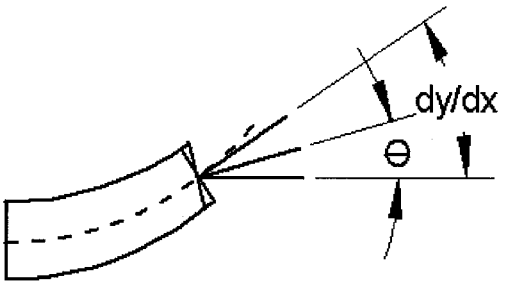
REVISED SIEVERS LATERAL DYNAMICS MODEL

by

R. L. Walton
Eastman Kodak Company
USA

NOMENCLATURE

A=cross-sectional area
E=Young's modulus
G=shear modulus
I=area moment of inertia
J=inertia per unit length
m=mass per unit length
n=shear factor
t=time
T=total tension
v=web longitudinal velocity
x=longitudinal position
y=lateral position
z=lateral position of a roller in a displacement guider frame (otherwise zero)
θ =face angle (centerline angle plus shear angle)
θ_r =roller misalignment angle (zero if perfectly aligned)



Dots above variables indicate time derivatives
Apostrophes following variables indicate spatial derivatives with respect to x

INTRODUCTION

In her thesis, Sievers[1] begins with fully-dynamic equations of motion for the potential and kinetic energy of moving beams. She applies Hamilton's principle, and these equations result:

$$-m(\ddot{y}+2v\dot{y}'+v^2y'')+(\frac{AG}{n}+T)y''-\frac{AG}{n}\theta'=0 \tag{1}$$

$$-J(\ddot{\theta}+2v\dot{\theta}'+v^2\theta'')+EI\theta''+\frac{AG}{n}(y'-\theta)=0 \tag{2}$$

By making the assumption of quasi-static behavior, the time derivatives drop out. The terms which include mass and inertia are negligibly small, and may be discarded. Manipulation of (1) and (2) gives (with the time dependence of the web shown):

$$\frac{\partial^4 y(x,t)}{\partial x^4}-k^2\frac{\partial^2 y(x,t)}{\partial x^2}=0 \tag{3}$$

$$\theta(x,t)=\frac{\partial y(x,t)}{\partial x}+f\frac{\partial^3 y(x,t)}{\partial x^3} \tag{4}$$

where:

$$k^2=\frac{T}{EI(1+\frac{nT}{AG})} \tag{5}$$

$$f=\frac{EIn}{AG}(1+\frac{nT}{AG}) \tag{6}$$

Note that web shear causes the web centerline angle to differ from the web face angle. It is the web face angle that is continuous across a roller, rather than the web centerline angle.

To describe the transient lateral behavior of a web treated as a tensioned Timoshenko beam in a web conveyance system, an equation is added to describe the web lateral behavior at the downstream roller interface:

$$\frac{\partial y(x,t)}{\partial t}=-v(\frac{\partial y(x,t)}{\partial x}-\theta_r)+\frac{\partial z}{\partial t} \tag{7}$$

Equation (7) introduces the assumption of no slippage on each roller.

---

[1] "Modeling and Control of Lateral Web Dynamics", Lisa Sievers, PhD thesis, Rensselaer Polytechnic Institute, 1987.

These above equations are insufficient to determine the transient behavior of the web. Sievers, following Shelton[2], goes on to differentiate (7) with respect to time, substituting (7) for the time part of $\frac{\partial^2 y(x,t)}{\partial t \partial x}$ which occurs, giving:

$$\frac{\partial^2 y(x,t)}{\partial t^2} = v^2 \frac{\partial^2 y(x,t)}{\partial x^2} + \frac{\partial^2 z}{\partial t^2} \tag{8}$$

VIOLATION OF THE NO-SLIP ASSUMPTION

Equations (3) (4) (5) (6) and (8) give rise to a set of ordinary differential equations (see Sievers thesis). Each downstream roller has two states associated with it, the web lateral position and the web lateral velocity. When solved, this set of equations gives rise to behavior that violates the assumption of no web slippage on the rollers. Specifically, a step in web lateral position on one roller will instantaneously result in a face angle change of the web at all the downstream rollers in a multi-roller set of web spans. In exploring the reason for this, it is apparent that (8) is inappropriate for use with a moving Timoshenko beam used to model web lateral behavior. This is because in the face of the instantaneous step change in upstream web lateral position, the web lateral velocity at the downstream roller remains zero in the first instant after the step (since it is a state). By (7), the centerline slope of the web at the downstream roller must remain zero. Since the web shear is non-zero, by (4), the face angle at the downstream roller must instantly become non-zero.

REVISION OF THE MODEL

It is proposed that equation (8) be replaced by the time derivative of (4):

$$\frac{\partial \theta(x,t)}{\partial t} = \frac{\partial^2 y(x,t)}{\partial t \partial x} + f \frac{\partial^4 y(x,t)}{\partial t \partial^3 x} \tag{9}$$

Appropriate manipulation and substitution of (7) and (3) results in:

$$\frac{\partial \theta(x,t)}{\partial t} = -v(1 + fk^2) \frac{\partial^2 y(x,t)}{\partial x^2} \tag{10}$$

This equation makes the web face angle at the downstream roller into a system state. Because it is a state, it cannot change its value instantly in the face of a finite disturbance, thereby preserving the no-slip assumption. Equations (3) (4) (5) (6) (7) and (10) then result in a set of ordinary differential equations (see Appendices A and B). Each downstream roller still has two states associated with it, the web lateral position and the web face angle. Now it is noted that when an upstream web lateral position is changed instantaneously, it is the web lateral velocity on the downstream roller which takes a step, rather than the face angle. This is appropriate, since the shear force applied to the web span causes the web centerline angle just upstream of the downstream roller to change (even though the face angle did not change), which will cause the web to instantaneously

[2] "Lateral Dynamics of a Moving Web", John Shelton, PdD thesis, Oklahoma State University, 1968.

begin moving on the downstream roller in the direction of the web lateral position change at the upstream roller.

VERIFICATION OF THE PROPOSED REVISION

An attempt to verify this model was undertaken by approximating the web with a finite-element membrane[3]. Since linear finite-element techniques are used, the geometric stiffening associated with web tension is not present. Thus, the finite-element technique used actually simulates a non-tensioned web. The estimated difference in results from this discrepancy is on the order of a few percent for typical wide webs at unit tension levels on the order of 1 pound/inch-width, based on Timoshenko tensioned beam theory. Another discrepancy in the model is the handling of the upstream constraints. The web should be free to contract or expand widthwise due to Poisson's ratio effects; in the model, it is fully constrained at the upstream roller. Although in principle this technique could be used with a multi-roller web span, for purposes of this validation, a single span was used, with an entrance and an exit roller. The entrance roller is the start of the finite-element mesh, and the exit roller is the end of the mesh. The left end of the mesh is constrained to the entrance roller, and the right end of the mesh is constrained to the exit roller. The mesh is rectangular, and all the elements are the same size. The solution technique (see Appendix C) involves moving the web by the length of one element. While the web is undergoing this motion, the displacements of the constrained nodes are moved by the amount each node displaces relative to a pure motion of strain-free web in the machine direction, in the direction at right angles to each roller. The stress-strain state of the web at the end of this motion is evaluated based on this set of constrained displacements. When this motion is complete, the strain occurring in the membrane is remembered, and another step is taken. This operation is repeated until the desired amount of time has passed. Estimates of web stress, moments and shear during web weave are developed. The resulting deflections, velocities, face angles, centerline angles, moments and shears may be compared with those of the revised Sievers model (see Figures 1 through 3). For typical wide-web cases, the results are within a few percent.

FUTURE WORK

One additional item is generated from the finite-element model: an estimate of distribution of web-to-roller traction forces required to support the required strain of the web (of course, both an upstream and a downstream roller must be present at each roller for which such forces are to be estimated, so a multi-roller implementation is required). These forces may be compared to the available traction. As future work, one could investigate the case in which the local traction is insufficient to support the required force. One may conclude that local slippage must occur. Such slippage could be implemented by assigning the node involved to the set of unconstrained nodes in the finite-element model, applying the traction force in the direction opposite to the slippage, and re-solving the finite-element problem. With iteration, the set of nodes which is slipping could be identified, and the behavior of the weaving web under slippage conditions could be estimated.

---

[3] "Introduction to Finite Element Analysis and Nastran Utilization", Lajos Imre Nagy, MacNeal Schwendler Corporation, 1985.

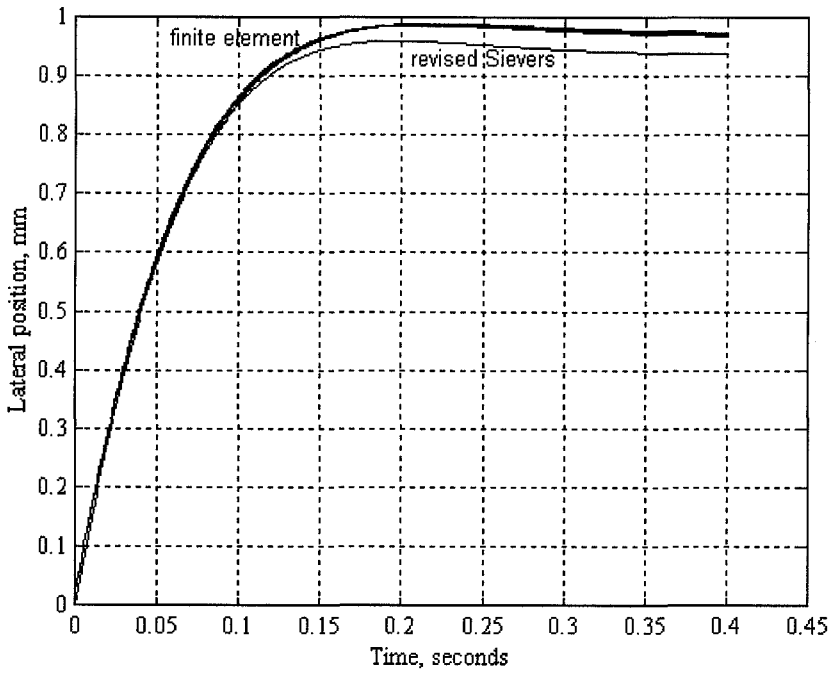


Figure 1: Lateral position of web at a misaligned downstream roller, revised Sievers model versus finite-element model.

Web width 1372 mm
Span length 1219 mm
Web thickness 0.178 mm
Web Young's modulus 4.688e9 N/m^2
Web Poisson's ratio 0.3
Web velocity 15.24 meters/second
Downstream roller misalignment 0.001 radian

The initial condition at time zero in both cases was an unstressed web starting at a straight upstream roller and ending at a misaligned downstream roller.

For the finite element model, the lateral position of each downstream node of the web is plotted. Since the lateral position at various widthwise positions in the web differs slightly, the displacements from the various nodes don't quite overlap, which causes the "fat line" effect noted above.

The finite element model achieved slightly more downstream lateral deflection than the revised Sievers model.

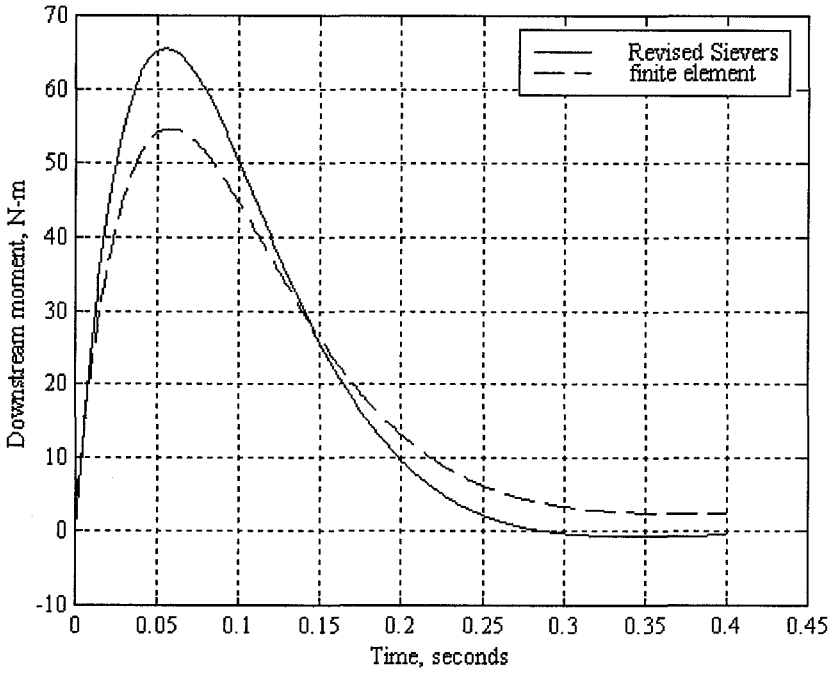


Figure 2: Downstream moment, revised Sievers model and finite element model.

One may note that the finite element model's moment did not quite approach zero asymptotically, like the revised Sievers model. The shapes are otherwise similar. The finite element model achieved somewhat less maximum downstream moment than the revised Sievers model.

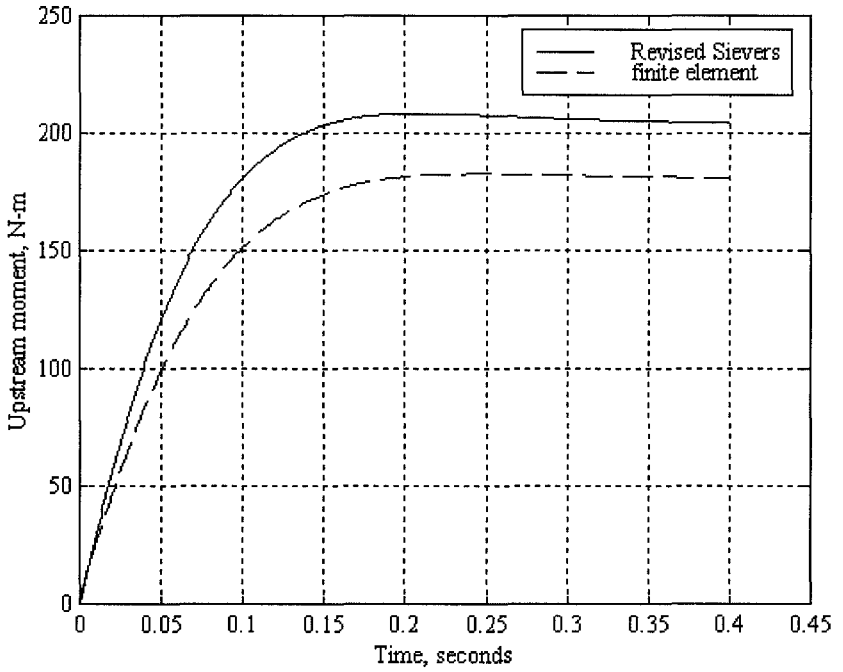


Figure 3: Upstream moment, revised Sievers model and finite element model.

The shapes of the plots are very similar, although the finite element model develops somewhat less upstream moment than the revised Sievers model.

APPENDIX A: Notes on derivation of a system model

The complete derivation is too lengthy to include here. The following Macsyma[4] batch file gives the derivation of a solvable set of ordinary differential equations for the revised Sievers model:

```
/*Sievers derivation using web-climbing equation
for dy/dt and an equation for the time derivative
of the face angle.
y=lateral position of web
x=longitudinal position of web (0 at upstream roller,
  lw at downstream roller)
lw=length of web span, upstream to downstream roller
kw="k" in d4y/dx4-k^2*d2y/dx2=0
y0=web lateral position at upstream roller
th0=web face angle at upstream roller
yl=web lateral position at downstream roller
thl=web face angle at downstream roller
fw="f" in th=dy/dx+f*d3y/dx3
th=web face angle
vw=web longitudinal velocity
z=downstream roller position*/
/*Start with a general solution of d4y/dx4-k^2*d2y/dx2=0*/
y:cc1*sinh(kw*x)+cc2*cosh(kw*x)+cc3*x+cc4;
/*Pick a set of basis solutions*/
y,x=0;
qq1:%=y0;
y,x=lw;
qq2:%=yl;
th:diff(y,x)+fw*diff(y,x,3);
th,x=0;
qq3:%=th0;
th,x=lw;
qq4:%=thl;
solve([qq1,qq2,qq3,qq4],[cc1,cc2,cc3,cc4]);
y,%;
y:trigsimp(%);
/*now derive expressions for the differential equations*/
/*For this derivation, let ddy=diff(y,t,2) and ssy=diff(y,x,2), etc.
Now, sx1... are the coefficients for the slope of y for each of
the boundary variables, ssx1 for the second derivative, etc:*/
sy:sx1*y0+sx2*th0+sx3*yl+sx4*thl;
ssy:ssx1*y0+ssx2*th0+ssx3*yl+ssx4*thl;
sssy:sssx1*y0+sssx2*th0+sssx3*yl+sssx4*thl;
/*right-angle rule equation at downstream roller:*/
q1:dy=-vw*(sy-thr)+dz;
/*downstream face angle from differentiating the defining equation for face angle:*/
q2:dth=-vw*(1+fw*kw^2)*ssy;
q3:rhs(q1);
q4:rhs(q2);
/*assign coefficients*/
w1:ratcoef(q3,y0);
w2:ratcoef(q3,th0);
w3:ratcoef(q3,yl);
w4:ratcoef(q3,thl);
w5:ratcoef(q3,thr);
w6:ratcoef(q3,dz);
```

[4] Macsyma version 2.2, Macsyma, Inc

```
z1:ratcoef(q4,y0);
z2:ratcoef(q4,th0);
z3:ratcoef(q4,yl);
z4:ratcoef(q4,thl);
sy:trigsimp(diff(y,x));
ssy:trigsimp(diff(y,x,2));
sssy:trigsimp(diff(y,x,3));
sy:sy,x=lw;
sy:trigsimp(sy);
ssy:ssy,x=lw;
ssy:trigsimp(ssy);
sssy:sssy,x=lw;
sssy:trigsimp(sssy);
sx1:trigsimp(ratcoef(sy,y0));
sx2:trigsimp(ratcoef(sy,th0));
sx3:trigsimp(ratcoef(sy,yl));
sx4:trigsimp(ratcoef(sy,thl));
ssx1:trigsimp(ratcoef(ssy,y0));
ssx2:trigsimp(ratcoef(ssy,th0));
ssx3:trigsimp(ratcoef(ssy,yl));
ssx4:trigsimp(ratcoef(ssy,thl));
sssx1:trigsimp(ratcoef(sssy,y0));
sssx2:trigsimp(ratcoef(sssy,th0));
sssx3:trigsimp(ratcoef(sssy,yl));
sssx4:trigsimp(ratcoef(sssy,thl));
/*substitute and simplify*/
w1:trigsimp(ev(w1));
w2:trigsimp(ev(w2));
w3:trigsimp(ev(w3));
w4:trigsimp(ev(w4));
w5:trigsimp(ev(w5));
w6:trigsimp(ev(w6));
z1:trigsimp(ev(z1));
z2:trigsimp(ev(z2));
z3:trigsimp(ev(z3));
z4:trigsimp(ev(z4));
optimize(['w1=w1,'w2=w2,'w3=w3,'w4=w4,'w5=w5,'w6=w6,'z1=z1,'z2=z2,'z3=z3,'z4=z4]);
```

Appendix B: An implementation of the revised Sievers model in an ACSL[5]-like language

```
program facesievers
!model of {$n=10} web spans (expressions inside {...} are preprocessor statements in Perl)
!web properties
constant ew=6.8e5 !Young's modulus, #/in
constant nuw=0.3 !Poisson's ratio
constant ww=54.0  !width, inches
constant xw=0.007 !thickness, inches
constant rhow=1.177e-4 !mass density, #-sec**2/inch**4
constant nw=0.7992  !Shear constant
constant vw=600.0 !velocity of web, in/sec
constant tw=40.0  !total tension of web, #
{dup 'constant lw1=\4\8.\0',$n} !{dup 'string',n} duplicates 'string' n times, incrementing unescaped integers
!in-plane misalignment angle of rollers, radians
{dup 'constant th1i=\0.\0',$n}
!out-of-plane misalignment angle of rollers, radians
{dup 'constant th1o=\0.\0',$n}
constant ay0t=0.0 !amplitude of incoming table weave, inches
constant ay0f=0.0 !amplitude of incoming sine weave, inches
constant ath0t=0.0 !amplitude of incoming table face angle, radians
constant ath0f=0.0 !amplitude of incoming sine face angle, radians
constant freq=0.1 !frequency of incoming sine weave, Hz
!incoming weave, inches, as a function of time, seconds
table y0t,1,4/0.0, 1.0, 1.0, 1000.0,&
        0.0, 0.0, 1.0, 1.0/
!incoming face angle, radians, as a function of time, seconds
table th0t,1,4/0.0, 1.0, 1.0, 1000.0,&
         0.0, 0.0, 0.01,0.01/
!initial web lateral positions
{dup 'constant y1ic=\0.\0',$n}
!roller wrap angles, radians
{dup 'constant th1w=\0.\0',$n}
!roller initial face angle rate of change, rad/sec
{dup 'constant dth1ic=\0.\0',$n}
!roller lateral accelerations, in/sec^2
constant z1ic=0.0 !initial lateral position for roller one
constant dz1ic=0.0 !initial lateral velocity for roller one
constant !lateral acceleration of roller 1, in/sec^2, as a function of time, seconds
table ddz1t,1,6/-1.0,0.0,0.01,0.03,0.04,100.0,&
           0.0,0.0,0.0, 0.0, 0.0 ,0.0/
constant z1f=1.0 !1.0 to make input lateral position follow along; zero otherwise
!roller lateral velocities, in/sec
{dup 'constant dz2=\0.\0',$n}
constant tstop=1.0 !stop time, seconds
cinterval cint=0.001
algorithm ialg=5
maxterval maxt=0.001
initial
pi=4.0*datan(1.0)
aw=ww*xw !area of web, in**2
iw=ww**3*xw/12.0 !area moment of inertia of web, in**4
gw=ew/2.0/(1.0+nuw) !shear modulus of web, #/in**2
jw=rhow*xw*ww**3/12.0 !inertia of 1-inch strip of web, #-sec**2
mw=rhow*ww*xw !mass of 1-inch strip of web, #*sec**2/in**2
!constant "k" in differential equation, 1/inch
kw=sqrt(tw/(ew*iw-jw*vw**2)/(1.0+nw*tw/aw/gw-mw*nw*vw**2/aw/gw))
```

[5] Advanced Continuous Simulation Language, Mitchell and Gauthier Associates

```
!factor on y''' in face angle equation:  th=y'+f*y''', inch**2
fw=(ew*iw-jw*vw**2)*nw/aw/gw*(1.0+nw*tw/aw/gw-nw*mw*vw**2/aw/gw)
f1=kw*(1.0+fw*kw**2) !convenient factor in equations, 1/inch
!w=coefficients for derivative of web lateral position, [yd thd yu thu]
!z=coefficients for face angle of web, [yd thd yu thu]
{dup 'call coef(wa1,wb1,wc1,wd1,za1,zb1,zc1,zd1=kw,lw1,fw,vw)',$n}
!cosine factors for roller wraps
{dup 'cw1=cos(th1w)',$n}
{dup 'sw1=sin(th1w)',$n}
end !of initial
derivative
!lateral web velocities, in/sec
{dup 'dy1=wa1*y0+wb1*th0d+wc1*y1+wd1*th1+vw*th1i+dz1',$n}
!lateral web positions, inches
{dup 'y1=integ(dy1,y1ic)',$n}
!face rotation rate of change, radian/sec, upstream side of each roller
{dup 'dth1=za1*y0+zb1*th0d+zc1*y1+zd1*th1',$n}
!face rotation, radians, upstream side of each roller
{dup 'th1=integ(dth1,dth1ic)',$n}
!face rotation, radians, downstream side of each roller
{dup 'th1d=th1*cw1+th1o*sw1',$n}
!incoming disturbances to upstream end of first web span
y0=z1f*z1+ay0t*y0t(t)+ay0f*sin(2*pi*freq*t)
th0d=ath0t*th0t(t)+ath0f*cos(2*pi*freq*t)
!roller 1 lateral acceleration, velocity, position
ddz1=ddz1t(t) !acceleration, in/sec^2
dz1=integ(ddz1,dz1ic) !velocity, in/sec^2
z1=integ(dz1,z1ic) !position, inches (used only to check for proper displacement)
termt(t.ge.tstop)
end !of derivative
end
subroutine coef(wa,wb,wc,wd,za,zb,zc,zd,kw,lw,fw,vw)
implicit double precision(a-z)
q1=kw**3
q2=kw*lw
q3=sinh(q2)
q4=cosh(q2)
q5=(fw*q1+kw)*lw*q3
q6=(1/(q5-2*q4+2))
q7=kw**2
q8=-fw*q7
q9=(1/((fw**2*kw**5+2*fw*q1+kw)*lw*q3+(-2*fw*q7-2)*q4+2*fw*q7+2))
q10=-q7
q11=kw**4
q12=-fw*q11
q13=(fw*q11+q7)*q4+q12+q10
q14=q12+q10
q15=kw*q3
wa=fw*q1*q3*q6*vw
wb=(fw*q7*q4+q8)*q9*vw
wc=-fw*q1*q3*q6*vw
wd=-(q5+(q8-2)*q4+fw*q7+2)*q9*vw
we=vw !we and wf not returned, since they don't depend
wf=1  !on the span number, and are so simple
za=-q13*q6*vw
zb=-(q15+q14*lw)*q6*vw
zc=q13*q6*vw
zd=(q15+q14*lw*q4)*q6*vw
return
end
```

Appendix C: Matlab[6] programs implementing finite-element verification

```
%do finite-element setup and computations for a single web span
%set up parameters: x=length direction, y=lateral direction
nx=61; %number of nodes in x
ny=16; %number of nodes in y
lw=48; %length of web between rollers, inches
ww=54; %width of web, inches
a=lw/(nx-1); %x spacing, inches
b=ww/(ny-1); %y spacing, inches
nu=.3; %Poisson's ratio
ew=6.8e5; %Young's modulus, #/in^2
th=.007; %thickness, inches
vw=600; %web velocity, in/sec
offset=0.0; %initial web offset on downstream roller
misalign=0.001; %downstream roller misalignment, radians
stoptime=.4; %time to stop run, seconds
%make unconstrained stiffness matrix, other stuff
[k,el,xx,yy]=buildk(nx,ny,a,b,nu,ew,th);
%identify points to constrain
constrain=find(xx<0.1); %node numbers
constrain=[constrain*2-1;constrain*2]; %convert to variable numbers
%constrained displacements
dispn=find(abs(xx-lw)<.1); %node  numbers
dispx=[dispn*2-1]; %convert to variable numbers
dispy=[dispn*2];
downstream=sort([dispx;dispy]);
%add these to constraints
constrain=sort([constrain;dispx;dispy]);
%form unconstrained subscripts
unconstrain=zeros(nx*ny*2,1);
unconstrain(constrain)=1;
unconstrain=unconstrain==0;
unconstrain=find(unconstrain);
%desired deflections at the "disp" points
u=zeros(nx*ny*2,1);
u(dispx)=0; %x-displacement
u(dispy)=offset+a*tan(misalign); %y-displacement
u=sparse(u);
%form force vector
f=zeros(nx*ny*2,1);
f=sparse(f);
%partition
kuu=k(unconstrain,unconstrain);
kuc=k(unconstrain,constrain);
kcu=k(constrain,unconstrain);
kcc=k(constrain,constrain);
disp('solving...')
u(unconstrain)=kuu\(f(unconstrain)-kuc*u(constrain));
disp('solved.')
%recover the forces of constraint
f(constrain)=kcu*u(unconstrain)+kcc*u(constrain);
%now package up the results so they can be plotted
x=zeros(nx,ny);
y=zeros(nx,ny);
ux=u(1:2:end);
uy=u(2:2:end);
maxc=max([max(abs(xx)) max(abs(yy))]); %maximum coordinate
```

[6] Matlab version 5.3, The Mathworks

```
maxd=max(abs(u)); %maximum deflection
ff=maxc/maxd/100; %deflection is 1% of plot
x(:)=xx+ff*ux;
y(:)=yy+ff*uy;
plot(x,y,'.')
title('Time=0')
drawnow
%now iterate on a moving web.  The same
%stiffness matrix is used, but the constrained
%displacement is stepped along.
time=0;
i=1;
upstream=find(abs(xx)<0.1)*2;
upstream=sort([upstream;upstream-1]);
outu=zeros(ny*2,1);
outf=zeros(ny*2,1);
outu(:,i)=u(downstream);
outf(:,i)=f(downstream);
outuu=zeros(ny*2,1);
outfu=zeros(ny*2,1);
outuu(:,i)=u(upstream);
outfu(:,i)=f(upstream);
outt=0;
downstreamm1=find(abs(xx-(lw-a))<.1)*2;
downstreamm1=sort([downstreamm1;downstreamm1-1]); %do x-displacements too
while time<stoptime
        i=i+1;
        u(downstream)=u(downstreamm1); %move the displacements back a node
        u(dispy)=u(dispy)+a*tan(misalign); %move web up the right amount for misalignment
        %solve again
        u(unconstrain)=kuu\(f(unconstrain)-kuc*u(constrain));
        %recover forces of constraint
        f(constrain)=kcu*u(unconstrain)+kcc*u(constrain);
        time=time+a/vw;
        outt(i)=time;
        outu(:,i)=u(downstream);
        outf(:,i)=f(downstream);
        outuu(:,i)=u(upstream); %recover upstream also
        outfu(:,i)=f(upstream);
        x=zeros(nx,ny);
        y=zeros(nx,ny);
        ux=u(1:2:end);
        uy=u(2:2:end);
        x(:)=xx+ff*ux;
        y(:)=yy+ff*uy;
        plot(x,y,'.')
        title(['Time=' num2str(time)])
        drawnow
end
%compute moments, shears
ymm=(0:b:ww)';
ymm=ymm-mean(ymm);
md=outf(1:2:end,:)'*ymm; %moments
mu=outfu(1:2:end,:)'*ymm;
sd=sum(outf(2:2:end,:)); %shears
su=sum(outfu(2:2:end,:));
twd=sum(outf(1:2:end,:)); %compute indicated web tensions
twu=sum(outfu(1:2:end,:));
```

```
function [k,el,xx,yy,ke]=buildk(nx,ny,a,b,nu,ew,th)
```

```
%[k,el,xx,yy,ke]=buildk(nx,ny,a,b,nu,ew,th) builds structures for
%a 2-D membrane model
%nx=number of nodes in x direction
%ny=number of nodes in y direction
%a=spacing of nodes in x direction, inches
%b=spacing of nodes in y direction, inches
%nu=Poisson's ratio
%ew=Young's modulus, #/in^2
%
%k=unconstrained stiffness matrix, 2*nx*ny x 2*nx*ny, sparse
%el=element node matrix (node number of each
%   corner of each element, arranged:  4   3
%                                      y
%                                      1 x 2
%xx=undeformed x location of each node, node number order
%yy=undeformed y location of each node, node number order
%ke=element stiffness matrix
%allocate memory for element array
ne=(nx-1)*(ny-1); %number of elements
el(ne).n1=0;
for i=1:ne
        r=fix((i-1)/(nx-1))+1; %element row
        c=i-(r-1)*(nx-1); %element column
        el(i).n1=(r-1)*nx+c; %element node 1
        el(i).n2=(r-1)*nx+c+1; %element node 2
        el(i).n3=r*nx+c+1; %element node 3
        el(i).n4=r*nx+c; %element node 4
end
%arrangement of displacement and force vectors
%(one x-y pair per node):
%[ux1;uy1;ux2;uy2;...]
%element stiffness matrix
ke=membraneelement(a,b,nu,ew,th);

%stiffness matrix
k=sparse(2*nx*ny,2*nx*ny);
disp('building stiffness matrix')
%add elements to stiffness matrix
for i=1:ne
        n1y=el(i).n1*2;
        n2y=el(i).n2*2;
        n3y=el(i).n3*2;
        n4y=el(i).n4*2;
        sub=[n1y-1 n1y n2y-1 n2y n3y-1 n3y n4y-1 n4y];
        k(sub,sub)=k(sub,sub)+ke;
end
x=[0:a:(nx-1)*a]';
y=[0:b:(ny-1)*b]';
[xx,yy]=meshgrid(x,y);
xx=xx';
yy=yy';
xx=xx(:);
yy=yy(:);
disp('stiffness matrix built')
```

```
function k=membraneelement(a,b,nu,ew,th)
%2-D rectangle membrane:
% e=membraneelement(a,b,nu,ew,th) where:
% a=distance between nodes 1 and 2, and 3 and 4
% b=distance between nodes 1 and 4, and 2 and 3
```

```
% nu=Poisson's ratio
% ew=Young's modulus
% th=thickness
%Node arrangement:  4  3
%
%                   1  2
%
%Equation: f=[f1x;f1y;f2x;f2y;f3x;f3y;f4x;f4y];
%          u=[u1x;u1y;u2x;u2y;u3x;u3y;u4x;u4y];
%          f=k*u
q1=(1/b);
q2=2*a*q1;
q3=(1/a);
q4=-2*a*q1*nu+4*q3*b+q2;
q5=((3*nu)/2)+1.5;
q6=-a*q1*nu-4*q3*b+a*q1;
q7=((9*nu)/2)-1.5;
q8=-2*q3*b;
q9=a*q1*nu+q8-a*q1;
q10=-((3*nu)/2)-1.5;
q11=-2*a*q1;
q12=2*q3*b;
q13=2*a*q1*nu+q12+q11;
q14=1.5-((9*nu)/2);
q15=-2*q3*b*nu+q12+4*a*q1;
q16=2*q3*b*nu+q8+q2;
q17=q3*b*nu-q3*b+q11;
q18=-q3*b*nu+q3*b-4*a*q1;
k=[q4,q5,q6,q7,q9,q10,q13,q14
q5,q15,q14,q16,q10,q17,q7,q18
q6,q14,q4,q10,q13,q7,q9,q5
q7,q16,q10,q15,q14,q18,q5,q17
q9,q10,q13,q14,q4,q5,q6,q7
q10,q17,q7,q18,q5,q15,q14,q16
q13,q7,q9,q5,q6,q14,q4,q10
q14,q18,q5,q17,q7,q16,q10,q15]*ew*th/12/(1-nu^2);
```

R. L. Walton
Revised Sievers Lateral Dynamics Model
6/9/99 Session 4 9:50 - 10:15 a.m.

Question - Ron Swanson, 3M
I had a question on the boundary conditions for the finite element. On the upstream roll, did you lock X and Y on all of them?

Answer – R. L. Walton, Kodak
That is correct.

Question – Ron Swanson, 3M
X and Y was locked on all of them? Downstream you forced them all to?

Answer – R. L. Walton
X and Y were all locked but then each time that I advanced the web by one mesh on the downstream side, I forced those points to follow the roller surface since, after all, they are adhered to the roller. And so, because it was misaligned, and they went up in Y by the amount of misalignment, and they went in X then by the web velocity. After that was done then I moved the attachment points to the rollers back one mesh and kept the displacement constraints for the forced X and Y displacements of the downstream constraint. Then I repeated that several hundred times. And that's really all it was. Incidentally, the full source core is included in the paper there so if there are questions about it, one can see exactly what I did.

Question - John Shelton, Oklahoma State University
First I have a question and then I would like to hold the mike for a comment. Lisa Sievers as I recall had a long wavelength plus and minus 0.165 inch sine wave input not a permanent deformation but a temporary sine wave input to a straight web. Her identification was R 3 the second roller on the displacement guide, the downstream roller on the displacement guide. The web guide took out all but three thousandths of that error. This was low frequency and a fairly high gain guide. Weave regeneration, between R 3 and the last roller on the web guide was observed, with 180 degrees of phase lag. Do you have any explanation for that larger phase lag? There would be some phase lag but 180 degrees is a little hard to conceive from the output of the web guide and the next roller.

Answer – R. L. Walton, Kodak
I guess I would have to not comment on that because I have not studied that particular problem with this model. I could do so but I have not applied this model to that particular case.

Comment –John Shelton, Oklahoma State University
Okay. But I do have some comments on my equations. The equation, I think in my thesis it's 4.14 but, that is not a derivative of 4.12 is not because the derivative didn't work the cross derivative was a term that wasn't verified by experimental data. The explanation was that all of my dynamic testing was with long spans and shear deflection was negligible. And in my equation, 4.14 which Sievers, I'm not sure whether she used it exactly or not, it neglects shear deflection but the next equation which I have not worked with much because of the complexity, 4.1.6, does consider shear in a kind of different

way, just trying to express what happens and account for shear deflection. All of my dynamic testing was with long enough spans that shear deflection was negligible.

Answer – R. L. Walton, Kodak
I believe in the Euler beam model the second derivative equation is correct. When you apply it to the Timoshenko beam model which includes angle shear deflection there is a problem. I think the problem comes from the fact that the web centerline angle is no longer continuous. There is a problem with taking the spatial derivative when the web centerline angle is discontinuous.