APPENDIX A

EES Programs for Transient Block Tests

```
"LUMPED CAPACITANCE MODEL FOR TWO THERMALLY CONNECTED
MASSES. "
"Constant heat loss parameter"
"Used to find R"_tc of sample, as well as loss parameters for each block"

"Block Information"
m_1 = 0.64  "kg"   ;                    m_2 = 0.64  "kg"
cp_al = 903 "J/kg-K" ;                  k_al = 170   "W/m-K"
rho_al = 2702     "kg/m^3";
L_al = 2 *convert(in,m)
"Biot # = Rcond / (Rconv = R_contact)"
Bi_1 =  L_al / (2*K_al * R``)
Bi_2 =  L_al / (2*K_al * R``)
Rcond = L_al / (2*K_al)
Rconv = 1/ (H_1*A_1)
Bi_ha = Rcond/Rconv
"Sample Info"
sample_name$ = 'polyester'
L_sample = .005*convert(in,m)
k_sample = 0.173     "W/m-K"
"Resistance Equations --- Used to minimize error below"
R``_sample = l_sample / k_sample
R`` = R``_sample +2* R``_tc
"Minimize error to find contact resistance"
{R``_tc=0.0006 }     "m^2-K/W"
R``_block = l_al/k_al
A_contact = pi * (3/2)^2 * convert(in^2,m^2)
T_surr = (70-32) * convert(F,C)+273.1
"-------------------------------------------------------------------------------"
"Find Constant Info"
"Set convection coefficients. Can also be found be minimizing error. "
{H_1 =6.68
 H_2 =2.89}
A_1 = (2*pi*3)*convert(in^2,m^2)+a_contact
A_2 = A_1
k1 = A_contact / (m_1 * cp_al * R``)
k2 =A_contact / (m_2 * cp_al  * R``)
k12 = H_1*A_1/ ( m_1*cp_al )
k22 = H_2*A_2 / ( m_2*cp_al )
"-------------------------------------------------------------------------------"
"Derivation found in chapter 2"
"Use analytical solution to block system to find constants "
"Quadratic Equation Constants - Solved Using Maple (Waterloo Software)"
a[1] = 1
a[2] = 1
```

```
b[1] = k22+k12+k1+k2
b[2] =  b[1]
c[1] = k12*(k2+k22) + k1*k22
{c[2] = k1*(k2+k22) + k12*(k2+k22) -k1^2}
c[2] = c[1]

"Find constants for general solution using quadratic formula"
"For Each Differential Equation"
Duplicate i=1,2
      m1_b[i] = (-b[i]+sqrt(b[i]^2- 4*a[i]*c[i])) / (2*a[i])
      m2_b[i] = (-b[i] - sqrt(b[i]^2- 4*a[i]*c[i])) / (2*a[i])
END
"-------------------------------------------------------------------------------"
"Prediction is compared with experimental data in lookup table"
"General Solution of Differential Equation"
"Two first order O.D.E's"
"Initiall Conditions = Initial temperatures "
Temp1_[0] = T_surr + M[1] + N[1]
Temp2_[0] = T_surr + M[2] + N[2]
"initial dTdt's from O.D.E."
m1_b[1]*M[1] + m2_b[1]*N[1] =-1/(m_1*cp_al)*((H_1*A_1*(Temp1_[0] - T_surr)) +
A_contact/R``*(Temp1_[0]-Temp2_[0]))
m1_b[2]*M[2] + m2_b[2]*N[2] =-1/(m_2*cp_al)*((H_2*A_2*(Temp2_[0] - T_surr)) +
A_contact/R``*(Temp2_[0]-Temp1_[0]))

“Set initial conditions”
Exp_time[0] = 0
T1_errmin = temp1_[0]
T2_errmin = temp2_[0]
"-------------------------------------------------------------------------------"
"Generate array values using analytical solution"
"n is counter variable"
m = 1 ; n =178
Duplicate t = m,n
      Exp_time[t] = lookup(t,#Time)
      "General solution with above constants"
      Temp1_[t]= T_surr + M[1]*exp(m1_b[1]*Exp_time[t]) +
      [1]*exp(m2_b[1]*Exp_time[t])
      Temp2_[t]= T_surr + M[2]*exp(m1_b[2]*Exp_time[t]) +
      N[2]*exp(m2_b[2]*Exp_time[t])

      "Compare Actual and predicted value to find error. All temperatures in K"
      T_meas_hot[t] = lookup(t,#T_hot)
      T_meas_cold[t] =lookup(t,#T_cold)
```

```
    "Minimize Sum of Error squared to find R``_tc"
    Step_error[t] =  sqrt((Temp1_[t] - T_meas_hot[t])^2 + (Temp2_[t] -
    T_meas_cold[t])^2)
END

Error = sum(Step_error[t],t=1,n)
"-------------------------------------------------------------------------------"
"Find Energy Loss during each step. Based On Experimental Data"
Duplicate t = m,n
    T_avg[t] = (T_meas_hot[t] + T_meas_cold[t]) / 2
END

Duplicate t = 2,n
    loss[t] = (m_1+m_2)*cp_al*(t_avg[t-1] - t_avg[t])/2   "Watts"
    loss[t] = h[t] *(A_1+A_2)* ((t_avg[t]+t_avg[t-1])/2 - T_surr)   "Watts"
    loss_predicted[t] = (h_1*a_1*(Temp1_[t] - T_surr) +h_2*a_2*(Temp2_[t] - T_surr))
        "Watts"
END

"Find Fraction of Energy Loss vs. Energy Transferred to Cold Block"
energy_cold = (m_2*cp_al*(T_meas_cold[n] - T_meas_cold[1]))          "J"
loss_fraction  = sum(loss[t],t=2,n) / energy_cold *100                "%"
"-------------------------------------------------------------------------------"
"Creat plot of dt/dt_init"
dt_init =  T_meas_hot[1] -  T_meas_cold[1]
duplicate n =1,100
ratio[n] = (Temp1_[n] -Temp2_[n])/dt_init
ratio_exp[n] = (T_meas_hot[n] -T_meas_cold[n])/dt_init
end
```

```
"ONE DIMENSIONAL TRANSIENT FINITE DIFFERENCE MODEL FOR BLOCK -
PLASTIC -BLOCK INTERFACE"
"Convection losses considered through thermal connection to distributed capacitance
insulation mass --- not constant heat sink"

"-----------------------------------------------------------------------"
"Control variables for counters"
n_al = 5                    "nodes in AL blocks"
n_pla = 2                   "nodes in plastic blocks"
T_inf = 295                 "K"
"-----------------------------------------------------------------------"
"Property Data"
k_al      = 170; k_plastic = .173            "W/m-K"
cp_al     = 903 ; cp_plastic = 1000          "J/mg-K"
rho_al    = 2702; rho_pla = 1000             "kg"
L_al      =  2*convert(in,m) / (n_al+1)      "n + .5 + .5 nodes per block"
L_pla     =  5e-3*convert(in,m) / (n_pla+1)      "5 mil,  + .5 + .5 nodes in plastic"
L_Rtc     = 1e-3*convert(in,m)               "1 mil , 1 node"
Perimeter_al = pi *3* convert(in,m)
"A_block for test"
m_block = l_al*rho_al*a_block
A_block = pi*(3/2)^2*convert(in^2,m^2)
m_al_node =    rho_al*A_block *L_al                "kg"
m_pla_node = rho_pla*A_block *L_pla                "kg"
m_Rtc_node = rho_Rtc*A_block * L_Rtc               "kg"
"-----------------------------------------------------------------------"
"Treat contact resistance as conduction node"
"Values taken from Dewitt example"
R_tc       = 0.0006          "m^2-K / W"
K_Rtc   =  L_Rtc / R_tc
rho_Rtc = 10           "kg / m^3                assigned low value"
cp_Rtc  = 80                "J/kg-K                  assigned low value"
"-----------------------------------------------------------------------"
"Insulation Parameters  --- Expanded Polystyrene values taken from Dewitt "
R_value       = 4.5 *convert (HR-FT^2- F/Btu, m^2-K/W)
L_Rvalue      = 1.5 * convert(in,m)
rho_ins        = 55                          "kg//m^3"
cp_ins         = 1210                        "J/kg-K"
R_value       = L_Rvalue / k_ins
"Radial conduction constants"
"-----------------------------------------------------------------------"
"Determine Constants for  loops"
Cap_al                = m_al_node *cp_al
Cap_pla               = m_pla_node *cp_plastic
Cap_Rtc               =m_Rtc_node *cp_Rtc
```

```
Con1_al                 = k_al / L_al*A_block
Con1_pla                = k_plastic / L_pla*A_block
Con1_Rtc                = K_Rtc/L_Rtc *A_block
Conv_al                 =l_al * Perimeter_al
"-----------------------------------------------------------------------"
"Critical Timesteps"
tau_critical = Cap_pla / (2*(con1_Rtc+con1_pla))
"-----------------------------------------------------------------------"
"Initial values are in the Parametric Table"
"Outer Loop is Time Progression"
Duplicate k=1,16+n_pla
T[k] =TableValue(1,#T_new[k])
END
"-----------------------------------------------------------------------"
"For insulation counter, first index is block number, second is radial node #"

"Adiabadic Initial node --- change for constant  temperature"
T_new[1]= T[1] + 2/cap_al*integral(-Con1_al *(T_new[1]-T_new[2]),Time)
a = 1 + n_al
Duplicate i = 2,a
          T_new[i]= T[i] + 1/cap_al*integral(-Con1_al * (T_new[i] - T_new[i+1])
         +Con1_al*(T_new[i-1]-T_new[i]) - q_block_ins[i-1],Time)
END
"Al block  ---     Rtc half (capacity & resistance) node"
b = a + 1
T_new[b] = T[b] + 2/cap_al*integral(-Con1_Rtc *(T_new[b]-T_new[b+1])+Con1_al
*(T_new[b-1]-T_new[b]),Time)
"Rtc half (capacity) node --   Plastic node "
c = b + 1
T_new[c] = T[c] + 2/cap_pla*Integral(-Con1_pla *(T_new[c]-T_new[c+1]) + Con1_Rtc
*(T_new[c-1]-T_new[c]),Time)
"Homogenous Plastic Slice  c+1 to c+1+n_pla"
d = c+n_pla
Duplicate i = c+1,d
          T_new[i]=T[i] +1/cap_pla*integral(Con1_pla * (T_new[i-1] + T_new[i+1] - 2 *
         T_new[i]),Time)
END

"Plastic       ---  Rtc half node "
e = d + 1
T_new[e] = T[e] + 2/cap_pla*integral(-Con1_Rtc *(T_new[e]-T_new[e+1])+Con1_pla
*(T_new[e-1]-T_new[e]),Time)
"Rtc half node  ---    Al block node"
ff = e + 1
```

```
T_new[ff] = T[ff] +2/cap_al*integral(-Con1_al *(T_new[ff]-T_new[ff+1])+Con1_Rtc
*(T_new[ff-1]- T_new[ff]),Time)

"Homogenous Al block"
g = ff + 1
Duplicate i = g,g+n_al  - 1
          T_new[i] = T[i] +1/cap_al*integral(Con1_al *(T_new[i-1] + T_new[i+1] - 2 *
          T_new[i])- q_block_ins[i-7],Time)
END
"diabadic node on End --- change for constant temperature"
h = g+n_al
T_new[h] = T[h]+2/cap_al*integral(Con1_al *(T_new[h-1]-T_new[h]) ,Time)
"-----------------------------------------------------------------------"
{"Calculate ratio for dt / dt init"
DELTA_T_ratio = (T_new[4] - T_new[15]) / (T[4]-T[15])}

"-----------------------------------------------------------------------"
"-----------------------------------------------------------------------"
"Insulation nodes 1 - D"
"First Set radius and capacities "
r[0] = 3/2 *convert(in,m)
Duplicate k = 1, 3
     r[k] = r[0] + k*.01 - .005
     cap[k] = cp_ins*rho_ins * l_al * pi * ((r[k] + .005)^2 - (r[k] - .005)^2)
END
"-----------------------------------------------------------------------"
"Initial values are in the Lookup Table"
"Outer Loop is Time Progression"
Duplicate i = 1,10
     T_ins[i,1] = lookup(i,#Node1)
     T_ins[i,2] = lookup(i,#Node2)
     T_ins[i,3] = lookup(i,#Node3)
END
"-----------------------------------------------------------------------"

"Outer loop is for Al blocks"
Duplicate j = 1,10
"Inner loop is for conduction through insulation"
     Duplicate k = 1,2
     T_new_ins[j,k] = T_ins[j,k] + (cond_radial[k]/cap[k]) * integral((T_new_ins[j,k-1] -
     T_new_ins[j,k])/ln(r[k]/r[k-1]) - (T_new_ins[j,k] -
     T_new_ins[j,k+1])/ln(r[k+1]/r[k]),Time)
     END
```

```
    T_new_ins[j,3] = T_ins[j,3] + 1/cap[3]*(cond_radial[3] * integral((T_new_ins[j,2] -
    T_new_ins[j,3])/ln(r[3]/r[2]),Time)+ conv_radial*integral((T_inf -
    T_new_ins[j,3]),Time))




    "Find Energy Taken from block"
    T_new_ins[j,0] = T_block[j,0]                          "Boundary condition"
    q_block_ins[j] =  (T_block[j,0] -T_new_ins[j,1]) / R_radial

END

DELTA_tau = tablevalue(tablerun#,#Time) - tablevalue(tablerun#-1,#Time)
R_radial =ln((r[1])/(r[0])) / cond_radial[0]
cond_radial[0] = 2 * pi * L_al *k_ins
cond_radial[1] = 2 * pi * (L_al+0.01) *k_ins
cond_radial[2] = 2 * pi * (L_al+0.02) *k_ins
cond_radial[3] = 2 * pi * (L_al+0.03) *k_ins

conv_radial =2*pi*(r[3]+0.02)*L_al*5          "Convection coefficient = 5 W/m^2-K. Not
sensitive at all"
"-----------------------------------------------------------------------"
"Find energy lost to insulation as fraction of energy transferred to cold block"
q_loss_insulation = sum(q_block_ins[i],i=1,10)
q_blok_block      = Con1_Rtc *(T_new[7]- T_new[8])
frac_loss           = q_loss_insulation / q_blok_block * 100


"Set Temperatures for T[j,0]'s"
T_block[1,0] = T_new[2]
T_block[2,0] = T_new[3]
T_block[3,0] = T_new[4]
T_block[4,0] = T_new[5]
T_block[5,0] = T_new[6]
T_block[6,0] = T_new[13]
T_block[7,0] = T_new[14]
T_block[8,0] = T_new[15]
T_block[9,0] = T_new[16]
T_block[10,0] = T_new[17]
```

```
"ONE DIMENSIONAL TRANSIENT FINITE DIFFERENCE MODEL FOR BLOCK -
PLASTIC -BLOCK INTERFACE"
"Convection losses considered through constant heat sink"

"-----------------------------------------------------------------------"
"Control variables for counters"
n_al = 5                    "nodes in AL blocks"
n_pla = 2                   "nodes in plastic blocks"
T_inf = 295                 "K"
"-----------------------------------------------------------------------"
"Propety Data"
k_al = 170; k_plastic = 0.173          "W/m-K"
cp_al = 903 ; cp_plastic = 1000        "J/mg-K"
rho_al = 2702; rho_pla = 1000          "kg"
L_al =  2*convert(in,m) / (n_al+1)    "n + .5 + .5 nodes per block"
L_pla =  5e-3*convert(in,m) / (n_pla+1)  "5 mil,  + .5 + .5 nodes in plastic"
L_Rtc = 1e-3*convert(in,m)  "1 mil , 1 node"
Perimeter_al = pi *3* convert(in,m)
"A_block for test"
m_block = l_al*rho_al*a_block
A_block = pi*(3/2)^2*convert(in^2,m^2)
m_al_node =    rho_al*A_block *L_al              "kg"
m_pla_node = rho_pla*A_block *L_pla              "kg"
m_Rtc_node = rho_Rtc*A_block * L_Rtc             "kg"
"-----------------------------------------------------------------------"
"Treat contact resistance as conduction node"
"Values taken from Dewitt example"
R_tc = 0.0006          "m^2-K / W"
K_Rtc =  L_Rtc / R_tc
rho_Rtc = 10           "kg / m^3                assigned low value"
cp_Rtc = 80            "J/kg-K                  assigned low value"
"-----------------------------------------------------------------------"
"Insulation Parameters  "
R_value = 4.5 *convert (HR-FT^2- F/Btu, m^2-K/W)
L_Rvalue = 1.5 * convert(in,m)
rho_ins = 55                          "kg//m^3"
cp_ins = 1210                         "J/kg-K"
R_value = L_Rvalue / k_ins
"Radial conduction constants"
"-----------------------------------------------------------------------"
"Determine Constants for  loops"
Cap_al                          = m_al_node *cp_al
Cap_pla                         = m_pla_node *cp_plastic
```

```
Cap_Rtc                    =m_Rtc_node *cp_Rtc
Con1_al                    = k_al / L_al*A_block
Con1_pla                   = k_plastic / L_pla*A_block
Con1_Rtc                   = K_Rtc/L_Rtc *A_block
Conv_al                    =l_al * Perimeter_al
"-----------------------------------------------------------------------"

"Initial values are in the Parametric Table"
"Outer Loop is Time Progression"
Duplicate k=1,16+n_pla
     T[k] =TableValue(1,#T_new[k])
END
"-----------------------------------------------------------------------"
"For insulation counter, first index is block number, second is radial node #"
"Adiabadic Initial node --- change for constant  temperature"
T_new[1]= T[1] + 2/cap_al*integral(-Con1_al *(T_new[1]-T_new[2]),Time)
a = 1 + n_al
Duplicate i = 2,a
          T_new[i]= T[i] + 1/cap_al*integral(-Con1_al * (T_new[i] - T_new[i+1])
         +Con1_al*(T_new[i-1]-T_new[i]) +h_top*conv_radial*(T_inf-T_new[i]),Time)
END
"Al block  ---     Rtc half (capacity & resistance) node"
b = a + 1
T_new[b] = T[b] + 2/cap_al*integral(-Con1_Rtc *(T_new[b]-T_new[b+1])+Con1_al
*(T_new[b-1]-T_new[b]),Time)
"Rtc half (capacity) node --   Plastic node "
c = b + 1
T_new[c] = T[c] + 2/cap_pla*Integral(-Con1_pla *(T_new[c]-T_new[c+1])+Con1_Rtc
*(T_new[c-1]-T_new[c]),Time)

"Homogenous Plastic Slice  c+1 to c+1+n_pla"
d = c+n_pla
Duplicate i = c+1,d
          T_new[i]=T[i] +1/cap_pla*integral(Con1_pla * (T_new[i-1] + T_new[i+1] - 2 *
         T_new[i]),Time)
END

"Plastic       ---  Rtc half node "
e = d + 1
T_new[e] = T[e] + 2/cap_pla*integral(-Con1_Rtc *(T_new[e]-T_new[e+1])+Con1_pla
*(T_new[e-1]-T_new[e]),Time)
"Rtc half node  ---    Al block node"
ff = e + 1
T_new[ff] = T[ff] +2/cap_al*integral(-Con1_al *(T_new[ff]-T_new[ff+1])+Con1_Rtc
*(T_new[ff-1]-T_new[ff]),Time)
```

```
"Homogenous Al block"
g = ff + 1


Duplicate i = g,g+n_al  - 1
      T_new[i] = T[i] +1/cap_al*integral(Con1_al *(T_new[i-1] + T_new[i+1] - 2 *
      T_new[i])+h_bottom*conv_radial*(T_inf-T_new[i]),Time)
END
"Adiabatic node on End --- change for constant temperature"
h = g+n_al
T_new[h] = T[h]+2/cap_al*integral(Con1_al *(T_new[h-1]-T_new[h]) ,Time)

"-----------------------------------------------------------------------"
"-----------------------------------------------------------------------"
"Convection coefficient set above"
conv_radial =2*pi*1.5*L_al*convert(in,m)*6/5          "6/5 to convert for half node
depth"
h_top = 7.7
h_bottom = 3.7
"Find Losses in Watts"
"Find Old T_avg"
T_avg_old = sum(tablevalue(tablerun#-1,#T_new[i]),i=1,18)/18
"Find New T_avg"
T_avg = sum(T_new[i],i=1,18)/18
q_loss_insulation = -cp_al*2*0.64*(T_avg - T_avg_old)/DELTA_tau
DELTA_tau = tablevalue(tablerun#,#Time) - tablevalue(tablerun#-1,#Time)
```

APPENDIX B

Roller Web Finite Difference Model

```
Procedure property_set(sample$:k_plastic, cp_plastic, rho_plastic)

"!To add additional plastics, they must be added here, as well in
list pick box in the diagram window setup.
List pick box names must match sample names exactly.
Plastic Property Data"
"rho = kg/m^3                  k =  W/m-K                 cp =  J/kg-K"
rho_pester := 1200 ;           k_pester := 0.173;         cp_pester := 1200;
rho_embossed_pe := 920;        k_embossed_pe := 0.33;     cp_embossed_pe := 2300
rho_pp := 905;                 k_pp := 0.24;              cp_pp := 1930;
rho_unc := 459;                k_unc := 0.127;            cp_unc:= 1450;
rho_cal := 658;                k_cal := 0.169;            cp_cal:= 1450;
rho_pvc:=1350;                 k_pvc:=0.15;               cp_pvc:=1350
"Set plastic Properties"
IF (sample$ = 'Polyester') THEN
k_plastic := k_pester
cp_plastic :=cp_pester
rho_plastic:=rho_pester
ENDIF
IF (sample$= 'Polyethylene') THEN
k_plastic := k_embossed_pe
cp_plastic :=cp_embossed_pe
rho_plastic:=rho_embossed_pe
ENDIF
IF (sample$ = 'Polypropylene') THEN
k_plastic := k_pp
cp_plastic :=cp_pp
rho_plastic:=rho_pp
ENDIF
IF (sample$ = 'Uncalendered') THEN
k_plastic := k_unc
cp_plastic :=cp_unc
rho_plastic:=rho_unc
ENDIF
IF (sample$ = 'Calendered') THEN
k_plastic := k_cal
cp_plastic :=cp_cal
rho_plastic:=rho_cal
ENDIF
IF (sample$ = 'PVC') THEN
k_plastic := k_pvc
cp_plastic :=cp_pvc
```

```
rho_plastic:=rho_pvc
ENDIF
END

“!F.D. model of web passing over surface heated roller.
Can model internally heated rollers as well"

"-------------------------------------------------------------------------------------------------------------
--------------------------------"
"v1.1 Variable contact resistance added"
"v1.2 Need to add circumfrential conduction--- neglected"
"v1.3 Crank Nicholson formulation on web nodes"
"v1.4 Surface heat flux considered --- internal convection dropped i.e. switched to
Thermalon."
" v1.5 Crank Nicholson formulation on TiO_2 nodes"
"v1.6 Convection coefficient  curve fit added"
"-------------------------------------------------------------------------------------------------------------
--------------------------------"
"Basic design variables"
dia_roll=dia_roll_inches*convert(in,m)                "m Outside radius of steel roller"
thick_steel=0.0206                                    "m Thickness of steel roller"
thick_tio2=0.0017                                     "m Thickness of Titanium Dioxide"
thick_web=web_thickness*convert(in,m)                 "m Thickness of web"
"!To include internal heating term must be included in steel F.D. equations"
"!Presently locked out"
h_i=1000                                              "Inner fluid to steel HTC"
T_i=400                                               "Heated fluid temperature"
Q` = flux_surface*width
flux_win = flux_surface*convert(W/m^2,W/in^2)
"-------------------------------------------------------------------------------------------------------------
--------------------------------"
"!Finite difference approximations"
n=72                                      "Number of divisions around cylinder"
m=Round(wrap_angle / 5)                   "Number of cylinder divisions that are
wrapped"
n_w=2                                     "Number of full slices in web material - note
that there are 1/2 slices top and bottom."
n_r=2                                      "Number of full slices in roller"
n_s=2                                     "Number of full slices in steel"

"Set element width, length really."
width=dia_roll*pi/n
"Time step DELTA"
width = web_speed * DELTA
"Back out rpm info"
```

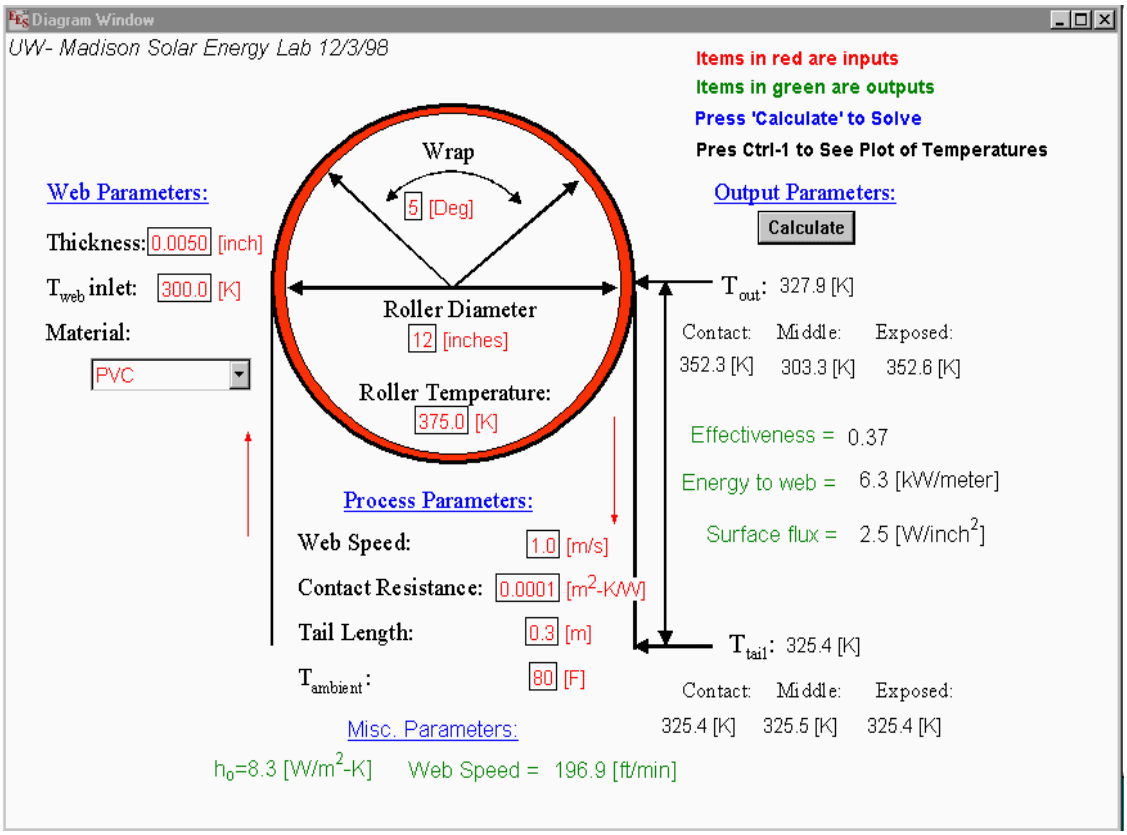
```
DELTA=60/(RPM*n)

"!Determine Convection coefficients for rotating cylinder"
T_o=(T_ambient - 32) *5/9 + 273.1                "K Surroundings temperature"
"Correlation based on average of rotating cylinder and flat plate. Web_speed in m/s, h in
w/m^2-K"
h_o= 1.752541 + 7.689549*web_speed-1.234470*web_speed^2 +
0.1087706*web_speed^3
"!Fix Roll properties and thermal masses"
"rho = kg/m^3                    k =  W/m-K            cp =  J/kg-K"
"TiO_2 roller Material"
rho_r=3000;                      k_r=10;               cp_r=1000
"Steel"
rho_s=7800;                      k_s=40;               cp_s=500

"Web Material"
"Sample data is set with a procedure. See start of program for info. "
call property_set(sample$:k_w,cp_w,rho_w) ;

"Contact Resistance Info"
con_R``_tc = 1 / R``_tc * width

"Element thermal masses"
cap_steel=Width*cp_s*thick_steel*rho_s/n_s
cap_web=width*cp_w*thick_web*rho_w/n_w
cap_tio2=width*cp_r*thick_tio2*rho_r/n_r

"Conductances"
con_steel=k_s*width*n_s/thick_steel
con_web=k_w*width*n_w/thick_web
con_tio2=k_r*width*n_r/thick_tio2

"!Boundary conditions for system"

"Incoming Temperatures are set according to T_web_in."
"Web incoming"
Duplicate i = 1, n_w + 1
T[i,1] =T_web_in
END

"Roll Incoming Temperatures= Roll outgoing temperatures for steady periodic state."
Duplicate i = n_w + 2,8
T[i,1] =  T[i,n+1]
END
```

```
"!Use F. D. Energy balances on web and roller nodes to calculate thermal history"

"Web section --- Use Crank Nicholson formulation"
Duplicate j=1, n
    "Outside of web"
    cap_web*(T[1,j+1]-T[1,j])=2*DELTA*(1/2*h_o*width*(2*T_o-T[1,j]-
T[1,j+1])+1/2*con_web*(T[2,j]-T[1,j]+T[2,j+1]-T[1,j+1]))
    "Web internal sections"
    Duplicate i=2,n_w
        cap_web*(T[i,j+1]-T[i,j])=1/2*DELTA*con_web*(T[i-1,j]+T[i+1,j]-
2*T[i,j]+T[i-1,j+1]+T[i+1,j+1]-2*T[i,j+1])
    end
T[9,j] = (T[1,j]+2*T[2,j]+T[3,j])/4
end

"Web-Roller interface - wrapped sections------ Use Crank Nicholson formulation"
{m=36}
duplicate j=1,m
    duplicate i=3, 3
        cap_web*(T[i,j+1]-T[i,j])=2*DELTA*( 1/2*con_web*(T[i-1,j]-T[i,j]+T[i-
1,j+1]-T[i,j+1])+1/2*con_R``_tc*(T[i+1,j]-T[i,j] + T[i+1,j+1]-T[i,j+1]))
        cap_tio2*(T[i+1,j+1] -T[i+1,j])=2*DELTA*( Q`+1/2*con_R``_tc*(+T[i,j]-
T[i+1,j]+T[i,j+1]-T[i+1,j+1])+1/2*con_tio2*(T[5,j]-T[4,j]+T[5,j+1]-T[4,j+1]))
    end
end


"Web-TiO_2 interface - unwrapped sections ------ Use Crank Nicholson formulation"
duplicate j=m+1,n
    duplicate i=3,3
        cap_web*(T[i,j+1]-T[i,j])=2*DELTA*(1/2*con_web*(T[i-1,j]-T[i,j]+T[i-1,j+1]-
T[i,j+1]) +h_o*width*(T_o-T[i,j]))
        cap_tio2*(T[i+1,j+1]-T[i+1,j])=2*DELTA*(Q`+con_tio2*(T[i+2,j]-
T[i+1,j])+h_o*width*(T_o-T[i+1,j]))
    end
end

"TIO2 section"
Duplicate j=1, n
    "Roller internal sections"
    Duplicate i=n_w+3,n_w+n_r+1
        cap_tio2*(T[i,j+1]-T[i,j])=DELTA*con_tio2*(T[i-1,j]+T[i+1,j]-2*T[i,j])
    end
```

```
end

"TIO2-Steel interface"
duplicate j=1,n
duplicate i=n_w+n_r+2, n_w+n_r+2
    (cap_tio2+cap_steel)*(T[i,j+1]-T[i,j])=2*DELTA*(con_tio2*(T[i-1,j]-
T[i,j])+con_steel*(T[i+1,j]-T[i,j]))
end
end

"Steel section"
Duplicate j=1, n
    "Steel internal sections"
    Duplicate i=n_w+n_r+3, n_w+n_r+n_s+1
        cap_steel*(T[i,j+1]-T[i,j])=DELTA*con_steel*(T[i-1,j]+T[i+1,j]-2*T[i,j])
    end
end

"Steel-hot fluid interface"
duplicate j=1,n
    duplicate i=n_w+n_r+n_s+2, n_w+n_r+n_s+2
        cap_steel*(T[i,j+1]-T[i,j])=2*DELTA*(con_steel*(T[i-1,j]-
        T[i,j]){+h_i*width*(T_i-T[i,j])})
    end
end

"!Use differential spacing to find cooling effect after leaving the roll"
contact_length              = m/n * pi*dia_roll
tail_length_corrected       =  tail_length - contact_length*(n-m)/n        "m"
DELTA_tail                   = tail_length_corrected / (web_speed*o)       "sec"
width_tail                  = tail_length_corrected / o                    "m"
o                           = 20                        "Number of nodes in tail"

"Find new web constants"
cap_web_tail=width_tail*cp_w*thick_web*rho_w/n_w
con_web_tail=k_w*width_tail*n_w/thick_web

"Write F.D. equations for tail section. Use C-N formulation"
duplicate j = n+1,n+1+o
    cap_web_tail*(T[1,j+1]-T[1,j])=2*DELTA_tail*(1/2*h_o*width_tail*(2*T_o-
    T[1,j]-T[1,j+1])+1/2 *con_web_tail*(T[2,j]-T[1,j]+T[2,j+1] - T[1,j+1]))
    cap_web_tail*(T[2,j+1]-T[2,j])=DELTA_tail*con_web_tail*1/2*(T[1,j]+T[3,j]-
    2*T[2,j] +T[1,j+1]+T[3,j+1]-2*T[2,j+1] )
    cap_web_tail*(T[3,j+1]-T[3,j])=2*DELTA_tail*(1/2*h_o*width_tail*(2*T_o-
    T[3,j]-T[3,j+1])+1/2 *con_web_tail*(T[2,j]-T[3,j]+T[2,j+1] - T[3,j+1]))
```

```
    T[9,j] = (T[1,j]+2*T[2,j]+T[3,j])/4
END


"Find Tail exit temperature"
T[9,n+o+2] = (T[1,n+o+2]+2*T[2,n+o+2]+T[3,n+o+2])/4
T_tail = T[9,n+o+2]
T_tail1 = T[1,n+o+2]
T_tail2 = T[2,n+o+2]
T_tail3 = T[3,n+o+2]
"!The following varables are in the diagram window.
and can be changed from there"
T[4,54] = T_roller                                          "K"

"Determine Power Draw of Thermalon"
Power_draw = flux_surface*pi*dia_roll/1000                  "kW"
Power_web = rho_w*cp_w*thick_web*web_speed*(T[9,73] - T[9,1])/1000 "kW"
T_web_out = T[9,m+1]
T_out1 = T[1,m+1]
T_out2 = T[2,m+1]
T_out3 = T[3,m+1]

web_speed_english = web_speed*convert(m/s,ft/min)

"!Try to predict epsilon"
epsilon = ( T_web_out - T_web_in) / (T_roller - T_web_in)


"!Make Plot for Reference"
"Flip Array variables"
Duplicate i = 1, n
Angle[i] = i*5
T_contact[i]      = T[3,i]
T_middle[i]       = T[2,i]
T_exposed[i]      = T[1,i]
T_surface[i]      = T[4,i]
END
```

The EES diagram window was used as the main import source for the above program as was as follows.



APPENDIX C

Experimental Equipment

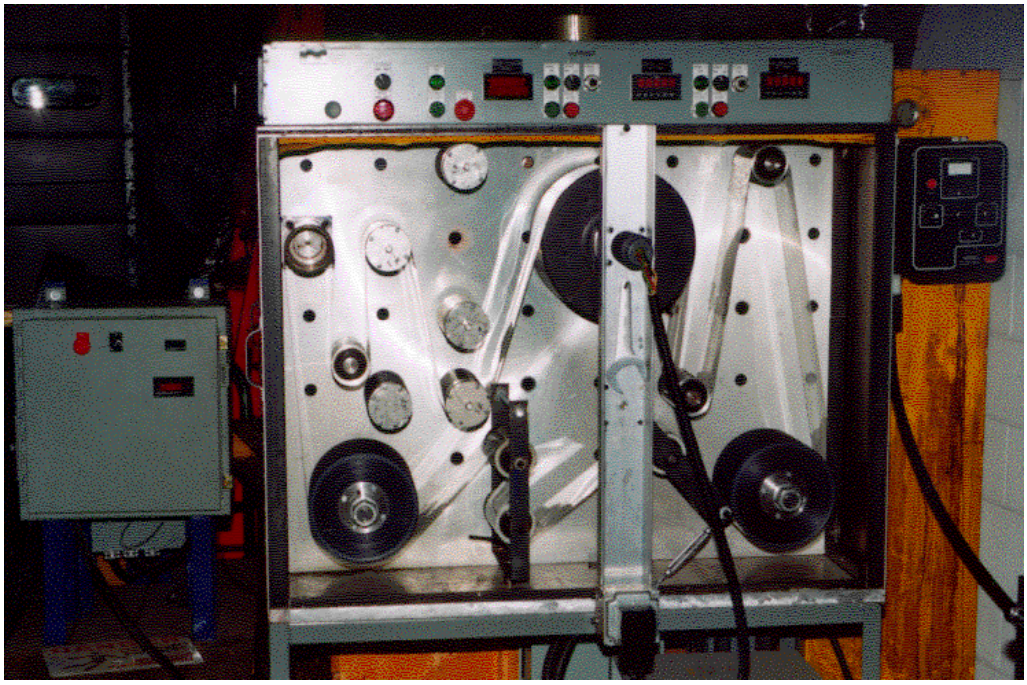


Figure C.1: Thermalon / Anti-Static Test Stand Used in Dynamic Tests at the American Roller Research and Development Facility, Union Grove, WI.

APPENDIX D

Pressure-Tension Relations

In roller terminology, the loading of a web is usually given in pounds per linear inch, or pli. Unfortunately, this can mean either of two things. For two rollers in contact in a nip, pli is the total normal force between the rollers divided by the length of the contact zone.

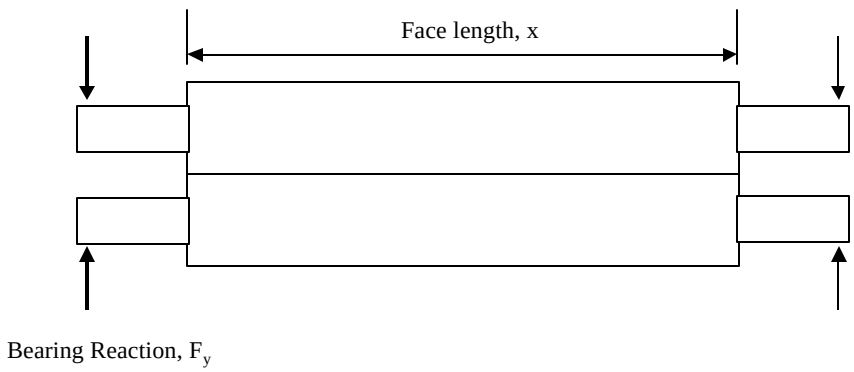


Figure D.1 Loading forces on a nipped roller

The loading would then be considered:

$$pli = \frac{2 \cdot F_y}{x} \tag{D.1}$$

For non covered rolls, the interface pressure can be determined from the Herz equation.

PLI can also be used to describe the tension on the web. In this study, the interface pressure is strictly a function of web tension and roller diameter. D.2 shows a web passing over the roller:

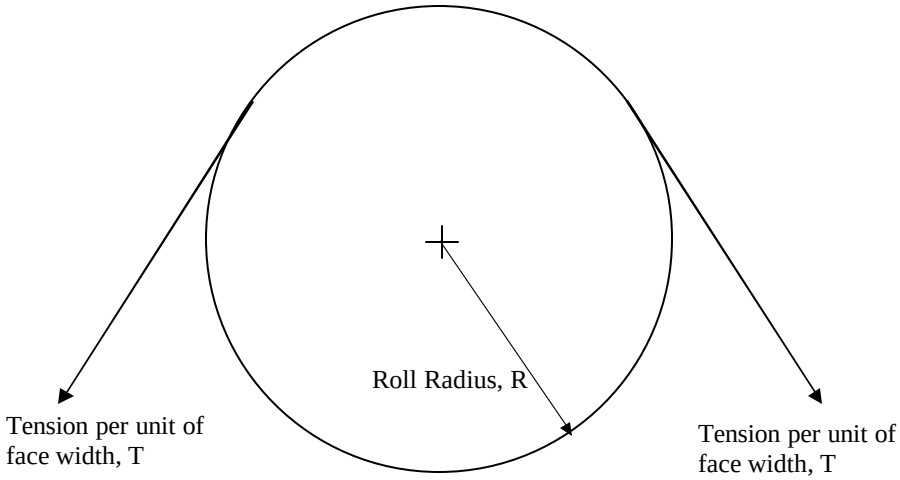


Figure D.2: Roller – Web Schematic

PLI here relates to the tension on the web per unit of face width. With the assumption of no slip at the interface, the pressure between the web and the roller can be found from:

$$P = \frac{T}{R} \tag{D.2}$$