This article was originally published at http://www.varg.unsw.edu.au/movingload.htm on 08 August 2008

Finite Element Modelling of Moving Loads on Structures

Author: Gareth Forbes
Created: 08/08/08
Contact: g.forbes10@gmail.com

Introduction/Background

The problem of moving loads on structures was first considered in the early 19th century when the traversing of bridges by locomotives was analysed, this has been followed by a considerable amount of research on this topic continuing to this day. This problem of a locomotive travelling over a bridge can be solved with various assumptions such as, i) the bridge has negligible mass compared to the moving locomotive, ii) the moving locomotive's mass is negligible compared to the bridge, or finally iii) the entire system can be dynamically analysed in full when considering the effects of the masses of both the bridge and locomotive.

Fryba's monograph [1], deals with all of these problems along with numerous others, and is the greatest single point of reference for moving loads on structures. The second type of loading ii), of a point force moving across a single span bridge, was first solved at the beginning of the 20th century by Krylov and continued to include many different types of loading by Timoshenko, cf. [1]. Despite the analytical solution for moving loads on a structure being completed for many different load types a substantial amount of time ago, being able to implement a solution for this type of loading within commercial Finite Element packages is still not a straight forward task. Outlined here will be a method of implementing a moving load, one which is both variable in time and space, in both a commercial Finite Element package, ANSYS (with accompanying script), as well as a simple Finite Element code for solution written for Matlab. This tutorial is not meant to be an exhaustive overview of how to model moving loads with finite element packages, but is aimed at providing enough information and example code to provide a good introduction to someone unfamiliar of this category of modelling.

Analytical formulations

The first critical speed of a force travelling across a simply supported beam is shown to occur when the load traverses the length of the beam in the time it takes the first natural frequency to oscillate through one quarter of its period. This can be seen in figure (1). Therefore, the 1st critical speed of a moving load can be shown to be:

$$c_{cr} = \frac{\omega_{(1)} l}{\pi} \qquad (1)$$

Where 'n' is the mode number, 'l' is the length of the beam and ω_n is the n^{th} mode natural frequency in rad/s:

$$\omega_n = \frac{n^2\pi^2}{l^2}\left(\frac{EI}{\rho A}\right)^{1/2} \qquad (2)$$

It can be easily shown that the non-dimensional speed parameter of the ratio of load speed to critical speed will be:

$$\alpha = \frac{c}{c_{cr}} = \frac{cl}{\pi}\left(\frac{\rho A}{EI}\right)^{1/2} \qquad (3)$$

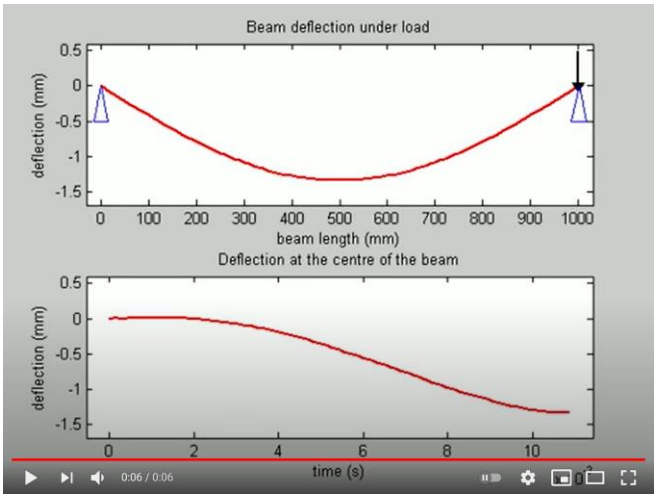

Figure 1 image i) moving force over a single span bridge, when the force velocity = first critical velocity, image ii) mid-span beam deflection. https://youtu.be/EdLTH7ymGJY

It is also common to have a non-dimensional term for damping which is denoted as:

$$\beta = \frac{\delta}{2\pi} = \frac{\omega_b}{\omega_{(1)}} \quad (4)$$

Where δ is the logarithmic decrement of damping of the beam. The circular angular frequency of the loads velocity, i.e. converting the horizontal load velocity into an angular velocity of rad/s, is:

$$\omega = \frac{\pi c}{l} \quad (5)$$

Derivation of the equation for motion for any point and moment in time, if $\alpha = \gamma$ and $\gamma = 1,2,3 \ldots$, then is can be shown to be (with the assumption of only light damping which is practical in most situations [1] β < 0.1):

$$u(x,t) \approx u_0 \frac{1}{2n^4}\left[e^{-\omega_b t} sinn\omega t - \frac{n^2}{\beta} cosn\omega t(1 - e^{-\omega_b t})\right] sin\frac{n\pi x}{l} + u_0 \sum_{\substack{n=1 \\ n\neq\gamma}}^{\infty} sin\frac{n\pi x}{l} \frac{1}{n^2(n^2-\alpha^2)}\left(sinn\omega t - \frac{\alpha}{n} e^{-\omega_b t} sin\omega_n t\right) \quad (6)$$

Where u_0 is the deflection of the beam at mid-span, being:

$$u_0 = \frac{Pl^3}{48EI} \quad (7)$$

Note: Equation (6) is only valid when the speed is equal to one of the critical speeds. See Ref [1] for further derivations

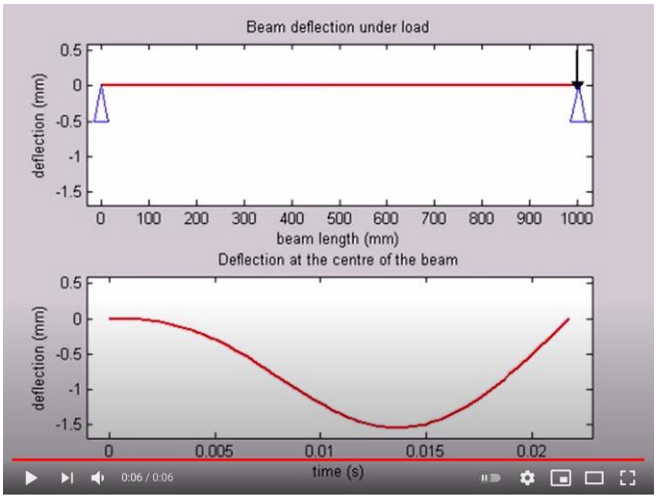

Figure 2 image i) moving force over a single span bridge, when the force velocity = half first critical velocity, image ii) mid-span beam deflection. https://youtu.be/XhU_IGx2CdU

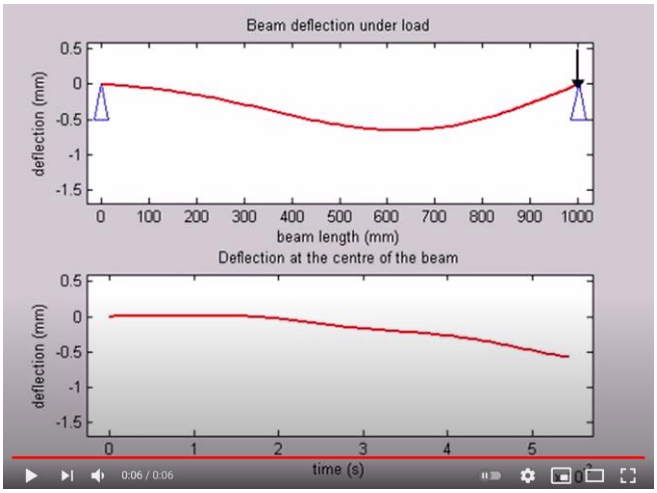

Figure 3 image i) moving force over a single span bridge, when the force velocity = twice first critical velocity, image ii) mid -span beam deflection https://youtu.be/fx3u416ehzw

If the maximum mid-span deflection of the beam is observed in figures (1)-(3), a curious result is observed. The maximum deflection of the mid-span of the beam occurs at just below the first critical velocity, however at speeds much greater than the lowest critical speed the deflection at the mid-span of the beam is lower than the static deflection, and maximum dynamic beam deflection. All three figures were produced using the matlab script available at Appendix A.

Use of finite elements for solution

Any method of modelling a moving load with finite elements, will require a transient solution method to be used, as a moving load is always transient in nature, i.e. no steady state dynamic motion occurs. Use of commercial finite element packages in general do not lend themselves easily towards modelling of such load cases. There exits however dedicated software applications for Finite element analysis of bridge structures under the influence of moving loads, such as BEDAS [2] or ANSYS CivilFEM. BEDAS is a mesh-free or exact stiffness finite element solver which is capable of providing solutions for 2D structures, it provides an GUI for specifying load speeds, magnitude and path. It does not however solve for the dynamics of the body traversing the structure, e.g. a vehicle passing over a bridge would be represented by a constant force, and the dynamics of the vehicle are not taken into account. Other in house codes have been developed by researchers for various dynamic loadings and to use with commercial packages, such as the DATIS GUI developed in [3]. Creation of a similar application script

should however be easily achieved with the implementation of techniques outlined here.

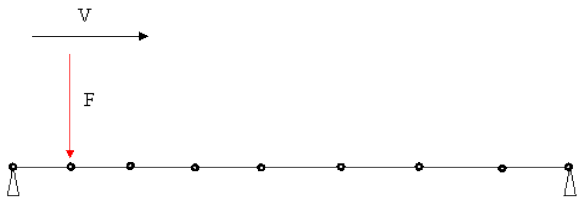

Figure 4 Simply supported beam with moving load 'F' at speed 'V'

If a simply supported beam is considered, with a moving point load, as shown in figure (4). Then the force for each node will change over time, such that a matrix of all the nodes of interest over the time span, in this case for the time the load is moving across the simply supported beam, can be constructed similar to what is shown in figure (5). (note the force vector does not actually jump in value from node to node as shown, see [4] for more details). This is essentially what the matlab script available in Appendix B implements with an explicit direct transient used to solve the beam motion for the respective time step.

$$\Delta t \Rightarrow \qquad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} nodes \\ \Downarrow \end{matrix}$$

Figure 5 Force matrix of nodes of interest for each time step

The method used in Appendix C, which provides a routine which can be used directly with the commercial finite element package ANSYS, differs slightly in that the load matrix is not pre-calculated, but the load on each node is calculated for each time step as the program solves for the respective load step.

The two files in Appendix B and C, both provide an example of how it is possible to model a moving load across a structure and includes some of the modelling subtleties which need to be considered. The results for the deflection of a single span beam, at load speed of half the first critical speed, using the finite element method in Appendix B is shown in figure (6) and can be compared to that which is obtained analytically in figure (2).

An explicit transient solution method is used in the model in Appendix B, however for more complex structures, it may often be more advantageous to seek a less computationally expensive modal transient solution or perhaps even an implicit solution method.

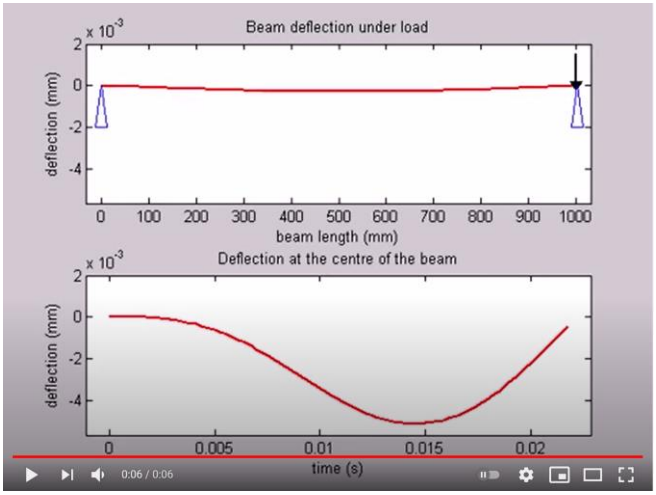

Figure 6 image i) moving force over a single span bridge, when the force velocity = half first critical velocity using the finite element method in Appendix B, image ii) mid-span beam deflection https://youtu.be/iM8w_3L9Zqs

References

[1] Fryba, L., Vibration of solids and structures under moving loads. 3rd ed. 1999, London: Thomas Telford. xxvii,494 p.

[2] http://www.mechatools.com/BDS/bedas.htm

[3] Liu, K., G. De Roeck, and E. Reynders. Experimental validation of the dynamic analysis of high speed composite railway bridge. in EURODYN 2008. 2008. Southampton, UK.

[4] Wu, J.-J., A.R. Whittaker, and M.P. Cartmell, Use of finite element techniques for calculating the dynamic response of structures to moving loads. Computers and Structures, 2000. 78(6): p. 789-799.

Appendix A – Analytical method (matlab)

```
% Moving force across a single span beam, solved analytically
clear all
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
% School of Mechanical and Manufacturing Engineering, University of New
% South Wales
% Author:   Gareth Forbes
% Date created: 1/2/08
% Date last modified: 10/11/08
% -------------------------------------------------------------------------
% -------------------------------------------------------------------------
% Revision number: 1
% Description of changes to latest revision:
% Updated to allow animated plotting for any input values. Change of plot
% y-axis labels to show relative displacement. Solution for all input
% values with 101 steps in direction and time plane.
% -------------------------------------------------------------------------
% -------------------------------------------------------------------------
% Description of Script:
% Creates the response of a single span beam under the influence of a
% moving force according to the analytical equations derived in [1]. Note,
% the formulation used here is only valid for light damping at various
% speeds, and with no damping at non-critical speeds. Two movies, at
% different moving load speeds have been created and uploaded at [2] and
% [3]
%
%
% References:
% [1]   Fryba, L., Vibration of solids and structures under moving loads.
% 3rd ed. 1999, London: Thomas Telford. xxvii,494 p.
% [2]   http://www.youtube.com/watch?v=XhU_IGx2CdU
% [3]   http://www.youtube.com/watch?v=jXIiQlnzChY
% -------------------------------------------------------------------------
% -------------------------------------------------------------------------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% input varibales for analysis
k = 5; %number of terms in expansion
P = 100; %load (N)
beta = 0.1; % non-dimensional damping parameter
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% material and geometric constants
L = 30480; %length of beam (mm)
E = 200000; %youngs modulus (MPa)
b = 2438; % width of cross section
h = 20; % height of cross section
A = b*h;% cross sectional area of beam (mm^2)
rho = 7.8E-9;% density of beam material (kg/mm^3)
I = (b*h^3)/12;% second moment of area (mm^4)
% critical speed (resonance of first mode)
cr = (pi/L)*(sqrt(E*I/rho/A));
% speed of load
c = 0.5*cr;% (mm/s)
alpha = c/cr; % non-dimensional speed parameter
omega = pi*c/L;
omegaj = ([1:k]*pi/L).^2*(sqrt(E*I/rho/A));
omegab = beta*omegaj(1);
% length vector;
x1 = 0:L/100:L;
x1 = x1';
x = repmat(x1,1,length(x1));
% time vector
```

```
tt = L/c;
step = tt/(length(x1)-1);
t1 = 0:step:tt;
t = repmat(t1,length(t1),1);
% static deflection of beam at mid span
u0 = (P*L^3)/(48*E*I); % (mm)
n = round(alpha);
if beta == 0;
    u2 = 0;
else
    if n > 0;
        if abs(n-alpha)<0.01;
            u2 = (u0/2/n^4)*(exp(-omegab*t).*sin(n*omega*t)-
(n^2/beta)*cos(n*omega*t).*(1-exp(-omegab*t))).*sin(n*pi*x/L);
        else
            u2 = 0;
        end
    else
        u2 = 0;
    end
end
u = 0;
for j = 1:k;
    if abs(j-alpha)<0.01;
        u1 = 0;
    else
        u1 = u0*(1/(j^2*(j^2-alpha^2)))*(sin(j*omega*t)-(alpha*exp(-
omegab*t)/j).*sin(omegaj(j)*t)).*sin(j*pi*x/L);
    end
    u = u +u1;
end
u3 = u +u2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% creation of movie
for i = 0:1:length(x1)-1;
    ax1 = (0:100)*L/100;
    subplot(2,1,1), plot(ax1,-u3(:,i+1)/u0,'linewidth',1.5,'color','r');
    xlabel('beam length (mm)')
    ylabel('deflection (u/u0)')
    title('Beam deflection under load')
    axis([-0.03*L 1.03*L -1.7 0.6])
    line([i*L/100 i*L/100],[0-(u3(i+1,i+1)/u0) 0.5-
(u3(i+1,i+1)/u0)],'Color','k','linewidth',2)
    line(i*L/100,(-
u3(i+1,i+1))/u0,'Color','k','marker','V','MarkerSize',3,'linewidth',3)
    % location of first supports
    a = [1 0];
    b = [L+1 0];
    % height of support
    h = 0.5;
    % support 1
    line([a(1) a(1)+h*L*0.03],[a(2) -h+a(2)],'linewidth',1);
    line([a(1)-h*L*0.03 a(1)+h*L*0.03],[-h+a(2) -h+a(2)],'linewidth',1);
    line([a(1)-h*L*0.03 a(1)],[-h+a(2) a(2)],'linewidth',1);
    % support 2
    line([b(1) b(1)+h*L*0.03],[b(2) -h+b(2)],'linewidth',1);
    line([b(1)-h*L*0.03 b(1)+h*L*0.03],[-h+b(2) -h+b(2)],'linewidth',1);
    line([b(1)-h*L*0.03 b(1)],[-h+b(2) b(2)],'linewidth',1);
    %
    ax2 = (0:100)*tt/100;
    subplot(2,1,2), plot(ax2(1:i+1),-u3(51,1:i+1)/u0,'linewidth',1.5,'color','r');
    xlabel('time (s)')
    ylabel('deflection (u/u0)')
```

```
    title('Deflection at the centre of the beam')
    axis([-tt/20 21*tt/20 -1.7 0.6])
    M(i+1) = getframe(gcf);
end
%movie2avi(M,'a','compression','none','quality',100)
```

Appendix B – FEA method (matlab)

```
clear all
close all
tic
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
% School of Mechanical and Manufacturing Engineering, University of New
% South Wales
% Author:   Gareth Forbes
% Date created: 1/2/08
% Date last modified: 12/8/08
% -------------------------------------------------------------------------
% -------------------------------------------------------------------------
% Description of Script:
% Solves the problem of a moving load across a single span simply supported
% beam, with the use of the finite element method. A transient direct
% central difference explicit integration scheme is used for solution, note
% this is often not the best solution method due to the amount of
% computations need for a direct solution method, transformation into
% the modal domain will often produce greater effeciency in the solution.
%
%
% References:
% [1]   Cook, R.D., et al., Concepts and applications of finite element
% analysis. 4th ed. 2002, New York, NY: Wiley. xvi, 719 p.
% [2]   Wu, J.-J., A.R. Whittaker, and M.P. Cartmell, Use of finite element
% techniques for calculating the dynamic response of structures to moving
% loads. Computers and Structures, 2000. 78(6): p. 789-799.
% -------------------------------------------------------------------------
% -------------------------------------------------------------------------
%speed of moving load mm/s
V = 0.25*9.1845e+004 % change this value to solve for different speeds
% -------------------------------------------------------------------------
% define material properties
rho = 7.8E-9; %density
b = 50; %width (mm)
h = 20; %height (mm)
A = b*h; %cross section
I = (b*h^3)/12; %second moment of area
E = 200000; %youngs modulus (MPa)
Lt = 1000; %length (mm)
%number of elements
EL = 30; % note the use of more than approx. 50 elements causes the
%solution time to become very large. The use of vector assignment instead
%of for loops may correct this
% number of nodes
N = EL + 1;
% forceing function in newtons, single pont force
P = -1;
% first critical speed of simply supported beam (mm/s)
SC = pi/Lt*sqrt(E*I/rho/A)
% load speed to critical speed ratio
crat = V/SC
% manual time step
tstepm = 1000; % minimum time step that is allowed
% all elements are defined as having the same length therefore the element
% lenght is;
L = Lt/EL;
% elemental mass matrix
%me = (rho*A*L/420)*[156 22*L 54 -13*L; 22*L 4*L^2 13*L -3*L^2; 54 13*L 156
%-22*L; -13*L -31*L^2 -22*L 4*L^2]; % distributed mass matrix
me = (rho*A*L/2)*[1 0 0 0;0 1/12*L^2 0 0;0 0 1 0; 0 0 0 1/12*L^2]; % lumped mass matrix
```

```
% elemental stiffness matrix
ke = (E*I/(L^3))*[12 6*L -12 6*L; 6*L 4*L^2 -6*L 2*L^2; -12 -6*L 12 -6*L; 6*L 2*L^2 -
6*L 4*L^2]; % distributed stiffness matrix
% create mass and stiffness matrix, note all propeties are constant
M1 = zeros(2*(EL+1),2*(EL+1));
M2 = zeros(2*(EL+1),2*(EL+1));
for i = 2:2:2*(EL+1);
    M2 = M2 + M1;
    M1 = zeros(2*(EL+1),2*(EL+1));
    M1(i-1:i+2,i-1:i+2) = me;
end
K1 = zeros(2*(EL+1),2*(EL+1));
K2 = zeros(2*(EL+1),2*(EL+1));
for i = 2:2:2*(EL+1);
    K2 = K2 + K1;
    K1 = zeros(2*(EL+1),2*(EL+1));
    K1(i-1:i+2,i-1:i+2) = ke;
end
% enforce boundary conditions
%  delete first and second last row and column
M = zeros(2*EL,2*EL);
K = zeros(2*EL,2*EL);
M(1:2*EL-1,1:2*EL-1) = M2(2:2*EL,2:2*EL);
M(2*EL,1:2*EL-1) = M2(2*(EL+1),2:2*EL);
M(1:2*EL-1,2*EL) = M2(2:2*EL,2*(EL+1));
M(2*EL,2*EL) = M2(2*(EL+1),2*(EL+1));
K(1:2*EL-1,1:2*EL-1) = K2(2:2*EL,2:2*EL);
K(2*EL,1:2*EL-1) = K2(2*(EL+1),2:2*EL);
K(1:2*EL-1,2*EL) = K2(2:2*EL,2*(EL+1));
K(2*EL,2*EL) = K2(2*(EL+1),2*(EL+1));
%create damping matrix, using proportional damping
alpha = 0.0002;
beta = 0.0002;
cm = alpha*M;
ck = beta*K;
ct = cm + ck;
% calculate the minimum time step
% solve eigen value problem such that deltat < 2/natfmax. see [1]
[V1,D1] = eig(K,M);
ttotal = Lt/V; % length of time record in seconds %0.1
E1 = sqrt(D1);
deltat1 = 2/(max(max(real(E1))));
deltat = (1/(1/deltat1+1));
tstep = max(round(ttotal/deltat),tstepm);

% create force vector for time step. see [2]
for i = 1:tstep;
    for j = 1:EL+1;
        xp = V*i*deltat;
        s = fix(xp/L)+1;
        zeta = (xp - (s-1)*L)/(L);
        N1 = 1 - 3*zeta^2 + 2*zeta^3;
        N2 = (zeta - 2*zeta^2 + zeta^3)*L;
        N3 = 3*zeta^2 - 2*zeta^3;
        N4 = (-zeta^2 + zeta^3)*L;
            if s == EL+1;
                F(j,i) = 0;
                MO(j,i) = 0;
            elseif j == s ;
                F(j,i) = P*N1;
                MO(j,i) = P*N2;
            elseif j == s+1;
                F(j,i) = P*N3;
```

```
                MO(j,i) = P*N4;
            else
                F(j,i) = 0;
                MO(j,i) = 0;
            end
    end
end
% set first and last rows of load vectors to zero, so that he load
% gradually enters and exits the structure, ie. does not jump on and off,
% this could be removed if a jump on and off is desired
F(1,:) = zeros;
F(end,:) = zeros;
MO(1,:) = zeros;
MO(end,:) = zeros;
    % create external load vector
    REXT2 = zeros(2*(EL+1),tstep);
    for i = 1:EL+1;
        REXT2(2*i-1,:) = F(i,:);
        REXT2(2*i,:) = MO(i,:);
    end
% enforce boundary conditions
% delete first and second last row and column
REXT = zeros(2*EL,tstep);
REXT(1:2*EL-1,:) = REXT2(2:2*EL,:);
REXT(1:2*EL-1,:) = REXT2(2:2*EL,:);
REXT(2*EL,:) = REXT2(2*(EL+1),:);
% use a central difference direct explicit integration method to find the
% response history
D = zeros(2*EL,tstep);
D2C = ((1/deltat^2)*M + (1/(2*deltat))*ct);
D1C = (2/deltat^2)*M;
D0C = ((1/deltat^2)*M - (1/(2*deltat))*ct);
RINT = K;
for i = 2:tstep-1;
    % enforce boundary conditions
    %D(1,i) = zero;
    %D(2*EL+1,i) = zero;
    D(:,i+1) = D2C\(REXT(:,i) - RINT*D(:,i) + D1C*D(:,i) - D0C*D(:,i-1));
end
% creat matrix of full element displacements
Df = zeros(2*(EL+1),tstep);
Df(2:2*EL,:) = D(1:2*EL-1,:);
Df(2*(EL+1),:) = D(2*EL,:);
% create matrix of just displacements and rotations
for i = 1:EL+1;
        Dx(i,:) = Df(2*i-1,:);
        Dtheta(i,:) = Df(2*i,:);
end
figure
ax = (0:length(Dx)-1)*ttotal/length(Dx);
fe = round(Lt/V/deltat); %time step when load leaves structure
plot(ax(fe),Dx(round(EL/2+1),fe),'ro')
hold
plot(ax,Dx(round(EL/2+1),:))
xlabel('time')
ylabel('displacement (mm)')
title('Displacement of centre element for whole time period')
legend('markes when load leaves structure')
hold
figure
% creation of movie. Note movie script is quite messy. Also the use of less
% than approx. 50 elements causes the movie to be a little stilted, however
% solution time for more than approx. 50 elements is quite long. Vector
% assignment instead of for loops may solve this problem.
```

```
for i = 0:1:100;
    ae = fe/100;
    ax1 = (0:EL)*Lt/EL;
    subplot(2,1,1), plot(ax1,Dx(:,round(i*ae)+1),'linewidth',1.5,'color','r');
    xlabel('beam length (mm)')
    ylabel('deflection (mm)')
    title('Beam deflection under load')
    ge(i+1) = Dx(round(round(ae*i)*deltat*V*EL/Lt+1),round(i*ae)+1);
    axis([-30 1030 -0.017/3 0.006/3])
    line([10*i 10*i],[ge(i+1) 0.0015+ge(i+1)],'Color','k','linewidth',2)
    line(i*10,ge(i+1),'Color','k','marker','V','MarkerSize',3,'linewidth',3)
    % location of first supports
    a = [1 0];
    b = [1001 0];
    % height of support
    h = 0.002;
    % support 1
    line([a(1) a(1)+h*6000],[a(2) -h+a(2)],'linewidth',1);
    line([a(1)-h*6000 a(1)+h*6000],[-h+a(2) -h+a(2)],'linewidth',1);
    line([a(1)-h*6000 a(1)],[-h+a(2) a(2)],'linewidth',1);
    % support 2
    line([b(1) b(1)+h*6000],[b(2) -h+b(2)],'linewidth',1);
    line([b(1)-h*6000 b(1)+h*6000],[-h+b(2) -h+b(2)],'linewidth',1);
    line([b(1)-h*6000 b(1)],[-h+b(2) b(2)],'linewidth',1);
    ax2 = (0:100)*Lt/V/100;
    subplot(2,1,2),
plot(ax2(1:i+1),Dx(EL/2+1,1:round(ae):i*round(ae)+1),'linewidth',1.5,'color','r');
    xlabel('time (s)')
    ylabel('deflection (mm)')
    title('Deflection at the centre of the beam')
    axis([-Lt/V/20 21*Lt/V/20 -0.017/3 0.006/3])
    Mov(i+1) = getframe(gcf);
end
toc
%movie2avi(Mov,'a','compression','none','quality',100)
```

Appendix C – FEA method (ANSYS APDL)

```
! Moving force across a single span beam, solved with ANSYS
!-------------------------------------------------------------------------
!-------------------------------------------------------------------------
! School of Mechanical and Manufacturing Engineering, University of New
! South Wales
! Author:   Gareth Forbes
! Date created: 13/8/08
! Date last modified: 13/8/08
! ------------------------------------------------------------------------
! ------------------------------------------------------------------------
! Description of Script:
! Creates the response of a single span beam under the influence of a
! moving force, for use with the ANSYS FEA software. More information can
! be found at www.varg.unsw.edu.au/movingload
!
!
! ------------------------------------------------------------------------
! ------------------------------------------------------------------------
! beam with 11 nodes consecutively numbered
! note script only works for a 10 element simply supported beam

/PREP7
ET,1,BEAM3
R,1,10,10000/12,10, , , ,
MPTEMP,,,,,,,,
MPTEMP,1,0
MPDATA,EX,1,,200000
MPDATA,PRXY,1,,0.3
MPTEMP,,,,,,,,
MPTEMP,1,0
MPDATA,DENS,1,,7.8E-9
mat,1
type,1
real,1
*DO,i,1,11,1
N,i,(i-1)*100,0,0
*ENDDO
*DO,i,1,10,1
EN,i,i,i+1
*ENDDO

FINISH


! define beam length in mm
Lt = 1000
!number of elements
```

```
EL = 10
! number of nodes
N = EL + 1
! forceing function in newtons, single pont force
P = -100
!speed of moving load mm/s
V = 10000
! manual time step
tstepm = 1000
! all elements are defined as having the same length therefore the element
! lenght is
L = Lt/EL
ttotal = Lt/V
deltat1 = L/V
deltat = min((1/(1/deltat1+1)),ttotal/tstepm)
tstep = max(NINT(ttotal/deltat),tstepm)

/CONFIG,NRES,tstep
/SOL
ANTYPE,4
TRNOPT,FULL
LUMPM,0
ALPHAD,0
BETAD,0.0005

D,1, ,0, , , ,UX,UY, , , ,
D,11, ,0, , , ,UX,UY, , , ,

TM_START=deltat          ! Starting time (must be > 0)
TM_END=ttotal            ! Ending time of the transient
TM_INCR=deltat           ! Time increment
nnode=11                 !number of nodes

*DO,TM,TM_START,TM_END,TM_INCR    ! Do for TM from TM_START to TM_END in
                    !  steps of TM_INCR
  TIME,TM             ! Time value
        *DO,i,1,nnode,1
                xp = V*TM
                ff = (xp/L+1)/NINT(xp/L+1)
                *IF,ff,GE,1,THEN
                        s1 = NINT(xp/L)
                *ELSE
                        s1 = NINT(xp/L)-1
                *ENDIF
                s = s1+1
        zeta = (xp - (s-1)*L)/(L)
        N1 = 1 - 3*zeta**2 + 2*zeta**3
        N2 = (zeta - 2*zeta**2 + zeta**3)*L
        N3 = 3*zeta**2 - 2*zeta**3
        N4 = (-zeta**2 + zeta**3)*L
```

```
        *IF,i,EQ,s,THEN
                Fy11 = P*N1
        *ELSEIF,i,EQ,s+1
                Fy11 = P*N3
        *ELSE
                Fy11 = 0
        *ENDIF
        *IF,i,EQ,s,THEN
                Mz11 = P*N2
        *ELSEIF,i,EQ,s+1
                Mz11 = P*N4
        *ELSE
                Mz11 = 0
        *ENDIF

                F,i,FY,Fy11
                F,i,MZ,Mz11

        *ENDDO
  SOLVE             ! Initiate solution calculations
*ENDDO
```