ME 7120: Project 3

Finite Element Method Applications

Due: December 9, 2016

Team Members:

Joel Summerfield

Michelle Callaham

Sainag Immidisetty

# Project Introduction

The purpose of this project was for us to apply the Newmark beta method to an existing problem in our book. For the L-beam structure shown in Figure 1, we were to apply a horizontal force of 100,000N to node 51 for 0.01 seconds. Prior to the load, the structure was at rest.

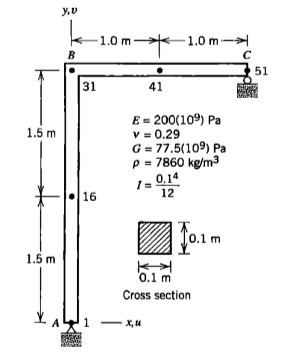


Figure 1: L-Beam Structure for Analysis

Using our MATLAB code in WFEM, we are tasked to compare all five of the Newmark beta methods. We also compared the responses for the displacement, velocity, and acceleration.

# L-Beam Element Analysis

We performed comparisons for the L-beam using all five of the Newmark beta methods from Table 1 below. We ran our code twice for each method to represent direct integration when ξ=0 and then with a damping coefficient of ξ=0.02.

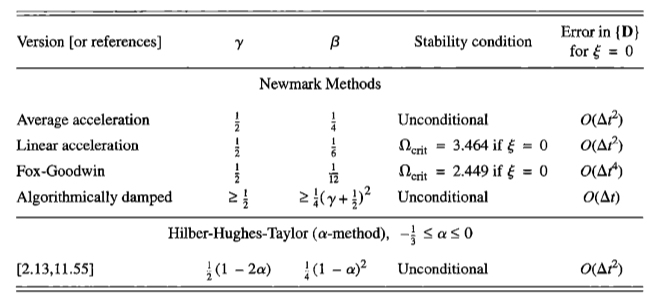


Table 1: Newmark Beta Method Table

Below are the plots for each of the different methods: average acceleration, linear acceleration, Fox-Goodwin, algorithmically damped, and Hilber-Hughes-Taylor method. Figures 2 through 11 show the displacement of the element.

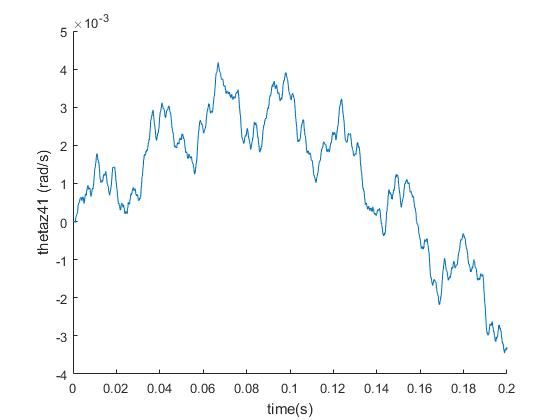


Figure 2: Average Acceleration Method, Displacement without Damping

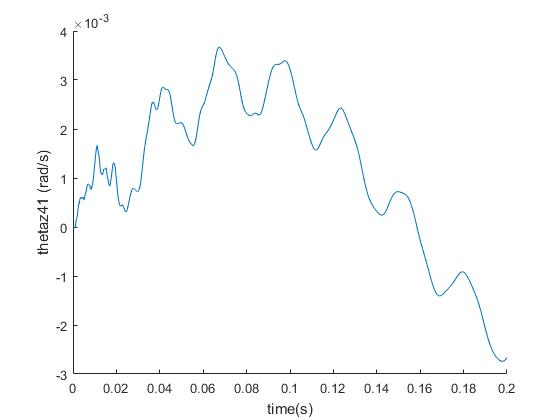


Figure 3: Average Acceleration Method, Displacement with Damping

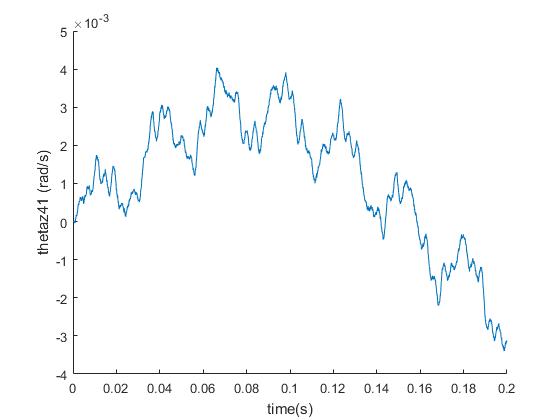


Figure 4: Linear Acceleration Method, Displacement without Damping

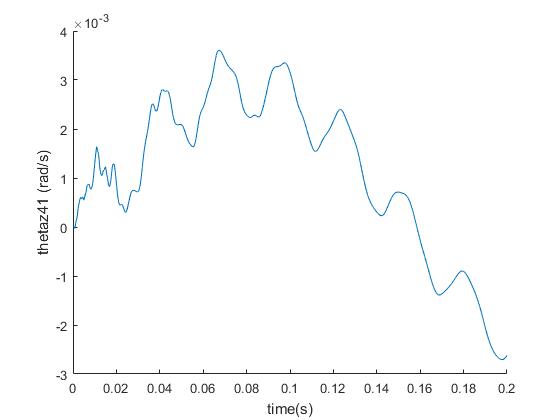


Figure 5: Linear Acceleration Method, Displacement with Damping

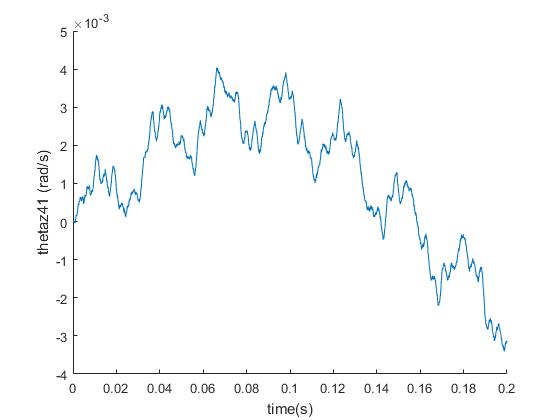


Figure 6: Fox-Goodwin Method, Displacement without Damping

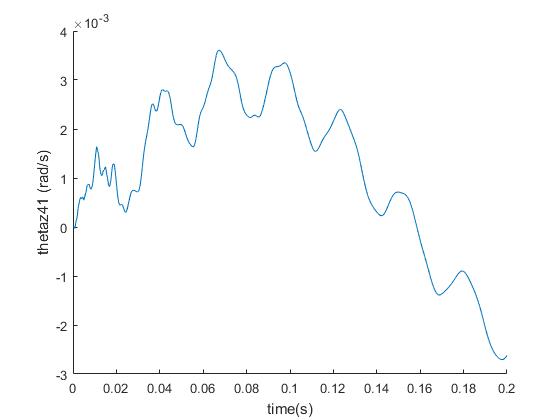


Figure 7: Fox-Goodwin Method, Displacement with Damping

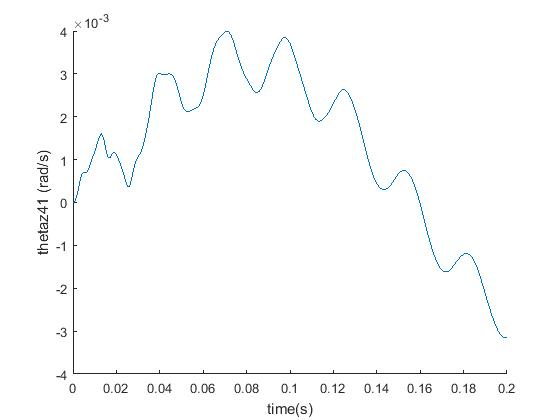


Figure 8: Algorithmically Damped Method, Displacement without Damping

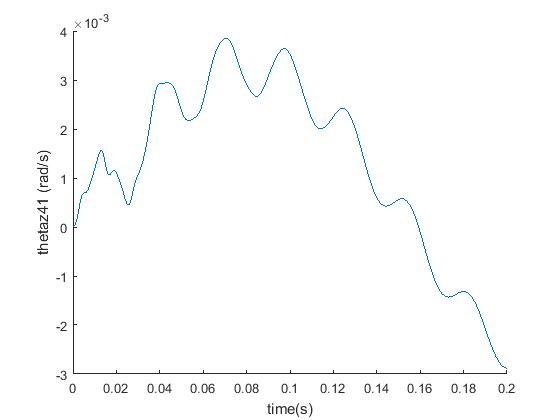


Figure 9: Algorithmically Damped Method, Displacement with Damping

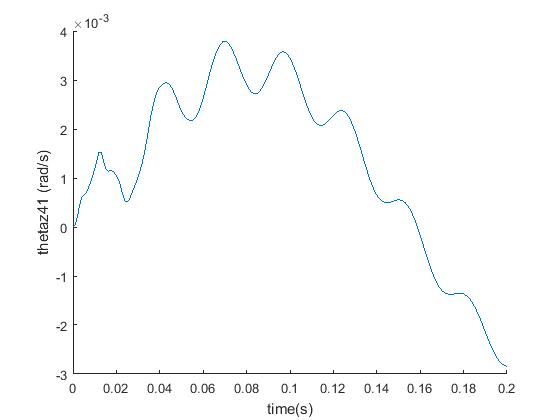


Figure 10: Hilber-Hughes-Taylor Method, Displacement without Damping

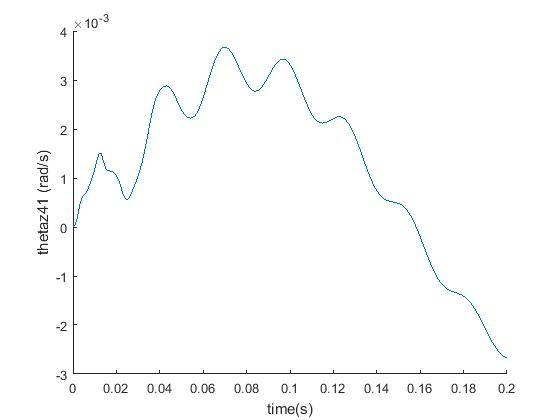


Figure 11: Hilber-Hughes-Taylor Method, Displacement with Damping

For each of the different methods, the damped figure has less peaks or does not have as much of a dramatic change between the peaks than when there was no damping. The tables below show the different Newmark method with their designated gamma and beta values. The maximum displacement, velocity, and acceleration for node 51 were found through our code. We also included what time steps and length of time used. Table 2 shows the results when there is no damping and Table 3 shows the results when there is 2% damping.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***0 Damping*** |  |  |  |  |  |  |  |
| **Method** | **Gamma** | **Beta** | **Max Disp.** | **Max Vel.** | **Max Acc.** | **Time Step** | **Time Range** |
| Average Acceleration | 0.5 | 0.25 | 0.0042 | 1.726 | 82312 | 0.0001 | 0 to 0.2 sec |
| Linear Acceleration | 0.5 | 0.1667 | 0.004 | 2.3235 | 129130 | 0.000001 | 0 to 0.2 sec |
| Fox - Goodwin | 0.5 | 0.0833 | 0.004 | 2.2972 | 143840 | 0.000001 | 0 to 0.2 sec |
| Algorithmically Damped | 0.6 | 0.4 | 0.004 | 0.3443 | 338.45 | 0.001 | 0 to 0.2 sec |
| Hilber-Hughes-Taylor Method (α=-0.25) | 0.75 | 0.3906 | 0.0038 | 0.2659 | 187.39 | 0.001 | 0 to 0.2 sec |

Table 2: Results from Example with no Damping

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***2% Damping*** |  |  |  |  |  |  |  |
| **Method** | **Gamma** | **Beta** | **Max Disp** | **Max Vel** | **Max Acc** | **Time Step** | **Time Range** |
| Average Acceleration | 0.5 | 0.25 | 0.0037 | 0.58 | 4812.1 | 0.0001 | 0 to 0.2 sec |
| Linear Acceleration | 0.5 | 0.1667 | 0.0036 | 0.5573 | 7786 | 0.000001 | 0 to 0.2 sec |
| Fox - Goodwin | 0.5 | 0.0833 | 0.0036 | 0.5571 | 7751.7 | 0.000001 | 0 to 0.2 sec |
| Algorithmically Damped | 0.6 | 0.4 | 0.0038 | 0.3709 | 278.636 | 0.001 | 0 to 0.2 sec |
| Hilber-Hughes-Taylor Method (α=-0.25) | 0.75 | 0.3906 | 0.0037 | 0.2415 | 178.73 | 0.001 | 0 to 0.2 sec |

Table 3: Results from Example with 2% Damping Factor

When we compare the maximum displacement, velocity, and acceleration from Table 2 and Table 3, we can confirm that the damping has an effect on the element. In most cases, there is a significant difference in these values. It was important to use the correct time steps because they gave more accurate results. It is interesting to see how each of the different methods produced different results based on which gamma and beta were used for the analysis. The Hilber-Hughes-Taylor method uses an alpha value whereas the other methods use a beta. The alpha method introduces less damping in lower modes, which preserves their accuracy. Based on our results, there was a less of a change among the values we saw for displacement, velocity, and acceleration between the damping and non-damping examples because of this.

# Conclusion

In conclusion, we were able to apply the Newmark beta method to the L-beam structure and find the maximum displacement, velocity, and acceleration. We applied these methods to two examples: one without damping, and one with a 2% damping factor. For each example, we plotted the displacement over time. As expected, the results for displacement, velocity, and acceleration for the damping factor were less than those without. The tables allowed us to see the affect the gamma, beta, and alpha values have on the Newmark method.

# Appendix

NewmarkBeta

function [ pos, vel, acc ] = NewmarkBeta( K, M, Beta, gamma, Zeta, deltat, t, tapp )

%%Finds displacement, velocity, acceleration using the newark beta method

clc

close all

run dof\_strip

%Reduce stiffness and mass from 3D to 2D by stripping displacement in z,

%rotation in x and rotation in y. Apply BCS (restrain x and y

%at node 1 and restrain y at node 51)

bcs = [1 2 152];

K = full(K);

K(stripdof,:) = [];

K(:,stripdof) = [];

K(bcs,:) = [];

K(:,bcs) = [];

M = full(M);

M(stripdof,:) = [];

M(:,stripdof) = [];

M(bcs,:) = [];

M(:,bcs) = [];

%Calculate Damping and from Force Matrix

C = Damps(K, Zeta);

R = zeros(size(K,1),1);

R(149,1) = 100000;

time = (0:deltat:t);

T = length(time);

R0 = zeros(size(R,1),size(R,2));

dn = zeros(size(K,1),1); %Initial position zero

ddn = zeros(size(K,1),1); %Initial velocity zero

dd2n = M\R; %F=ma == a=F/m

for i = 1:T

pos(:,i) = dn(:);

vel(:,i) = ddn(:);

acc(:,i) = dd2n(:);

step = time(i);

%If still within pluck use R, else use R0

if step <= tapp

Dn = dnplus1(R, M, Beta, deltat, dn, ddn, dd2n, C, gamma, K);

else

Dn = dnplus1(R0, M, Beta, deltat, dn, ddn, dd2n, C, gamma, K);

end

Ddn = ddnplus1(Beta,deltat,Dn,dn,ddn,dd2n,gamma);

Dd2n = dd2nplus1(Beta, deltat, dn, ddn, dd2n, Dn);

dn = real(Dn);

ddn = real(Ddn);

dd2n = real(Dd2n);

end

theta = pos(121,:);

dtheta = vel(121,:);

ddtheta = acc(121,:);

hold on

figure(1)

plot(time,theta)

ylabel('thetaz41 (rad/s)')

xlabel('time(s)')

figure(2)

plot(time,dtheta)

ylabel('dthetaz41 (rad/s)')

xlabel('time(s)')

figure(3)

plot(time,ddtheta)

ylabel('ddthetaz41 (rad/s)')

xlabel('time(s)')

%Working Input

%[pos, vel, acc] = NewmarkBeta(K, M, 0.25, 0.5, 0.0001, 0.15, 0.01)

end

dnplus1

function dnplus1 = dnplus1(Rnplus1, M, Beta, deltat, dn, ddn, dd2n, C, gamma, K )

%Find the D\_N+1 result

a=(1/(Beta\*deltat^2))\*M+(gamma/(Beta\*deltat))\*C+K;

b=Rnplus1;

size(M);

size(dn);

size(ddn);

size(dd2n);

c=(M\*((1/(Beta\*deltat^2))\*dn+(1/(Beta\*deltat))\*ddn+(1/(2\*Beta)-1)\*dd2n));

d=(C\*((gamma/(Beta\*deltat))\*dn+(gamma/Beta-1)\*ddn+(gamma/Beta-2)\*(deltat/2)\*dd2n));

size(a);

size(b);

size(c);

size(d);

test = (b+c+d);

size(test);

dnplus1t =a\(b+c+d);

dnplus1 = real(dnplus1t);

end

ddnplus1

function ddnplus1 = ddnplus1(Beta,deltat,dnplus1,dn,ddn,dd2n,gamma)

%Find the Ddn\_N+1 result

a = (gamma/(Beta\*deltat))\*(dnplus1-dn);

b = ((gamma/Beta)-1)\*ddn;

c = deltat\*((gamma/(2\*Beta))-1)\*dd2n;

ddnplus1t = a-b-c;

ddnplus1 = real(ddnplus1t);

end

dd2nplus1

function dd2nplus1 = dd2nplus1(Beta, deltat, dn, ddn, dd2n, dnplus1)

%Find the Dd2n\_N+1 result

a = 1/(Beta\*deltat^2)\*(dnplus1-dn-deltat\*ddn);

b = (1/(2\*Beta)-1)\*dd2n;

dd2nplus1t = a-b;

dd2nplus1 = real(dd2nplus1t);

end

Damps

function C = Damps( K, Zeta )

%Finds Damping matrix from Zeta, K, M

%Determine Mode Shapes and Eigenvalues

[mode, lam ]= eig(K);

w = sqrt(lam);

lam\_mat = 2\*Zeta\*w;

C = (mode')^-1\*lam\_mat\*mode^-1;

end

dofstrip

clc

stripdof = [3 4 5];

nodes = 50;

for i = 1:nodes

strip1 = stripdof(1)+6\*i;

strip2 = strip1+1;

strip3 = strip2+1;

strip = [strip1 strip2 strip3];

stripdof = [stripdof strip];

end