(current.ind.ineq.eig_real.ind(m),current.ind.theta.ind) = ...
    real(eigderivatives.deig.dtheta);
J_g(current.ind.ineq.eig_real.ind(m),current.ind.Id.ind) = ...
    real(eigderivatives.deig.dId);
J_g(current.ind.ineq.eig_real.ind(m),current.ind.Iq.ind) = ...
    real(eigderivatives.deig.dIq);
J_g(current.ind.ineq.eig_real.ind(m),current.ind.Edp.ind) = ...
    real(eigderivatives.deig.dEdp);
J_g(current.ind.ineq.eig_real.ind(m),current.ind.Eqp.ind) = ...
    real(eigderivatives.deig.dEqp);
J_g(current.ind.ineq.eig_real.ind(m),current.ind.Efd.ind) = ...
    real(eigderivatives.deig.dEfd);
J_g(current.ind.ineq.eig_real.ind(m),current.ind.delta.ind) = ...
    real(eigderivatives.deig.ddelta);
hess_eig = zeros(ind.Y_len,ind.Y_len);
hess_eig(current.ind.Id.ind,current.ind.V.ind) = eigderivatives.d2eig.dId_dV;
hess_eig(current.ind.V.ind,current.ind.Id.ind) = eigderivatives.d2eig.dV_dId;
hess_eig(current.ind.Iq.ind,current.ind.V.ind) = eigderivatives.d2eig.dIq_dV;
hess_eig(current.ind.V.ind,current.ind.Iq.ind) = eigderivatives.d2eig.dV_dIq;
%-----
hess_eig(current.ind.Id.ind,current.ind.theta.ind) = ...
    eigderivatives.d2eig.dId_dtheta;
hess_eig(current.ind.theta.ind,current.ind.Id.ind) = ...
    eigderivatives.d2eig.dtheta_dId;
hess_eig(current.ind.Iq.ind,current.ind.theta.ind) = ...
    eigderivatives.d2eig.dIq_dtheta;
hess_eig(current.ind.theta.ind,current.ind.Iq.ind) = ...
    eigderivatives.d2eig.dtheta_dIq;
%-----
hess_eig(current.ind.Id.ind,current.ind.delta.ind) = ...
    eigderivatives.d2eig.dId_ddelta;
hess_eig(current.ind.delta.ind,current.ind.Id.ind) = ...
    eigderivatives.d2eig.ddelta_dId;
hess_eig(current.ind.Iq.ind,current.ind.delta.ind) = ...
    eigderivatives.d2eig.dIq_ddelta;
hess_eig(current.ind.delta.ind,current.ind.Iq.ind) = ...
    eigderivatives.d2eig.ddelta_dIq;
%-----
hess_eig(current.ind.Efd.ind,current.ind.Efd.ind) = ...
    diag(eigderivatives.d2eig.dEfd_dEfd);
%-----
hess_eig(current.ind.V.ind,current.ind.Edp.ind) = ...
    eigderivatives.d2eig.dV_dEdp;
hess_eig(current.ind.Edp.ind,current.ind.V.ind) = ...
    eigderivatives.d2eig.dEdp_dV;
hess_eig(current.ind.V.ind,current.ind.Eqp.ind) = ...
    eigderivatives.d2eig.dV_dEqp;
hess_eig(current.ind.Eqp.ind,current.ind.V.ind) = ...
    eigderivatives.d2eig.dEqp_dV;
%-----

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

hess_eig(current.ind.theta.ind,current.ind.Edp.ind) = ...
    eigderivatives.d2eig.dtheta_dEdp;
hess_eig(current.ind.Edp.ind,current.ind.theta.ind) = ...
    eigderivatives.d2eig.dEdp_dtheta;
hess_eig(current.ind.theta.ind,current.ind.Eqp.ind) = ...
    eigderivatives.d2eig.dtheta_dEqp;
hess_eig(current.ind.Eqp.ind,current.ind.theta.ind) = ...
    eigderivatives.d2eig.dEqp_dtheta;
%-----
hess_eig(current.ind.delta.ind,current.ind.Edp.ind) = ...
    eigderivatives.d2eig.ddelta_dEdp;
hess_eig(current.ind.Edp.ind,current.ind.delta.ind) = ...
    eigderivatives.d2eig.dEdp_ddelta;
hess_eig(current.ind.delta.ind,current.ind.Eqp.ind) = ...
    eigderivatives.d2eig.ddelta_dEqp;
hess_eig(current.ind.Eqp.ind,current.ind.delta.ind) = ...
    eigderivatives.d2eig.dEqp_ddelta;
%-----
hess_eig(current.ind.Id.ind,current.ind.Edp.ind) = ...
    eigderivatives.d2eig.dId_dEdp;
hess_eig(current.ind.Edp.ind,current.ind.Id.ind) = ...
    eigderivatives.d2eig.dEdp_dId;
hess_eig(current.ind.Id.ind,current.ind.Eqp.ind) = ...
    eigderivatives.d2eig.dId_dEqp;
hess_eig(current.ind.Eqp.ind,current.ind.Id.ind) = ...
    eigderivatives.d2eig.dEqp_dId;
%-----
hess_eig(current.ind.Iq.ind,current.ind.Edp.ind) = eigderivatives.d2eig.dIq_dEdp;
hess_eig(current.ind.Edp.ind,current.ind.Iq.ind) = eigderivatives.d2eig.dEdp_dIq;
hess_eig(current.ind.Iq.ind,current.ind.Eqp.ind) = eigderivatives.d2eig.dIq_dEqp;
hess_eig(current.ind.Eqp.ind,current.ind.Iq.ind) = eigderivatives.d2eig.dEqp_dIq;
%-----
hess_eig(current.ind.V.ind,current.ind.V.ind) = eigderivatives.d2eig.dV_dV;
hess_eig(current.ind.theta.ind,current.ind.V.ind) = ...
    eigderivatives.d2eig.dtheta_dV;
hess_eig(current.ind.V.ind,current.ind.theta.ind) = ...
    eigderivatives.d2eig.dV_dtheta;
hess_eig(current.ind.theta.ind,current.ind.theta.ind) = ...
    eigderivatives.d2eig.dtheta_dtheta;
%-----
hess_eig(current.ind.delta.ind,current.ind.V.ind) = ...
    eigderivatives.d2eig.ddelta_dV;
hess_eig(current.ind.V.ind,current.ind.delta.ind) = ...
    eigderivatives.d2eig.dV_ddelta;
hess_eig(current.ind.delta.ind,current.ind.delta.ind) = ...
    eigderivatives.d2eig.ddelta_ddelta;
%-----
hess_eig(current.ind.delta.ind,current.ind.theta.ind) = ...
    eigderivatives.d2eig.ddelta_dtheta;
hess_eig(current.ind.theta.ind,current.ind.delta.ind) = ...
    eigderivatives.d2eig.dtheta_ddelta;
hess_eig(current.ind.delta.ind,current.ind.delta.ind) = ...
    eigderivatives.d2eig.ddelta_ddelta;
%-----
%a.H_YY = a.H_YY - var.lambda.eigen_real(m)*real(hess_eig);
%a.H_YY = a.H_YY - var.lambda.eigen_imag(m)*imag(hess_eig);
H_g = H_g + var.ineq.lagrange(current.ind.ineq.eig_real.ind(m))*real(hess_eig);
% a.gYY_mu = a.gYY_mu - x.var.mu(m)*imag(hess_eig);
end%for
end%for

```

H.21 asysderiv.m

```

function a=asysderiv(data,var,ind)

%*****
% var.eigenvalue derivatives
%*****

ejTheta = exp(j*var.theta);
ejgamma = exp(j*var.gamma);
ejdelta_c = exp(-j*var.delta);
Vp = var.V .* ejTheta;
Ig = var.Ig.* ejgamma;
Ig = var.Id+j*var.Iq;
Ig_c = var.Id-j*var.Iq;
% Adelt = var.Adelt.* ejdelta;
Ip = data.Ybus * Vp;
Sp = Vp.*conj(Ip);
Iq_ind = 2:2:2*data.nPgen;
Id_ind = Iq_ind-1;
imag_ind = 2:2:2*data.nPgen;
real_ind = imag_ind-1;
delta_ind = 1:7:data.nPgen*7;
nongen_ind = 1:data.nbuses;
nongen_ind(data.Pgenbuses) = [];
M = data.machine.H/pi/60;
for k=1:data.nPgen
%*****
a.dA1.dIq{k} = zeros(7*data.nPgen,7*data.nPgen);
a.dA1.dId{k} = zeros(7*data.nPgen,7*data.nPgen);
a.dA1.dEfd{k} = a.dA1.dIq{k};
a.d2A1.dEfd_dEfd{k} = a.dA1.dIq{k};
%*****

%*****
a.dB1.dIq{k} = zeros(7*data.nPgen,2*data.nPgen);
a.dB1.dId{k} = a.dB1.dIq{k};
a.dB1.dEdp{k} = a.dB1.dIq{k};
a.dB1.dEqp{k} = a.dB1.dIq{k};
%*****

%*****
a.dC1.dV{k} = zeros(2*data.nPgen,7*data.nPgen);
a.dC1.dtheta{k} = a.dC1.dV{k};
a.dC1.ddelta{k} = a.dC1.dV{k};
% a.dC1.ddelta_dV{k} = a.dC1.dV{k};
% a.dC1.dV_ddelta{k} = a.dC1.dV{k};
% a.dC1.ddelta_dtheta{k} = a.dC1.dV{k};
% a.dC1.dtheta_ddelta{k} = a.dC1.dV{k};
% a.dC1.dtheta_dV{k} = a.dC1.dV{k};
% a.dC1.dV_dtheta{k} = a.dC1.dV{k};
a.dC1.dV_dV{k} = a.dC1.dV{k};
% a.dC1.dtheta_dtheta{k} = a.dC1.dV{k};
for m=1:data.nPgen
a.d2C1.ddelta_ddelta{k,m} = a.dC1.dV{k};
end%for
for m=1:data.nbuses
a.dC1.dV{m} = zeros(2*data.nPgen,7*data.nPgen);
a.dC1.dtheta{m} = a.dC1.dV{k};
for n=1:data.nbuses
a.d2C1.dtheta_dV{m,n} = a.dC1.dV{k};
a.d2C1.dV_dtheta{m,n} = a.dC1.dV{k};
a.d2C1.dtheta_dtheta{m,n} = a.dC1.dV{k};
end%for
for mn=1:data.nPgen

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

    a.d2C1.ddelta_dV{mn,m} = a.dC1.dV{k};
    a.d2C1.dV_ddelta{m,mn} = a.dC1.dV{k};
    a.d2C1.ddelta_dtheta{mn,m} = a.dC1.dV{k};
    a.d2C1.dtheta_ddelta{m,mn} = a.dC1.dV{k};
    end%for
end%for
%*****

%*****
a.dD2.dV{k} = zeros(2*data.nPgen,2*data.nPgen);
a.dD2.dtheta{k} = a.dD2.dV{k};
a.dD2.ddelta{k} = a.dD2.dV{k};
% a.d2D2.dV_dtheta{k} = a.dD2.dV{k};
% a.d2D2.dtheta_dV{k} = a.dD2.dV{k};
% a.d2D2.dV_ddelta{k} = a.dD2.dV{k};
% a.d2D2.ddelta_dV{k} = a.dD2.dV{k};
% a.d2D2.dtheta_ddelta{k} = a.dD2.dV{k};
% a.d2D2.ddelta_dtheta{k} = a.dD2.dV{k};
% a.d2D2.dtheta_dtheta{k} = a.dD2.dV{k};
for m=1:data.nPgen
    a.d2D2.ddelta_ddelta{k,m} = a.dD2.dV{k};
end%for
    for m=1:data.nbuses
        a.dD2.dV{m} = zeros(2*data.nPgen,2*data.nPgen);
        a.dD2.dtheta{m} = a.dD2.dV{k};
        for n=1:data.nbuses
            a.d2D2.dV_dtheta{m,n} = a.dD2.dV{k};
            a.d2D2.dtheta_dV{m,n} = a.dD2.dV{k};
            a.d2D2.dtheta_dtheta{m,n} = a.dD2.dV{k};
        end%for
        for mn=1:data.nPgen
            a.d2D2.dV_ddelta{m,mn} = a.dD2.dV{k};
            a.d2D2.ddelta_dV{mn,m} = a.dD2.dV{k};
            a.d2D2.dtheta_ddelta{m,mn} = a.dD2.dV{k};
            a.d2D2.ddelta_dtheta{mn,m} = a.dD2.dV{k};
        end%for
    end%for
%*****

%*****
a.dC2.dV{k} = zeros(2*data.nPgen,7*data.nPgen);
a.dC2.dtheta{k} = a.dC2.dV{k};
a.dC2.ddelta{k} = a.dC2.dV{k};
a.dC2.dId{k} = a.dC2.dV{k};
a.dC2.dIq{k} = a.dC2.dV{k};
% a.d2C2.dV_dId{k} = a.dC2.dV{k};
% a.d2C2.dId_dV{k} = a.dC2.dV{k};
% a.d2C2.dV_dIq{k} = a.dC2.dV{k};
% a.d2C2.dIq_dV{k} = a.dC2.dV{k};
% a.d2C2.dtheta_dId{k} = a.dC2.dV{k};
% a.d2C2.dId_dtheta{k} = a.dC2.dV{k};
% a.d2C2.dtheta_dIq{k} = a.dC2.dV{k};
% a.d2C2.dIq_dtheta{k} = a.dC2.dV{k};
a.d2C2.ddelta_dId{k} = a.dC2.dV{k};
a.d2C2.dId_ddelta{k} = a.dC2.dV{k};
a.d2C2.ddelta_dIq{k} = a.dC2.dV{k};
a.d2C2.dIq_ddelta{k} = a.dC2.dV{k};
% a.d2C2.dV_ddelta{k} = a.dC2.dV{k};
% a.d2C2.ddelta_dV{k} = a.dC2.dV{k};
% a.d2C2.dV_dtheta{k} = a.dC2.dV{k};
% a.d2C2.dtheta_dV{k} = a.dC2.dV{k};
% a.d2C2.dtheta_ddelta{k} = a.dC2.dV{k};
% a.d2C2.ddelta_dtheta{k} = a.dC2.dV{k};
for m=1:data.nPgen
    a.d2C2.ddelta_ddelta{k,m} = a.dC2.dV{k};

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

end%for
for m=1:data.nbuses
a.dC2.dV{m} = zeros(2*data.nPgen,7*data.nPgen);
a.dC2.dtheta{m} = a.dC2.dV{m};
for n=1:data.nbuses
a.d2C2.dV_dtheta{m,n} = a.dC2.dV{m};
a.d2C2.dtheta_dV{m,n} = a.dC2.dV{m};
a.d2C2.dtheta_dtheta{m,n} = a.dC2.dV{k};
a.d2C2.dV_dV{m,n} = a.dC2.dV{k};
end%for
for mn=1:data.nPgen
a.d2C2.dV_dId{m,mn} = a.dC2.dV{k};
a.d2C2.dId_dV{mn,m} = a.dC2.dV{k};
a.d2C2.dV_dIq{m,mn} = a.dC2.dV{k};
a.d2C2.dIq_dV{mn,m} = a.dC2.dV{k};
a.d2C2.dtheta_dId{m,mn} = a.dC2.dV{k};
a.d2C2.dId_dtheta{mn,m} = a.dC2.dV{k};
a.d2C2.dtheta_dIq{m,mn} = a.dC2.dV{k};
a.d2C2.dIq_dtheta{mn,m} = a.dC2.dV{k};
a.d2C2.dV_ddelta{m,mn} = a.dC2.dV{k};
a.d2C2.ddelta_dV{mn,m} = a.dC2.dV{k};
a.d2C2.dtheta_ddelta{m,mn} = a.dC2.dV{k};
a.d2C2.ddelta_dtheta{mn,m} = a.dC2.dV{k};
end%for
end%for
%*****

%*****
a.dD3.dV{k} = zeros(2*data.nPgen,2*data.nPgen);
a.dD3.dtheta{k} = a.dD3.dV{k};
a.dD3.ddelta{k} = a.dD3.dV{k};
% a.d2D3.dV_ddelta{k} = a.dD3.dV{k};
% a.d2D3.ddelta_dV{k} = a.dD3.dV{k};
% a.d2D3.dV_dtheta{k} = a.dD3.dV{k};
% a.d2D3.dtheta_dV{k} = a.dD3.dV{k};
% a.d2D3.dtheta_ddelta{k} = a.dD3.dV{k};
% a.d2D3.ddelta_dtheta{k} = a.dD3.dV{k};
a.d2D3.dV_dV{k} = a.dD3.dV{k};
% a.d2D3.dtheta_dtheta{k} = a.dD3.dV{k};
for m=1:data.nPgen
a.d2D3.ddelta_ddelta{k,m} = a.dD3.dV{k};
end%for
for m=1:data.nbuses
a.dD3.dV{m} = zeros(2*data.nPgen,2*data.nPgen);
a.dD3.dtheta{m} = a.dD3.dV{m};
for n=1:data.nbuses
a.d2D3.dV_dtheta{m,n} = a.dD3.dV{m};
a.d2D3.dtheta_dV{m,n} = a.dD3.dV{m};
a.d2D3.dtheta_dtheta{m,n} = a.dD3.dV{k};
end%for
for mn=1:data.nPgen
a.d2D3.dV_ddelta{m,mn} = a.dD3.dV{k};
a.d2D3.ddelta_dV{mn,m} = a.dD3.dV{k};
a.d2D3.dtheta_ddelta{m,mn} = a.dD3.dV{k};
a.d2D3.ddelta_dtheta{mn,m} = a.dD3.dV{k};
end%for
end%for
%*****

%*****
a.dD4.dV{k} = zeros(2*data.nPgen,2*data.nPgen);
a.dD4.ddelta{k} = a.dD4.dV{k};
a.dD4.dId{k} = a.dD4.dV{k};
a.dD4.dIq{k} = a.dD4.dV{k};
% a.d2D4.dV_ddelta{k} = a.dD4.dV{k};

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

% a.d2D4.ddelta_dV{k} = a.dD4.dV{k};
% a.d2D4.dV_dtheta{k} = a.dD4.dV{k};
% a.d2D4.dtheta_dV{k} = a.dD4.dV{k};
% a.d2D4.dtheta_ddelta{k} = a.dD4.dV{k};
% a.d2D4.ddelta_dtheta{k} = a.dD4.dV{k};
a.d2D4.dV_dV{k} = a.dD4.dV{k};
% a.d2D4.dtheta_dtheta{k} = a.dD4.dV{k};
for m=1:data.nPgen
a.d2D4.ddelta_ddelta{k,m} = a.dD4.dV{k};
end%for
% a.d2D4.dV_dId{k} = a.dD4.dV{k};
% a.d2D4.dId_dV{k} = a.dD4.dV{k};
% a.d2D4.dV_dIq{k} = a.dD4.dV{k};
% a.d2D4.dIq_dV{k} = a.dD4.dV{k};
% a.d2D4.dtheta_dId{k} = a.dD4.dV{k};
% a.d2D4.dId_dtheta{k} = a.dD4.dV{k};
% a.d2D4.dtheta_dIq{k} = a.dD4.dV{k};
% a.d2D4.dIq_dtheta{k} = a.dD4.dV{k};
a.d2D4.ddelta_dId{k} = a.dD4.dV{k};
a.d2D4.dId_ddelta{k} = a.dD4.dV{k};
a.d2D4.ddelta_dIq{k} = a.dD4.dV{k};
a.d2D4.dIq_ddelta{k} = a.dD4.dV{k};
for m=1:data.nbuses
a.dD4.dV{m} = zeros(2*data.nPgen,2*data.nPgen);
a.dD4.dtheta{m} = a.dD4.dV{m};
for n=1:data.nbuses
a.d2D4.dV_dtheta{m,n} = a.dD4.dV{m};
a.d2D4.dtheta_dV{m,n} = a.dD4.dV{m};
a.d2D4.dtheta_dtheta{m,n} = a.dD4.dV{k};
end%for
for mn=1:data.nPgen
a.d2D4.dV_ddelta{m,mn} = a.dD4.dV{k};
a.d2D4.ddelta_dV{mn,m} = a.dD4.dV{k};
a.d2D4.dtheta_ddelta{m,mn} = a.dD4.dV{k};
a.d2D4.ddelta_dtheta{mn,m} = a.dD4.dV{k};
a.d2D4.dV_dId{m,mn} = a.dD4.dV{k};
a.d2D4.dId_dV{mn,m} = a.dD4.dV{k};
a.d2D4.dV_dIq{m,mn} = a.dD4.dV{k};
a.d2D4.dIq_dV{mn,m} = a.dD4.dV{k};
a.d2D4.dtheta_dId{m,mn} = a.dD4.dV{k};
a.d2D4.dId_dtheta{mn,m} = a.dD4.dV{k};
a.d2D4.dtheta_dIq{m,mn} = a.dD4.dV{k};
end%for
end%for
%*****

%*****
for m=1:data.nbuses
a.dD5.dV{m} = zeros(2*data.nPgen,2*data.nbuses-2*data.nPgen);
a.dD5.dtheta{m} = a.dD5.dV{m};
for n=1:data.nbuses
a.d2D5.dV_dtheta{m,n} = a.dD5.dV{m};
a.d2D5.dtheta_dV{m,n} = a.dD5.dV{m};
end%for
end%for
%*****

%*****
for m=1:data.nbuses
a.dD6.dV{m} = zeros(2*data.nPgen,2*data.nbuses-2*data.nPgen);
a.dD6.dtheta{m} = a.dD6.dV{m};
for n=1:data.nbuses
a.d2D6.dV_dtheta{m,n} = a.dD6.dV{m};
a.d2D6.dtheta_dV{m,n} = a.dD6.dV{m};
end%for

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

end%for
%*****

%*****
for m=1:data.nbuses
a.dD7.dV{m} = zeros(2*data.nPgen,2*data.nbuses-2*data.nPgen);
a.dD7.dtheta{m} = a.dD7.dV{m};
for n=1:data.nbuses
a.d2D7.dV_dtheta{m,n} = a.dD7.dV{m};
a.d2D7.dtheta_dV{m,n} = a.dD7.dV{m};
end%for
end%for
%*****

%*****
a.dK3.dV{k} = zeros(size(var.eigen.K3));
a.dK3.dtheta{k} = a.dK3.dV{k};
a.dK3.ddelta{k} = a.dK3.dV{k};
a.dK3.dId{k} = a.dK3.dV{k};
a.dK3.dIq{k} = a.dK3.dV{k};
% a.d2K3.ddelta_dId{k} = a.dK3.dV{k};
% a.d2K3.dId_ddelta{k} = a.dK3.dV{k};
% a.d2K3.ddelta_dIq{k} = a.dK3.dV{k};
% a.d2K3.dIq_ddelta{k} = a.dK3.dV{k};
for m=1:data.nPgen
a.d2K3.ddelta_dId{k,m}=a.dK3.dV{k};
a.d2K3.dId_ddelta{k,m}=a.dK3.dV{k};
a.d2K3.ddelta_dIq{k,m}=a.dK3.dV{k};
a.d2K3.dIq_ddelta{k,m}=a.dK3.dV{k};
a.d2K3.dEdp_ddelta{k,m}=a.dK3.dV{k};
a.d2K3.ddelta_dEdp{k,m}=a.dK3.dV{k};
a.d2K3.dEdp_ddelta{k,m}=a.dK3.dV{k};
a.d2K3.ddelta_dEdp{k,m}=a.dK3.dV{k};
a.d2K3.dEqp_ddelta{k,m}=a.dK3.dV{k};
a.d2K3.ddelta_dEqp{k,m}=a.dK3.dV{k};
a.d2K3.dEqp_ddelta{k,m}=a.dK3.dV{k};
a.d2K3.ddelta_dEqp{k,m}=a.dK3.dV{k};
a.d2K3.ddelta_ddelta{k,m} = a.dK3.dV{k};
end%for
for m=1:data.nbuses
a.dK3.dV{m} = zeros(size(var.eigen.K3));
a.dK3.dtheta{m} = a.dK3.dV{m};
for n=1:data.nbuses
a.d2K3.dV_dV{m,n} = a.dK3.dV{m};
a.d2K3.dV_dtheta{m,n} = a.dK3.dV{m};
a.d2K3.dtheta_dV{m,n} = a.dK3.dV{m};
a.d2K3.dtheta_dtheta{m,n} = a.dK3.dV{k};
end%for
for mn=1:data.nPgen
a.d2K3.dV_dId{m,mn} = a.dK3.dV{k};
a.d2K3.dId_dV{mn,m} = a.dK3.dV{k};
a.d2K3.dV_dIq{m,mn} = a.dK3.dV{k};
a.d2K3.dIq_dV{mn,m} = a.dK3.dV{k};
a.d2K3.dtheta_dId{m,mn} = a.dK3.dV{k};
a.d2K3.dId_dtheta{mn,m} = a.dK3.dV{k};
a.d2K3.dtheta_dIq{m,mn} = a.dK3.dV{k};
a.d2K3.dIq_dtheta{mn,m} = a.dK3.dV{k};
a.d2K3.dV_ddelta{m,mn} = a.dK3.dV{k};
a.d2K3.ddelta_dV{mn,m} = a.dK3.dV{k};
a.d2K3.dtheta_ddelta{m,mn} = a.dK3.dV{k};
a.d2K3.ddelta_dtheta{mn,m} = a.dK3.dV{k};
end%for
end%for
%*****

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

%*****
a.dK4.dV{k} = zeros(size(var.eigen.K4));
a.dK4.dtheta{k} = a.dK4.dV{k};
a.dK4.ddelta{k} = a.dK4.dV{k};
a.dK4.dId{k} = a.dK4.dV{k};
a.dK4.dIq{k} = a.dK4.dV{k};
% a.d2K4.ddelta_dId{k} = a.dK4.dV{k};
% a.d2K4.dId_ddelta{k} = a.dK4.dV{k};
% a.d2K4.ddelta_dIq{k} = a.dK4.dV{k};
% a.d2K4.dIq_ddelta{k} = a.dK4.dV{k};
% a.d2K4.ddelta_ddelta{k} = a.dK4.dV{k};
for m=1:data.nPgen
a.d2K4.ddelta_dId{k,m}=a.dK4.dV{k};
a.d2K4.dId_ddelta{k,m}=a.dK4.dV{k};
a.d2K4.ddelta_dIq{k,m}=a.dK4.dV{k};
a.d2K4.dIq_ddelta{k,m}=a.dK4.dV{k};
a.d2K4.dEdp_ddelta{k,m}=a.dK4.dV{k};
a.d2K4.ddelta_dEdp{k,m}=a.dK4.dV{k};
a.d2K4.dEdp_ddelta{k,m}=a.dK4.dV{k};
a.d2K4.ddelta_dEdp{k,m}=a.dK4.dV{k};
a.d2K4.dEqp_ddelta{k,m}=a.dK4.dV{k};
a.d2K4.ddelta_dEqp{k,m}=a.dK4.dV{k};
a.d2K4.dEqp_ddelta{k,m}=a.dK4.dV{k};
a.d2K4.ddelta_dEqp{k,m}=a.dK4.dV{k};
a.d2K4.ddelta_ddelta{k,m} = a.dK4.dV{k};
end%for
for m=1:data.nbuses
a.dK4.dV{m} = zeros(size(var.eigen.K4));
a.dK4.dtheta{m} = a.dK4.dV{m};
for n=1:data.nbuses
a.d2K4.dV_dtheta{m,n} = a.dK4.dV{m};
a.d2K4.dtheta_dV{m,n} = a.dK4.dV{m};
a.d2K4.dtheta_dtheta{m,n} = a.dK4.dV{k};
a.d2K4.dV_dV{m,n} = a.dK4.dV{k};
end%for
for mn=1:data.nPgen
a.d2K4.dV_dId{m,mn} = a.dK4.dV{k};
a.d2K4.dId_dV{mn,m} = a.dK4.dV{k};
a.d2K4.dV_dIq{m,mn} = a.dK4.dV{k};
a.d2K4.dIq_dV{mn,m} = a.dK4.dV{k};
a.d2K4.dtheta_dId{m,mn} = a.dK4.dV{k};
a.d2K4.dId_dtheta{mn,m} = a.dK4.dV{k};
a.d2K4.dtheta_dIq{m,mn} = a.dK4.dV{k};
a.d2K4.dIq_dtheta{mn,m} = a.dK4.dV{k};
a.d2K4.dV_ddelta{m,mn} = a.dK4.dV{k};
a.d2K4.ddelta_dV{mn,m} = a.dK4.dV{k};
a.d2K4.dtheta_ddelta{m,mn} = a.dK4.dV{k};
a.d2K4.ddelta_dtheta{mn,m} = a.dK4.dV{k};
end%for
end%for
%*****

%*****
a.dK5.dV{k} = zeros(size(var.eigen.K5));
a.dK5.dtheta{k} = a.dK5.dV{k};
a.dK5.ddelta{k} = a.dK5.dV{k};
a.dK5.dId{k} = a.dK5.dV{k};
a.dK5.dIq{k} = a.dK5.dV{k};
%a.d2K5.ddelta_dId{k} = a.dK5.dV{k};
%a.d2K5.dId_ddelta{k} = a.dK5.dV{k};
%a.d2K5.ddelta_dIq{k} = a.dK5.dV{k};
%a.d2K5.dIq_ddelta{k} = a.dK5.dV{k};
%a.d2K5.dV_dV{k} = a.dK5.dV{k};
%a.d2K5.ddelta_ddelta{k} = a.dK5.dV{k};
for m=1:data.nPgen

```


APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

a.d2K5.ddelta_dId{k,m}=a.dK5.dV{k};
a.d2K5.dId_ddelta{k,m}=a.dK5.dV{k};
a.d2K5.ddelta_dIq{k,m}=a.dK5.dV{k};
a.d2K5.dIq_ddelta{k,m}=a.dK5.dV{k};
a.d2K5.dEdp_ddelta{k,m}=a.dK5.dV{k};
a.d2K5.ddelta_dEdp{k,m}=a.dK5.dV{k};
a.d2K5.dEdp_ddelta{k,m}=a.dK5.dV{k};
a.d2K5.ddelta_dEdp{k,m}=a.dK5.dV{k};
a.d2K5.dEqp_ddelta{k,m}=a.dK5.dV{k};
a.d2K5.ddelta_dEqp{k,m}=a.dK5.dV{k};
a.d2K5.dEqp_ddelta{k,m}=a.dK5.dV{k};
a.d2K5.ddelta_dEqp{k,m}=a.dK5.dV{k};
a.d2K5.ddelta_ddelta{k,m} = a.dK5.dV{k};
end%for
for m=1:data.nbuses
a.dK5.dV{m} = zeros(size(var.eigen.K5));
a.dK5.dtheta{m} = a.dK5.dV{m};
for n=1:data.nbuses
a.d2K5.dV_dtheta{m,n} = a.dK5.dV{m};
a.d2K5.dtheta_dV{m,n} = a.dK5.dV{m};
a.d2K5.dtheta_dtheta{m,n} = a.dK5.dV{k};
a.d2K5.dV_dV{m,n} = a.dK5.dV{k};
end%for
for mn=1:data.nPgen
a.d2K5.dV_dId{m,mn} = a.dK5.dV{k};
a.d2K5.dId_dV{mn,m} = a.dK5.dV{k};
a.d2K5.dV_dIq{m,mn} = a.dK5.dV{k};
a.d2K5.dIq_dV{mn,m} = a.dK5.dV{k};
a.d2K5.dtheta_dId{m,mn} = a.dK5.dV{k};
a.d2K5.dId_dtheta{mn,m} = a.dK5.dV{k};
a.d2K5.dtheta_dIq{m,mn} = a.dK5.dV{k};
a.d2K5.dIq_dtheta{mn,m} = a.dK5.dV{k};
a.d2K5.dV_ddelta{m,mn} = a.dK5.dV{k};
a.d2K5.ddelta_dV{mn,m} = a.dK5.dV{k};
a.d2K5.dtheta_ddelta{m,mn} = a.dK5.dV{k};
a.d2K5.ddelta_dtheta{mn,m} = a.dK5.dV{k};
end%for
end%for
%*****
end%for

for k=1:data.nPgen
imag_ind = 2:2:2*data.nPgen;
real_ind = imag_ind-1;
%*****
A1Id_row_ind = 2:7:7*data.nPgen;
A1Id_col_ind = 3:7:7*data.nPgen;
A1Efd_row_ind = 5:7:7*data.nPgen;
A1Efd_col_ind = 5:7:7*data.nPgen;
a.dA1.dIq{k}(A1Id_row_ind(k),A1Id_col_ind(k)) = -1/M(k);
a.dA1.dId{k}(A1Id_row_ind(k),A1Id_col_ind(k)+1) = -1/M(k);
a.dA1.dEfd{k}(A1Efd_row_ind(k),A1Efd_col_ind(k)) = ...
    -1/data.machine.TE(k)*(2*sederiv(var.Efd(k))+ ...
    var.Efd(k)*se2deriv(var.Efd(k)));
a.d2A1.dEfd_dEfd{k}(A1Efd_row_ind(k),A1Efd_col_ind(k)) = ...
    -1/data.machine.TE(k)*(3*se2deriv(var.Efd(k))+ ...
    var.Efd(k)*se3deriv(var.Efd(k)));
%*****
%*****
B1row_ind = 2:7:7*data.nPgen;
B1col_ind = 1:2:2*data.nPgen;
a.dB1.dIq{k}(B1row_ind(k),B1col_ind(k)) = (data.machine.Xdp(k) - data.machine.Xqp(k))/M(k);

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

a.dB1.dId{k}(B1row_ind(k),B1col_ind(k)+1) = (data.machine.Xdp(k) - data.machine.Xqp(k))/M(k);
a.dB1.dEdp{k}(B1row_ind(k),B1col_ind(k)) = -1/M(k);
a.dB1.dEqp{k}(B1row_ind(k),B1col_ind(k)+1) = -1/M(k);
%*****

%*****
C1row_ind = 1:2:2*data.nPgen;
C1col_ind = 1:7:7*data.nPgen;
a.dC1.dV{k}(C1row_ind(k),C1col_ind(k)) = real(-ejTheta(k)*ejdelta_c(k));
a.dC1.dV{k}(C1row_ind(k)+1,C1col_ind(k)) = imag(-ejTheta(k)*ejdelta_c(k));
%-----
a.dC1.dtheta{k}(C1row_ind(k),C1col_ind(k)) = real(-j*Vp(k)*ejdelta_c(k));
a.dC1.dtheta{k}(C1row_ind(k)+1,C1col_ind(k)) = imag(-j*Vp(k)*ejdelta_c(k));
%-----
a.dC1.ddelta{k}(C1row_ind(k),C1col_ind(k)) = real(j*Vp(k)*ejdelta_c(k));
a.dC1.ddelta{k}(C1row_ind(k)+1,C1col_ind(k)) = imag(j*Vp(k)*ejdelta_c(k));
%-----
a.d2C1.ddelta_dV{k,k}(C1row_ind(k),C1col_ind(k)) = ...
    real(j*ejTheta(k)*ejdelta_c(k));
a.d2C1.ddelta_dV{k,k}(C1row_ind(k)+1,C1col_ind(k)) = ...
    imag(j*ejTheta(k)*ejdelta_c(k));
a.d2C1.dV_ddelta{k,k}(C1row_ind(k),C1col_ind(k)) = ...
    a.d2C1.ddelta_dV{k,k}(C1row_ind(k),C1col_ind(k));
a.d2C1.dV_ddelta{k,k}(C1row_ind(k)+1,C1col_ind(k)) = ...
    a.d2C1.ddelta_dV{k,k}(C1row_ind(k)+1,C1col_ind(k));
%-----
a.d2C1.ddelta_dtheta{k,k}(C1row_ind(k),C1col_ind(k)) = ...
    real(-Vp(k)*ejdelta_c(k));
a.d2C1.ddelta_dtheta{k,k}(C1row_ind(k)+1,C1col_ind(k)) = ...
    imag(-Vp(k)*ejdelta_c(k));
a.d2C1.dtheta_ddelta{k,k}(C1row_ind(k),C1col_ind(k)) = ...
    a.d2C1.ddelta_dtheta{k,k}(C1row_ind(k),C1col_ind(k));
a.d2C1.dtheta_ddelta{k,k}(C1row_ind(k)+1,C1col_ind(k)) = ...
    a.d2C1.ddelta_dtheta{k,k}(C1row_ind(k)+1,C1col_ind(k));
%-----
a.d2C1.dtheta_dV{k,k}(C1row_ind(k),C1col_ind(k)) = ...
    real(-j*ejTheta(k)*ejdelta_c(k));
a.d2C1.dtheta_dV{k,k}(C1row_ind(k)+1,C1col_ind(k)) = ...
    imag(-j*ejTheta(k)*ejdelta_c(k));
a.d2C1.dV_dtheta{k,k}(C1row_ind(k),C1col_ind(k)) = ...
    a.d2C1.dtheta_dV{k,k}(C1row_ind(k),C1col_ind(k));
a.d2C1.dV_dtheta{k,k}(C1row_ind(k)+1,C1col_ind(k)) = ...
    a.d2C1.dtheta_dV{k,k}(C1row_ind(k)+1,C1col_ind(k));
%-----
a.d2C1.dV_dV{k,k}(C1row_ind(k),C1col_ind(k)) = 0;
a.d2C1.dV_dV{k,k}(C1row_ind(k)+1,C1col_ind(k)) = 0;
%-----
a.d2C1.dtheta_dtheta{k,k}(C1row_ind(k),C1col_ind(k)) = ...
    real(Vp(k)*ejdelta_c(k));
a.d2C1.dtheta_dtheta{k,k}(C1row_ind(k)+1,C1col_ind(k)) = ...
    imag(Vp(k)*ejdelta_c(k));
%-----
a.d2C1.ddelta_ddelta{k,k}(C1row_ind(k),C1col_ind(k)) = ...
    real(Vp(k)*ejdelta_c(k));
a.d2C1.ddelta_ddelta{k,k}(C1row_ind(k)+1,C1col_ind(k)) = ...
    imag(Vp(k)*ejdelta_c(k));
%*****

%*****
a.dD2.dV{k}(real_ind(k),k+data.nPgen) = real(ejTheta(k)*ejdelta_c(k));
a.dD2.dV{k}(imag_ind(k),k+data.nPgen) = imag(ejTheta(k)*ejdelta_c(k));
%-----
a.dD2.dtheta{k}(real_ind(k),k+data.nPgen) = real(j*Vp(k)*ejdelta_c(k));
a.dD2.dtheta{k}(imag_ind(k),k+data.nPgen) = imag(j*Vp(k)*ejdelta_c(k));
a.dD2.dtheta{k}(real_ind(k),k) = real(ejTheta(k)*ejdelta_c(k));

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

a.dD2.dtheta{k}(imag_ind(k),k) = imag(ejTheta(k)*ejdelta_c(k));
%-----
a.dD2.ddelta{k}(real_ind(k),k+data.nPgen) = real(-j*Vp(k)*ejdelta_c(k));
a.dD2.ddelta{k}(imag_ind(k),k+data.nPgen) = imag(-j*Vp(k)*ejdelta_c(k));
a.dD2.ddelta{k}(real_ind(k),k) = real(-ejTheta(k)*ejdelta_c(k));
a.dD2.ddelta{k}(imag_ind(k),k) = imag(-ejTheta(k)*ejdelta_c(k));
%-----
a.d2D2.dV_dtheta{k,k}(real_ind(k),k+data.nPgen) = ...
    real(j*ejTheta(k)*ejdelta_c(k));
a.d2D2.dV_dtheta{k,k}(imag_ind(k),k+data.nPgen) = ...
    imag(j*ejTheta(k)*ejdelta_c(k));
a.d2D2.dtheta_dV{k,k}(real_ind(k),k+data.nPgen) = ...
    a.d2D2.dV_dtheta{k,k}(real_ind(k),k+data.nPgen);
a.d2D2.dtheta_dV{k,k}(imag_ind(k),k+data.nPgen) = ...
    a.d2D2.dV_dtheta{k,k}(imag_ind(k),k+data.nPgen);
%-----
a.d2D2.dV_ddelta{k,k}(real_ind(k),k+data.nPgen) = ...
    real(-j*ejTheta(k)*ejdelta_c(k));
a.d2D2.dV_ddelta{k,k}(imag_ind(k),k+data.nPgen) = ...
    imag(-j*ejTheta(k)*ejdelta_c(k));
a.d2D2.ddelta_dV{k,k}(real_ind(k),k+data.nPgen) = ...
    a.d2D2.dV_ddelta{k,k}(real_ind(k),k+data.nPgen);
a.d2D2.ddelta_dV{k,k}(imag_ind(k),k+data.nPgen) = ...
    a.d2D2.dV_ddelta{k,k}(imag_ind(k),k+data.nPgen);
%-----
a.d2D2.dtheta_ddelta{k,k}(real_ind(k),k+data.nPgen) = ...
    real(Vp(k)*ejdelta_c(k));
a.d2D2.dtheta_ddelta{k,k}(imag_ind(k),k+data.nPgen) = ...
    imag(Vp(k)*ejdelta_c(k));
a.d2D2.ddelta_dtheta{k,k}(real_ind(k),k+data.nPgen) = ...
    a.d2D2.dtheta_ddelta{k,k}(real_ind(k),k+data.nPgen);
a.d2D2.ddelta_dtheta{k,k}(imag_ind(k),k+data.nPgen) = ...
    a.d2D2.dtheta_ddelta{k,k}(imag_ind(k),k+data.nPgen);
%-----
a.d2D2.dtheta_ddelta{k,k}(real_ind(k),k) = ...
    real(-j*ejTheta(k)*ejdelta_c(k));
a.d2D2.dtheta_ddelta{k,k}(imag_ind(k),k) = ...
    imag(-j*ejTheta(k)*ejdelta_c(k));
a.d2D2.ddelta_dtheta{k,k}(real_ind(k),k) = ...
    a.d2D2.dtheta_ddelta{k,k}(real_ind(k),k+data.nPgen);
a.d2D2.ddelta_dtheta{k,k}(imag_ind(k),k) = ...
    a.d2D2.dtheta_ddelta{k,k}(imag_ind(k),k+data.nPgen);
%-----
a.d2D2.dtheta_dtheta{k,k}(real_ind(k),k+data.nPgen) = ...
    real(-Vp(k)*ejdelta_c(k));
a.d2D2.dtheta_dtheta{k,k}(imag_ind(k),k+data.nPgen) = ...
    imag(-Vp(k)*ejdelta_c(k));
a.d2D2.dtheta_dtheta{k,k}(real_ind(k),k) = ...
    real(j*ejTheta(k)*ejdelta_c(k));
a.d2D2.dtheta_dtheta{k,k}(imag_ind(k),k) = ...
    imag(j*ejTheta(k)*ejdelta_c(k));
%-----
a.d2D2.ddelta_ddelta{k,k}(real_ind(k),k+data.nPgen) = ...
    real(-Vp(k)*ejdelta_c(k));
a.d2D2.ddelta_ddelta{k,k}(imag_ind(k),k+data.nPgen) = ...
    imag(-Vp(k)*ejdelta_c(k));
a.d2D2.ddelta_ddelta{k,k}(real_ind(k),k) = ...
    real(j*ejTheta(k)*ejdelta_c(k));
a.d2D2.ddelta_ddelta{k,k}(imag_ind(k),k) = ...
    imag(j*ejTheta(k)*ejdelta_c(k));
%*****

real_ind = 1:data.nPgen;
imag_ind = real_ind+data.nPgen;
%*****

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

a.dC2.dV{k}(real_ind(k),delta_ind(k)) = real(Ig_c(k)*ejTheta(k)*ejdelta_c(k));
a.dC2.dV{k}(imag_ind(k),delta_ind(k)) = imag(Ig_c(k)*ejTheta(k)*ejdelta_c(k));
a.dC2.dtheta{k}(real_ind(k),delta_ind(k)) = ...
    real(Vp(k)*ejdelta_c(k)*j*Ig_c(k));
a.dC2.dtheta{k}(imag_ind(k),delta_ind(k)) = ...
    imag(Vp(k)*ejdelta_c(k)*j*Ig_c(k));
a.dC2.ddelta{k}(real_ind(k),delta_ind(k)) = ...
    real(-Vp(k)*ejdelta_c(k)*j*Ig_c(k));
a.dC2.ddelta{k}(imag_ind(k),delta_ind(k)) = ...
    imag(-Vp(k)*ejdelta_c(k)*j*Ig_c(k));
a.dC2.dId{k}(real_ind(k),delta_ind(k)) = ...
    real(Vp(k)*ejdelta_c(k));
a.dC2.dId{k}(imag_ind(k),delta_ind(k)) = ...
    imag(Vp(k)*ejdelta_c(k));
a.dC2.dIq{k}(real_ind(k),delta_ind(k)) = ...
    real(-j*Vp(k)*ejdelta_c(k));
a.dC2.dIq{k}(imag_ind(k),delta_ind(k)) = ...
    imag(-j*Vp(k)*ejdelta_c(k));
%-----
a.d2C2.dV_dId{k,k}(real_ind(k),delta_ind(k)) = ...
    real(ejTheta(k)*ejdelta_c(k));
a.d2C2.dV_dId{k,k}(imag_ind(k),delta_ind(k)) = ...
    imag(ejTheta(k)*ejdelta_c(k));
a.d2C2.dId_dV{k,k}(real_ind(k),delta_ind(k)) = ...
    a.d2C2.dV_dId{k,k}(real_ind(k),delta_ind(k));
a.d2C2.dId_dV{k,k}(imag_ind(k),delta_ind(k)) = ...
    a.d2C2.dV_dId{k,k}(imag_ind(k),delta_ind(k));
%-----
a.d2C2.dV_dIq{k,k}(real_ind(k),delta_ind(k)) = ...
    real(-j*ejTheta(k)*ejdelta_c(k));
a.d2C2.dV_dIq{k,k}(imag_ind(k),delta_ind(k)) = ...
    imag(-j*ejTheta(k)*ejdelta_c(k));
a.d2C2.dIq_dV{k,k}(real_ind(k),delta_ind(k)) = ...
    a.d2C2.dV_dIq{k,k}(real_ind(k),delta_ind(k));
a.d2C2.dIq_dV{k,k}(imag_ind(k),delta_ind(k)) = ...
    a.d2C2.dV_dIq{k,k}(imag_ind(k),delta_ind(k));
%-----
a.d2C2.dtheta_dId{k,k}(real_ind(k),delta_ind(k)) = ...
    real(j*Vp(k)*ejdelta_c(k));
a.d2C2.dtheta_dId{k,k}(imag_ind(k),delta_ind(k)) = ...
    imag(j*Vp(k)*ejdelta_c(k));
a.d2C2.dId_dtheta{k,k}(real_ind(k),delta_ind(k)) = ...
    a.d2C2.dtheta_dId{k,k}(real_ind(k),delta_ind(k));
a.d2C2.dId_dtheta{k,k}(imag_ind(k),delta_ind(k)) = ...
    a.d2C2.dtheta_dId{k,k}(imag_ind(k),delta_ind(k));
%-----
a.d2C2.dtheta_dIq{k,k}(real_ind(k),delta_ind(k)) = ...
    real(Vp(k)*ejdelta_c(k));
a.d2C2.dtheta_dIq{k,k}(imag_ind(k),delta_ind(k)) = ...
    imag(Vp(k)*ejdelta_c(k));
a.d2C2.dIq_dtheta{k,k}(real_ind(k),delta_ind(k)) = ...
    a.d2C2.dtheta_dIq{k,k}(real_ind(k),delta_ind(k));
a.d2C2.dIq_dtheta{k,k}(imag_ind(k),delta_ind(k)) = ...
    a.d2C2.dtheta_dIq{k,k}(imag_ind(k),delta_ind(k));
%-----
a.d2C2.ddelta_dId{k}(real_ind(k),delta_ind(k)) = ...
    real(-j*Vp(k)*ejdelta_c(k));
a.d2C2.ddelta_dId{k}(imag_ind(k),delta_ind(k)) = ...
    imag(-j*Vp(k)*ejdelta_c(k));
a.d2C2.dId_ddelta{k}(real_ind(k),delta_ind(k)) = ...
    a.d2C2.ddelta_dId{k}(real_ind(k),delta_ind(k));
a.d2C2.dId_ddelta{k}(imag_ind(k),delta_ind(k)) = ...
    a.d2C2.ddelta_dId{k}(imag_ind(k),delta_ind(k));
%-----
a.d2C2.ddelta_dIq{k}(real_ind(k),delta_ind(k)) = ...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

        real(-Vp(k)*ejdelta_c(k));
a.d2C2.ddelta_dIq{k}(imag_ind(k),delta_ind(k)) = ...
        imag(-Vp(k)*ejdelta_c(k));
a.d2C2.dIq_ddelta{k}(real_ind(k),delta_ind(k)) = ...
        a.d2C2.ddelta_dIq{k}(real_ind(k),delta_ind(k));
a.d2C2.dIq_ddelta{k}(imag_ind(k),delta_ind(k)) = ...
        a.d2C2.ddelta_dIq{k}(imag_ind(k),delta_ind(k));
%-----
a.d2C2.ddelta_dV{k,k}(real_ind(k),delta_ind(k)) = ...
        real(-j*Ig_c(k)*ejTheta(k)*ejdelta_c(k));
a.d2C2.ddelta_dV{k,k}(imag_ind(k),delta_ind(k)) = ...
        imag(-j*Ig_c(k)*ejTheta(k)*ejdelta_c(k));
a.d2C2.dV_ddelta{k,k}(real_ind(k),delta_ind(k)) = ...
        a.d2C2.ddelta_dV{k,k}(real_ind(k),delta_ind(k));
a.d2C2.dV_ddelta{k,k}(imag_ind(k),delta_ind(k)) = ...
        a.d2C2.ddelta_dV{k,k}(imag_ind(k),delta_ind(k));
%-----
a.d2C2.dtheta_dV{k,k}(real_ind(k),delta_ind(k)) = ...
        real(j*Ig_c(k)*ejTheta(k)*ejdelta_c(k));
a.d2C2.dtheta_dV{k,k}(imag_ind(k),delta_ind(k)) = ...
        imag(j*Ig_c(k)*ejTheta(k)*ejdelta_c(k));
a.d2C2.dV_dtheta{k,k}(real_ind(k),delta_ind(k)) = ...
        a.d2C2.dtheta_dV{k,k}(real_ind(k),delta_ind(k));
a.d2C2.dV_dtheta{k,k}(imag_ind(k),delta_ind(k)) = ...
        a.d2C2.dtheta_dV{k,k}(imag_ind(k),delta_ind(k));
%-----
a.d2C2.ddelta_dtheta{k,k}(real_ind(k),delta_ind(k)) = ...
        real(Ig_c(k)*Vp(k)*ejdelta_c(k));
a.d2C2.ddelta_dtheta{k,k}(imag_ind(k),delta_ind(k)) = ...
        imag(Ig_c(k)*Vp(k)*ejdelta_c(k));
a.d2C2.dtheta_ddelta{k,k}(real_ind(k),delta_ind(k)) = ...
        a.d2C2.ddelta_dtheta{k,k}(real_ind(k),delta_ind(k));
a.d2C2.dtheta_ddelta{k,k}(imag_ind(k),delta_ind(k)) = ...
        a.d2C2.ddelta_dtheta{k,k}(imag_ind(k),delta_ind(k));
%-----
a.d2C2.dtheta_dtheta{k,k}(real_ind(k),delta_ind(k)) = ...
        real(-Ig_c(k)*Vp(k)*ejdelta_c(k));
a.d2C2.dtheta_dtheta{k,k}(imag_ind(k),delta_ind(k)) = ...
        imag(-Ig_c(k)*Vp(k)*ejdelta_c(k));
%-----
a.d2C2.ddelta_ddelta{k}(real_ind(k),delta_ind(k)) = ...
        real(-Ig_c(k)*Vp(k)*ejdelta_c(k));
a.d2C2.ddelta_ddelta{k}(imag_ind(k),delta_ind(k)) = ...
        imag(-Ig_c(k)*Vp(k)*ejdelta_c(k));
%*****

%*****
a.dD3.dV{k}(real_ind(k),Id_ind(k)) = ...
        real(j*ejTheta(k)*ejdelta_c(k));
a.dD3.dV{k}(imag_ind(k),Id_ind(k)) = ...
        imag(j*ejTheta(k)*ejdelta_c(k));
a.dD3.dV{k}(real_ind(k),Iq_ind(k)) = ...
        real(ejTheta(k)*ejdelta_c(k));
a.dD3.dV{k}(imag_ind(k),Iq_ind(k)) = ...
        imag(ejTheta(k)*ejdelta_c(k));
%-----
a.dD3.dtheta{k}(real_ind(k),Id_ind(k)) = ...
        real(-Vp(k)*ejdelta_c(k));
a.dD3.dtheta{k}(imag_ind(k),Id_ind(k)) = ...
        imag(-Vp(k)*ejdelta_c(k));
a.dD3.dtheta{k}(real_ind(k),Iq_ind(k)) = ...
        real(j*Vp(k)*ejdelta_c(k));
a.dD3.dtheta{k}(imag_ind(k),Iq_ind(k)) = ...
        imag(j*Vp(k)*ejdelta_c(k));
%-----

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

a.dD3.ddelta{k}(real_ind(k),Id_ind(k)) = ...
    real(Vp(k)*ejdelta_c(k));
a.dD3.ddelta{k}(imag_ind(k),Id_ind(k)) = ...
    imag(Vp(k)*ejdelta_c(k));
a.dD3.ddelta{k}(real_ind(k),Iq_ind(k)) = ...
    real(-j*Vp(k)*ejdelta_c(k));
a.dD3.ddelta{k}(imag_ind(k),Iq_ind(k)) = ...
    imag(-j*Vp(k)*ejdelta_c(k));
%-----
a.d2D3.dV_ddelta{k,k}(real_ind(k),Id_ind(k)) = ...
    real(ejTheta(k)*ejdelta_c(k));
a.d2D3.dV_ddelta{k,k}(imag_ind(k),Id_ind(k)) = ...
    imag(ejTheta(k)*ejdelta_c(k));
a.d2D3.ddelta_dV{k,k}(real_ind(k),Id_ind(k)) = ...
    a.d2D3.dV_ddelta{k,k}(real_ind(k),Id_ind(k));
a.d2D3.ddelta_dV{k,k}(imag_ind(k),Id_ind(k)) = ...
    a.d2D3.dV_ddelta{k,k}(imag_ind(k),Id_ind(k));
%-----
a.d2D3.dV_ddelta{k,k}(real_ind(k),Iq_ind(k)) = ...
    real(-j*ejTheta(k)*ejdelta_c(k));
a.d2D3.dV_ddelta{k,k}(imag_ind(k),Iq_ind(k)) = ...
    imag(-j*ejTheta(k)*ejdelta_c(k));
a.d2D3.ddelta_dV{k,k}(real_ind(k),Iq_ind(k)) = ...
    a.d2D3.dV_ddelta{k,k}(real_ind(k),Iq_ind(k));
a.d2D3.ddelta_dV{k,k}(imag_ind(k),Iq_ind(k)) = ...
    a.d2D3.dV_ddelta{k,k}(imag_ind(k),Iq_ind(k));
%-----
a.d2D3.dV_dtheta{k,k}(real_ind(k),Id_ind(k)) = ...
    real(-ejTheta(k)*ejdelta_c(k));
a.d2D3.dV_dtheta{k,k}(imag_ind(k),Id_ind(k)) = ...
    imag(-ejTheta(k)*ejdelta_c(k));
a.d2D3.dtheta_dV{k,k}(real_ind(k),Id_ind(k)) = ...
    a.d2D3.dV_dtheta{k,k}(real_ind(k),Id_ind(k));
a.d2D3.dtheta_dV{k,k}(imag_ind(k),Id_ind(k)) = ...
    a.d2D3.dV_dtheta{k,k}(imag_ind(k),Id_ind(k));
%-----
a.d2D3.dV_dtheta{k,k}(real_ind(k),Iq_ind(k)) = ...
    real(j*ejTheta(k)*ejdelta_c(k));
a.d2D3.dV_dtheta{k,k}(imag_ind(k),Iq_ind(k)) = ...
    imag(j*ejTheta(k)*ejdelta_c(k));
a.d2D3.dtheta_dV{k,k}(real_ind(k),Iq_ind(k)) = ...
    a.d2D3.dV_dtheta{k,k}(real_ind(k),Iq_ind(k));
a.d2D3.dtheta_dV{k,k}(imag_ind(k),Iq_ind(k)) = ...
    a.d2D3.dV_dtheta{k,k}(imag_ind(k),Iq_ind(k));
%-----
a.d2D3.ddelta_dtheta{k,k}(real_ind(k),Id_ind(k)) = ...
    real(j*Vp(k)*ejdelta_c(k));
a.d2D3.ddelta_dtheta{k,k}(imag_ind(k),Id_ind(k)) = ...
    imag(j*Vp(k)*ejdelta_c(k));
a.d2D3.dtheta_ddelta{k,k}(real_ind(k),Id_ind(k)) = ...
    a.d2D3.ddelta_dtheta{k,k}(real_ind(k),Id_ind(k));
a.d2D3.dtheta_ddelta{k,k}(imag_ind(k),Id_ind(k)) = ...
    a.d2D3.ddelta_dtheta{k,k}(imag_ind(k),Id_ind(k));
%-----
a.d2D3.ddelta_dtheta{k,k}(real_ind(k),Iq_ind(k)) = ...
    real(Vp(k)*ejdelta_c(k));
a.d2D3.ddelta_dtheta{k,k}(imag_ind(k),Iq_ind(k)) = ...
    imag(Vp(k)*ejdelta_c(k));
a.d2D3.dtheta_ddelta{k,k}(real_ind(k),Iq_ind(k)) = ...
    a.d2D3.ddelta_dtheta{k,k}(real_ind(k),Iq_ind(k));
a.d2D3.dtheta_ddelta{k,k}(imag_ind(k),Iq_ind(k)) = ...
    a.d2D3.ddelta_dtheta{k,k}(imag_ind(k),Iq_ind(k));
%-----
a.d2D3.dtheta_dtheta{k,k}(real_ind(k),Id_ind(k)) = ...
    real(-j*Vp(k)*ejdelta_c(k));

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

a.d2D3.dtheta_dtheta{k,k}(imag_ind(k),Id_ind(k)) = ...
    imag(-j*Vp(k)*ejdelta_c(k));
a.d2D3.dtheta_dtheta{k,k}(real_ind(k),Iq_ind(k)) = ...
    real(-Vp(k)*ejdelta_c(k));
a.d2D3.dtheta_dtheta{k,k}(imag_ind(k),Iq_ind(k)) = ...
    imag(-Vp(k)*ejdelta_c(k));
%-----
a.d2D3.ddelta_ddelta{k,k}(real_ind(k),Id_ind(k)) = ...
    real(-j*Vp(k)*ejdelta_c(k));
a.d2D3.ddelta_ddelta{k,k}(imag_ind(k),Id_ind(k)) = ...
    imag(-j*Vp(k)*ejdelta_c(k));
a.d2D3.ddelta_ddelta{k,k}(real_ind(k),Iq_ind(k)) = ...
    real(-Vp(k)*ejdelta_c(k));
a.d2D3.ddelta_ddelta{k,k}(imag_ind(k),Iq_ind(k)) = ...
    imag(-Vp(k)*ejdelta_c(k));
%*****

end%for

a.dA1.dId{1}(9,4) = +1/M(1);
a.dA1.dIq{1}(9,3) = +1/M(1);
a.dA1.dId{1}(16,4) = +1/M(1);
a.dA1.dIq{1}(16,3) = +1/M(1);

a.dB1.dIq{1}(9,1) = -(data.machine.Xdp(1) - data.machine.Xqp(k))/M(1);
a.dB1.dId{1}(9,2) = -(data.machine.Xdp(1) - data.machine.Xqp(k))/M(1);
a.dB1.dEdp{1}(9,1) = +1/M(1);
a.dB1.dEqp{1}(9,2) = +1/M(1);
a.dB1.dIq{1}(16,1) = -(data.machine.Xdp(1) - data.machine.Xqp(k))/M(1);
a.dB1.dId{1}(16,2) = -(data.machine.Xdp(1) - data.machine.Xqp(k))/M(1);
a.dB1.dEdp{1}(16,1) = +1/M(1);
a.dB1.dEqp{1}(16,2) = +1/M(1);

%*****
% Derivative of Jacobian wrt decision variables
%*****
fr = data.fr;
to = data.to;
ejTheta = exp(j*var.theta);
ejgamma = exp(j*var.gamma);
ejdelta = exp(j*var.delta);
Vp = var.V .* ejTheta;
Ig = var.Ig.* ejgamma;
Adelt = var.Adelt.* ejdelta;
Ip = data.Ybus * Vp;
Sp = Vp.*conj(Ip);
Sij = Vp(fr).*conj(var.Iij);
line_ind_fr = sparse(fr,1:data.nlines,ones(data.nlines,1),data.nbuses,data.nlines);
line_ind_to = sparse(to,1:data.nlines,ones(data.nlines,1),data.nbuses,data.nlines);
Yvec_nc = conj(-data.Yvec);
ejTheta_c = conj(ejTheta);
Ybus_diag = line_ind_fr*data.Yvec + line_ind_fr*j*data.Bvec/2 ...
    + line_ind_to*data.Yvec+line_ind_to*j*data.Bvec/2;

a.JasysVV = cell(data.nbuses,1);
a.Jasystt = cell(data.nbuses,1);
a.JasysV = cell(data.nbuses,1);
a.JasysVt = cell(data.nbuses,1);
a.HasysVV = cell(data.nbuses,data.nbuses);
for k=1:data.nbuses
    fr_ind = find(data.fr==k);
    to_ind = find(data.to==k);
    fr = [data.fr(fr_ind); data.to(to_ind)];

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

to = [data.to(fr_ind); data.fr(to_ind)];
ind = [fr_ind; to_ind];
a.JasysVV{k} = sparse([to; k; to], ...
                    [fr; k; to], ...
                    [ejTheta(fr).*Yvec_nc(ind).*ejTheta_c(to); ...
                    2*conj(Ybus_diag(k)); ...
                    ejTheta(to).*Yvec_nc(ind).*ejTheta_c(fr)],data.nbuses,data.nbuses);
a.JasysVV{k} = a.JasysVV{k}.';
%*****
a.Jasystt{k} = ...
    sparse([to; k; to; fr], ...
          [fr; k; to; to], ...
          [Vp(fr).*Yvec_nc(ind).*conj(Vp(to)); ...
          -Sp(k)+Vp(k)*conj(Ybus_diag(k))*conj(Vp(k)); ...
          Vp(to).*Yvec_nc(ind).*conj(Vp(fr)); ...
          -Vp(to).*Yvec_nc(ind).*conj(Vp(fr))], ...
          data.nbuses,data.nbuses);
a.Jasystt{k} = a.Jasystt{k}.';
%*****
a.JasystV{k} = ...
    sparse([to; k; to; fr], ...
          [fr; k; to; to], ...
          [j*Vp(fr).*Yvec_nc(ind).*ejTheta_c(to); ...
          j*ejTheta(k)*conj(Ip(k))-j*var.V(k)*conj(Ybus_diag(k)); ...
          -j*ejTheta(to).*Yvec_nc(ind).*conj(Vp(fr)); ...
          -j*Vp(to).*Yvec_nc(ind).*ejTheta_c(fr)], ...
          data.nbuses,data.nbuses);
a.JasystV{k} = a.JasystV{k}.';
%*****
a.JasysVt{k} = ...
    sparse([to; k; to; fr], ...
          [fr; k; to; to], ...
          [-j*ejTheta(fr).*Yvec_nc(ind).*conj(Vp(to)); ...
          j*ejTheta(k)*conj(Ip(k))-j*var.V(k)*conj(Ybus_diag(k)); ...
          j*Vp(to).*Yvec_nc(ind).*ejTheta_c(fr); ...
          -j*Vp(to).*Yvec_nc(ind).*ejTheta_c(fr)], ...
          data.nbuses,data.nbuses);
a.JasysVt{k} = a.JasysVt{k}.';
%*****
% Second Derivative of Jacobian wrt decision variables
%*****
for m=1:data.nbuses
    if m==k
        % a.HasystVW = dHasysVV{k}/dtheta(m)
        a.HasystVW{k,m}= sparse([to; to], ...
                                [fr; to], ...
                                [j*ejTheta(fr).*Yvec_nc(ind).*ejTheta_c(to); ...
                                -j*ejTheta(to).*Yvec_nc(ind).*ejTheta_c(fr)],data.nbuses,data.nbuses);
%*****
a.HasystVtV{k,m} = ...
    sparse([to; to], ...
          [fr; to], ...
          [j*ejTheta(fr).*Yvec_nc(ind).*ejTheta_c(to); ...
          -j*ejTheta(to).*Yvec_nc(ind).*ejTheta_c(fr)], ...
          data.nbuses,data.nbuses);
%*****
a.HasysVtV{k,m} = ...
    sparse(data.nbuses,data.nbuses);
%*****
a.HasysttV{k,m} = ...
    sparse([to; k; to; fr], ...
          [fr; k; to; to], ...
          [-Vp(fr).*Yvec_nc(ind).*ejTheta_c(to); ...
          -ejTheta(k)*conj(Ip(k))+var.V(k)*conj(Ybus_diag(k)); ...
          -ejTheta(to).*Yvec_nc(ind).*conj(Vp(fr)); ...

```


APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

-Vp(to).*Yvec_nc(ind).*ejTheta_c(fr)], ...
data.nbuses,data.nbuses);
%*****
a.HasysVtt{k,m} = sparse([to; k; to; fr], ...
    [fr; k; to; to], ...
    [ejTheta(fr).*Yvec_nc(ind).*conj(Vp(to)); ...
    -ejTheta(k).*conj(Ip(k))+var.V(k).*conj(Ybus_diag(k)); ...
    Vp(to).*Yvec_nc(ind).*ejTheta_c(fr); ...
    -Vp(to).*Yvec_nc(ind).*ejTheta_c(fr)], ...
    data.nbuses,data.nbuses);
%*****
a.HasystVt{k,m} = sparse([to; k; to; fr], ...
    [fr; k; to; to], ...
    [ejTheta(fr).*Yvec_nc(ind).*conj(Vp(to)); ...
    -ejTheta(k).*conj(Ip(k))+var.V(k).*conj(Ybus_diag(k)); ...
    Vp(to).*Yvec_nc(ind).*ejTheta_c(fr); ...
    -Vp(to).*Yvec_nc(ind).*ejTheta_c(fr)], ...
    data.nbuses,data.nbuses);
%*****
a.Hasysttt{k,m} = sparse([to; k; to; fr], ...
    [fr; k; to; to], ...
    [ j*Vp(fr).*Yvec_nc(ind).*conj(Vp(to)); ...
    -j*Sp(k)+j*Vp(k).*conj(Ybus_diag(k)).*conj(Vp(k)); ...
    -j*Vp(to).*Yvec_nc(ind).*conj(Vp(fr)); ...
    j*Vp(to).*Yvec_nc(ind).*conj(Vp(fr))], ...
    data.nbuses,data.nbuses);
end%if
if m~=k
if isempty(find(to==m))==0
%*****
a.HasystVV{k,m} = sparse([k; k],[m; k], ...
    [-j*ejTheta(m).*Yvec_nc(ind(find(to==m))).*ejTheta_c(k); ...
    *ejTheta_c(k); ...
    j*ejTheta(k).*Yvec_nc(ind(find(to==m))).*ejTheta_c(m); ...
    *ejTheta_c(m)],data.nbuses,data.nbuses);
%*****
a.HasysVtt{k,m} = sparse([m; k; m; k], ...
    [k; k; m; m], ...
    [-ejTheta(k).*Yvec_nc(ind(find(to==m))).*conj(Vp(m)); ...
    ejTheta(k).*Yvec_nc(ind(find(to==m))).*conj(Vp(m)); ...
    -Vp(m).*Yvec_nc(ind(find(to==m))).*ejTheta_c(k); ...
    Vp(m).*Yvec_nc(ind(find(to==m))).*ejTheta_c(k)], ...
    data.nbuses,data.nbuses);
%*****
a.HasystVt{k,m} = sparse([m; k; m; k], ...
    [k; k; m; m], ...
    [ Vp(k).*Yvec_nc(ind(find(to==m))).*ejTheta_c(m); ...
    ejTheta(m).*Yvec_nc(ind(find(to==m))).*conj(Vp(k)); ...
    -Vp(k).*Yvec_nc(ind(find(to==m))).*ejTheta_c(m); ...
    -ejTheta(m).*Yvec_nc(ind(find(to==m))).*conj(Vp(k))], ...
    data.nbuses,data.nbuses);
%*****
a.HasysVvt{k,m} = sparse([m; k; m; k], ...
    [k; k; m; m], ...
    [ -j*ejTheta(k).*Yvec_nc(ind(find(to==m))).*ejTheta_c(m); ...
    ejTheta(m).*Yvec_nc(ind(find(to==m))).*conj(Vp(k)); ...
    j*ejTheta(k).*Yvec_nc(ind(find(to==m))).*ejTheta_c(m); ...
    -j*ejTheta(m).*Yvec_nc(ind(find(to==m))).*ejTheta_c(k)], ...
    data.nbuses,data.nbuses);
%*****
a.Hasysttt{k,m} = sparse([m; k; m; k], ...
    [k; k; m; m], ...
    [-j*Vp(k).*Yvec_nc(ind(find(to==m))).*conj(Vp(m)); ...
    j*Vp(k).*Yvec_nc(ind(find(to==m))).*conj(Vp(m)); ...
    j*Vp(m).*Yvec_nc(ind(find(to==m))).*conj(Vp(k)); ...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

        -j*Vp(m).*Yvec_nc(ind(find(to==m))).*conj(Vp(k))], ...
        data.nbuses,data.nbuses);
%*****
a.HasysVtV{k,m} = ...
    sparse([ k; k], ...
           [ k; m], ...
           [ j*ejTheta(k)*Yvec_nc(ind(find(to==m))).*ejTheta_c(m); ...
             -j*ejTheta(m).*Yvec_nc(ind(find(to==m))).*ejTheta_c(k)], ...
           data.nbuses,data.nbuses);
%*****
a.HasysttV{k,m} = ...
    sparse([m; k; m; k], ...
           [m; k; k; m], ...
           [ejTheta(m).*Yvec_nc(ind(find(to==m))).*conj(Vp(k)); ...
            ejTheta(k).*Yvec_nc(ind(find(to==m))).*conj(Vp(m)); ...
            Vp(k).*Yvec_nc(ind(find(to==m))).*ejTheta_c(m); ...
            Vp(m).*Yvec_nc(ind(find(to==m))).*ejTheta_c(k)], ...
           data.nbuses,data.nbuses);
end%if
if isempty(find(to==m))==1
    a.HasystVV{k,m}=sparse(data.nbuses,data.nbuses);
    a.HasysVtV{k,m}=sparse(data.nbuses,data.nbuses);
    a.HasysttV{k,m}=sparse(data.nbuses,data.nbuses);
    a.HasysVtt{k,m}=sparse(data.nbuses,data.nbuses);
    a.HasystVt{k,m}=sparse(data.nbuses,data.nbuses);
    a.HasysVvt{k,m}=sparse(data.nbuses,data.nbuses);
    a.Hasysttt{k,m}=sparse(data.nbuses,data.nbuses);
end%if
    a.HasysVVV{k,m}=sparse(data.nbuses,data.nbuses);
end%if
a.HasystVV{k,m} = a.HasystVV{k,m}.';
a.HasysVtV{k,m} = a.HasysVtV{k,m}.';
a.HasysttV{k,m} = a.HasysttV{k,m}.';
a.HasysVvt{k,m} = a.HasysVvt{k,m}.';
a.HasystVt{k,m} = a.HasystVt{k,m}.';
a.HasysVtt{k,m} = a.HasysVtt{k,m}.';
a.Hasysttt{k,m} = a.Hasysttt{k,m}.';
end%for
end%for
%*****
% Test Jasysxx
%*****
% a.h_S_VV=zeros(data.nbuses,data.nbuses);
% a.h_S_tt=zeros(data.nbuses,data.nbuses);
% a.h_S_tV=zeros(data.nbuses,data.nbuses);
% a.h_S_Vt=zeros(data.nbuses,data.nbuses);
% for k=1:data.nbuses
% a.h_S_VV(:,k) = a.h_S_VV(:,k) ...
%     +real(a.JasysVV{k}.')*var.lambda.P+imag(a.JasysVV{k}.')*var.lambda.Q;
% a.h_S_tt(:,k) = a.h_S_tt(:,k) ...
%     +real(a.Jasystt{k}.')*var.lambda.P+imag(a.Jasystt{k}.')*var.lambda.Q;
% a.h_S_tV(:,k) = a.h_S_tV(:,k) ...
%     +real(a.JasystV{k}.')*var.lambda.P+imag(a.JasystV{k}.')*var.lambda.Q;
% a.h_S_Vt(:,k) = a.h_S_Vt(:,k) ...
%     +real(a.JasysVt{k}.')*var.lambda.P+imag(a.JasysVt{k}.')*var.lambda.Q;
% end%for

%*****
% D4-D7 derivatives
%*****
for k=1:data.nPgen
%*****
a.dD4.dV{k} = ...
    -[real(a.JasysVV{k}(data.Pgenbuses,data.Pgenbuses)) ...
      real(a.JasysVt{k}(data.Pgenbuses,data.Pgenbuses))

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

    imag(a.JasysVv{k}(data.Pgenbuses,data.Pgenbuses)) ...
    imag(a.JasysVt{k}(data.Pgenbuses,data.Pgenbuses))] ...
+sparse([real_ind(k) imag_ind(k)],[k+data.nPgen k+data.nPgen], ...
    [real(-Ig_c(k)*ejTheta(k)*ejdelta_c(k)) ...
    imag(-Ig_c(k)*ejTheta(k)*ejdelta_c(k))], ...
    2*data.nPgen,2*data.nPgen);
%*****
a.dD4.dtheta{k} = ...
-[real(a.JasystV{k}(data.Pgenbuses,data.Pgenbuses)) ...
real(a.Jasystt{k}(data.Pgenbuses,data.Pgenbuses))
imag(a.JasystV{k}(data.Pgenbuses,data.Pgenbuses)) ...
imag(a.Jasystt{k}(data.Pgenbuses,data.Pgenbuses))] ...
+sparse([real_ind(k) imag_ind(k) real_ind(k) imag_ind(k)],...
    [k k k+data.nPgen k+data.nPgen], ...
    [real(-Ig_c(k)*ejTheta(k)*ejdelta_c(k)) ...
    imag(-Ig_c(k)*ejTheta(k)*ejdelta_c(k)) ...
    real(-j*Ig_c(k)*Vp(k)*ejdelta_c(k)) ...
    imag(-j*Ig_c(k)*Vp(k)*ejdelta_c(k))], ...
    2*data.nPgen,2*data.nPgen);
%*****
a.dD4.ddelta{k} = ...
sparse([real_ind(k) imag_ind(k) real_ind(k) imag_ind(k)],...
    [k k k+data.nPgen k+data.nPgen], ...
    [real(Ig_c(k)*ejTheta(k)*ejdelta_c(k)) ...
    imag(Ig_c(k)*ejTheta(k)*ejdelta_c(k)) ...
    real(j*Ig_c(k)*Vp(k)*ejdelta_c(k)) ...
    imag(j*Ig_c(k)*Vp(k)*ejdelta_c(k))], ...
    2*data.nPgen,2*data.nPgen);
%*****
a.dD4.dId{k} = ...
sparse([real_ind(k) imag_ind(k) real_ind(k) imag_ind(k)],...
    [k k k+data.nPgen k+data.nPgen], ...
    [real(j*ejTheta(k)*ejdelta_c(k)) ...
    imag(j*ejTheta(k)*ejdelta_c(k)) ...
    real(-Vp(k)*ejdelta_c(k)) ...
    imag(-Vp(k)*ejdelta_c(k))], ...
    2*data.nPgen,2*data.nPgen);
%[real((j+var.Iq(k))*ejTheta(k)*ejdelta_c(k)) ...
% imag((j+var.Iq(k))*Ig_c(k)*ejTheta(k)*ejdelta_c(k)) ...
% real((-1+j*var.Iq(k))*Vp(k)*ejdelta_c(k)) ...
% imag((-1+j*var.Iq(k))*Vp(k)*ejdelta_c(k))], ...
% 2*data.nPgen,2*data.nPgen);
%*****
a.dD4.dIq{k} = ...
sparse([real_ind(k) imag_ind(k) real_ind(k) imag_ind(k)],...
    [k k k+data.nPgen k+data.nPgen], ...
    [real(ejTheta(k)*ejdelta_c(k)) ...
    imag(ejTheta(k)*ejdelta_c(k)) ...
    real(j*Vp(k)*ejdelta_c(k)) ...
    imag(j*Vp(k)*ejdelta_c(k))], ...
    2*data.nPgen,2*data.nPgen);
%[real((1+j*var.Id(k))*ejTheta(k)*ejdelta_c(k)) ...
% imag((1+j*var.Id(k))*Ig_c(k)*ejTheta(k)*ejdelta_c(k)) ...
% real((j-var.Id(k))*Vp(k)*ejdelta_c(k)) ...
% imag((j-var.Id(k))*Vp(k)*ejdelta_c(k))], ...
% 2*data.nPgen,2*data.nPgen);
%-----
a.d2D4.dtheta_dId{k,k} = ...
sparse([real_ind(k) imag_ind(k) real_ind(k) imag_ind(k)],...
    [k k k+data.nPgen k+data.nPgen], ...
    [real(-ejTheta(k)*ejdelta_c(k)) ...
    imag(-ejTheta(k)*ejdelta_c(k)) ...
    real(-j*Vp(k)*ejdelta_c(k)) ...
    imag(-j*Vp(k)*ejdelta_c(k))], ...
    2*data.nPgen,2*data.nPgen);

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

a.d2D4.dId_dtheta{k,k} = ...
    a.d2D4.dtheta_dId{k,k};
%-----
a.d2D4.dtheta_dIq{k,k} = ...
    sparse([real_ind(k) imag_ind(k) real_ind(k) imag_ind(k)],...
           [k k k+data.nPgen k+data.nPgen], ...
           [real(j*ejTheta(k)*ejdelta_c(k)) ...
            imag(j*ejTheta(k)*ejdelta_c(k)) ...
            real(-Vp(k)*ejdelta_c(k)) ...
            imag(-Vp(k)*ejdelta_c(k))], ...
           2*data.nPgen,2*data.nPgen);
a.d2D4.dIq_dtheta{k,k} = ...
    a.d2D4.dtheta_dIq{k,k};
%-----
a.d2D4.ddelta_dId{k} = ...
    sparse([real_ind(k) imag_ind(k) real_ind(k) imag_ind(k)],...
           [k k k+data.nPgen k+data.nPgen], ...
           [real(ejTheta(k)*ejdelta_c(k)) ...
            imag(ejTheta(k)*ejdelta_c(k)) ...
            real(j*Vp(k)*ejdelta_c(k)) ...
            imag(j*Vp(k)*ejdelta_c(k))], ...
           2*data.nPgen,2*data.nPgen);
a.d2D4.dId_ddelta{k} = ...
    a.d2D4.ddelta_dId{k};
%-----
a.d2D4.ddelta_dIq{k} = ...
    sparse([real_ind(k) imag_ind(k) real_ind(k) imag_ind(k)],...
           [k k k+data.nPgen k+data.nPgen], ...
           [real(-j*ejTheta(k)*ejdelta_c(k)) ...
            imag(-j*ejTheta(k)*ejdelta_c(k)) ...
            real(Vp(k)*ejdelta_c(k)) ...
            imag(Vp(k)*ejdelta_c(k))], ...
           2*data.nPgen,2*data.nPgen);
a.d2D4.dIq_ddelta{k} = ...
    a.d2D4.ddelta_dIq{k};
%-----
a.d2D4.ddelta_ddelta{k} = ...
    sparse([real_ind(k) imag_ind(k) real_ind(k) imag_ind(k)],...
           [k k k+data.nPgen k+data.nPgen], ...
           [real(-j*Ig_c(k)*ejTheta(k)*ejdelta_c(k)) ...
            imag(-j*Ig_c(k)*ejTheta(k)*ejdelta_c(k)) ...
            real(-j*j*Ig_c(k)*Vp(k)*ejdelta_c(k)) ...
            imag(-j*j*Ig_c(k)*Vp(k)*ejdelta_c(k))], ...
           2*data.nPgen,2*data.nPgen);
%-----
a.d2D4.dtheta_ddelta{k,k} = ...
    sparse([real_ind(k) imag_ind(k) real_ind(k) imag_ind(k)],...
           [k k k+data.nPgen k+data.nPgen], ...
           [real(j*Ig_c(k)*ejTheta(k)*ejdelta_c(k)) ...
            imag(j*Ig_c(k)*ejTheta(k)*ejdelta_c(k)) ...
            real(j*j*Ig_c(k)*Vp(k)*ejdelta_c(k)) ...
            imag(j*j*Ig_c(k)*Vp(k)*ejdelta_c(k))], ...
           2*data.nPgen,2*data.nPgen);
a.d2D4.ddelta_dtheta{k,k} = a.d2D4.dtheta_ddelta{k,k};
%-----
a.d2D4.dV_dId{k,k} = ...
    sparse([real_ind(k) imag_ind(k)],...
           [k+data.nPgen k+data.nPgen], ...
           [real(-ejTheta(k)*ejdelta_c(k)) ...
            imag(-ejTheta(k)*ejdelta_c(k))], ...
           2*data.nPgen,2*data.nPgen);
a.d2D4.dId_dV{k,k} = a.d2D4.dV_dId{k,k};
%-----
a.d2D4.dV_dIq{k,k} = ...
    sparse([real_ind(k) imag_ind(k)],...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

[k+data.nPgen k+data.nPgen], ...
[real(j*ejTheta(k)*ejdelta_c(k)) ...
 imag(j*ejTheta(k)*ejdelta_c(k))], ...
2*data.nPgen,2*data.nPgen);
a.d2D4.dIq_dV{k,k} = a.d2D4.dV_dIq{k,k};
%-----
a.d2D4.dV_ddelta{k,k} = ...
sparse([real_ind(k) imag_ind(k)],...
[k+data.nPgen k+data.nPgen], ...
[real(j*Ig_c(k)*ejTheta(k)*ejdelta_c(k)) ...
 imag(j*Ig_c(k)*ejTheta(k)*ejdelta_c(k))], ...
2*data.nPgen,2*data.nPgen);
a.d2D4.ddelta_dV{k,k} = a.d2D4.dV_ddelta{k,k};
%*****
%*****
end%for
%*****
%*****
for k=1:data.nbuses
%*****
a.dD4.dV{k} = ...
-[real(a.JasysVV{k}(data.Pgenbuses,data.Pgenbuses)) ...
 real(a.JasysVt{k}(data.Pgenbuses,data.Pgenbuses))
 imag(a.JasysVV{k}(data.Pgenbuses,data.Pgenbuses)) ...
 imag(a.JasysVt{k}(data.Pgenbuses,data.Pgenbuses))];
%*****
a.dD4.dtheta{k} = ...
-[real(a.JasystV{k}(data.Pgenbuses,data.Pgenbuses)) ...
 real(a.Jasystt{k}(data.Pgenbuses,data.Pgenbuses))
 imag(a.JasystV{k}(data.Pgenbuses,data.Pgenbuses)) ...
 imag(a.Jasystt{k}(data.Pgenbuses,data.Pgenbuses))];
%*****
if isempty(find(data.Pgenbuses==k))==0
a.dD4.dV{k} = ...
a.dD4.dV{k} ...
+sparse([real_ind(k) imag_ind(k)],[k+data.nPgen k+data.nPgen], ...
[real(-Ig_c(k)*ejTheta(k)*ejdelta_c(k)) ...
 imag(-Ig_c(k)*ejTheta(k)*ejdelta_c(k))], ...
2*data.nPgen,2*data.nPgen);
%*****
a.dD4.dtheta{k} = ...
a.dD4.dtheta{k} ...
+sparse([real_ind(k) imag_ind(k) real_ind(k) imag_ind(k)],...
[k k k+data.nPgen k+data.nPgen], ...
[real(-Ig_c(k)*ejTheta(k)*ejdelta_c(k)) ...
 imag(-Ig_c(k)*ejTheta(k)*ejdelta_c(k)) ...
 real(-j*Ig_c(k)*Vp(k)*ejdelta_c(k)) ...
 imag(-j*Ig_c(k)*Vp(k)*ejdelta_c(k))], ...
2*data.nPgen,2*data.nPgen);
%*****
%a.d2K3.dV_ddelta{k} = ...
end%if
%*****
a.dD5.dV{k} = ...
-[real(a.JasysVV{k}(data.Pgenbuses,nongen_ind)) ...
 real(a.JasysVt{k}(data.Pgenbuses,nongen_ind))
 imag(a.JasysVV{k}(data.Pgenbuses,nongen_ind)) ...
 imag(a.JasysVt{k}(data.Pgenbuses,nongen_ind))];
%*****
a.dD5.dtheta{k} = ...
-[real(a.JasystV{k}(data.Pgenbuses,nongen_ind)) ...
 real(a.Jasystt{k}(data.Pgenbuses,nongen_ind))
 imag(a.JasystV{k}(data.Pgenbuses,nongen_ind)) ...
 imag(a.Jasystt{k}(data.Pgenbuses,nongen_ind))];
%*****

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

a.dD6.dV{k} = ...
-[real(a.JasysVW{k}(nongen_ind,data.Pgenbuses)) ...
 real(a.JasysVt{k}(nongen_ind,data.Pgenbuses))
 imag(a.JasysVW{k}(nongen_ind,data.Pgenbuses)) ...
 imag(a.JasysVt{k}(nongen_ind,data.Pgenbuses))];
%*****
a.dD6.dtheta{k} = ...
-[real(a.JasystV{k}(nongen_ind,data.Pgenbuses)) ...
 real(a.Jasystt{k}(nongen_ind,data.Pgenbuses))
 imag(a.JasystV{k}(nongen_ind,data.Pgenbuses)) ...
 imag(a.Jasystt{k}(nongen_ind,data.Pgenbuses))];
%*****
a.dD7.dV{k} = ...
-[real(a.JasysVW{k}(nongen_ind,nongen_ind)) ...
 real(a.JasysVt{k}(nongen_ind,nongen_ind))
 imag(a.JasysVW{k}(nongen_ind,nongen_ind)) ...
 imag(a.JasysVt{k}(nongen_ind,nongen_ind))];
%*****
a.dD7.dtheta{k} = ...
-[real(a.JasystV{k}(nongen_ind,nongen_ind)) ...
 real(a.Jasystt{k}(nongen_ind,nongen_ind))
 imag(a.JasystV{k}(nongen_ind,nongen_ind)) ...
 imag(a.Jasystt{k}(nongen_ind,nongen_ind))];
%*****
%-----
%*****
%*****
%*****
%*****
for m=1:data.nbuses
if m==k
if isempty(find(data.Pgenbuses==k))==0
a.d2D4.dtheta_dV{k,m} = ...
-[real(a.HasystVW{k,m}(data.Pgenbuses,data.Pgenbuses)) ...
 real(a.HasystVt{k,m}(data.Pgenbuses,data.Pgenbuses))
 imag(a.HasystVW{k,m}(data.Pgenbuses,data.Pgenbuses)) ...
 imag(a.HasystVt{k,m}(data.Pgenbuses,data.Pgenbuses))] ...
+sparse([real_ind(k) imag_ind(k)], [k+data.nPgen k+data.nPgen], ...
 [real(-j*Ig_c(k)*ejTheta(k)*ejdelta_c(k)) ...
 imag(-j*Ig_c(k)*ejTheta(k)*ejdelta_c(k))], ...
 2*data.nPgen,2*data.nPgen);
%-----
a.d2D4.dV_dtheta{k,m} = ...
-[real(a.HasysVtV{k,m}(data.Pgenbuses,data.Pgenbuses)) ...
 real(a.HasysVtt{k,m}(data.Pgenbuses,data.Pgenbuses))
 imag(a.HasysVtV{k,m}(data.Pgenbuses,data.Pgenbuses)) ...
 imag(a.HasysVtt{k,m}(data.Pgenbuses,data.Pgenbuses))] ...
+sparse([real_ind(k) imag_ind(k)], ...
 [k+data.nPgen k+data.nPgen], ...
 [real(-j*Ig_c(k)*ejTheta(k)*ejdelta_c(k)) ...
 imag(-j*Ig_c(k)*ejTheta(k)*ejdelta_c(k))], ...
 2*data.nPgen,2*data.nPgen);
%-----
a.d2D4.dtheta_dtheta{k,m} = ...
-[real(a.HasysttV{k,m}(data.Pgenbuses,data.Pgenbuses)) ...
 real(a.Hasysttt{k,m}(data.Pgenbuses,data.Pgenbuses))
 imag(a.HasysttV{k,m}(data.Pgenbuses,data.Pgenbuses)) ...
 imag(a.Hasysttt{k,m}(data.Pgenbuses,data.Pgenbuses))] ...
+sparse([real_ind(k) imag_ind(k) real_ind(k) imag_ind(k)], ...
 [k k k+data.nPgen k+data.nPgen], ...
 [real(-j*Ig_c(k)*ejTheta(k)*ejdelta_c(k)) ...
 imag(-j*Ig_c(k)*ejTheta(k)*ejdelta_c(k)) ...
 real(Ig_c(k)*Vp(k)*ejdelta_c(k)) ...
 imag(Ig_c(k)*Vp(k)*ejdelta_c(k))], ...
 2*data.nPgen,2*data.nPgen);
end%if

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

end%if
%-----
a.d2D4.dtheta_dV{k,m} = ...
-[real(a.HasystVV{k,m}(data.Pgenbuses,data.Pgenbuses)) ...
 real(a.HasystVt{k,m}(data.Pgenbuses,data.Pgenbuses))
 imag(a.HasystVV{k,m}(data.Pgenbuses,data.Pgenbuses)) ...
 imag(a.HasystVt{k,m}(data.Pgenbuses,data.Pgenbuses))];
%-----
a.d2D4.dV_dtheta{k,m} = ...
-[real(a.HasysVtV{k,m}(data.Pgenbuses,data.Pgenbuses)) ...
 real(a.HasysVtt{k,m}(data.Pgenbuses,data.Pgenbuses))
 imag(a.HasysVtV{k,m}(data.Pgenbuses,data.Pgenbuses)) ...
 imag(a.HasysVtt{k,m}(data.Pgenbuses,data.Pgenbuses))];
%-----
a.d2D4.dtheta_dtheta{k,m} = ...
-[real(a.HasysttV{k,m}(data.Pgenbuses,data.Pgenbuses)) ...
 real(a.Hasysttt{k,m}(data.Pgenbuses,data.Pgenbuses))
 imag(a.HasysttV{k,m}(data.Pgenbuses,data.Pgenbuses)) ...
 imag(a.Hasysttt{k,m}(data.Pgenbuses,data.Pgenbuses))];
%-----
a.d2D5.dtheta_dV{k,m} = ...
-[real(a.HasystVV{k,m}(data.Pgenbuses,nongen_ind)) ...
 real(a.HasystVt{k,m}(data.Pgenbuses,nongen_ind))
 imag(a.HasystVV{k,m}(data.Pgenbuses,nongen_ind)) ...
 imag(a.HasystVt{k,m}(data.Pgenbuses,nongen_ind))];
%-----
a.d2D5.dV_dtheta{k,m} = ...
-[real(a.HasysVtV{k,m}(data.Pgenbuses,nongen_ind)) ...
 real(a.HasysVtt{k,m}(data.Pgenbuses,nongen_ind))
 imag(a.HasysVtV{k,m}(data.Pgenbuses,nongen_ind)) ...
 imag(a.HasysVtt{k,m}(data.Pgenbuses,nongen_ind))];
%-----
a.d2D5.dtheta_dtheta{k,m} = ...
-[real(a.HasysttV{k,m}(data.Pgenbuses,nongen_ind)) ...
 real(a.Hasysttt{k,m}(data.Pgenbuses,nongen_ind))
 imag(a.HasysttV{k,m}(data.Pgenbuses,nongen_ind)) ...
 imag(a.Hasysttt{k,m}(data.Pgenbuses,nongen_ind))];
%-----
a.d2D6.dtheta_dV{k,m} = ...
-[real(a.HasystVV{k,m}(nongen_ind,data.Pgenbuses)) ...
 real(a.HasystVt{k,m}(nongen_ind,data.Pgenbuses))
 imag(a.HasystVV{k,m}(nongen_ind,data.Pgenbuses)) ...
 imag(a.HasystVt{k,m}(nongen_ind,data.Pgenbuses))];
%-----
a.d2D6.dV_dtheta{k,m} = ...
-[real(a.HasysVtV{k,m}(nongen_ind,data.Pgenbuses)) ...
 real(a.HasysVtt{k,m}(nongen_ind,data.Pgenbuses))
 imag(a.HasysVtV{k,m}(nongen_ind,data.Pgenbuses)) ...
 imag(a.HasysVtt{k,m}(nongen_ind,data.Pgenbuses))];
%-----
a.d2D6.dtheta_dtheta{k,m} = ...
-[real(a.HasysttV{k,m}(nongen_ind,data.Pgenbuses)) ...
 real(a.Hasysttt{k,m}(nongen_ind,data.Pgenbuses))
 imag(a.HasysttV{k,m}(nongen_ind,data.Pgenbuses)) ...
 imag(a.Hasysttt{k,m}(nongen_ind,data.Pgenbuses))];
%-----
a.d2D7.dtheta_dV{k,m} = ...
-[real(a.HasystVV{k,m}(nongen_ind,nongen_ind)) ...
 real(a.HasystVt{k,m}(nongen_ind,nongen_ind))
 imag(a.HasystVV{k,m}(nongen_ind,nongen_ind)) ...
 imag(a.HasystVt{k,m}(nongen_ind,nongen_ind))];
%-----
a.d2D7.dV_dtheta{k,m} = ...
-[real(a.HasysVtV{k,m}(nongen_ind,nongen_ind)) ...
 real(a.HasysVtt{k,m}(nongen_ind,nongen_ind))

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

    imag(a.HasysVtV{k,m}(nongen_ind,nongen_ind)) ...
    imag(a.HasysVtt{k,m}(nongen_ind,nongen_ind));
%-----
a.d2D7.dtheta_dtheta{k,m} = ...
-[real(a.HasysttV{k,m}(nongen_ind,nongen_ind)) ...
 real(a.Hasysttt{k,m}(nongen_ind,nongen_ind))
 imag(a.HasysttV{k,m}(nongen_ind,nongen_ind)) ...
 imag(a.Hasysttt{k,m}(nongen_ind,nongen_ind))];
end%for
end%for
%*****
% Eliminate first delta rows and columns
%*****
for k=1:data.nPgen
%*****
a.dA1.dIq{k}(1,:) = [];
a.dA1.dIq{k}(:,1) = [];
a.dA1.dId{k}(1,:) = [];
a.dA1.dId{k}(:,1) = [];
a.dA1.dEfd{k}(1,:) = [];
a.dA1.dEfd{k}(:,1) = [];
a.d2A1.dEfd_dEfd{k}(1,:) = [];
a.d2A1.dEfd_dEfd{k}(:,1) = [];

a.dA1.dIq{k}(1,:) = [];
a.dA1.dIq{k}(:,1) = [];
a.dA1.dId{k}(1,:) = [];
a.dA1.dId{k}(:,1) = [];
a.dA1.dEfd{k}(1,:) = [];
a.dA1.dEfd{k}(:,1) = [];
a.d2A1.dEfd_dEfd{k}(1,:) = [];
a.d2A1.dEfd_dEfd{k}(:,1) = [];
%*****

%*****
a.dB1.dIq{k}(1,:) = [];
a.dB1.dId{k}(1,:) = [];
a.dB1.dEdp{k}(1,:) = [];
a.dB1.dEqp{k}(1,:) = [];

a.dB1.dIq{k}(1,:) = [];
a.dB1.dId{k}(1,:) = [];
a.dB1.dEdp{k}(1,:) = [];
a.dB1.dEqp{k}(1,:) = [];
%*****

%*****
a.dC1.ddelta{k}(:,1) = [];
a.dC1.ddelta{k}(:,1) = [];
for m=1:data.nPgen
a.d2C1.ddelta_ddelta{k,m}(:,1) = [];
a.d2C1.ddelta_ddelta{k,m}(:,1) = [];
end%for
end%for
for m=1:data.nbuses
a.dC1.dV{m}(:,1) = [];
a.dC1.dtheta{m}(:,1) = [];

a.dC1.dV{m}(:,1) = [];
a.dC1.dtheta{m}(:,1) = [];
for n=1:data.nbuses
a.d2C1.dtheta_dV{m,n}(:,1) = [];
a.d2C1.dV_dtheta{m,n}(:,1) = [];
a.d2C1.dtheta_dtheta{m,n}(:,1) = [];

```


APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

a.d2C1.dtheta_dV{m,n}(:,1) = [];
a.d2C1.dV_dtheta{m,n}(:,1) = [];
a.d2C1.dtheta_dtheta{m,n}(:,1) = [];
end%for
for mn=1:data.nPgen
a.d2C1.ddelta_dV{mn,m}(:,1) = [];
a.d2C1.dV_ddelta{m,mn}(:,1) = [];
a.d2C1.ddelta_dtheta{mn,m}(:,1) = [];
a.d2C1.dtheta_ddelta{m,mn}(:,1) = [];

a.d2C1.ddelta_dV{mn,m}(:,1) = [];
a.d2C1.dV_ddelta{m,mn}(:,1) = [];
a.d2C1.ddelta_dtheta{mn,m}(:,1) = [];
a.d2C1.dtheta_ddelta{m,mn}(:,1) = [];
end%for
end%for

%*****

for k=1:data.nPgen
%*****
a.dC2.ddelta{k}(:,1) = [];
a.dC2.dId{k}(:,1) = [];
a.dC2.dIq{k}(:,1) = [];
a.d2C2.ddelta_dId{k}(:,1) = [];
a.d2C2.dId_ddelta{k}(:,1) = [];
a.d2C2.ddelta_dIq{k}(:,1) = [];
a.d2C2.dIq_ddelta{k}(:,1) = [];

a.dC2.ddelta{k}(:,1) = [];
a.dC2.dId{k}(:,1) = [];
a.dC2.dIq{k}(:,1) = [];
a.d2C2.ddelta_dId{k}(:,1) = [];
a.d2C2.dId_ddelta{k}(:,1) = [];
a.d2C2.ddelta_dIq{k}(:,1) = [];
a.d2C2.dIq_ddelta{k}(:,1) = [];
for m=1:data.nPgen
a.d2C2.ddelta_ddelta{k,m}(:,1) = [];
a.d2C2.ddelta_ddelta{k,m}(:,1) = [];
end%for
end%for
for m=1:data.nbuses
a.dC2.dV{m}(:,1) = [];
a.dC2.dtheta{m}(:,1) = [];

a.dC2.dV{m}(:,1) = [];
a.dC2.dtheta{m}(:,1) = [];
for n=1:data.nbuses
a.d2C2.dV_dtheta{m,n}(:,1) = [];
a.d2C2.dtheta_dV{m,n}(:,1) = [];
a.d2C2.dtheta_dtheta{m,n}(:,1) = [];
a.d2C2.dV_dV{m,n}(:,1) = [];

a.d2C2.dV_dtheta{m,n}(:,1) = [];
a.d2C2.dtheta_dV{m,n}(:,1) = [];
a.d2C2.dtheta_dtheta{m,n}(:,1) = [];
a.d2C2.dV_dV{m,n}(:,1) = [];
end%for
for mn=1:data.nPgen
a.d2C2.dV_dId{m,mn}(:,1) = [];
a.d2C2.dId_dV{mn,m}(:,1) = [];
a.d2C2.dV_dIq{m,mn}(:,1) = [];
a.d2C2.dIq_dV{mn,m}(:,1) = [];
a.d2C2.dtheta_dId{m,mn}(:,1) = [];

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

a.d2C2.dId_dtheta{mn,m}(:,1) = [];
a.d2C2.dtheta_dIq{m,mn}(:,1) = [];
a.d2C2.dIq_dtheta{mn,m}(:,1) = [];
a.d2C2.dV_ddelta{m,mn}(:,1) = [];
a.d2C2.ddelta_dV{mn,m}(:,1) = [];
a.d2C2.dtheta_ddelta{m,mn}(:,1) = [];
a.d2C2.ddelta_dtheta{mn,m}(:,1) = [];

a.d2C2.dV_dId{m,mn}(:,1) = [];
a.d2C2.dId_dV{mn,m}(:,1) = [];
a.d2C2.dV_dIq{m,mn}(:,1) = [];
a.d2C2.dIq_dV{mn,m}(:,1) = [];
a.d2C2.dtheta_dId{m,mn}(:,1) = [];
a.d2C2.dId_dtheta{mn,m}(:,1) = [];
a.d2C2.dtheta_dIq{m,mn}(:,1) = [];
a.d2C2.dIq_dtheta{mn,m}(:,1) = [];
a.d2C2.dV_ddelta{m,mn}(:,1) = [];
a.d2C2.ddelta_dV{mn,m}(:,1) = [];
a.d2C2.dtheta_ddelta{m,mn}(:,1) = [];
a.d2C2.ddelta_dtheta{mn,m}(:,1) = [];
end%for
end%for

%*****
% K derivatives
%*****
for k=1:data.nbuses
if isempty(find(data.Pgenbuses==k))==0
a.dK3.ddelta{k} = ...
    [-var.eigen.B1/var.eigen.D1*a.dD2.ddelta{k} var.eigen.zerok3];
%-----
a.dK3.dId{k} = ...
    [-a.dB1.dId{k}/var.eigen.D1*var.eigen.D2 var.eigen.zerok3];
%-----
a.dK3.dIq{k} = ...
    [-a.dB1.dIq{k}/var.eigen.D1*var.eigen.D2 var.eigen.zerok3];
%-----
a.dK3.dEdp{k} = ...
    [-a.dB1.dEdp{k}/var.eigen.D1*var.eigen.D2 var.eigen.zerok3];
%-----
a.dK3.dEqp{k} = ...
    [-a.dB1.dEqp{k}/var.eigen.D1*var.eigen.D2 var.eigen.zerok3];
%-----
a.dK4.ddelta{k} = ...
    [a.dD4.ddelta{k}-a.dD3.ddelta{k}/var.eigen.D1*var.eigen.D2- ...
    var.eigen.D3/var.eigen.D1*a.dD2.ddelta{k} var.eigen.zerok5
    var.eigen.zerok6 var.eigen.zerok7];
%-----
a.dK4.dId{k} = ...
    [a.dD4.dId{k} var.eigen.zerok5
    var.eigen.zerok6 var.eigen.zerok7];
%-----
a.dK4.dIq{k} = ...
    [a.dD4.dIq{k} var.eigen.zerok5
    var.eigen.zerok6 var.eigen.zerok7];
%-----
a.dK5.dId{k} = ...
    [a.dC2.dId{k}; var.eigen.zerok5];
%-----
a.dK5.dIq{k} = ...
    [a.dC2.dIq{k}; var.eigen.zerok5];
%-----
a.dK5.ddelta{k} = ...
    [a.dC2.ddelta{k}-a.dD3.ddelta{k}/var.eigen.D1*var.eigen.C1- ...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

var.eigen.D3/var.eigen.D1*a.dC1.ddelta{k}
var.eigen.zerok5];
%-----
for m=1:data.nPgen
%-----
a.d2K3.ddelta_dEdp{k,m} = [-a.dB1.dEdp{m}/var.eigen.D1*a.dD2.ddelta{k} ...
var.eigen.zerok3];
a.d2K3.dEdp_ddelta{m,k} = [-a.dB1.dEdp{m}/var.eigen.D1*a.dD2.ddelta{k} ...
var.eigen.zerok3];
%-----
a.d2K3.ddelta_dEqp{k,m} = [-a.dB1.dEqp{m}/var.eigen.D1*a.dD2.ddelta{k} ...
var.eigen.zerok3];
a.d2K3.dEqp_ddelta{m,k} = [-a.dB1.dEqp{m}/var.eigen.D1*a.dD2.ddelta{k} ...
var.eigen.zerok3];
%-----
a.d2K3.ddelta_dId{k,m} = [-a.dB1.dId{m}/var.eigen.D1*a.dD2.ddelta{k} ...
var.eigen.zerok3];
a.d2K3.dId_ddelta{m,k} = [-a.dB1.dId{m}/var.eigen.D1*a.dD2.ddelta{k} ...
var.eigen.zerok3];
%-----
a.d2K3.ddelta_dIq{k,m} = [-a.dB1.dIq{m}/var.eigen.D1*a.dD2.ddelta{k} ...
var.eigen.zerok3];
a.d2K3.dIq_ddelta{m,k} = [-a.dB1.dIq{m}/var.eigen.D1*a.dD2.ddelta{k} ...
var.eigen.zerok3];
%-----
d2K411 = ...
    a.d2D4.ddelta_ddelta{k,m} ...
    - a.d2D3.ddelta_ddelta{k,m}/var.eigen.D1*var.eigen.D2 ...
    -a.dD3.ddelta{m}/var.eigen.D1*a.dD2.ddelta{k} ...
    -a.dD3.ddelta{k}/var.eigen.D1*a.dD2.ddelta{m} ...
    -var.eigen.D3/var.eigen.D1*a.d2D2.ddelta_ddelta{k,m};
a.d2K4.ddelta_ddelta{k,m} = ...
[d2K411          var.eigen.zerod5
 var.eigen.zerod6          var.eigen.zerod7];
%-----
d2K511 = ...
    a.d2C2.ddelta_ddelta{k,m} ...
    - a.d2D3.ddelta_ddelta{k,m}/var.eigen.D1*var.eigen.C1 ...
    -a.dD3.ddelta{m}/var.eigen.D1*a.dC1.ddelta{k} ...
    -a.dD3.ddelta{k}/var.eigen.D1*a.dC1.ddelta{m} ...
    -var.eigen.D3/var.eigen.D1*a.d2C1.ddelta_ddelta{k,m};
a.d2K5.ddelta_ddelta{k,m} = [d2K511;          var.eigen.zerok5];
%-----
if m==k
%-----
a.d2K3.ddelta_ddelta{k,k} = ...
[-var.eigen.B1/var.eigen.D1*a.d2D2.ddelta_ddelta{k,k} ...
var.eigen.zerok3];
%-----
a.d2K4.dId_ddelta{k,k} = ...
[a.d2D4.dId_ddelta{k} var.eigen.zerod5
 var.eigen.zerod6 var.eigen.zerod7 ];
%-----
a.d2K4.ddelta_dId{k,k} = ...
[a.d2D4.ddelta_dId{k} var.eigen.zerod5
 var.eigen.zerod6 var.eigen.zerod7 ];
%-----
a.d2K4.dIq_ddelta{k,k} = ...
[a.d2D4.dIq_ddelta{k} var.eigen.zerod5
 var.eigen.zerod6 var.eigen.zerod7 ];
%-----
a.d2K4.ddelta_dIq{k,k} = ...
[a.d2D4.dIq_ddelta{k} var.eigen.zerod5
 var.eigen.zerod6 var.eigen.zerod7 ];
%-----

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

end%if
end%for
end%if
%-----
a.dK3.dV{k} = ...
                [-var.eigen.B1/var.eigen.D1*a.dD2.dV{k} var.eigen.zerok3];
%-----
a.dK3.dtheta{k} = ...
                [-var.eigen.B1/var.eigen.D1*a.dD2.dtheta{k} var.eigen.zerok3];
%-----
a.dK4.dV{k} = ...
[a.dD4.dV{k}-a.dD3.dV{k}/var.eigen.D1*var.eigen.D2-var.eigen.D3/var.eigen.D1*a.dD2.dV{k} ...
a.dD5.dV{k}
a.dD6.dV{k}      a.dD7.dV{k}];
%-----
a.dK4.dtheta{k} = ...
[a.dD4.dtheta{k}-a.dD3.dtheta{k}/var.eigen.D1*var.eigen.D2 ...
-var.eigen.D3/var.eigen.D1*a.dD2.dtheta{k} ...
a.dD5.dV{k}
a.dD6.dV{k}      a.dD7.dV{k}];
%-----
a.dK5.dV{k} = ...
[a.dC2.dV{k}-a.dD3.dV{k}/var.eigen.D1*var.eigen.C1-var.eigen.D3/var.eigen.D1*a.dC1.dV{k}
var.eigen.zerok5];
%-----
a.dK5.dtheta{k} = ...
[a.dC2.dtheta{k}-a.dD3.dtheta{k}/var.eigen.D1*var.eigen.C1 ...
-var.eigen.D3/var.eigen.D1*a.dC1.dtheta{k}
var.eigen.zerok5];
%-----
for m=1:data.nbuses
if isempty(find(data.Pgenbuses==m))==0
%-----
a.d2K3.dV_dId{k,m} = [-a.dB1.dId{m}/var.eigen.D1*a.dD2.dV{k} var.eigen.zerok3];
a.d2K3.dId_dV{m,k} = [-a.dB1.dId{m}/var.eigen.D1*a.dD2.dV{k} var.eigen.zerok3];
%-----
a.d2K3.dtheta_dId{k,m} = [-a.dB1.dId{m}/var.eigen.D1*a.dD2.dtheta{k} var.eigen.zerok3];
a.d2K3.dId_dtheta{m,k} = [-a.dB1.dId{m}/var.eigen.D1*a.dD2.dtheta{k} var.eigen.zerok3];
%-----
a.d2K3.dV_dIq{k,m} = [-a.dB1.dIq{m}/var.eigen.D1*a.dD2.dV{k} var.eigen.zerok3];
a.d2K3.dIq_dV{m,k} = [-a.dB1.dIq{m}/var.eigen.D1*a.dD2.dV{k} var.eigen.zerok3];
%-----
a.d2K3.dtheta_dIq{k,m} = [-a.dB1.dIq{m}/var.eigen.D1*a.dD2.dtheta{k} var.eigen.zerok3];
a.d2K3.dIq_dtheta{m,k} = [-a.dB1.dIq{m}/var.eigen.D1*a.dD2.dtheta{k} var.eigen.zerok3];
%-----
a.d2K3.dV_dEdp{k,m} = [-a.dB1.dEdp{m}/var.eigen.D1*a.dD2.dV{k} var.eigen.zerok3];
a.d2K3.dEdp_dV{m,k} = [-a.dB1.dEdp{m}/var.eigen.D1*a.dD2.dV{k} var.eigen.zerok3];
%-----
a.d2K3.dtheta_dEdp{k,m} = [-a.dB1.dEdp{m}/var.eigen.D1*a.dD2.dtheta{k} var.eigen.zerok3];
a.d2K3.dEdp_dtheta{m,k} = [-a.dB1.dEdp{m}/var.eigen.D1*a.dD2.dtheta{k} var.eigen.zerok3];
%-----
a.d2K3.dV_dEq{k,m} = [-a.dB1.dEq{m}/var.eigen.D1*a.dD2.dV{k} var.eigen.zerok3];
a.d2K3.dEq_dV{m,k} = [-a.dB1.dEq{m}/var.eigen.D1*a.dD2.dV{k} var.eigen.zerok3];
%-----
a.d2K3.dtheta_dEq{k,m} = [-a.dB1.dEq{m}/var.eigen.D1*a.dD2.dtheta{k} var.eigen.zerok3];
a.d2K3.dEq_dtheta{m,k} = [-a.dB1.dEq{m}/var.eigen.D1*a.dD2.dtheta{k} var.eigen.zerok3];
%-----
a.d2K3.dtheta_ddelta{k,m} = [-var.eigen.B1/var.eigen.D1*a.d2D2.dtheta_ddelta{k,m} ...
var.eigen.zerok3];
a.d2K3.ddelta_dtheta{m,k} = [-var.eigen.B1/var.eigen.D1*a.d2D2.dtheta_ddelta{k,m} ...
var.eigen.zerok3];
%-----
d2K411 = ...
a.d2D4.dV_ddelta{k,m} ...
-a.d2D3.dV_ddelta{k,m}/var.eigen.D1*var.eigen.D2 ...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

-a.dD3.ddelta{m}/var.eigen.D1*a.dD2.dV{k} ...
-a.dD3.dV{k}/var.eigen.D1*a.dD2.ddelta{m} ...
-var.eigen.D3/var.eigen.D1*a.d2D2.dV_ddelta{k,m};
a.d2K4.dV_ddelta{k,m} = ...
[d2K411          var.eigen.zeroD5
 var.eigen.zeroD6          var.eigen.zeroD7];
%-----
d2K411 = ...
    a.d2D4.ddelta_dV{m,k} ...
- a.d2D3.ddelta_dV{m,k}/var.eigen.D1*var.eigen.D2 ...
-a.dD3.dV{k}/var.eigen.D1*a.dD2.ddelta{m} ...
-a.dD3.ddelta{m}/var.eigen.D1*a.dD2.dV{k} ...
-var.eigen.D3/var.eigen.D1*a.d2D2.ddelta_dV{m,k};
a.d2K4.ddelta_dV{m,k} = ...
[d2K411          var.eigen.zeroD5
 var.eigen.zeroD6          var.eigen.zeroD7];
%-----
d2K411 = ...
    a.d2D4.dtheta_ddelta{k,m} ...
- a.d2D3.dtheta_ddelta{k,m}/var.eigen.D1*var.eigen.D2 ...
-a.dD3.ddelta{m}/var.eigen.D1*a.dD2.dtheta{k} ...
-a.dD3.dtheta{k}/var.eigen.D1*a.dD2.ddelta{m} ...
-var.eigen.D3/var.eigen.D1*a.d2D2.dtheta_ddelta{k,m};
a.d2K4.dtheta_ddelta{k,m} = ...
[d2K411          var.eigen.zeroD5
 var.eigen.zeroD6          var.eigen.zeroD7];
%-----
d2K411 = ...
    a.d2D4.ddelta_dtheta{m,k} ...
- a.d2D3.ddelta_dtheta{m,k}/var.eigen.D1*var.eigen.D2 ...
-a.dD3.dtheta{k}/var.eigen.D1*a.dD2.ddelta{m} ...
-a.dD3.ddelta{m}/var.eigen.D1*a.dD2.dtheta{k} ...
-var.eigen.D3/var.eigen.D1*a.d2D2.ddelta_dtheta{m,k};
a.d2K4.ddelta_dtheta{m,k} = ...
[d2K411          var.eigen.zeroD5
 var.eigen.zeroD6          var.eigen.zeroD7];
%-----
d2K511 = ...
    a.d2C2.dV_ddelta{k,m} ...
- a.d2D3.dV_ddelta{k,m}/var.eigen.D1*var.eigen.C1 ...
-a.dD3.ddelta{m}/var.eigen.D1*a.dC1.dV{k} ...
-a.dD3.dV{k}/var.eigen.D1*a.dC1.ddelta{m} ...
-var.eigen.D3/var.eigen.D1*a.d2C1.dV_ddelta{k,m};
a.d2K5.dV_ddelta{k,m} = [d2K511;          var.eigen.zerok5];
%-----
d2K511 = ...
    a.d2C2.ddelta_dV{m,k} ...
- a.d2D3.ddelta_dV{m,k}/var.eigen.D1*var.eigen.C1 ...
-a.dD3.dV{k}/var.eigen.D1*a.dC1.ddelta{m} ...
-a.dD3.ddelta{m}/var.eigen.D1*a.dC1.dV{k} ...
-var.eigen.D3/var.eigen.D1*a.d2C1.ddelta_dV{m,k};
a.d2K5.ddelta_dV{m,k} = [d2K511;          var.eigen.zerok5];
%-----
d2K511 = ...
    a.d2C2.dtheta_ddelta{k,m} ...
- a.d2D3.dtheta_ddelta{k,m}/var.eigen.D1*var.eigen.C1 ...
-a.dD3.ddelta{m}/var.eigen.D1*a.dC1.dtheta{k} ...
-a.dD3.dtheta{k}/var.eigen.D1*a.dC1.ddelta{m} ...
-var.eigen.D3/var.eigen.D1*a.d2C1.dtheta_ddelta{k,m};
a.d2K5.dtheta_ddelta{k,m} = [d2K511;          var.eigen.zerok5];
%-----
d2K511 = ...
    a.d2C2.ddelta_dtheta{m,k} ...
- a.d2D3.ddelta_dtheta{m,k}/var.eigen.D1*var.eigen.C1 ...
-a.dD3.dtheta{k}/var.eigen.D1*a.dC1.ddelta{m} ...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

-a.dD3.ddelta{m}/var.eigen.D1*a.dC1.dtheta{k} ...
-var.eigen.D3/var.eigen.D1*a.d2C1.ddelta_dtheta{m,k};
a.d2K5.ddelta_dtheta{m,k} = [d2K5i1; var.eigen.zerok5];
%-----
%-----
if m==k
%-----
a.d2K4.dId_dV{k,k} = ...
[a.d2D4.dId_dV{k,k} var.eigen.zeroD5
var.eigen.zeroD6 var.eigen.zeroD7 ];
%-----
a.d2K4.dV_dId{k,k} = ...
[a.d2D4.dV_dId{k,k} var.eigen.zeroD5
var.eigen.zeroD6 var.eigen.zeroD7 ];
%-----
a.d2K4.dIq_dV{k,k} = ...
[a.d2D4.dIq_dV{k,k} var.eigen.zeroD5
var.eigen.zeroD6 var.eigen.zeroD7 ];
%-----
a.d2K4.dV_dIq{k,k} = ...
[a.d2D4.dIq_dV{k,k} var.eigen.zeroD5
var.eigen.zeroD6 var.eigen.zeroD7 ];
%-----
a.d2K4.dId_dtheta{k,k} = ...
[a.d2D4.dId_dtheta{k,k} var.eigen.zeroD5
var.eigen.zeroD6 var.eigen.zeroD7 ];
%-----
a.d2K4.dtheta_dId{k,k} = ...
[a.d2D4.dtheta_dId{k,k} var.eigen.zeroD5
var.eigen.zeroD6 var.eigen.zeroD7 ];
%-----
a.d2K4.dIq_dtheta{k,k} = ...
[a.d2D4.dIq_dtheta{k,k} var.eigen.zeroD5
var.eigen.zeroD6 var.eigen.zeroD7 ];
%-----
a.d2K4.dtheta_dIq{k,k} = ...
[a.d2D4.dIq_dtheta{k,k} var.eigen.zeroD5
var.eigen.zeroD6 var.eigen.zeroD7 ];
%-----
a.d2K4.dId_ddelta{k,k} = ...
[a.d2D4.dId_ddelta{k} var.eigen.zeroD5
var.eigen.zeroD6 var.eigen.zeroD7 ];
%-----
a.d2K4.ddelta_dId{k,k} = ...
[a.d2D4.ddelta_dId{k} var.eigen.zeroD5
var.eigen.zeroD6 var.eigen.zeroD7 ];
%-----
a.d2K4.dIq_ddelta{k,k} = ...
[a.d2D4.dIq_ddelta{k} var.eigen.zeroD5
var.eigen.zeroD6 var.eigen.zeroD7 ];
%-----
a.d2K4.ddelta_dIq{k,k} = ...
[a.d2D4.dIq_ddelta{k} var.eigen.zeroD5
var.eigen.zeroD6 var.eigen.zeroD7 ];
%-----
a.d2K5.dId_dV{k,k} = ...
[a.d2C2.dId_dV{k}; var.eigen.zerok5];
%-----
a.d2K5.dV_dId{k,k} = ...
[a.d2C2.dV_dId{k}; var.eigen.zerok5];
%-----
a.d2K5.dIq_dV{k,k} = ...
[a.d2C2.dIq_dV{k}; var.eigen.zerok5];
%-----
a.d2K5.dV_dIq{k,k} = ...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

    [a.d2C2.dV_dIq{k}; var.eigen.zerok5];
%-----
a.d2K5.dId_dtheta{k,k} = ...
    [a.d2C2.dId_dtheta{k}; var.eigen.zerok5];
%-----
a.d2K5.dtheta_dId{k,k} = ...
    [a.d2C2.dtheta_dId{k}; var.eigen.zerok5];
%-----
a.d2K5.dIq_dtheta{k,k} = ...
    [a.d2C2.dIq_dtheta{k}; var.eigen.zerok5];
%-----
a.d2K5.dtheta_dIq{k,k} = ...
    [a.d2C2.dtheta_dIq{k}; var.eigen.zerok5];
%-----
a.d2K5.dId_ddelta{k,k} = ...
    [a.d2C2.dId_ddelta{k}; var.eigen.zerok5];
%-----
a.d2K5.ddelta_dId{k,k} = ...
    [a.d2C2.ddelta_dId{k}; var.eigen.zerok5];
%-----
a.d2K5.dIq_ddelta{k,k} = ...
    [a.d2C2.dIq_ddelta{k}; var.eigen.zerok5];
%-----
a.d2K5.ddelta_dIq{k,k} = ...
    [a.d2C2.ddelta_dIq{k}; var.eigen.zerok5];
%-----
end%if
end%if
%-----
a.d2K3.dV_dtheta{k,m} = [-var.eigen.B1/var.eigen.D1*a.d2D2.dV_dtheta{k,m} ...
    var.eigen.zerok3];
%-----
a.d2K3.dtheta_dV{k,m} = [-var.eigen.B1/var.eigen.D1*a.d2D2.dtheta_dV{k,m} ...
    var.eigen.zerok3];
%-----
a.d2K3.dtheta_dtheta{k,m} = [-var.eigen.B1/var.eigen.D1*a.d2D2.dtheta_dtheta{k,m} ...
    var.eigen.zerok3];
%-----
a.d2K3.dtheta_dtheta{m,k} = [-var.eigen.B1/var.eigen.D1*a.d2D2.dtheta_dtheta{m,k} ...
    var.eigen.zerok3];
%-----
d2K411 = ...
    a.d2D4.dV_dtheta{k,m} ...
    - a.d2D3.dV_dtheta{k,m}/var.eigen.D1*var.eigen.D2 ...
    -a.dD3.dtheta{m}/var.eigen.D1*a.dD2.dV{k} ...
    -a.dD3.dV{k}/var.eigen.D1*a.dD2.dtheta{m} ...
    -var.eigen.D3/var.eigen.D1*a.d2D2.dV_dtheta{k,m};
a.d2K4.dV_dtheta{k,m} = ...
[d2K411      a.d2D5.dV_dtheta{k,m}
 a.d2D6.dV_dtheta{k,m}  a.d2D7.dV_dtheta{k,m}];
%-----
d2K411 = ...
    a.d2D4.dtheta_dV{k,m} ...
    - a.d2D3.dtheta_dV{k,m}/var.eigen.D1*var.eigen.D2 ...
    -a.dD3.dV{m}/var.eigen.D1*a.dD2.dtheta{k} ...
    -a.dD3.dtheta{k}/var.eigen.D1*a.dD2.dV{m} ...
    -var.eigen.D3/var.eigen.D1*a.d2D2.dtheta_dV{k,m};
a.d2K4.dtheta_dV{k,m} = ...
[d2K411      a.d2D5.dtheta_dV{k,m}
 a.d2D6.dtheta_dV{k,m}  a.d2D7.dtheta_dV{k,m}];
%-----
k2K411 = ...
    a.d2D4.dtheta_dtheta{k,m} ...
    -a.d2D3.dtheta_dtheta{k,m}/var.eigen.D1*var.eigen.D2 ...
    -a.dD3.dtheta{m}/var.eigen.D1*a.dD2.dtheta{k} ...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

-a.dD3.dtheta{k}/var.eigen.D1*a.dD2.dtheta{m} ...
-var.eigen.D3/var.eigen.D1*a.dD2.dtheta_dtheta{k,m};
a.d2K4.dtheta_dtheta{k,m} = ...
[k2K411 a.d2D5.dtheta_dtheta{k,m}
 a.d2D6.dtheta_dtheta{k,m} a.d2D7.dtheta_dtheta{k,m}];
%-----
k2K411 = ...
a.d2D4.dtheta_dtheta{m,k} ...
-a.d2D3.dtheta_dtheta{m,k}/var.eigen.D1*var.eigen.D2 ...
-a.dD3.dtheta{k}/var.eigen.D1*a.dD2.dtheta{m} ...
-a.dD3.dtheta{m}/var.eigen.D1*a.dD2.dtheta{k} ...
-var.eigen.D3/var.eigen.D1*a.dD2.dtheta_dtheta{m,k};
a.d2K4.dtheta_dtheta{m,k} = ...
[k2K411 a.d2D5.dtheta_dtheta{m,k}
 a.d2D6.dtheta_dtheta{m,k} a.d2D7.dtheta_dtheta{m,k}];
%-----
a.d2K4.dV_dV{k,m} = ...
[-2*a.dD3.dV{m}/var.eigen.D1*a.dD2.dV{k} var.eigen.zeroD5
 var.eigen.zeroD6 var.eigen.zeroD7];
%-----
d2K511 = ...
a.d2C2.dV_dtheta{k,m} ...
-a.d2D3.dV_dtheta{k,m}/var.eigen.D1*var.eigen.C1 ...
-a.dD3.dtheta{m}/var.eigen.D1*a.dC1.dV{k} ...
-a.dD3.dV{k}/var.eigen.D1*a.dC1.dtheta{m} ...
-var.eigen.D3/var.eigen.D1*a.d2C1.dV_dtheta{k,m};
a.d2K5.dV_dtheta{k,m} = [d2K511; var.eigen.zerok5];
%-----
d2K511 = ...
a.d2C2.dtheta_dV{k,m} ...
-a.d2D3.dtheta_dV{k,m}/var.eigen.D1*var.eigen.C1 ...
-a.dD3.dV{m}/var.eigen.D1*a.dC1.dtheta{k} ...
-a.dD3.dtheta{k}/var.eigen.D1*a.dC1.dV{m} ...
-var.eigen.D3/var.eigen.D1*a.d2C1.dtheta_dV{k,m};
a.d2K5.dtheta_dV{k,m} = [d2K511; var.eigen.zerok5];
%-----
d2K511 = ...
a.d2C2.dtheta_dtheta{k,m} ...
-a.d2D3.dtheta_dtheta{k,m}/var.eigen.D1*var.eigen.C1 ...
-a.dD3.dtheta{m}/var.eigen.D1*a.dC1.dtheta{k} ...
-a.dD3.dtheta{k}/var.eigen.D1*a.dC1.dtheta{m} ...
-var.eigen.D3/var.eigen.D1*a.d2C1.dtheta_dtheta{k,m};
a.d2K5.dtheta_dtheta{k,m} = [d2K511; var.eigen.zerok5];
%-----
a.d2K5.dV_dV{k,m} = ...
[-2*a.dD3.dV{m}/var.eigen.D1*a.dC1.dV{k}; var.eigen.zerok5];
%-----
end%for
end%for
%*****
% Asys derivatives
%*****
K4inv = eye(length(var.eigen.K4))/var.eigen.K4;
%psi = eye(length(var.eigen.phi))/var.eigen.phi;
psi = inviteration(var.eigen);
zeroD5 = sparse(zeros(size(var.eigen.D5)));
zeroD6 = sparse(zeros(size(var.eigen.D6)));
zeroD7 = sparse(zeros(size(var.eigen.D7)));
for k=1:data.nbuses
if isempty(find(data.Pgenbuses==k))==0
% a.dK3.dId{k} = [-a.dB1.dId{k}/var.eigen.D1*var.eigen.D2 var.eigen.zerok3];
% a.dK3.dIq{k} = [-a.dB1.dIq{k}/var.eigen.D1*var.eigen.D2 var.eigen.zerok3];
% a.dK4.dId{k} = [a.dD4.dId{k} zeroD5
% zeroD6 zeroD7];
% a.dK4.dIq{k} = [a.dD4.dIq{k} zeroD5

```


APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

%          zeroD6          zeroD7);
% a.dK5.dId{k} = [a.dC2.dId{k}; var.eigen.zerok5];
% a.dK5.dIq{k} = [a.dC2.dIq{k}; var.eigen.zerok5];
a.dAsys.dId{k} = a.dA1.dId{k}-a.dB1.dId{k}/var.eigen.D1*var.eigen.C1 ...
-a.dK3.dId{k}/var.eigen.K4*var.eigen.K5 ...
+var.eigen.K3/var.eigen.K4*a.dK4.dId{k}/var.eigen.K4*var.eigen.K5 ...
-var.eigen.K3/var.eigen.K4*a.dK5.dId{k};
a.dAsys.dIq{k} = a.dA1.dIq{k}-a.dB1.dIq{k}/var.eigen.D1*var.eigen.C1 ...
-a.dK3.dIq{k}/var.eigen.K4*var.eigen.K5 ...
+var.eigen.K3/var.eigen.K4*a.dK4.dIq{k}/var.eigen.K4*var.eigen.K5 ...
-var.eigen.K3/var.eigen.K4*a.dK5.dIq{k};
a.deig.dId(k) = psi(1,:)*a.dAsys.dId{k}*var.eigen.phi(:,1) ...
/(psi(1,:)*var.eigen.phi(:,1));
a.deig.dIq(k) = psi(1,:)*a.dAsys.dIq{k}*var.eigen.phi(:,1) ...
/(psi(1,:)*var.eigen.phi(:,1));
%-----
a.dAsys.dEfd{k} = a.dA1.dEfd{k};
a.dAsys.dEdp{k} = -a.dB1.dEdp{k}/var.eigen.D1*var.eigen.C1 ...
-a.dK3.dEdp{k}/var.eigen.K4*var.eigen.K5;
a.dAsys.dEqp{k} = -a.dB1.dEqp{k}/var.eigen.D1*var.eigen.C1 ...
-a.dK3.dEqp{k}/var.eigen.K4*var.eigen.K5;
a.dAsys.ddelta{k} = -var.eigen.B1/var.eigen.D1*a.dC1.ddelta{k} ...
-a.dK3.ddelta{k}/var.eigen.K4*var.eigen.K5 ...
+var.eigen.K3/var.eigen.K4*a.dK4.ddelta{k}/var.eigen.K4*var.eigen.K5 ...
-var.eigen.K3/var.eigen.K4*a.dK5.ddelta{k};
a.d2Asys.dEfd_dEfd{k} = a.d2A1.dEfd_dEfd{k};
%-----
for m=1:data.nPgen
%-----
a.d2Asys.ddelta_ddelta{k,m} = ...
-var.eigen.B1/var.eigen.D1*a.d2C1.ddelta_ddelta{k,m} ...
-a.d2K3.ddelta_ddelta{k,m}/var.eigen.K4*var.eigen.K5 ...
+a.dK3.ddelta{m}/var.eigen.K4*a.dK4.ddelta{k}/var.eigen.K4*var.eigen.K5 ...
-a.dK3.ddelta{m}/var.eigen.K4*a.dK5.ddelta{k} ...
+a.dK3.ddelta{k}/var.eigen.K4*a.dK4.ddelta{m}/var.eigen.K4*var.eigen.K5 ...
-var.eigen.K3*( ...
-K4inv*a.d2K4.ddelta_ddelta{k,m}/var.eigen.K4 ...
+K4inv*a.dK4.ddelta{m}/var.eigen.K4*a.dK4.ddelta{k}/var.eigen.K4 ...
+K4inv*a.dK4.ddelta{k}/var.eigen.K4*a.dK4.ddelta{m}/var.eigen.K4 ...
)*var.eigen.K5 ...
+var.eigen.K3/var.eigen.K4*a.dK4.ddelta{m}/var.eigen.K4*a.dK5.ddelta{k} ...
-a.dK3.ddelta{k}/var.eigen.K4*a.dK5.ddelta{m} ...
+var.eigen.K3/var.eigen.K4*a.dK4.ddelta{k}/var.eigen.K4*a.dK5.ddelta{m} ...
-var.eigen.K3/var.eigen.K4*a.d2K5.ddelta_ddelta{k,m};
%-----
a.d2Asys.dId_ddelta{m,k} = -a.dB1.dId{m}/var.eigen.D1*a.dC1.ddelta{k} ...
-a.d2K3.dId_ddelta{m,k}/var.eigen.K4*var.eigen.K5 ...
+a.dK3.dId{m}/var.eigen.K4*a.dK4.ddelta{k}/var.eigen.K4*var.eigen.K5 ...
-a.dK3.dId{m}/var.eigen.K4*a.dK5.ddelta{k} ...
+a.dK3.ddelta{k}/var.eigen.K4*a.dK4.dId{m}/var.eigen.K4*var.eigen.K5 ...
-var.eigen.K3*( ...
-K4inv*a.d2K4.dId_ddelta{m,k}/var.eigen.K4 ...
+K4inv*a.dK4.dId{m}/var.eigen.K4*a.dK4.ddelta{k}/var.eigen.K4 ...
+K4inv*a.dK4.ddelta{k}/var.eigen.K4*a.dK4.dId{m}/var.eigen.K4 ...
)*var.eigen.K5 ...
+var.eigen.K3/var.eigen.K4*a.dK4.dId{m}/var.eigen.K4*a.dK5.ddelta{k} ...
-a.dK3.ddelta{k}/var.eigen.K4*a.dK5.dId{m} ...
+var.eigen.K3/var.eigen.K4*a.dK4.ddelta{k}/var.eigen.K4*a.dK5.dId{m} ...
-var.eigen.K3/var.eigen.K4*a.d2K5.dId_ddelta{m,k};
a.d2Asys.ddelta_dId{k,m} = -a.dB1.dId{m}/var.eigen.D1*a.dC1.ddelta{k} ...
-a.d2K3.ddelta_dId{k,m}/var.eigen.K4*var.eigen.K5 ...
+a.dK3.dId{m}/var.eigen.K4*a.dK4.ddelta{k}/var.eigen.K4*var.eigen.K5 ...
-a.dK3.dId{m}/var.eigen.K4*a.dK5.ddelta{k} ...
+a.dK3.ddelta{k}/var.eigen.K4*a.dK4.dId{m}/var.eigen.K4*var.eigen.K5 ...
-var.eigen.K3*( ...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

-K4inv*a.d2K4.ddelta_dId{k,m}/var.eigen.K4 ...
+K4inv*a.dK4.dId{m}/var.eigen.K4*a.dK4.ddelta{k}/var.eigen.K4 ...
+K4inv*a.dK4.ddelta{k}/var.eigen.K4*a.dK4.dId{m}/var.eigen.K4 ...
)*var.eigen.K5 ...
    +var.eigen.K3/var.eigen.K4*a.dK4.dId{m}/var.eigen.K4*a.dK5.ddelta{k} ...
      -a.dK3.ddelta{k}/var.eigen.K4*a.dK5.dId{m} ...
      +var.eigen.K3/var.eigen.K4*a.dK4.ddelta{k}/var.eigen.K4*a.dK5.dId{m} ...
    -var.eigen.K3/var.eigen.K4*a.d2K5.ddelta_dId{k,m};
a.d2Asys.dIq_ddelta{m,k} = -a.dB1.dIq{m}/var.eigen.D1*a.dC1.ddelta{k} ...
    -a.d2K3.dIq_ddelta{m,k}/var.eigen.K4*var.eigen.K5 ...
    +a.dK3.dIq{m}/var.eigen.K4*a.dK4.ddelta{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dIq{m}/var.eigen.K4*a.dK5.ddelta{k} ...
    +a.dK3.ddelta{k}/var.eigen.K4*a.dK4.dIq{m}/var.eigen.K4*var.eigen.K5 ...
    -var.eigen.K3*( ...
-K4inv*a.d2K4.dIq_ddelta{m,k}/var.eigen.K4 ...
+K4inv*a.dK4.dIq{m}/var.eigen.K4*a.dK4.ddelta{k}/var.eigen.K4 ...
+K4inv*a.dK4.ddelta{k}/var.eigen.K4*a.dK4.dIq{m}/var.eigen.K4 ...
)*var.eigen.K5 ...
    +var.eigen.K3/var.eigen.K4*a.dK4.dIq{m}/var.eigen.K4*a.dK5.ddelta{k} ...
      -a.dK3.ddelta{k}/var.eigen.K4*a.dK5.dIq{m} ...
      +var.eigen.K3/var.eigen.K4*a.dK4.ddelta{k}/var.eigen.K4*a.dK5.dIq{m} ...
    -var.eigen.K3/var.eigen.K4*a.d2K5.dIq_ddelta{m,k};
a.d2Asys.ddelta_dIq{k,m} = -a.dB1.dIq{m}/var.eigen.D1*a.dC1.ddelta{k} ...
    -a.d2K3.ddelta_dIq{k,m}/var.eigen.K4*var.eigen.K5 ...
    +a.dK3.dIq{m}/var.eigen.K4*a.dK4.ddelta{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dIq{m}/var.eigen.K4*a.dK5.ddelta{k} ...
    +a.dK3.ddelta{k}/var.eigen.K4*a.dK4.dIq{m}/var.eigen.K4*var.eigen.K5 ...
    -var.eigen.K3*( ...
-K4inv*a.d2K4.ddelta_dIq{k,m}/var.eigen.K4 ...
+K4inv*a.dK4.dIq{m}/var.eigen.K4*a.dK4.ddelta{k}/var.eigen.K4 ...
+K4inv*a.dK4.ddelta{k}/var.eigen.K4*a.dK4.dIq{m}/var.eigen.K4 ...
)*var.eigen.K5 ...
    +var.eigen.K3/var.eigen.K4*a.dK4.dIq{m}/var.eigen.K4*a.dK5.ddelta{k} ...
      -a.dK3.ddelta{k}/var.eigen.K4*a.dK5.dIq{m} ...
      +var.eigen.K3/var.eigen.K4*a.dK4.ddelta{k}/var.eigen.K4*a.dK5.dIq{m} ...
    -var.eigen.K3/var.eigen.K4*a.d2K5.ddelta_dIq{k,m};
%-----
a.d2Asys.dEdp_dId{k,m} = ...
    +a.dK3.dEdp{m}/var.eigen.K4*a.dK4.dId{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dEdp{m}/var.eigen.K4*a.dK5.dId{k};
a.d2Asys.dId_dEdp{k,m} = ...
    +a.dK3.dEdp{m}/var.eigen.K4*a.dK4.dId{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dEdp{m}/var.eigen.K4*a.dK5.dId{k};
%-----
a.d2Asys.dEqp_dId{k,m} = ...
    +a.dK3.dEqp{m}/var.eigen.K4*a.dK4.dId{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dEqp{m}/var.eigen.K4*a.dK5.dId{k};
a.d2Asys.dId_dEqp{k,m} = ...
    +a.dK3.dEqp{m}/var.eigen.K4*a.dK4.dId{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dEqp{m}/var.eigen.K4*a.dK5.dId{k};
%-----
a.d2Asys.dEdp_dIq{k,m} = ...
    +a.dK3.dEdp{m}/var.eigen.K4*a.dK4.dIq{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dEdp{m}/var.eigen.K4*a.dK5.dIq{k};
a.d2Asys.dIq_dEdp{k,m} = ...
    +a.dK3.dEdp{m}/var.eigen.K4*a.dK4.dIq{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dEdp{m}/var.eigen.K4*a.dK5.dIq{k};
%-----
a.d2Asys.dEqp_dIq{k,m} = ...
    +a.dK3.dEqp{m}/var.eigen.K4*a.dK4.dIq{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dEqp{m}/var.eigen.K4*a.dK5.dIq{k};
a.d2Asys.dIq_dEqp{k,m} = ...
    +a.dK3.dEqp{m}/var.eigen.K4*a.dK4.dIq{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dEqp{m}/var.eigen.K4*a.dK5.dIq{k};
%-----

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

a.d2Asys.ddelta_dEdp{k,m} = -a.dB1.dEdp{m}/var.eigen.D1*a.dC1.ddelta{k} ...
    -a.d2K3.ddelta_dEdp{k,m}/var.eigen.K4*var.eigen.K5 ...
    +a.dK3.dEdp{m}/var.eigen.K4*a.dK4.ddelta{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dEdp{m}/var.eigen.K4*a.dK5.ddelta{k};
a.d2Asys.dEdp_ddelta{m,k} = -a.dB1.dEdp{m}/var.eigen.D1*a.dC1.ddelta{k} ...
    -a.d2K3.dEdp_ddelta{m,k}/var.eigen.K4*var.eigen.K5 ...
    +a.dK3.dEdp{m}/var.eigen.K4*a.dK4.ddelta{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dEdp{m}/var.eigen.K4*a.dK5.ddelta{k};
%-----
a.d2Asys.ddelta_dEqp{k,m} = -a.dB1.dEqp{m}/var.eigen.D1*a.dC1.ddelta{k} ...
    -a.d2K3.ddelta_dEqp{k,m}/var.eigen.K4*var.eigen.K5 ...
    +a.dK3.dEqp{m}/var.eigen.K4*a.dK4.ddelta{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dEqp{m}/var.eigen.K4*a.dK5.ddelta{k};
a.d2Asys.dEqp_ddelta{m,k} = -a.dB1.dEqp{m}/var.eigen.D1*a.dC1.ddelta{k} ...
    -a.d2K3.dEqp_ddelta{m,k}/var.eigen.K4*var.eigen.K5 ...
    +a.dK3.dEqp{m}/var.eigen.K4*a.dK4.ddelta{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dEqp{m}/var.eigen.K4*a.dK5.ddelta{k};
%-----
end%for
end%if
%-----
a.dAsys.dV{k} = -var.eigen.B1/var.eigen.D1*a.dC1.dV{k} ...
    -a.dK3.dV{k}/var.eigen.K4*var.eigen.K5 ...
    +var.eigen.K3/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*var.eigen.K5 ...
    -var.eigen.K3/var.eigen.K4*a.dK5.dV{k};
%-----
a.dAsys.dtheta{k} = -var.eigen.B1/var.eigen.D1*a.dC1.dtheta{k} ...
    -a.dK3.dtheta{k}/var.eigen.K4*var.eigen.K5 ...
    +var.eigen.K3/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4*var.eigen.K5 ...
    -var.eigen.K3/var.eigen.K4*a.dK5.dtheta{k};
%-----
for m=1:data.nbuses
if isempty(find(data.Pgenbuses==m))==0
a.d2Asys.dId_dV{m,k} = -a.dB1.dId{m}/var.eigen.D1*a.dC1.dV{k} ...
    -a.d2K3.dId_dV{m,k}/var.eigen.K4*var.eigen.K5 ...
    +a.dK3.dId{m}/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dId{m}/var.eigen.K4*a.dK5.dV{k} ...
    +a.dK3.dV{k}/var.eigen.K4*a.dK4.dId{m}/var.eigen.K4*var.eigen.K5 ...
    -var.eigen.K3*( ...
    -K4inv*a.d2K4.dId_dV{m,k}/var.eigen.K4 ...
    +K4inv*a.dK4.dId{m}/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4 ...
    +K4inv*a.dK4.dV{k}/var.eigen.K4*a.dK4.dId{m}/var.eigen.K4 ...
    )*var.eigen.K5 ...
    +var.eigen.K3/var.eigen.K4*a.dK4.dId{m}/var.eigen.K4*a.dK5.dV{k} ...
    -a.dK3.dV{k}/var.eigen.K4*a.dK5.dId{m} ...
    +var.eigen.K3/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*a.dK5.dId{m} ...
    -var.eigen.K3/var.eigen.K4*a.d2K5.dId_dV{m,k};
a.d2Asys.dV_dId{k,m} = -a.dB1.dId{m}/var.eigen.D1*a.dC1.dV{k} ...
    -a.d2K3.dV_dId{k,m}/var.eigen.K4*var.eigen.K5 ...
    +a.dK3.dId{m}/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dId{m}/var.eigen.K4*a.dK5.dV{k} ...
    +a.dK3.dV{k}/var.eigen.K4*a.dK4.dId{m}/var.eigen.K4*var.eigen.K5 ...
    -var.eigen.K3*( ...
    -K4inv*a.d2K4.dV_dId{k,m}/var.eigen.K4 ...
    +K4inv*a.dK4.dId{m}/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4 ...
    +K4inv*a.dK4.dV{k}/var.eigen.K4*a.dK4.dId{m}/var.eigen.K4 ...
    )*var.eigen.K5 ...
    +var.eigen.K3/var.eigen.K4*a.dK4.dId{m}/var.eigen.K4*a.dK5.dV{k} ...
    -a.dK3.dV{k}/var.eigen.K4*a.dK5.dId{m} ...
    +var.eigen.K3/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*a.dK5.dId{m} ...
    -var.eigen.K3/var.eigen.K4*a.d2K5.dV_dId{k,m};
a.d2Asys.dIq_dV{m,k} = -a.dB1.dIq{m}/var.eigen.D1*a.dC1.dV{k} ...
    -a.d2K3.dIq_dV{m,k}/var.eigen.K4*var.eigen.K5 ...
    +a.dK3.dIq{m}/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dIq{m}/var.eigen.K4*a.dK5.dV{k} ...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

+ a.dK3.dV{k}/var.eigen.K4*a.dK4.dIq{m}/var.eigen.K4*var.eigen.K5 ...
-var.eigen.K3*( ...
-K4inv*a.d2K4.dIq_dV{m,k}/var.eigen.K4 ...
+K4inv*a.dK4.dIq{m}/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4 ...
+K4inv*a.dK4.dV{k}/var.eigen.K4*a.dK4.dIq{m}/var.eigen.K4 ...
)*var.eigen.K5 ...
+var.eigen.K3/var.eigen.K4*a.dK4.dIq{m}/var.eigen.K4*a.dK5.dV{k} ...
-a.dK3.dV{k}/var.eigen.K4*a.dK5.dIq{m} ...
+var.eigen.K3/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*a.dK5.dIq{m} ...
-var.eigen.K3/var.eigen.K4*a.d2K5.dIq_dV{m,k};
a.d2Asys.dV_dIq{k,m} = -a.dB1.dIq{m}/var.eigen.D1*a.dC1.dV{k} ...
-a.d2K3.dV_dIq{k,m}/var.eigen.K4*var.eigen.K5 ...
+a.dK3.dIq{m}/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*var.eigen.K5 ...
-a.dK3.dIq{m}/var.eigen.K4*a.dK5.dV{k} ...
+a.dK3.dV{k}/var.eigen.K4*a.dK4.dIq{m}/var.eigen.K4*var.eigen.K5 ...
-var.eigen.K3*( ...
-K4inv*a.d2K4.dV_dIq{k,m}/var.eigen.K4 ...
+K4inv*a.dK4.dIq{m}/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4 ...
+K4inv*a.dK4.dV{k}/var.eigen.K4*a.dK4.dIq{m}/var.eigen.K4 ...
)*var.eigen.K5 ...
+var.eigen.K3/var.eigen.K4*a.dK4.dIq{m}/var.eigen.K4*a.dK5.dV{k} ...
-a.dK3.dV{k}/var.eigen.K4*a.dK5.dIq{m} ...
+var.eigen.K3/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*a.dK5.dIq{m} ...
-var.eigen.K3/var.eigen.K4*a.d2K5.dV_dIq{k,m};
%-----
a.d2Asys.dId_dtheta{m,k} = -a.dB1.dId{m}/var.eigen.D1*a.dC1.dtheta{k} ...
-a.d2K3.dId_dtheta{m,k}/var.eigen.K4*var.eigen.K5 ...
+a.dK3.dId{m}/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4*var.eigen.K5 ...
-a.dK3.dId{m}/var.eigen.K4*a.dK5.dtheta{k} ...
+a.dK3.dtheta{k}/var.eigen.K4*a.dK4.dId{m}/var.eigen.K4*var.eigen.K5 ...
-var.eigen.K3*( ...
-K4inv*a.d2K4.dId_dtheta{m,k}/var.eigen.K4 ...
+K4inv*a.dK4.dId{m}/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4 ...
+K4inv*a.dK4.dtheta{k}/var.eigen.K4*a.dK4.dId{m}/var.eigen.K4 ...
)*var.eigen.K5 ...
+var.eigen.K3/var.eigen.K4*a.dK4.dId{m}/var.eigen.K4*a.dK5.dtheta{k} ...
-a.dK3.dtheta{k}/var.eigen.K4*a.dK5.dId{m} ...
+var.eigen.K3/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4*a.dK5.dId{m} ...
-var.eigen.K3/var.eigen.K4*a.d2K5.dId_dtheta{m,k};
a.d2Asys.dtheta_dId{k,m} = -a.dB1.dId{m}/var.eigen.D1*a.dC1.dtheta{k} ...
-a.d2K3.dtheta_dId{k,m}/var.eigen.K4*var.eigen.K5 ...
+a.dK3.dId{m}/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4*var.eigen.K5 ...
-a.dK3.dId{m}/var.eigen.K4*a.dK5.dtheta{k} ...
+a.dK3.dtheta{k}/var.eigen.K4*a.dK4.dId{m}/var.eigen.K4*var.eigen.K5 ...
-var.eigen.K3*( ...
-K4inv*a.d2K4.dtheta_dId{k,m}/var.eigen.K4 ...
+K4inv*a.dK4.dId{m}/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4 ...
+K4inv*a.dK4.dtheta{k}/var.eigen.K4*a.dK4.dId{m}/var.eigen.K4 ...
)*var.eigen.K5 ...
+var.eigen.K3/var.eigen.K4*a.dK4.dId{m}/var.eigen.K4*a.dK5.dtheta{k} ...
-a.dK3.dtheta{k}/var.eigen.K4*a.dK5.dId{m} ...
+var.eigen.K3/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4*a.dK5.dId{m} ...
-var.eigen.K3/var.eigen.K4*a.d2K5.dtheta_dId{k,m};
a.d2Asys.dIq_dtheta{m,k} = -a.dB1.dIq{m}/var.eigen.D1*a.dC1.dtheta{k} ...
-a.d2K3.dIq_dtheta{m,k}/var.eigen.K4*var.eigen.K5 ...
+a.dK3.dIq{m}/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4*var.eigen.K5 ...
-a.dK3.dIq{m}/var.eigen.K4*a.dK5.dtheta{k} ...
+a.dK3.dtheta{k}/var.eigen.K4*a.dK4.dIq{m}/var.eigen.K4*var.eigen.K5 ...
-var.eigen.K3*( ...
-K4inv*a.d2K4.dIq_dtheta{m,k}/var.eigen.K4 ...
+K4inv*a.dK4.dIq{m}/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4 ...
+K4inv*a.dK4.dtheta{k}/var.eigen.K4*a.dK4.dIq{m}/var.eigen.K4 ...
)*var.eigen.K5 ...
+var.eigen.K3/var.eigen.K4*a.dK4.dIq{m}/var.eigen.K4*a.dK5.dtheta{k} ...
-a.dK3.dtheta{k}/var.eigen.K4*a.dK5.dIq{m} ...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

+var.eigen.K3/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4*a.dK5.dIq{m} ...
-var.eigen.K3/var.eigen.K4*a.d2K5.dIq_dtheta{m,k};
a.d2Asys.dtheta_dIq{k,m} = -a.dB1.dIq{m}/var.eigen.D1*a.dC1.dtheta{k} ...
-a.d2K3.dtheta_dIq{k,m}/var.eigen.K4*var.eigen.K5 ...
+a.dK3.dIq{m}/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4*var.eigen.K5 ...
-a.dK3.dIq{m}/var.eigen.K4*a.dK5.dtheta{k} ...
+a.dK3.dtheta{k}/var.eigen.K4*a.dK4.dIq{m}/var.eigen.K4*var.eigen.K5 ...
-var.eigen.K3*( ...
-K4inv*a.d2K4.dtheta_dIq{k,m}/var.eigen.K4 ...
+K4inv*a.dK4.dIq{m}/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4 ...
+K4inv*a.dK4.dtheta{k}/var.eigen.K4*a.dK4.dIq{m}/var.eigen.K4 ...
)*var.eigen.K5 ...
+var.eigen.K3/var.eigen.K4*a.dK4.dIq{m}/var.eigen.K4*a.dK5.dtheta{k} ...
-a.dK3.dtheta{k}/var.eigen.K4*a.dK5.dIq{m} ...
+var.eigen.K3/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4*a.dK5.dIq{m} ...
-var.eigen.K3/var.eigen.K4*a.d2K5.dtheta_dIq{k,m};
%-----
a.d2Asys.ddelta_dV{m,k} = -var.eigen.B1/var.eigen.D1*a.d2C1.ddelta_dV{m,k} ...
-a.d2K3.ddelta_dV{m,k}/var.eigen.K4*var.eigen.K5 ...
+a.dK3.ddelta{m}/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*var.eigen.K5 ...
-a.dK3.ddelta{m}/var.eigen.K4*a.dK5.dV{k} ...
+a.dK3.dV{k}/var.eigen.K4*a.dK4.ddelta{m}/var.eigen.K4*var.eigen.K5 ...
-var.eigen.K3*( ...
-K4inv*a.d2K4.ddelta_dV{m,k}/var.eigen.K4 ...
+K4inv*a.dK4.ddelta{m}/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4 ...
+K4inv*a.dK4.dV{k}/var.eigen.K4*a.dK4.ddelta{m}/var.eigen.K4 ...
)*var.eigen.K5 ...
+var.eigen.K3/var.eigen.K4*a.dK4.ddelta{m}/var.eigen.K4*a.dK5.dV{k} ...
-a.dK3.dV{k}/var.eigen.K4*a.dK5.ddelta{m} ...
+var.eigen.K3/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*a.dK5.ddelta{m} ...
-var.eigen.K3/var.eigen.K4*a.d2K5.ddelta_dV{m,k};
a.d2Asys.dV_ddelta{k,m} = -var.eigen.B1/var.eigen.D1*a.d2C1.dV_ddelta{k,m} ...
-a.d2K3.dV_ddelta{k,m}/var.eigen.K4*var.eigen.K5 ...
+a.dK3.ddelta{m}/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*var.eigen.K5 ...
-a.dK3.ddelta{m}/var.eigen.K4*a.dK5.dV{k} ...
+a.dK3.dV{k}/var.eigen.K4*a.dK4.ddelta{m}/var.eigen.K4*var.eigen.K5 ...
-var.eigen.K3*( ...
-K4inv*a.d2K4.dV_ddelta{k,m}/var.eigen.K4 ...
+K4inv*a.dK4.ddelta{m}/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4 ...
+K4inv*a.dK4.dV{k}/var.eigen.K4*a.dK4.ddelta{m}/var.eigen.K4 ...
)*var.eigen.K5 ...
+var.eigen.K3/var.eigen.K4*a.dK4.ddelta{m}/var.eigen.K4*a.dK5.dV{k} ...
-a.dK3.dV{k}/var.eigen.K4*a.dK5.ddelta{m} ...
+var.eigen.K3/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*a.dK5.ddelta{m} ...
-var.eigen.K3/var.eigen.K4*a.d2K5.dV_ddelta{k,m};
%-----
a.d2Asys.ddelta_dtheta{m,k} = -var.eigen.B1/var.eigen.D1*a.d2C1.ddelta_dtheta{m,k} ...
-a.d2K3.ddelta_dtheta{m,k}/var.eigen.K4*var.eigen.K5 ...
+a.dK3.ddelta{m}/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4*var.eigen.K5 ...
-a.dK3.ddelta{m}/var.eigen.K4*a.dK5.dtheta{k} ...
+a.dK3.dtheta{k}/var.eigen.K4*a.dK4.ddelta{m}/var.eigen.K4*var.eigen.K5 ...
-var.eigen.K3*( ...
-K4inv*a.d2K4.ddelta_dtheta{m,k}/var.eigen.K4 ...
+K4inv*a.dK4.ddelta{m}/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4 ...
+K4inv*a.dK4.dtheta{k}/var.eigen.K4*a.dK4.ddelta{m}/var.eigen.K4 ...
)*var.eigen.K5 ...
+var.eigen.K3/var.eigen.K4*a.dK4.ddelta{m}/var.eigen.K4*a.dK5.dtheta{k} ...
-a.dK3.dtheta{k}/var.eigen.K4*a.dK5.ddelta{m} ...
+var.eigen.K3/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4*a.dK5.ddelta{m} ...
-var.eigen.K3/var.eigen.K4*a.d2K5.ddelta_dtheta{m,k};
a.d2Asys.dtheta_ddelta{k,m} = -var.eigen.B1/var.eigen.D1*a.d2C1.dtheta_ddelta{k,m} ...
-a.d2K3.dtheta_ddelta{k,m}/var.eigen.K4*var.eigen.K5 ...
+a.dK3.ddelta{m}/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4*var.eigen.K5 ...
-a.dK3.ddelta{m}/var.eigen.K4*a.dK5.dtheta{k} ...
+a.dK3.dtheta{k}/var.eigen.K4*a.dK4.ddelta{m}/var.eigen.K4*var.eigen.K5 ...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

    -var.eigen.K3*( ...
    -K4inv*a.d2K4.dtheta_ddelta{k,m}/var.eigen.K4 ...
    +K4inv*a.dK4.ddelta{m}/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4 ...
    +K4inv*a.dK4.dtheta{k}/var.eigen.K4*a.dK4.ddelta{m}/var.eigen.K4 ...
    )*var.eigen.K5 ...
    +var.eigen.K3/var.eigen.K4*a.dK4.ddelta{m}/var.eigen.K4*a.dK5.dtheta{k} ...
    -a.dK3.dtheta{k}/var.eigen.K4*a.dK5.ddelta{m} ...
    +var.eigen.K3/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4*a.dK5.ddelta{m} ...
    -var.eigen.K3/var.eigen.K4*a.d2K5.dtheta_ddelta{k,m};
%-----
a.d2Asys.dV_dEdp{k,m} = -a.dB1.dEdp{m}/var.eigen.D1*a.dC1.dV{k} ...
    -a.d2K3.dV_dEdp{k,m}/var.eigen.K4*var.eigen.K5 ...
    +a.dK3.dEdp{m}/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dEdp{m}/var.eigen.K4*a.dK5.dV{k};
a.d2Asys.dEdp_dV{m,k} = -a.dB1.dEdp{m}/var.eigen.D1*a.dC1.dV{k} ...
    -a.d2K3.dEdp_dV{m,k}/var.eigen.K4*var.eigen.K5 ...
    +a.dK3.dEdp{m}/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dEdp{m}/var.eigen.K4*a.dK5.dV{k};
%-----
a.d2Asys.dV_dEqp{k,m} = -a.dB1.dEqp{m}/var.eigen.D1*a.dC1.dV{k} ...
    -a.d2K3.dV_dEqp{k,m}/var.eigen.K4*var.eigen.K5 ...
    +a.dK3.dEqp{m}/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dEqp{m}/var.eigen.K4*a.dK5.dV{k};
a.d2Asys.dEqp_dV{m,k} = -a.dB1.dEqp{m}/var.eigen.D1*a.dC1.dV{k} ...
    -a.d2K3.dEqp_dV{m,k}/var.eigen.K4*var.eigen.K5 ...
    +a.dK3.dEqp{m}/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dEqp{m}/var.eigen.K4*a.dK5.dV{k};
%-----
a.d2Asys.dtheta_dEdp{k,m} = -a.dB1.dEdp{m}/var.eigen.D1*a.dC1.dtheta{k} ...
    -a.d2K3.dtheta_dEdp{k,m}/var.eigen.K4*var.eigen.K5 ...
    +a.dK3.dEdp{m}/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dEdp{m}/var.eigen.K4*a.dK5.dtheta{k};
a.d2Asys.dEdp_dtheta{m,k} = -a.dB1.dEdp{m}/var.eigen.D1*a.dC1.dtheta{k} ...
    -a.d2K3.dEdp_dtheta{m,k}/var.eigen.K4*var.eigen.K5 ...
    +a.dK3.dEdp{m}/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dEdp{m}/var.eigen.K4*a.dK5.dtheta{k};
%-----
a.d2Asys.dtheta_dEqp{k,m} = -a.dB1.dEqp{m}/var.eigen.D1*a.dC1.dtheta{k} ...
    -a.d2K3.dtheta_dEqp{k,m}/var.eigen.K4*var.eigen.K5 ...
    +a.dK3.dEqp{m}/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dEqp{m}/var.eigen.K4*a.dK5.dtheta{k};
a.d2Asys.dEqp_dtheta{m,k} = -a.dB1.dEqp{m}/var.eigen.D1*a.dC1.dtheta{k} ...
    -a.d2K3.dEqp_dtheta{m,k}/var.eigen.K4*var.eigen.K5 ...
    +a.dK3.dEqp{m}/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dEqp{m}/var.eigen.K4*a.dK5.dtheta{k};
%-----
end%if
%-----
a.d2Asys.dV_dtheta{k,m} = -var.eigen.B1/var.eigen.D1*a.d2C1.dV_dtheta{k,m} ...
    -a.d2K3.dV_dtheta{k,m}/var.eigen.K4*var.eigen.K5 ...
    +a.dK3.dtheta{m}/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*var.eigen.K5 ...
    -a.dK3.dtheta{m}/var.eigen.K4*a.dK5.dV{k} ...
    +a.dK3.dV{k}/var.eigen.K4*a.dK4.dtheta{m}/var.eigen.K4*var.eigen.K5 ...
    -var.eigen.K3*( ...
    -K4inv*a.d2K4.dV_dtheta{k,m}/var.eigen.K4 ...
    +K4inv*a.dK4.dtheta{m}/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4 ...
    +K4inv*a.dK4.dV{k}/var.eigen.K4*a.dK4.dtheta{m}/var.eigen.K4 ...
    )*var.eigen.K5 ...
    +var.eigen.K3/var.eigen.K4*a.dK4.dtheta{m}/var.eigen.K4*a.dK5.dV{k} ...
    -a.dK3.dV{k}/var.eigen.K4*a.dK5.dtheta{m} ...
    +var.eigen.K3/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*a.dK5.dtheta{m} ...
    -var.eigen.K3/var.eigen.K4*a.d2K5.dV_dtheta{k,m};
%-----
a.d2Asys.dtheta_dV{m,k} = -var.eigen.B1/var.eigen.D1*a.d2C1.dtheta_dV{k,m} ...
    -a.d2K3.dtheta_dV{m,k}/var.eigen.K4*var.eigen.K5 ...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

+ a.dK3.dtheta{m}/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*var.eigen.K5 ...
-a.dK3.dtheta{m}/var.eigen.K4*a.dK5.dV{k} ...
+ a.dK3.dV{k}/var.eigen.K4*a.dK4.dtheta{m}/var.eigen.K4*var.eigen.K5 ...
-var.eigen.K3*( ...
-K4inv*a.d2K4.dtheta_dV{m,k}/var.eigen.K4 ...
+K4inv*a.dK4.dtheta{m}/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4 ...
+K4inv*a.dK4.dV{k}/var.eigen.K4*a.dK4.dtheta{m}/var.eigen.K4 ...
)*var.eigen.K5 ...
+var.eigen.K3/var.eigen.K4*a.dK4.dtheta{m}/var.eigen.K4*a.dK5.dV{k} ...
-a.dK3.dV{k}/var.eigen.K4*a.dK5.dtheta{m} ...
+var.eigen.K3/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*a.dK5.dtheta{m} ...
-var.eigen.K3/var.eigen.K4*a.d2K5.dtheta_dV{m,k};
%-----
a.d2Asys.dV_dV{k,m} = ...
+ a.dK3.dV{m}/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*var.eigen.K5 ...
-a.dK3.dV{m}/var.eigen.K4*a.dK5.dV{k} ...
+ a.dK3.dV{k}/var.eigen.K4*a.dK4.dV{m}/var.eigen.K4*var.eigen.K5 ...
-var.eigen.K3*( ...
-K4inv*a.d2K4.dV_dV{k,m}/var.eigen.K4 ...
+K4inv*a.dK4.dV{m}/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4 ...
+K4inv*a.dK4.dV{k}/var.eigen.K4*a.dK4.dV{m}/var.eigen.K4 ...
)*var.eigen.K5 ...
+var.eigen.K3/var.eigen.K4*a.dK4.dV{m}/var.eigen.K4*a.dK5.dV{k} ...
-a.dK3.dV{k}/var.eigen.K4*a.dK5.dV{m} ...
+var.eigen.K3/var.eigen.K4*a.dK4.dV{k}/var.eigen.K4*a.dK5.dV{m} ...
-var.eigen.K3/var.eigen.K4*a.d2K5.dV_dV{k,m};
%-----
a.d2Asys.dtheta_dtheta{k,m} = ...
-a.d2K3.dtheta_dtheta{m,k}/var.eigen.K4*var.eigen.K5 ...
+ a.dK3.dtheta{m}/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4*var.eigen.K5 ...
-a.dK3.dtheta{m}/var.eigen.K4*a.dK5.dtheta{k} ...
+ a.dK3.dtheta{k}/var.eigen.K4*a.dK4.dtheta{m}/var.eigen.K4*var.eigen.K5 ...
-var.eigen.K3*( ...
-K4inv*a.d2K4.dtheta_dtheta{k,m}/var.eigen.K4 ...
+K4inv*a.dK4.dtheta{m}/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4 ...
+K4inv*a.dK4.dtheta{k}/var.eigen.K4*a.dK4.dtheta{m}/var.eigen.K4 ...
)*var.eigen.K5 ...
+var.eigen.K3/var.eigen.K4*a.dK4.dtheta{m}/var.eigen.K4*a.dK5.dtheta{k} ...
-a.dK3.dtheta{k}/var.eigen.K4*a.dK5.dtheta{m} ...
+var.eigen.K3/var.eigen.K4*a.dK4.dtheta{k}/var.eigen.K4*a.dK5.dtheta{m} ...
-var.eigen.K3/var.eigen.K4*a.d2K5.dtheta_dtheta{k,m};
%-----
end%for
end%for
% a.dAsys.dIq = a.dh.dA1.dIq - a.dh.dB1.dIq/eig.D1*eig.C1 - ...
% - a.dh.dK3.dIq/eig.K4*eig.K5 ...
% - eig.K3/eig.K4*a.dh.dK4.dIq/eig.K4*eig.K5 ...
% - eig.K3/eig.K4*a.dh.dK5.dIq;
%*****
% Right var.eigenvector derivatives
%*****
%-----
% Id, Iq, Edp, Eqp, Efd, delta
%-----

```

H.22 getstate.m

```

function current=getstate(data,index,var,k)
if k==0
current.data.pi = data.line.pi_0;
current.data.fr = data.fr;
current.data.to = data.to;
current.data.Ybus = data.pre.Ybus;
current.data.Yvec = data.Yvec;
current.data.Bvec = data.Bvec;
current.data.nlines = data.nlines;
current.data.nbuses = data.nbuses;
current.data.Pgenbuses = data.Pgenbuses;
current.data.Ploadbuses = data.Ploadbuses;
current.data.Qgenbuses = data.Qgenbuses;
current.data.nPgen = data.nPgen;
current.data.nQgen = data.nQgen;
current.data.nload = data.nload;
current.data.kpf = data.kpf;
current.data.slackbus = data.slackbus;
current.data.limits.Pgmax = data.limits.Pgmax;
current.data.limits.Pgmin = data.limits.Pgmin;
current.data.limits.Plmin = data.limits.Plmin;
current.data.limits.Qgmax = data.limits.Qgmax;
current.data.limits.Qgmin = data.limits.Qgmin;
current.data.limits.volt_max = data.limits.volt_max;
current.data.limits.volt_min = data.limits.volt_min;
current.data.limits.Sijmax = data.limits.Sijmax;
current.data.machine.Xd = data.machine.Xd;
current.data.machine.Xdp = data.machine.Xdp;
current.data.machine.Xdpp = data.machine.Xdpp;
current.data.machine.Xq = data.machine.Xq;
current.data.machine.Xqp = data.machine.Xqp;
current.data.machine.Xqpp = data.machine.Xqpp;
current.data.machine.XL = data.machine.XL;
current.data.machine.Tdop = data.machine.Tdop;
current.data.machine.Tdopp = data.machine.Tdopp;
current.data.machine.Tqop = data.machine.Tqop;
current.data.machine.Tqopp = data.machine.Tqopp;
current.data.machine.Ra = data.machine.Ra;
current.data.machine.H = data.machine.H;
current.data.machine.KD = data.machine.KD;
current.data.machine.KA = data.machine.KA;
current.data.machine.TA = data.machine.TA;
current.data.machine.KE = data.machine.KE;
current.data.machine.TE = data.machine.TE;
current.data.machine.KF = data.machine.KF;
current.data.machine.TF = data.machine.TF;
current.data.machine.D = data.machine.D;
current.var.V = var.pre.V;
current.var.theta = var.pre.theta;
current.var.Pg = var.pre.Pg;
current.var.Qg = var.pre.Qg;
current.var.Pl = var.pre.Pl;
current.var.Pij = var.pre.Pij;
current.var.Qij = var.pre.Qij;
current.var.Ig = var.pre.Ig;
current.var.gamma = var.pre.gamma;
current.var.Adelt = var.pre.Adelt;
current.var.delta = var.pre.delta;
current.var.Id = var.pre.Id;
current.var.Iq = var.pre.Iq;
current.var.Vd = var.pre.Vd;
current.var.Vq = var.pre.Vq;
current.var.Edp = var.pre.Edp;

```


APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

current.var.Eqp = var.pre.Eqp;
current.var.Efd = var.pre.Efd;
current.var.Sij = var.pre.Sij;
current.var.Iij = var.pre.Iij;
current.var.eigen = var.pre.eigen;
current.var.eigenvalues = var.pre.eigenvalues;
current.var.eq.lagrange.P = var.pre.eq.lagrange.P ;
current.var.eq.lagrange.Q = var.pre.eq.lagrange.Q ;
current.var.eq.lagrange.Pij = var.pre.eq.lagrange.Pij ;
current.var.eq.lagrange.Qij = var.pre.eq.lagrange.Qij ;
current.var.eq.lagrange.Sij = var.pre.eq.lagrange.Sij ;
current.var.eq.lagrange.Ig_real = var.pre.eq.lagrange.Ig_real ;
current.var.eq.lagrange.Ig_imag = var.pre.eq.lagrange.Ig_imag ;
current.var.eq.lagrange.delta_real = var.pre.eq.lagrange.delta_real ;
current.var.eq.lagrange.delta_imag = var.pre.eq.lagrange.delta_imag ;
current.var.eq.lagrange.Idq_real = var.pre.eq.lagrange.Idq_real ;
current.var.eq.lagrange.Idq_imag = var.pre.eq.lagrange.Idq_imag ;
current.var.eq.lagrange.Vdq_real = var.pre.eq.lagrange.Vdq_real ;
current.var.eq.lagrange.Vdq_imag = var.pre.eq.lagrange.Vdq_imag ;
current.var.eq.lagrange.Edp = var.pre.eq.lagrange.Edp ;
current.var.eq.lagrange.Eqp = var.pre.eq.lagrange.Eqp ;
current.var.eq.lagrange.Efd = var.pre.eq.lagrange.Efd ;
current.var.eq.lagrange.slack = var.pre.eq.lagrange.slack ;
current.ind.V.ind = index.pre.V.ind;
current.ind.theta.ind = index.pre.theta.ind;
current.ind.Pg.ind = index.pre.Pg.ind;
current.ind.Qg.ind = index.pre.Qg.ind;
current.ind.Pl.ind = index.pre.Pl.ind;
current.ind.Pij.ind = index.pre.Pij.ind;
current.ind.Qij.ind = index.pre.Qij.ind;
current.ind.Sij.ind = index.pre.Sij.ind;
current.ind.Ig.ind = index.pre.Ig.ind;
current.ind.gamma.ind = index.pre.gamma.ind;
current.ind.Adelt.ind = index.pre.Adelt.ind;
current.ind.delta.ind = index.pre.delta.ind;
current.ind.Id.ind = index.pre.Id.ind;
current.ind.Iq.ind = index.pre.Iq.ind;
current.ind.Vd.ind = index.pre.Vd.ind;
current.ind.Vq.ind = index.pre.Vq.ind;
current.ind.Edp.ind = index.pre.Edp.ind;
current.ind.Eqp.ind = index.pre.Eqp.ind;
current.ind.Efd.ind = index.pre.Efd.ind;
%current.ind.Y_len = index.pre.Y_len;
current.ind.eq.P.ind = index.pre.eq.P.ind;
current.ind.eq.Q.ind = index.pre.eq.Q.ind;
current.ind.eq.Pij.ind = index.pre.eq.Pij.ind;
current.ind.eq.Qij.ind = index.pre.eq.Qij.ind;
current.ind.eq.Sij.ind = index.pre.eq.Sij.ind;
current.ind.eq.Ig_real.ind = index.pre.eq.Ig_real.ind;
current.ind.eq.Ig_imag.ind = index.pre.eq.Ig_imag.ind;
current.ind.eq.delta_real.ind = index.pre.eq.delta_real.ind;
current.ind.eq.delta_imag.ind = index.pre.eq.delta_imag.ind;
current.ind.eq.Idq_real.ind = index.pre.eq.Idq_real.ind;
current.ind.eq.Idq_imag.ind = index.pre.eq.Idq_imag.ind;
current.ind.eq.Vdq_real.ind = index.pre.eq.Vdq_real.ind;
current.ind.eq.Vdq_imag.ind = index.pre.eq.Vdq_imag.ind;
current.ind.eq.Edp.ind = index.pre.eq.Edp.ind;
current.ind.eq.Eqp.ind = index.pre.eq.Eqp.ind;
current.ind.eq.Efd.ind = index.pre.eq.Efd.ind;
current.ind.eq.slack.ind = index.pre.eq.slack.ind;
current.ind.eq.lagrange.ind = index.pre.eq.lagrange.ind;
current.ind.ineq.Pgmax.ind = index.pre.ineq.Pgmax.ind;
current.ind.ineq.Pgmin.ind = index.pre.ineq.Pgmin.ind;
current.ind.ineq.Vmax.ind = index.pre.ineq.Vmax.ind;
current.ind.ineq.Vmin.ind = index.pre.ineq.Vmin.ind;

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

current.ind.ineq.Qgmax.ind = index.pre.ineq.Qgmax.ind;
current.ind.ineq.Qgmin.ind = index.pre.ineq.Qgmin.ind;
current.ind.ineq.eig_real.ind = index.pre.ineq.eig_real.ind;
current.ind.ineq.Plmin.ind = index.pre.ineq.Plmin.ind;
current.ind.ineq.Sijmax.ind = index.pre.ineq.Sijmax.ind;
current.ind.ineq.Igmin.ind = index.pre.ineq.Igmin.ind;
current.ind.ineq.Adeltmin.ind = index.pre.ineq.Adeltmin.ind;
end%if

if k>0
current.data.pi = data.line.pi_k(k);
current.data.fr = data.fr;
current.data.to = data.to;
current.data.fr(k) = [];
current.data.to(k) = [];
current.data.nlines = data.nlines-1;
current.data.nbuses = data.nbuses;
current.data.nload = data.nload;
current.data.slackbus = data.slackbus;
current.data.kpf = data.kpf;
current.data.Pgenbuses = data.Pgenbuses;
current.data.Ploadbuses = data.Ploadbuses;
current.data.Qgenbuses = data.Qgenbuses;
current.data.nPgen = data.nPgen;
current.data.nQgen = data.nQgen;
current.data.Yvec = data.Yvec;
current.data.Bvec = data.Bvec;
current.data.Yvec(k) = [];
current.data.Bvec(k) = [];
current.data.Ybus = data.post.Ybus[k];
current.data.limits.Pgmax = data.limits.Pgmax;
current.data.limits.Pgmin = data.limits.Pgmin;
current.data.limits.Plmin = data.limits.Plmin;
current.data.limits.Qgmax = data.limits.Qgmax;
current.data.limits.Qgmin = data.limits.Qgmin;
current.data.limits.volt_max = data.limits.volt_max;
current.data.limits.volt_min = data.limits.volt_min;
current.data.limits.Sijmax = data.limits.Sijmax_emerg;
current.data.limits.Sijmax(k) = [];
current.data.machine.Xd = data.machine.Xd;
current.data.machine.Xdp = data.machine.Xdp;
current.data.machine.Xdpp = data.machine.Xdpp;
current.data.machine.Xq = data.machine.Xq;
current.data.machine.Xqp = data.machine.Xqp;
current.data.machine.Xqpp = data.machine.Xqpp;
current.data.machine.XL = data.machine.XL;
current.data.machine.Tdop = data.machine.Tdop;
current.data.machine.Tdopp = data.machine.Tdopp;
current.data.machine.Tqop = data.machine.Tqop;
current.data.machine.Tqopp = data.machine.Tqopp;
current.data.machine.Ra = data.machine.Ra;
current.data.machine.H = data.machine.H;
current.data.machine.KD = data.machine.KD;
current.data.machine.KA = data.machine.KA;
current.data.machine.TA = data.machine.TA;
current.data.machine.KE = data.machine.KE;
current.data.machine.TE = data.machine.TE;
current.data.machine.KF = data.machine.KF;
current.data.machine.TF = data.machine.TF;
current.data.machine.D = data.machine.D;
current.var.V = var.post.V(:,k);
current.var.theta = var.post.theta(:,k);
current.var.Pg = var.post.Pg(:,k);
current.var.Qg = var.post.Qg(:,k);
current.var.Pl = var.post.Pl(:,k);

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

current.var.Pij = var.post.Pij(:,k);
current.var.Qij = var.post.Qij(:,k);
current.var.Sij = var.post.Sij(:,k);
current.var.Ig = var.post.Ig(:,k);
current.var.gamma = var.post.gamma(:,k);
current.var.Adelt = var.post.Adelt(:,k);
current.var.delta = var.post.delta(:,k);
current.var.Id = var.post.Id(:,k);
current.var.Iq = var.post.Iq(:,k);
current.var.Vd = var.post.Vd(:,k);
current.var.Vq = var.post.Vq(:,k);
current.var.Edp = var.post.Edp(:,k);
current.var.Eqp = var.post.Eqp(:,k);
current.var.Efd = var.post.Efd(:,k);
current.var.Iij = var.post.Iij(:,k);
current.var.eigen = var.post.eigen{k};
current.var.eigenvalues = var.post.eigenvalues(:,k);
current.var.eq.lagrange.P = var.post.eq.lagrange.P(:,k);
current.var.eq.lagrange.Q = var.post.eq.lagrange.Q(:,k);
current.var.eq.lagrange.Pij = var.post.eq.lagrange.Pij(:,k);
current.var.eq.lagrange.Qij = var.post.eq.lagrange.Qij(:,k);
current.var.eq.lagrange.Sij = var.post.eq.lagrange.Sij(:,k);
current.var.eq.lagrange.Ig_real = var.post.eq.lagrange.Ig_real(:,k);
current.var.eq.lagrange.Ig_imag = var.post.eq.lagrange.Ig_imag(:,k);
current.var.eq.lagrange.delta_real = var.post.eq.lagrange.delta_real(:,k);
current.var.eq.lagrange.delta_imag = var.post.eq.lagrange.delta_imag(:,k);
current.var.eq.lagrange.Idq_real = var.post.eq.lagrange.Idq_real(:,k);
current.var.eq.lagrange.Idq_imag = var.post.eq.lagrange.Idq_imag(:,k);
current.var.eq.lagrange.Vdq_real = var.post.eq.lagrange.Vdq_real(:,k);
current.var.eq.lagrange.Vdq_imag = var.post.eq.lagrange.Vdq_imag(:,k);
current.var.eq.lagrange.Edp = var.post.eq.lagrange.Edp(:,k);
current.var.eq.lagrange.Eqp = var.post.eq.lagrange.Eqp(:,k);
current.var.eq.lagrange.Efd = var.post.eq.lagrange.Efd(:,k);
current.var.eq.lagrange.slack = var.post.eq.lagrange.slack(:,k);
current.ind.V.ind = index.post.V.ind(k,:);
current.ind.theta.ind = index.post.theta.ind(k,:);
current.ind.Pg.ind = index.post.Pg.ind(k,:);
current.ind.Qg.ind = index.post.Qg.ind(k,:);
current.ind.Pl.ind = index.post.Pl.ind(k,:);
current.ind.Pij.ind = index.post.Pij.ind(k,:);
current.ind.Qij.ind = index.post.Qij.ind(k,:);
current.ind.Sij.ind = index.post.Sij.ind(k,:);
current.ind.Ig.ind = index.post.Ig.ind(k,:);
current.ind.gamma.ind = index.post.gamma.ind(k,:);
current.ind.Adelt.ind = index.post.Adelt.ind(k,:);
current.ind.delta.ind = index.post.delta.ind(k,:);
current.ind.Id.ind = index.post.Id.ind(k,:);
current.ind.Iq.ind = index.post.Iq.ind(k,:);
current.ind.Vd.ind = index.post.Vd.ind(k,:);
current.ind.Vq.ind = index.post.Vq.ind(k,:);
current.ind.Edp.ind = index.post.Edp.ind(k,:);
current.ind.Eqp.ind = index.post.Eqp.ind(k,:);
current.ind.Efd.ind = index.post.Efd.ind(k,:);
%current.ind.Y_len = index.post.Y_len;
current.ind.eq.P.ind = index.post.eq.P.ind+index.pre.neq ...
    +(k-1)*index.post.neq;
current.ind.eq.Q.ind = index.post.eq.Q.ind+index.pre.neq ...
    +(k-1)*index.post.neq;
current.ind.eq.Pij.ind = index.post.eq.Pij.ind+index.pre.neq ...
    +(k-1)*index.post.neq;
current.ind.eq.Qij.ind = index.post.eq.Qij.ind+index.pre.neq ...
    +(k-1)*index.post.neq;
current.ind.eq.Sij.ind = index.post.eq.Sij.ind+index.pre.neq ...
    +(k-1)*index.post.neq;
current.ind.eq.Ig_real.ind = index.post.eq.Ig_real.ind+index.pre.neq ...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

        +(k-1)*index.post.neq;
current.ind.eq.Ig_imag.ind = index.post.eq.Ig_imag.ind+index.pre.neq ...
        +(k-1)*index.post.neq;
current.ind.eq.delta_real.ind = index.post.eq.delta_real.ind+index.pre.neq ...
        +(k-1)*index.post.neq;
current.ind.eq.delta_imag.ind = index.post.eq.delta_imag.ind+index.pre.neq ...
        +(k-1)*index.post.neq;
current.ind.eq.Idq_real.ind = index.post.eq.Idq_real.ind+index.pre.neq ...
        +(k-1)*index.post.neq;
current.ind.eq.Idq_imag.ind = index.post.eq.Idq_imag.ind+index.pre.neq ...
        +(k-1)*index.post.neq;
current.ind.eq.Vdq_real.ind = index.post.eq.Vdq_real.ind+index.pre.neq ...
        +(k-1)*index.post.neq;
current.ind.eq.Vdq_imag.ind = index.post.eq.Vdq_imag.ind+index.pre.neq ...
        +(k-1)*index.post.neq;
current.ind.eq.Edp.ind = index.post.eq.Edp.ind+index.pre.neq ...
        +(k-1)*index.post.neq;
current.ind.eq.Eqp.ind = index.post.eq.Eqp.ind+index.pre.neq ...
        +(k-1)*index.post.neq;
current.ind.eq.Efd.ind = index.post.eq.Efd.ind+index.pre.neq ...
        +(k-1)*index.post.neq;
current.ind.eq.slack.ind = index.post.eq.slack.ind+index.pre.neq ...
        +(k-1)*index.post.neq;
current.ind.eq.lagrange.ind = index.post.eq.lagrange.ind(k,:);
current.ind.ineq.Pgmax.ind = index.post.ineq.Pgmax.ind+index.pre.nineq ...
        +(k-1)*index.post.nineq;
current.ind.ineq.Pgmin.ind = index.post.ineq.Pgmin.ind+index.pre.nineq ...
        +(k-1)*index.post.nineq;
current.ind.ineq.Vmax.ind = index.post.ineq.Vmax.ind+index.pre.nineq ...
        +(k-1)*index.post.nineq;
current.ind.ineq.Vmin.ind = index.post.ineq.Vmin.ind+index.pre.nineq ...
        +(k-1)*index.post.nineq;
current.ind.ineq.Qgmax.ind = index.post.ineq.Qgmax.ind+index.pre.nineq ...
        +(k-1)*index.post.nineq;
current.ind.ineq.Qgmin.ind = index.post.ineq.Qgmin.ind+index.pre.nineq ...
        +(k-1)*index.post.nineq;
current.ind.ineq.eig_real.ind = index.post.ineq.eig_real.ind+index.pre.nineq ...
        +(k-1)*index.post.nineq;
current.ind.ineq.Plmin.ind = index.post.ineq.Plmin.ind+index.pre.nineq ...
        +(k-1)*index.post.nineq;
current.ind.ineq.Sijmax.ind = index.post.ineq.Sijmax.ind+index.pre.nineq ...
        +(k-1)*index.post.nineq;
current.ind.ineq.Igmin.ind = index.post.ineq.Igmin.ind+index.pre.nineq ...
        +(k-1)*index.post.nineq;
current.ind.ineq.Adeltmin.ind = index.post.ineq.Adeltmin.ind+index.pre.nineq ...
        +(k-1)*index.post.nineq;
current.ind.ineq.Pgrampup.ind = index.post.ineq.Pgrampup.ind+index.pre.nineq ...
        +(k-1)*index.post.nineq;
current.ind.ineq.Pgrampdwn.ind = index.post.ineq.Pgrampdwn.ind+index.pre.nineq ...
        +(k-1)*index.post.nineq;
current.ind.ineq.Plmax.ind = index.post.ineq.Plmax.ind+index.pre.nineq ...
        +(k-1)*index.post.nineq;
end%if

```

H.23 eigderiv.m

```

function y=eigderiv(data,a,m,eigen)

%*****
% First-order eigenvalue derivatives
%*****
%-----
% Id, Iq, Edp, Eqp, Efd, delta
%-----

neig = length(eigen.values);
sizeA = length(eigen.Asys);
%psi = eye(length(eigen.phi))/eigen.phi;
psi = inviteration(eigen);
AminusLamI = eigen.Asys-eye(length(eigen.Asys))*eigen.values(m);
Q = inv(AminusLamI+eigen.phi(:,m)*eigen.phi(:,m)');
AminusLam = eigen.Asys-eye(length(eigen.Asys))*real(eigen.values(m));
LamI = imag(eigen.values(m))*eye(length(eigen.Asys));
for k=1:data.nbuses
if isempty(find(data.Pgenbuses==k))==0
y.deig.dId(k) = psi(m,:)*a.dAsys.dId{k}*eigen.phi(:,m) ...
/(psi(m,:)*eigen.phi(:,m));
y.deig.dIq(k) = psi(m,:)*a.dAsys.dIq{k}*eigen.phi(:,m) ...
/(psi(m,:)*eigen.phi(:,m));
y.deig.dEdp(k) = psi(m,:)*a.dAsys.dEdp{k}*eigen.phi(:,m) ...
/(psi(m,:)*eigen.phi(:,m));
y.deig.dEqp(k) = psi(m,:)*a.dAsys.dEqp{k}*eigen.phi(:,m) ...
/(psi(m,:)*eigen.phi(:,m));
y.deig.dEfd(k) = psi(m,:)*a.dAsys.dEfd{k}*eigen.phi(:,m) ...
/(psi(m,:)*eigen.phi(:,m));
y.deig.ddelta(k) = psi(m,:)*a.dAsys.ddelta{k}*eigen.phi(:,m) ...
/(psi(m,:)*eigen.phi(:,m));
z = -(a.dAsys.dId{k}-y.deig.dId(k)*eye(length(eigen.Asys)))*eigen.phi(:,m);
W = [AminusLam LamI
-LamI AminusLam
real((eigen.phi(:,m))' imag((eigen.phi(:,m))')]);
%phiderivdId = W\[real(z); imag(z); 0];
phideriv.dId(:,k)=eigen.phi(:,m)*y.deig.dId(k)-Q*a.dAsys.dId{k}*eigen.phi(:,m);
% phideriv.dId(:,k) = phiderivdId(1:length(eigen.phi)) + ...
% j*phiderivdId(length(eigen.phi)+1:2*length(eigen.phi));
z = -(a.dAsys.dIq{k}-y.deig.dIq(k)*eye(length(eigen.Asys)))*eigen.phi(:,m);
W = [AminusLam LamI
-LamI AminusLam
real((eigen.phi(:,m))' imag((eigen.phi(:,m))')]);
%phiderivdIq = W\[real(z); imag(z); 0];
phideriv.dIq(:,k) = eigen.phi(:,m)*y.deig.dIq(k)-Q*a.dAsys.dIq{k}*eigen.phi(:,m);
% phideriv.dIq(:,k) = phiderivdIq(1:length(eigen.phi)) + ...
% j*phiderivdIq(length(eigen.phi)+1:2*length(eigen.phi));
z = -(a.dAsys.dEdp{k}-y.deig.dEdp(k)*eye(length(eigen.Asys)))*eigen.phi(:,m);
W = [AminusLam LamI
-LamI AminusLam
real((eigen.phi(:,m))' imag((eigen.phi(:,m))')]);
%phiderivdEdp = W\[real(z); imag(z); 0];
phideriv.dEdp(:,k) = eigen.phi(:,m)*y.deig.dEdp(k)-Q*a.dAsys.dEdp{k}*eigen.phi(:,m);
% phideriv.dEdp(:,k) = phiderivdEdp(1:length(eigen.phi)) + ...
% j*phiderivdEdp(length(eigen.phi)+1:2*length(eigen.phi));
z = -(a.dAsys.dEqp{k}-y.deig.dEqp(k)*eye(length(eigen.Asys)))*eigen.phi(:,m);
W = [AminusLam LamI
-LamI AminusLam
real((eigen.phi(:,m))' imag((eigen.phi(:,m))')]);
%phiderivdEqp = W\[real(z); imag(z); 0];
phideriv.dEqp(:,k) = eigen.phi(:,m)*y.deig.dEqp(k)-Q*a.dAsys.dEqp{k}*eigen.phi(:,m);
% phideriv.dEqp(:,k) = phiderivdEqp(1:length(eigen.phi)) + ...
% j*phiderivdEqp(length(eigen.phi)+1:2*length(eigen.phi));
z = -(a.dAsys.dEfd{k}-y.deig.dEfd(k)*eye(length(eigen.Asys)))*eigen.phi(:,m);

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

W = [AminusLam LamI
     -LamI AminusLam
     real((eigen.phi(:,m))) imag((eigen.phi(:,m)))];
%phiderivdEfd = W\[real(z); imag(z); 0];
phideriv.dEfd(:,k) = eigen.phi(:,m)*y.deig.dEfd(k)-Q*a.dAsys.dEfd{k}*eigen.phi(:,m);
% phideriv.dEfd(:,k) = phiderivdEfd(1:length(eigen.phi)) + ...
% j*phiderivdEfd(length(eigen.phi)+1:2*length(eigen.phi));
z = -(a.dAsys.ddelta{k}-y.deig.ddelta(k)*eye(length(eigen.Asys)))*eigen.phi(:,m);
W = [AminusLam LamI
     -LamI AminusLam
     real((eigen.phi(:,m))) imag((eigen.phi(:,m)))];
%phiderivddelta = W\[real(z); imag(z); 0];
phideriv.ddelta(:,k) = eigen.phi(:,m)*y.deig.ddelta(k)-Q*a.dAsys.ddelta{k}*eigen.phi(:,m);
% phideriv.ddelta(:,k) = phiderivddelta(1:length(eigen.phi)) + ...
% j*phiderivddelta(length(eigen.phi)+1:2*length(eigen.phi));
y.d2eig.dEfd_dEfd(k) = ...
    1/(psi(m,)*eigen.phi(:,m)) ...
*( psi(m,)*a.d2Asys.dEfd_dEfd{k}*eigen.phi(:,m) ...
+psi(m,)*a.dAsys.dEfd{k}*phideriv.dEfd(:,k) ...
+psi(m,)*a.dAsys.dEfd{k}*phideriv.dEfd(:,k) );
end%if
y.deig.dV(k) = psi(m,)*a.dAsys.dV{k}*eigen.phi(:,m) ...
/(psi(m,)*eigen.phi(:,m));
y.deig.dtheta(k) = psi(m,)*a.dAsys.dtheta{k}*eigen.phi(:,m) ...
/(psi(m,)*eigen.phi(:,m));
z = -(a.dAsys.dV{k}-y.deig.dV(k)*eye(length(eigen.Asys)))*eigen.phi(:,m);
W = [AminusLam LamI
     -LamI AminusLam
     real((eigen.phi(:,m))) imag((eigen.phi(:,m)))];
%phiderivdV = W\[real(z); imag(z); 0];
phideriv.dV(:,k) = eigen.phi(:,m)*y.deig.dV(k)-Q*a.dAsys.dV{k}*eigen.phi(:,m);
% phideriv.dV(:,k) = phiderivdV(1:length(eigen.phi)) + ...
% j*phiderivdV(length(eigen.phi)+1:2*length(eigen.phi));
z = -(a.dAsys.dtheta{k}-y.deig.dtheta(k)*eye(length(eigen.Asys)))*eigen.phi(:,m);
W = [AminusLam LamI
     -LamI AminusLam
     real((eigen.phi(:,m))) imag((eigen.phi(:,m)))];
%phiderivdtheta = W\[real(z); imag(z); 0];
phideriv.dtheta(:,k) = eigen.phi(:,m)*y.deig.dtheta(k)-Q*a.dAsys.dtheta{k}*eigen.phi(:,m);
% phideriv.dtheta(:,k) = phiderivdtheta(1:length(eigen.phi)) + ...
% j*phiderivdtheta(length(eigen.phi)+1:2*length(eigen.phi));
end%for
W*[real(phideriv.dtheta(:,1)); imag(phideriv.dtheta(:,1))];
for k=1:data.nbuses
if isempty(find(data.Pgenbuses==k))==0
for n=1:data.nPgen
y.d2eig.dId_ddelta(k,n) = ...
    1/(psi(m,)*eigen.phi(:,m)) ...
*( psi(m,)*a.d2Asys.dId_ddelta{k,n}*eigen.phi(:,m) ...
+psi(m,)*(a.dAsys.dId{k}-eye(sizeA)*y.deig.dId(k))*phideriv.ddelta(:,n) ...
+psi(m,)*(a.dAsys.ddelta{n}-eye(sizeA)*y.deig.ddelta(n))*phideriv.dId(:,k) );
y.d2eig.dIq_ddelta(k,n) = ...
    1/(psi(m,)*eigen.phi(:,m)) ...
*( psi(m,)*a.d2Asys.dIq_ddelta{k,n}*eigen.phi(:,m) ...
+psi(m,)*(a.dAsys.dIq{k}-eye(sizeA)*y.deig.dIq(k))*phideriv.ddelta(:,n) ...
+psi(m,)*(a.dAsys.ddelta{n}-eye(sizeA)*y.deig.ddelta(n))*phideriv.dIq(:,k) );
y.d2eig.ddelta_dId(n,k) = ...
    1/(psi(m,)*eigen.phi(:,m)) ...
*( psi(m,)*a.d2Asys.ddelta_dId{n,k}*eigen.phi(:,m) ...
+psi(m,)*(a.dAsys.ddelta{n}-eye(sizeA)*y.deig.ddelta(n))*phideriv.dId(:,k) ...
+psi(m,)*(a.dAsys.dId{k}-eye(sizeA)*y.deig.dId(k))*phideriv.ddelta(:,n) );
y.d2eig.ddelta_dIq(n,k) = ...
    1/(psi(m,)*eigen.phi(:,m)) ...
*( psi(m,)*a.d2Asys.ddelta_dIq{n,k}*eigen.phi(:,m) ...
+psi(m,)*(a.dAsys.ddelta{n}-eye(sizeA)*y.deig.ddelta(n))*phideriv.dIq(:,k) ...
+psi(m,)*(a.dAsys.dIq{k}-eye(sizeA)*y.deig.dIq(k))*phideriv.ddelta(:,n) );

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

+psi(m,:)*(a.dAsys.dIq{k}-eye(sizeA)*y.deig.dIq(k))*phideriv.ddelta(:,n) );
%-----
y.d2eig.ddelta_ddelta(n,k) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.ddelta_ddelta{n,k}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.ddelta{n}-eye(sizeA)*y.deig.ddelta(n))*phideriv.ddelta(:,k) ...
+psi(m,:)*(a.dAsys.ddelta{k}-eye(sizeA)*y.deig.ddelta(k))*phideriv.ddelta(:,n) );
%-----
y.d2eig.dEdp_ddelta(k,n) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dEdp_ddelta{k,n}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dEdp{k}-eye(sizeA)*y.deig.dEdp(k))*phideriv.ddelta(:,n) ...
+psi(m,:)*(a.dAsys.ddelta{n}-eye(sizeA)*y.deig.ddelta(n))*phideriv.dEdp(:,k) );
y.d2eig.ddelta_dEdp(n,k) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.ddelta_dEdp{n,k}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dEdp{k}-eye(sizeA)*y.deig.dEdp(k))*phideriv.ddelta(:,n) ...
+psi(m,:)*(a.dAsys.ddelta{n}-eye(sizeA)*y.deig.ddelta(n))*phideriv.dEdp(:,k) );
%-----
y.d2eig.dEqp_ddelta(k,n) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dEqp_ddelta{k,n}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dEqp{k}-eye(sizeA)*y.deig.dEqp(k))*phideriv.ddelta(:,n) ...
+psi(m,:)*(a.dAsys.ddelta{n}-eye(sizeA)*y.deig.ddelta(n))*phideriv.dEqp(:,k) );
y.d2eig.ddelta_dEqp(n,k) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.ddelta_dEqp{n,k}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dEqp{k}-eye(sizeA)*y.deig.dEqp(k))*phideriv.ddelta(:,k) ...
+psi(m,:)*(a.dAsys.ddelta{n}-eye(sizeA)*y.deig.ddelta(n))*phideriv.dEqp(:,k) );
%-----
y.d2eig.dId_dEdp(k,n) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dId_dEdp{k,n}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dId{k}-eye(sizeA)*y.deig.dId(k))*phideriv.dEdp(:,n) ...
+psi(m,:)*(a.dAsys.dEdp{n}-eye(sizeA)*y.deig.dEdp(n))*phideriv.dId(:,k) );
y.d2eig.dEdp_dId(n,k) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dEdp_dId{n,k}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dEdp{n}-eye(sizeA)*y.deig.dEdp(n))*phideriv.dId(:,k) ...
+psi(m,:)*(a.dAsys.dId{k}-eye(sizeA)*y.deig.dId(k))*phideriv.dEdp(:,n) );
%-----
y.d2eig.dIq_dEdp(k,n) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dIq_dEdp{k,n}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dIq{k}-eye(sizeA)*y.deig.dIq(k))*phideriv.dEdp(:,n) ...
+psi(m,:)*(a.dAsys.dEdp{n}-eye(sizeA)*y.deig.dEdp(n))*phideriv.dIq(:,k) );
y.d2eig.dEdp_dIq(n,k) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dEdp_dIq{n,k}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dEdp{n}-eye(sizeA)*y.deig.dEdp(n))*phideriv.dIq(:,k) ...
+psi(m,:)*(a.dAsys.dIq{k}-eye(sizeA)*y.deig.dIq(k))*phideriv.dEdp(:,n) );
%-----
y.d2eig.dId_dEqp(k,n) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dId_dEqp{k,n}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dId{k}-eye(sizeA)*y.deig.dId(k))*phideriv.dEqp(:,n) ...
+psi(m,:)*(a.dAsys.dEqp{n}-eye(sizeA)*y.deig.dEqp(n))*phideriv.dId(:,k) );
y.d2eig.dEqp_dId(n,k) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dEqp_dId{n,k}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dEqp{n}-eye(sizeA)*y.deig.dEqp(n))*phideriv.dId(:,k) ...
+psi(m,:)*(a.dAsys.dId{k}-eye(sizeA)*y.deig.dId(k))*phideriv.dEqp(:,n) );
%-----
y.d2eig.dIq_dEqp(k,n) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

*( psi(m,:)*a.d2Asys.dIq_dEq{k,n}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dIq{k}-eye(sizeA)*y.deig.dIq(k))*phideriv.dEq(:,n) ...
+psi(m,:)*(a.dAsys.dEq{n}-eye(sizeA)*y.deig.dEq(n))*phideriv.dIq(:,k) );
y.d2eig.dEq_dIq(n,k) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dEq_dIq{n,k}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dEq{n}-eye(sizeA)*y.deig.dEq(n))*phideriv.dIq(:,k) ...
+psi(m,:)*(a.dAsys.dIq{k}-eye(sizeA)*y.deig.dIq(k))*phideriv.dEq(:,n) );
%-----
end%for
end%if
for n=1:data.nbuses
%-----
if isempty(find(data.nPgen==n))==0
%-----
y.d2eig.dV_dId(k,n) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dV_dId{k,n}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dV{k}-eye(sizeA)*y.deig.dV(k))*phideriv.dId(:,n) ...
+psi(m,:)*(a.dAsys.dId{n}-eye(sizeA)*y.deig.dId(n))*phideriv.dV(:,k) );
y.d2eig.dId_dV(n,k) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dId_dV{n,k}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dV{k}-eye(sizeA)*y.deig.dV(k))*phideriv.dId(:,n) ...
+psi(m,:)*(a.dAsys.dId{n}-eye(sizeA)*y.deig.dId(n))*phideriv.dV(:,k) );
%-----
y.d2eig.dtheta_dId(k,n) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dtheta_dId{k,n}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dtheta{k}-eye(sizeA)*y.deig.dtheta(k))*phideriv.dId(:,n) ...
+psi(m,:)*(a.dAsys.dId{n}-eye(sizeA)*y.deig.dId(n))*phideriv.dtheta(:,k) );
y.d2eig.dId_dtheta(n,k) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dId_dtheta{n,k}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dtheta{k}-eye(sizeA)*y.deig.dtheta(k))*phideriv.dId(:,n) ...
+psi(m,:)*(a.dAsys.dId{n}-eye(sizeA)*y.deig.dId(n))*phideriv.dtheta(:,k) );
%-----
y.d2eig.dV_dIq(k,n) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dV_dIq{k,n}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dV{k}-eye(sizeA)*y.deig.dV(k))*phideriv.dIq(:,n) ...
+psi(m,:)*(a.dAsys.dIq{n}-eye(sizeA)*y.deig.dIq(n))*phideriv.dV(:,k) );
y.d2eig.dIq_dV(n,k) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dIq_dV{n,k}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dV{k}-eye(sizeA)*y.deig.dV(k))*phideriv.dIq(:,n) ...
+psi(m,:)*(a.dAsys.dIq{n}-eye(sizeA)*y.deig.dIq(n))*phideriv.dV(:,k) );
%-----
y.d2eig.dtheta_dIq(k,n) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dtheta_dIq{k,n}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dtheta{k}-eye(sizeA)*y.deig.dtheta(k))*phideriv.dIq(:,n) ...
+psi(m,:)*(a.dAsys.dIq{n}-eye(sizeA)*y.deig.dIq(n))*phideriv.dtheta(:,k) );
y.d2eig.dIq_dtheta(n,k) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dIq_dtheta{n,k}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dtheta{k}-eye(sizeA)*y.deig.dtheta(k))*phideriv.dIq(:,n) ...
+psi(m,:)*(a.dAsys.dIq{n}-eye(sizeA)*y.deig.dIq(n))*phideriv.dtheta(:,k) );
%-----
y.d2eig.dV_dEdp(k,n) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dV_dEdp{k,n}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dV{k}-eye(sizeA)*y.deig.dV(k))*phideriv.dEdp(:,n) ...
+psi(m,:)*(a.dAsys.dEdp{n}-eye(sizeA)*y.deig.dEdp(n))*phideriv.dV(:,k) );
y.d2eig.dEdp_dV(n,k) = ...

```


APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dEdp_dV{n,k}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dV{k}-eye(sizeA)*y.deig.dV(k))*phideriv.dEdp(:,n) ...
+psi(m,:)*(a.dAsys.dEdp{n}-eye(sizeA)*y.deig.dEdp(n))*phideriv.dV(:,k) );
%-----
y.d2eig.dtheta_dEdp(k,n) = ...
1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dtheta_dEdp{k,n}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dtheta{k}-eye(sizeA)*y.deig.dtheta(k))*phideriv.dEdp(:,n) ...
+psi(m,:)*(a.dAsys.dEdp{n}-eye(sizeA)*y.deig.dEdp(n))*phideriv.dtheta(:,k) );
y.d2eig.dEdp_dtheta(n,k) = ...
1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dEdp_dtheta{n,k}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dtheta{k}-eye(sizeA)*y.deig.dtheta(k))*phideriv.dEdp(:,n) ...
+psi(m,:)*(a.dAsys.dEdp{n}-eye(sizeA)*y.deig.dEdp(n))*phideriv.dtheta(:,k) );
%-----
y.d2eig.dV_dEq(k,n) = ...
1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dV_dEq{k,n}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dV{k}-eye(sizeA)*y.deig.dV(k))*phideriv.dEq(:,n) ...
+psi(m,:)*(a.dAsys.dEq{n}-eye(sizeA)*y.deig.dEq(n))*phideriv.dV(:,k) );
y.d2eig.dEq_dV(n,k) = ...
1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dEq_dV{n,k}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dV{k}-eye(sizeA)*y.deig.dV(k))*phideriv.dEq(:,n) ...
+psi(m,:)*(a.dAsys.dEq{n}-eye(sizeA)*y.deig.dEq(n))*phideriv.dV(:,k) );
%-----
y.d2eig.dtheta_dEq(k,n) = ...
1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dtheta_dEq{k,n}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dtheta{k}-eye(sizeA)*y.deig.dtheta(k))*phideriv.dEq(:,n) ...
+psi(m,:)*(a.dAsys.dEq{n}-eye(sizeA)*y.deig.dEq(n))*phideriv.dtheta(:,k) );
y.d2eig.dEq_dtheta(n,k) = ...
1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dEq_dtheta{n,k}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dtheta{k}-eye(sizeA)*y.deig.dtheta(k))*phideriv.dEq(:,n) ...
+psi(m,:)*(a.dAsys.dEq{n}-eye(sizeA)*y.deig.dEq(n))*phideriv.dtheta(:,k) );
%-----
y.d2eig.dV_ddelta(k,n) = ...
1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dV_ddelta{k,n}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dV{k}-eye(sizeA)*y.deig.dV(k))*phideriv.ddelta(:,n) ...
+psi(m,:)*(a.dAsys.ddelta{n}-eye(sizeA)*y.deig.ddelta(n))*phideriv.dV(:,k) );
y.d2eig.ddelta_dV(n,k) = ...
1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.ddelta_dV{n,k}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dV{k}-eye(sizeA)*y.deig.dV(k))*phideriv.ddelta(:,n) ...
+psi(m,:)*(a.dAsys.ddelta{n}-eye(sizeA)*y.deig.ddelta(n))*phideriv.dV(:,k) );
%-----
y.d2eig.dtheta_ddelta(k,n) = ...
1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dtheta_ddelta{k,n}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dtheta{k}-eye(sizeA)*y.deig.dtheta(k))*phideriv.ddelta(:,n) ...
+psi(m,:)*(a.dAsys.ddelta{n}-eye(sizeA)*y.deig.ddelta(n))*phideriv.dtheta(:,k) );
y.d2eig.ddelta_dtheta(n,k) = ...
1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.ddelta_dtheta{n,k}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dtheta{k}-eye(sizeA)*y.deig.dtheta(k))*phideriv.ddelta(:,n) ...
+psi(m,:)*(a.dAsys.ddelta{n}-eye(sizeA)*y.deig.ddelta(n))*phideriv.dtheta(:,k) );
%-----
end%if
%-----
y.d2eig.dV_dtheta(k,n) = ...
1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dV_dtheta{k,n}*eigen.phi(:,m) ...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

+psi(m,:)*(a.dAsys.dV{k}-eye(sizeA)*y.deig.dV(k))*phideriv.dtheta(:,n) ...
+psi(m,:)*(a.dAsys.dtheta{n}-eye(sizeA)*y.deig.dtheta(n))*phideriv.dV(:,k) );
y.d2eig.dtheta_dV(n,k) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dtheta_dV{n,k}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dtheta{n}-eye(sizeA)*y.deig.dtheta(n))*phideriv.dV(:,k) ...
+psi(m,:)*(a.dAsys.dV{k}-eye(sizeA)*y.deig.dV(k))*phideriv.dtheta(:,n) );
y.d2eig.dV_dV(k,n) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dV_dV{k,n}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dV{k}-eye(sizeA)*y.deig.dV(k))*phideriv.dV(:,n) ...
+psi(m,:)*(a.dAsys.dV{n}-eye(sizeA)*y.deig.dV(n))*phideriv.dV(:,k) );
y.d2eig.dtheta_dtheta(k,n) = ...
    1/(psi(m,:)*eigen.phi(:,m)) ...
*( psi(m,:)*a.d2Asys.dtheta_dtheta{k,n}*eigen.phi(:,m) ...
+psi(m,:)*(a.dAsys.dtheta{k}-eye(sizeA)*y.deig.dtheta(k))*phideriv.dtheta(:,n) ...
+psi(m,:)*(a.dAsys.dtheta{n}-eye(sizeA)*y.deig.dtheta(n))*phideriv.dtheta(:,k) );
end%for
end%for

```

H.24 update.m

```

function new = update(data,var,index);

current = getstate(data,index,var,0);
%-----
% Y variables
%-----
new.pre.V = var.allvariables(current.ind.V.ind);
new.pre.theta = var.allvariables(current.ind.theta.ind);
new.pre.Pg = var.allvariables(current.ind.Pg.ind);
new.pre.Pl = var.allvariables(current.ind.Pl.ind);
new.pre.Qg = var.allvariables(current.ind.Qg.ind);
new.pre.Ig = var.allvariables(current.ind.Ig.ind);
new.pre.gamma = var.allvariables(current.ind.gamma.ind);
new.pre.Adelt = var.allvariables(current.ind.Adelt.ind);
new.pre.delta = var.allvariables(current.ind.delta.ind);
new.pre.Id = var.allvariables(current.ind.Id.ind);
new.pre.Iq = var.allvariables(current.ind.Iq.ind);
new.pre.Vd = var.allvariables(current.ind.Vd.ind);
new.pre.Vq = var.allvariables(current.ind.Vq.ind);
new.pre.Edp = var.allvariables(current.ind.Edp.ind);
new.pre.Eqp = var.allvariables(current.ind.Eqp.ind);
new.pre.Efd = var.allvariables(current.ind.Efd.ind);
new.pre.Pij = var.allvariables(current.ind.Pij.ind);
new.pre.Qij = var.allvariables(current.ind.Qij.ind);
new.pre.Sij = var.allvariables(current.ind.Sij.ind);

Vp = new.pre.V.*exp(j*new.pre.theta);
new.pre.Iij = (Vp(current.data.fr)-Vp(current.data.to)).*current.data.Yvec+ ...
Vp(current.data.fr).*(j*current.data.Bvec/2);
%-----
% Equality constraint Lagrange multipliers
%-----
new.pre.eq.lagrange.P = ...
    var.allvariables(current.ind.eq.lagrange.ind(index.pre.eq.P.ind));
new.pre.eq.lagrange.Q = ...
    var.allvariables(current.ind.eq.lagrange.ind(index.pre.eq.Q.ind));
new.pre.eq.lagrange.Pij = ...
    var.allvariables(current.ind.eq.lagrange.ind(index.pre.eq.Pij.ind));
new.pre.eq.lagrange.Qij = ...
    var.allvariables(current.ind.eq.lagrange.ind(index.pre.eq.Qij.ind));
new.pre.eq.lagrange.Ig_real = ...
    var.allvariables(current.ind.eq.lagrange.ind(index.pre.eq.Ig_real.ind));
new.pre.eq.lagrange.Ig_imag = ...
    var.allvariables(current.ind.eq.lagrange.ind(index.pre.eq.Ig_imag.ind));
new.pre.eq.lagrange.delta_real = ...
    var.allvariables(current.ind.eq.lagrange.ind(index.pre.eq.delta_real.ind));
new.pre.eq.lagrange.delta_imag = ...
    var.allvariables(current.ind.eq.lagrange.ind(index.pre.eq.delta_imag.ind));
new.pre.eq.lagrange.Idq_real = ...
    var.allvariables(current.ind.eq.lagrange.ind(index.pre.eq.Idq_real.ind));
new.pre.eq.lagrange.Idq_imag = ...
    var.allvariables(current.ind.eq.lagrange.ind(index.pre.eq.Idq_imag.ind));
new.pre.eq.lagrange.Vdq_real = ...
    var.allvariables(current.ind.eq.lagrange.ind(index.pre.eq.Vdq_real.ind));
new.pre.eq.lagrange.Vdq_imag = ...
    var.allvariables(current.ind.eq.lagrange.ind(index.pre.eq.Vdq_imag.ind));
new.pre.eq.lagrange.Edp = ...
    var.allvariables(current.ind.eq.lagrange.ind(index.pre.eq.Edp.ind));
new.pre.eq.lagrange.Eqp = ...
    var.allvariables(current.ind.eq.lagrange.ind(index.pre.eq.Eqp.ind));
new.pre.eq.lagrange.Efd = ...
    var.allvariables(current.ind.eq.lagrange.ind(index.pre.eq.Efd.ind));
new.pre.eq.lagrange.Sij = ...

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```

        var.allvariables(current.ind.eq.lagrange.ind(index.pre.eq.Sij.ind));
new.pre.eq.lagrange.slack = ...
        var.allvariables(current.ind.eq.lagrange.ind(index.pre.eq.slack.ind));
new.eq.lagrange.total = var.allvariables(current.ind.eq.lagrange.ind);
%-----
% Inequality constraint Lagrange multipliers and slacks
%-----
%new.pre.ineq.lagrange = var.allvariables(current.ind.ineq.lagrange.ind);
%new.pre.ineq.slack = var.allvariables(current.ind.ineq.slack.ind);
%new.pre.ineq.Lagrange = diag(new.pre.ineq.lagrange);
%new.pre.ineq.Slack = diag(new.pre.ineq.slack);

for k=1:data.ncont
current = getstate(data,index,var,k);

%-----
% Y variables
%-----
new.post.V(:,k) = var.allvariables(current.ind.V.ind);
new.post.theta(:,k) = var.allvariables(current.ind.theta.ind);
new.post.Pg(:,k) = var.allvariables(current.ind.Pg.ind);
new.post.Pl(:,k) = var.allvariables(current.ind.Pl.ind);
new.post.Qg(:,k) = var.allvariables(current.ind.Qg.ind);
new.post.Pij(:,k) = var.allvariables(current.ind.Pij.ind);
new.post.Qij(:,k) = var.allvariables(current.ind.Qij.ind);
new.post.Sij(:,k) = var.allvariables(current.ind.Sij.ind);
new.post.Ig(:,k) = var.allvariables(current.ind.Ig.ind);
new.post.gamma(:,k) = var.allvariables(current.ind.gamma.ind);
new.post.Adelt(:,k) = var.allvariables(current.ind.Adelt.ind);
new.post.delta(:,k) = var.allvariables(current.ind.delta.ind);
new.post.Id(:,k) = var.allvariables(current.ind.Id.ind);
new.post.Iq(:,k) = var.allvariables(current.ind.Iq.ind);
new.post.Vd(:,k) = var.allvariables(current.ind.Vd.ind);
new.post.Vq(:,k) = var.allvariables(current.ind.Vq.ind);
new.post.Edp(:,k) = var.allvariables(current.ind.Edp.ind);
new.post.Eqp(:,k) = var.allvariables(current.ind.Eqp.ind);
new.post.Efd(:,k) = var.allvariables(current.ind.Efd.ind);
Vp = new.post.V(:,k).*exp(j*new.post.theta(:,k));
new.post.Iij(:,k) = (Vp(current.data.fr)-Vp(current.data.to)).*current.data.Yvec+ ...
Vp(current.data.fr).*(j*current.data.Bvec/2);
%-----
% Equality constraint Lagrange multipliers
%-----
new.post.eq.lagrange.P(:,k) = ...
        var.allvariables(current.ind.eq.lagrange.ind(index.post.eq.P.ind));
new.post.eq.lagrange.Q(:,k) = ...
        var.allvariables(current.ind.eq.lagrange.ind(index.post.eq.Q.ind));
new.post.eq.lagrange.Pij(:,k) = ...
        var.allvariables(current.ind.eq.lagrange.ind(index.post.eq.Pij.ind));
new.post.eq.lagrange.Qij(:,k) = ...
        var.allvariables(current.ind.eq.lagrange.ind(index.post.eq.Qij.ind));
new.post.eq.lagrange.Ig_real(:,k) = ...
        var.allvariables(current.ind.eq.lagrange.ind(index.post.eq.Ig_real.ind));
new.post.eq.lagrange.Ig_imag(:,k) = ...
        var.allvariables(current.ind.eq.lagrange.ind(index.post.eq.Ig_imag.ind));
new.post.eq.lagrange.delta_real(:,k) = ...
        var.allvariables(current.ind.eq.lagrange.ind(index.post.eq.delta_real.ind));
new.post.eq.lagrange.delta_imag(:,k) = ...
        var.allvariables(current.ind.eq.lagrange.ind(index.post.eq.delta_imag.ind));
new.post.eq.lagrange.Idq_real(:,k) = ...
        var.allvariables(current.ind.eq.lagrange.ind(index.post.eq.Idq_real.ind));
new.post.eq.lagrange.Idq_imag(:,k) = ...
        var.allvariables(current.ind.eq.lagrange.ind(index.post.eq.Idq_imag.ind));
new.post.eq.lagrange.Vdq_real(:,k) = ...
        var.allvariables(current.ind.eq.lagrange.ind(index.post.eq.Vdq_real.ind));

```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

```
new.post.eq.lagrange.Vdq_imag(:,k) = ...
    var.allvariables(current.ind.eq.lagrange.ind(index.post.eq.Vdq_imag.ind));
new.post.eq.lagrange.Edp(:,k) = ...
    var.allvariables(current.ind.eq.lagrange.ind(index.post.eq.Edp.ind));
new.post.eq.lagrange.Eqp(:,k) = ...
    var.allvariables(current.ind.eq.lagrange.ind(index.post.eq.Eqp.ind));
new.post.eq.lagrange.Efd(:,k) = ...
    var.allvariables(current.ind.eq.lagrange.ind(index.post.eq.Efd.ind));
new.post.eq.lagrange.Sij(:,k) = ...
    var.allvariables(current.ind.eq.lagrange.ind(index.post.eq.Sij.ind));
new.post.eq.lagrange.slack(:,k) = ...
    var.allvariables(current.ind.eq.lagrange.ind(index.post.eq.slack.ind));
new.eq.lagrange.total = [new.eq.lagrange.total; var.allvariables(index.post.eq.lagrange.ind(k,:))];
%-----
% Inequality constraint Lagrange multipliers and slacks
%-----
%new.post.ineq.lagrange = var.allvariables(current.ind.ineq.lagrange.ind);
%new.post.ineq.slack = var.allvariables(current.ind.ineq.slack.ind);
%new.post.ineq.Lagrange = diag(new.post.ineq.lagrange);
%new.post.ineq.Slack = diag(new.post.ineq.slack);
end%for

new.ineq.lagrange = var.allvariables(index.ineq.lagrange.ind);
new.ineq.slack = var.allvariables(index.ineq.slack.ind);
new.ineq.Lagrange = diag(new.ineq.lagrange);
new.ineq.Slack = diag(new.ineq.slack);

new.allvariables = var.allvariables;
```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

H.25 `sederiv.m`

```
function f=sederiv(Efd)
% First derivative of exciter saturation function.
f = 0.0039*1.55*exp(1.55*Efd);
```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

H.26 se2deriv.m

```
function f=se2deriv(Efd)
% Second derivative of exciter saturation function.
f = 0.0039*1.55^2*exp(1.55*Efd);
```

APPENDIX H. SMALL-SIGNAL STABILITY ESCOPF MATLAB CODE

H.27 se3deriv.m

```
function f=se2deriv(Efd)
```

```
% Second derivative of exciter saturation function.
```

```
f = 0.0039*1.553*exp(1.55*Efd);
```


VITA 2

John Condren

Candidate for the Degree of

Doctor of Philosophy

Thesis: EXPECTED-SECURITY-COST OPTIMAL POWER FLOW WITH
SMALL-SIGNAL STABILITY CONSTRAINTS

Major Field: Electrical Engineering

Education: Graduated from Roland High School, Roland, Oklahoma in May 1994; attended Westark Community College from 1994-1996; received Bachelor of Science degree in Electrical Engineering from Oklahoma State University, Stillwater, OK in May 1999. Completed the requirements for the Doctor of Philosophy degree with a major in Electrical Engineering at Oklahoma State University in August 2003.

Experience: Employed by Baldor Electric as a lab technician during summers of 1996-1998; Employed by Oklahoma State University as a research assistant from 1998-2000 and as a teaching assistant from 2000-2002.

Professional Memberships: Institute of Electrical and Electronics Engineers, Phi Kappa Phi honor society, Eta Kappa Nu electrical engineering honor society, Tau Beta Pi engineering honor society.