**Wilson-Theta Problem**

*CEE 526 Finite Elements*

*Michael Justice*

Contents

[The Problem at Hand 2](#_Toc449385980)

[A *Very Brief* Analysis 2](#_Toc449385981)

[The Code 4](#_Toc449385982)

[Figure 1 - Problem Diagram and Loading Function 2](#_Toc449385817)

[Figure 2 - Acceleration Over Time 3](#_Toc449385818)

[Figure 3 - Velocity Over Time 3](#_Toc449385819)

[Figure 4 - Displacement Over Time 3](#_Toc449385820)

[Table 1- Max and Min TDP's for DOFY 2](#_Toc449385824)

[Table 2 - Max and Min TDP's for DOFR 2](#_Toc449385825)

# The Problem at Hand

The Wilson-Theta technique was used in this problem to determine the acceleration, velocity and displacement at the tip of a cantilever beam. A point load was placed at the tip under a timed forcing function.

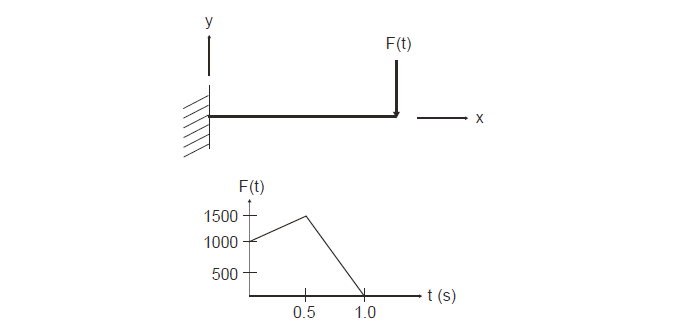


Figure - Problem Diagram and Loading Function

The cantilever beam was made of steel with the following properties:

A 1-element model was used to approximate the time dependent parameters. From the above diagram, it is worth noting that the tip of the beam only displaces in the y-direction. The forcing function was essentially a piecewise function over a 1-second domain. Below are two tables (Table 1 and Table 2) that contain the max and minimum values of the time dependent parameters (TDP’s) of the system.

(Note that displacement, velocity and acceleration values are zero in the x-direction)

# A *Very Brief* Analysis

Table - Max and Min TDP's for DOFY

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| DOF Y | | | | | | |
|  |  |  |  |  |  |  |
| 0.5 | 0 | 2.7298 | 0 | 2.7297 | 6.9775 | 92.667 |
| 0.4 | 0 | 1.7471 | 0 | 2.1837 | 3.2915 | 92.667 |
| 0.3 | 0 | 0.9827 | 0 | 1.6377 | 0.7324 | 92.666 |
| 0.2 | 0 | 0.4368 | 0 | 1.0917 | 0.0363 | 92.664 |

Table - Max and Min TDP's for DOFR

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| DOF ROT | | | | | | |
|  |  |  |  |  |  |  |
| 0.5 | 0 | 7.5068 | 0 | 7.5068 | 19.188 | 254.837 |
| 0.4 | 0 | 4.8044 | 0 | 6.0054 | 9.0515 | 254.837 |
| 0.3 | 0 | 2.7025 | 0 | 4.5040 | 2.0141 | 254.836 |
| 0.2 | 0 | 1.2011 | 0 | 3.0026 | 0.0997 | 254.835 |

From interpolating the data above, it can be seen that the values are generally reaching a converging state. It is a safe assumption that a continual decrease in the time step will produce a better converging trend. Below are plots of the acceleration, velocity and displacement of the beam using a time step of

|  |  |
| --- | --- |
|  | As can be seen from the plots immediately to the right (Figure 2), the acceleration is excited for the first second due to the timed forcing function, then begins to die out after the end of the forcing function. The damped nature of the system after a “long time” () is apparent, and was expected. The energy in the system eventually dissipates into the reaction at the end of the beam. It is worth noting that the rotational DOF has a larger amplitude than the y-direction DOF. The beam is essentially rotating faster than it is displacing in the vertical direction. Note that the beam is not moving in the x-direction. |
|  | Figure 2 - Acceleration Over Time |
|  |  |
| Figure 3 - Velocity Over Time | Figure 4 - Displacement Over Time |

The two plots above (Figure 3 and 4) display the same findings as in Figure 1, but it is worth noting the initial condition put in place for this problem. The initial velocity and displacement were set to zero, which is evident in the above figures. Again, natural damping of the system occurs approximately at the 6 second mark.

# The Code

|  |
| --- |
| % MICHAEL JUSTICE (C) %  % WILSON-THETA METHOD FOR PROBLEM 1 %    % material constants  E = 200e9; % elastic modulus [Pa]  I = 10e-4; % moment of inertia [m^4]  L = 2.0; % length of beam [m]  A = 0.01; % constant x/s area of beam [m^2]  rho = 7850; % density [kg/m^3]    % time dependent parameters  delt = 0.2; % time step  theta = 1.4; % wilson theta value > 1.37  t0 = 0.0; % initial time (can't be changed!)  tf = 10.0; % final time    % create storage arrays for a,v,x,t  size = floor((tf-t0)/(theta\*delt));  aStore = zeros(size,3);  vStore = zeros(size,3);  xStore = zeros(size,3);  tStore = zeros(size,1);    % allocate t into storage array  for i=2:size  tStore(i,1) = tStore(i-1,1) + delt\*theta;  end    % initialize time dependent variables  a0 = zeros(3,1); % acceleration at time zero  v0 = zeros(3,1); % velocity at time zero  x0 = zeros(3,1); % displacement at time zero  F0 = zeros(3,1); % forcing function at time zero    an = zeros(3,1); % acceleration at time n  vn = zeros(3,1); % velocity at time n  xn = zeros(3,1); % displacement at time n  Fn = zeros(3,1); % forcing function at time n    an1 = zeros(3,1); % acceleration at time n+1  vn1 = zeros(3,1); % velocity at time n+1  xn1 = zeros(3,1); % displacement at time n+1    atheta = zeros(3,1); % displacement at time theta  Ftheta = zeros(3,1); % forcing function at time theta    % initialize wilson recurrence values  a0wr = 6.0/(theta\*delt)^2;  a1wr = 3.0/(theta\*delt);  a2wr = 2.0\*a1wr;  a3wr = theta\*delt/a2wr;  a4wr = a0wr/theta;  a5wr = -a2wr/theta;  a6wr = 1.0-3.0/theta;  a7wr = delt/2.0;  a8wr = delt^2/6.0;    % input stiffness and mass matrix here  % initialize K and M  K = zeros(3,3);  M = zeros(3,3);    K = [E\*A/L 0 0 ; 0 12\*E\*I/L^3 -6\*E\*I/L^2 ; 0 -6\*E\*I/L^2 4\*E\*I/L];  M = rho\*A\*L/420\*[140 0 0 ; 0 156 -22\*L ; 0 -22\*L 4\*L^2];    % calculate initial acceleration (assume zero initial disp. & velocity)  F0 = [0; 1000; 0];  a0 = M\F0;    % allocate initial a,v,x into storage arrays  for i=1:3  aStore(1,i) = a0(i);  vStore(1,i) = v0(i);  xStore(1,i) = x0(i);  end    % form effective stiffness matrix  K = K + a0wr\*M;    % initialize t for storage scheme  t = 2;    % begin loop over prescribed time domain  for nsteps = t0:theta\*delt:tf    % check for initial values if time t = 0  if nsteps==0  Fn = F0;  an = a0;  vn = v0;  xn = x0;  end    % check over time domain for proper forcing function  if nsteps<=0.5  Ft = 1000 + nsteps\*1000;  else  Ft = 3000 - nsteps\*3000;  if Ft<0.0  Ft = 0.0;  end  end    % form effective load vector  Ftheta = [0; Ft; 0]; % update forcing function  F = Fn + theta\*(Ftheta - Fn) + M\*(a0wr\*x0 + a2wr\*v0 + 2.0\*a0);    % solving for displacements at current time step  atheta = K\F;    % solve for acceleration, velocity & displacement at next time step  an1 = a4wr\*(atheta-xn) + a5wr\*vn + a6wr\*an;  vn1 = vn + a7wr\*(an1 + an);  xn1 = xn + delt\*vn + a8wr\*(an1 + 2.0\*an);    % update time dependent variables for next time step  Fn = Ftheta;  an = an1;  vn = vn1;  xn = xn1;    % allocate next a,v,x into storage arrays  for i=1:3  aStore(t,i) = an(i);  vStore(t,i) = vn(i);  xStore(t,i) = xn(i);  end  % increment t for storage scheme  if t==size;  break;  else  t = t + 1;  end  end    % plot results  acceleration = 1; % =1 if you want acceleration  velocity = 1; % =1 if you want velocity  displacement = 1; % =1 if you want displacement  report = 1; % =1 if you want to print report to CW    if (acceleration == 1)  figa=figure(1); clf; grid on; axis square; hold on;  xlabel('t'); ylabel('a'); title('ACCELERATION OVER TIME');  for i=1:3  p = plot(tStore(:,1),aStore(:,i));  if (i==1)  hold on; set(p,'Color','green','LineWidth',0.5);  elseif (i==2)  hold on; set(p,'Color','blue','LineWidth',0.5);  else  hold on; set(p,'Color','red','LineWidth',0.5);  end  end  legend('X DOF [m/s^2]','Y DOF [m/s^2]','ROT DOF [rad/s^2]')  end    if (velocity == 1)  figv=figure(2); clf; grid on; axis square; hold on;  xlabel('t'); ylabel('v'); title('VELOCITY OVER TIME');  for i=1:3  p = plot(tStore(:,1),vStore(:,i));  if (i==1)  hold on; set(p,'Color','green','LineWidth',0.5);  elseif (i==2)  hold on; set(p,'Color','blue','LineWidth',0.5);  else  hold on; set(p,'Color','red','LineWidth',0.5);  end  end  legend('X DOF [m/s]','Y DOF [m/s]','ROT DOF [rad/s]')  end    if (displacement == 1)  figx=figure(3); clf; grid on; axis square; hold on;  xlabel('t'); ylabel('x'); title('DISPLACEMENT OVER TIME');  for i=1:3  p = plot(tStore(:,1),xStore(:,i));  if (i==1)  hold on; set(p,'Color','green','LineWidth',0.5);  elseif (i==2)  hold on; set(p,'Color','blue','LineWidth',0.5);  else  hold on; set(p,'Color','red','LineWidth',0.5);  end  end  legend('X DOF [m]','Y DOF [m]','ROT DOF [rad]')  end    % report  if (report==1)  % display important values, data, parameters  xStore = abs(xStore);  vStore = abs(vStore);  aStore = abs(aStore);    display(delt);  aminDOFX = min(aStore(:,1))  amaxDOFX = max(aStore(:,1))  vminDOFX = min(vStore(:,1))  vmaxDOFX = max(vStore(:,1))  xminDOFX = min(xStore(:,1))  xmaxDOFX = max(xStore(:,1))    aminDOFY = min(aStore(:,2))  amaxDOFY = max(aStore(:,2))  vminDOFY = min(vStore(:,2))  vmaxDOFY = max(vStore(:,2))  xminDOFY = min(xStore(:,2))  xmaxDOFY = max(xStore(:,2))    aminDOFR = min(aStore(:,3))  amaxDOFR = max(aStore(:,3))  vminDOFR = min(vStore(:,3))  vmaxDOFR = max(vStore(:,3))  xminDOFR = min(xStore(:,3))  xmaxDOFR = max(xStore(:,3))  end |