4.2 Brute-force time-stepping method

1. Newmark method

(1) The general idea of the Newmark method is introduced as follows in a single degree of freedom system without any damping mechanism for simplicity. Given values of u_i^a and $\partial u_i^a/\partial t$ at some time t, we want to determine values at a slightly later time $t+\Delta t$, using the governing equations of motion. There are of course many ways to do this. Here, the popular Newmark time integration scheme will be presented.

(2) The equation of motion for a forced and undamped spring-mass system is:

$$m\,\ddot{u} + k\,u - f(t) = 0 \tag{4.16}$$

where $f(t)$ and the initial conditions of the displacement $u(t=0)$ and velocity $\dot{u}(t=0)$ are given.

(3) Suppose that the estimates for the accelerations $\ddot{u}(t)$ and $\ddot{u}(t+\Delta t)$ both at the start and end of a general time step Δt. We could then use a *Taylor series expansion* to obtain estimates of displacement and velocity at time $t+\Delta t$:

$$\dot{u}(t+\Delta t) \approx \dot{u}(t) + \Delta t\,[(1-\beta_1)\ddot{u}(t) + \beta_1\ddot{u}(t+\Delta t)] \tag{4.17}$$

$$u(t+\Delta t) \approx u(t) + \Delta t\,\dot{u}(t) + \frac{\Delta t^2}{2}[(1-\beta_2)\ddot{u}(t) + \beta_2\ddot{u}(t+\Delta t)] \tag{4.18}$$

where, β_1 and β_2 are two adjustable parameters that determine the nature of the time integration scheme.

① If we set $\beta_1 = 0$ and $\beta_2 = 0$, the acceleration is estimated based on its value at time t. This is known as an *explicit time integration* scheme.

② If we put $\beta_1 = 1$ and $\beta_2 = 1$, the acceleration is estimated from its value at time $t+\Delta t$. This is known as an *implicit time integration* scheme.

③ If we put $\beta_1 = \frac{1}{2}$ and $\beta_2 = \frac{1}{2}$, the acceleration is estimated as the mean of its values at time t and $t+\Delta t$. This is known as a *constant average of acceleration* scheme.

④ If we put $\beta_1 = \frac{1}{2}$ and $\beta_2 = \frac{1}{3}$, the acceleration is estimated as the linear variation from its values at time t to $t+\Delta t$. This is known as a *linear variation of acceleration* scheme.

(4) The accelerations at the start and end of the time step are computed using the equation of motion. At time t,

$$\ddot{u}(t) = \frac{1}{m}[-k\,u(t) + f(t)] \tag{4.19}$$

whereas, at time $t+\Delta t$,

$$\begin{aligned} &m\ddot{u}(t+\Delta t) + ku(t+\Delta t) - f(t+\Delta t) = 0 \\ &m\ddot{u}(t+\Delta t) + k\left\{u(t) + \Delta t\,\dot{u}(t) + \frac{\Delta t^2}{2}[(1-\beta_2)\ddot{u}(t) + \beta_2\ddot{u}(t+\Delta t)]\right\} - f(t+\Delta t) = 0 \\ &\ddot{u}(t+\Delta t) = \frac{1}{m + k\beta_2\Delta t^2/2}\left\{-k\left[u(t) + \Delta t\,\dot{u}(t) + \frac{\Delta t^2}{2}(1-\beta_2)\ddot{u}(t)\right] + f(t+\Delta t)\right\} \end{aligned} \tag{4.20}$$

This is the basic formula forming the **time-stepping scheme** for given k, m, $f(t)$, $u(0)$, and $\dot{u}(0)$:

① At $t=0$, compute:

$$\ddot{u}(0) = \frac{1}{m}[-k\,u(0) + f(0)] \tag{4.21}$$

② For succesive time steps, compute the accelerations:

$$\ddot{u}(t+\Delta t) = \frac{1}{m + k\beta_2\Delta t^2/2}\left\{-k\left[u(t) + \Delta t\,\dot{u}(t) + \frac{\Delta t^2}{2}(1-\beta_2)\ddot{u}(t)\right] + f(t+\Delta t)\right\} \tag{4.22}$$

③ For succesive time steps, compute the displacements:

$$u(t+\Delta t) \approx u(t) + \Delta t\, \dot{u}(t) + \frac{\Delta t^2}{2}[(1-\beta_2)\ddot{u}(t) + \beta_2 \ddot{u}(t+\Delta t)] \tag{4.23}$$

④ For succesive time steps, compute the velocities:

$$\dot{u}(t+\Delta t) \approx \dot{u}(t) + \Delta t\,[(1-\beta_1)\ddot{u}(t) + \beta_1 \ddot{u}(t+\Delta t)] \tag{4.24}$$

⑤ Go back to the step 2). for evaluating the accelerations, displacements, and velocities of the consequent time steps.

2. Newmark method applied to finite element equations

(1) The time-stepping scheme developed in the Newmark method can immediately be extended to the general n degrees of freedom case with given M_{ab}, K_{aibk}, $F_i^a(t)$, $u_i^a(0)$, and $\dot{u}_i^a(0)$.

(2) As for the finite element mass matrix (Eq. 4.11), the mass matrix of each element can be evaluated by using numerial Gaussian quadrature as before:

$$\begin{aligned} m_{ab}^{(\ell)} &= \int_{V_e^{(\ell)}} \rho\, N^a(\boldsymbol{x}) N^b(\boldsymbol{x})\, dV \\ &= \int_{\hat{V}_e^{(\ell)}} \rho\, \hat{N}^a(\boldsymbol{\xi}) \hat{N}^b(\boldsymbol{\xi}) |\boldsymbol{J}(\boldsymbol{\xi})|\, dV \\ &= \sum_{I=1}^{M} w_I\, \rho\, N^a(\boldsymbol{\xi}^I) N^b(\boldsymbol{\xi}^I) |\boldsymbol{J}(\boldsymbol{\xi}^I)| \end{aligned} \tag{4.25}$$

and the global mass matrix of numbers M_{ab} can be then assembled from the mass matrix of each element above as usual.

(3) In the beginning (at $t = 0$), compute

$$\ddot{u}_i^a(0) = M_{ab}^{-1}\left[K_{bick} u_k^c(0) + F_i^b(0)\right] \tag{4.26}$$

where M_{ab}^{-1} is the number at a-th row and b-th row of the inverse matrix of the mass matrix $\boldsymbol{M}$.

(4) For successive time steps, solve the accelerations $\ddot{u}_i^a(t+\Delta t)$ at $t+\Delta t$ by solving

$$M_{ab}\,\ddot{u}_i^b(t+\Delta t) + \frac{\beta_2 \Delta t^2}{2} K_{aibk}\,\ddot{u}_k^b(t+\Delta t) = -K_{aibk}\left[u_k^b(t) + \Delta t \dot{u}_k^b(t) + \frac{\Delta t^2}{2}(1-\beta_2)\,\ddot{u}_k^b(t)\right] + F_i^a(t+\Delta t) \tag{4.27}$$

(5) Then compute the velocities $\dot{u}_i^a(t+\Delta t)$ at $t+\Delta t$ by

$$\dot{u}_i^a(t+\Delta t) \approx \dot{u}_i^a(t) + \Delta t\,[(1-\beta_1)\,\ddot{u}_i^a(t) + \beta_1 \ddot{u}_i^a(t+\Delta t)] \tag{4.28}$$

(6) Also compute the displacements $u_i^a(t+\Delta t)$ at $t+\Delta t$ by

$$u_i^a(t+\Delta t) \approx u_i^a(t) + \Delta t\, \dot{u}_i^a(t) + \frac{\Delta t^2}{2}[(1-\beta_2)\,\ddot{u}_i^a(t) + \beta_2 \ddot{u}_i^a(t+\Delta t)] \tag{4.29}$$

(7) The stable condition of the Newmark method:

$$\frac{\Delta t}{T} \le \frac{1}{\pi\sqrt{2}} \frac{1}{\sqrt{\beta_1 - \beta_2}} \tag{4.30}$$

where T is the natural frequency of the particular mode of interest. For the case of $\beta_1 = \dfrac{1}{2}$ and $\beta_2 = \dfrac{1}{2}$, this condition becomes:

$$\frac{\Delta t}{T} < \infty \tag{4.31}$$

This implies that the constant average of acceleration scheme is stable for any Δt, no matter how large; however, it is accurate only if Δt is small enough. For the linear variation of acceleration scheme ($\beta_1 = \dfrac{1}{2}$ and $\beta_2 = \dfrac{1}{3}$), the stable condition is:

$$\frac{\Delta t}{T} < 0.551 \tag{4.32}$$

4.3 Modal method

1. Modal analysis

(1) The modal method for time integration is introduced in this section. Consider the finite element equations (Eq. 4.14):

$$M_{ab}\,\frac{\partial^2 u_i^b}{\partial t^2} + K_{aibk}\,u_k^b - F_i^a = 0 \quad \forall (i,a) : x_k^a \text{ not on } \partial_1 R$$

are a standard set of coupled, second-order linear differential equations such as one would meet in analyzing forced vibrations in mechanical systems. They can thus be solved using the usual *modal techniques*. The procedure is to rearrange the system of equations to yield n *decoupled* second-order differential equations, by means of the following substitutions.

(2) Modal analysis applied to finite element equations:

① Write the governing equation as:

$$\boldsymbol{M}\ddot{\boldsymbol{u}} + \boldsymbol{K}\boldsymbol{u} = \boldsymbol{F}(t) \tag{4.33}$$

② Let

$$\boldsymbol{q} = \boldsymbol{M}^{1/2}\,\boldsymbol{u} \tag{4.34}$$

and

$$\boldsymbol{H} = \boldsymbol{M}^{-1/2}\,\boldsymbol{K}\,\boldsymbol{M}^{-1/2} \tag{4.35}$$

and the governing equation can be rewritten as:

$$\ddot{\boldsymbol{q}} + \boldsymbol{H}\,\boldsymbol{q} = \boldsymbol{M}^{-1/2}\,\boldsymbol{F} \tag{4.36}$$

③ Note that $\boldsymbol{H}$ is a symmetric, positive definite matrix. Consequently, the *spectral decomposition* can be performed as:

$$\boldsymbol{H} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T \tag{4.37}$$

where $\boldsymbol{Q}$ is an orthogonal matrix ($\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$) and $\boldsymbol{\Lambda}$ is a diagonal matrix.

④ In the spectral decomposition, first compute the the eigenvalues λ_i and corresponding normalized eigenvectors $\boldsymbol{r}^{(i)}$ ($\boldsymbol{r}^{(i)} \cdot \boldsymbol{r}^{(i)} = 1$) of the $\boldsymbol{H}$ matrix. Then, the $\boldsymbol{\Lambda}$ and $\boldsymbol{Q}$ matrices are:

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \tag{4.38}$$

$$\boldsymbol{Q} = \begin{bmatrix} r_1^{(1)} & r_1^{(2)} & \cdots & r_1^{(n)} \\ r_2^{(1)} & r_2^{(2)} & \cdots & r_2^{(n)} \\ \vdots & \vdots & \cdots & \vdots \\ r_n^{(1)} & r_n^{(2)} & \cdots & r_n^{(n)} \end{bmatrix} \tag{4.39}$$

⑤ Next, let $\boldsymbol{w} = \boldsymbol{Q}^T\,\boldsymbol{q}$, and rearrange the governing equation as:

$$\ddot{\boldsymbol{w}} + \boldsymbol{\Lambda}\,\boldsymbol{w} = \boldsymbol{Q}^T\,\boldsymbol{M}^{-1/2}\,\boldsymbol{F} \tag{4.40}$$

Because $\boldsymbol{\Lambda}$ is diagonal, this is now a set of n *uncoupled ODE*s for the general displacements $\boldsymbol{w}$.

⑥ The initial conditions for the general displacements $\boldsymbol{w}$ are:

$$\boldsymbol{w} = \boldsymbol{Q}^T\,\boldsymbol{q} = \boldsymbol{Q}^T\,\boldsymbol{M}^{1/2}\,\boldsymbol{u}(0) \tag{4.41}$$

$$\dot{\boldsymbol{w}} = \boldsymbol{Q}^T\,\dot{\boldsymbol{q}} = \boldsymbol{Q}^T\,\boldsymbol{M}^{1/2}\,\dot{\boldsymbol{u}}(0) \tag{4.42}$$

⑦ By using the initial conditions of $\boldsymbol{w}$, the decoupled ODEs (Eq. 4.40) can be solved, and then the displacements are recovered as:

$$\boldsymbol{u} = \boldsymbol{M}^{-1/2}\boldsymbol{Q}\boldsymbol{w} \tag{4.43}$$

⑧ Many methods can be applied to solve the decoupled ODEs for $\boldsymbol{w}$:

a. In harmonic forcing cases, you could use the exact solution.

b. For general $\boldsymbol{F}(t)$ cases, you could use Fourier transforms to present the original decoupled ODE by using Fourier coefficients of

$\boldsymbol{F}(t)$ in such harmonic forcing problems.

c. The Newmark algorithm could be also used.

2. Natural frequencies and mode shapes

(1) The *natural frequencies* (the reciprocal of the natural periods) and mode shapes for a vibrating, linear elastic solid can be extracted by the following procedures.

(2) By definition, the natural frequencies and mode shapes of a continuous solid or structure are special deformations and frequencies for which the solid will vibrate harmonically. Again, the governing equations for the general displacement $\boldsymbol{w}$ can be regarded as describing the vibration of n uncoupled spring-mass systems.

(3) If we solve the problem with $\boldsymbol{F} = \boldsymbol{0}$ and excite only the i-th spring-mass system (e.g., by choosing appropriate initial conditions), the solution will be a harmonic vibration at the natural frequency of the i-th spring-mass system.

(4) The natural frequencies of the n decoupled systems are therefore the natural frequencies of vibration of the solids or structure. The corresponding eigenvectors $\boldsymbol{r}^{(i)}$ are related to the deformations $\boldsymbol{u}^{(i)}$ associated with the i-th vibration mode through $\boldsymbol{u}^{(i)} = \boldsymbol{M}^{-1/2}\,\boldsymbol{r}^{(i)}$.

(5) Therefore, procedures for determining the natural frequencies and mode shapes of a linear elastic solid are:

① Compute the finite element mass and stiffness matrices $\boldsymbol{M}$ and $\boldsymbol{K}$.

② Find $\boldsymbol{H} = \boldsymbol{M}^{-1/2}\,\boldsymbol{K}\,\boldsymbol{M}^{-1/2}$.

③ Compute the eigenvalues λ_i and corresponding eigenvectors $\boldsymbol{r}^{(i)}$ of $\boldsymbol{H}$.

④ The natural frequencies ω_i are related to the eigenvalues of $\boldsymbol{H}$ by $\lambda_i = \omega_i^2$.

⑤ The deflections $\boldsymbol{u}^{(i)}$ associated with the i-th vibration mode follow as $\boldsymbol{u}^{(i)} = \boldsymbol{M}^{-1/2}\,\boldsymbol{r}^{(i)}$. The deflections calculated in this way will have some random magnitude, so we are suggested to normalize the deflection vectors appropriately.

(6) There two issues to be noticed in practice. First, we need to find the square root of the mass matrix by computing a spectral decomposition of $\boldsymbol{M}$. See the MATLAB code demonstration for details. Second, computing all the eigenvalues and eigenvectors of a general matrix $\boldsymbol{H}$ is an exceedingly time-consuming process. To make this viable, only the lowest few eigenvalues and the corresponding eigenvectors (e.g., 10 ~ 20) are normally extracted, and the others are reasonably discarded.

3. Comparison of the Newmark and modal methods for time integration

(1) For linear problems, the modal decomposition method usually beats direct time integration in computation cost.

(2) If you are interested in looking in detail at transient wave propagation through the solid, there is not much difference between the two methods; however, in this case you need to retain a huge number of terms in the modal expansion for accurate results.

(3) For most vibration problems, however, it is generally only necessary to retain a small number of modes in the expansion, in which case the modal decomposition method will be vastly preferable to direct time integration. Moreover, knowledge of the vibration modes can itself be valuable information. You can use also use the modal approach to handle very complex forcing, including random vibrations, directly.

(4) The main limitation of the modal decomposition approach is that it can only handle undamped, linear systems (one can introduce modal damping for vibration applications, but this does not model any real energy dissipation mechanism). This is, of course, far too restrictive for any real application except the simplest possible problems in vibration analysis.

4. MATLAB code demonstration

Please see the MATLAB code demonstrations:

(1) Newmark method

Note: Tha mass density of the material has to be specified as the 4-th number of the `mate` array in your `ReadInput` function.

① *GlobMass* function

```
function mglob = GlobMass(ndime,nnode,nelem,nelnd,mate,coor,conn)
  mglob = zeros(ndime*nnode,ndime*nnode);
  for j = 1:nelem
    mel = ElemMass(j,ndime,nelnd,coor,conn,mate);
    for a = 1:nelnd
      for i = 1:ndime
        for b = 1:nelnd
          for k = 1:ndime
            ir = ndime*(conn(a,j)-1)+i;
            ic = ndime*(conn(b,j)-1)+k;
            mglob(ir,ic) = mglob(ir,ic) + mel(ndime*(a-1)+i,ndime*(b-1)+k);
          end
        end
```

```
      end
    end
  end
end
```

② *ElemMass* function

```
function mel = ElemMass(iel,ndime,nelnd,coor,conn,mate)
  mel = zeros(ndime*nelnd,ndime*nelnd);
  coorie = zeros(ndime,nelnd);
  rho = mate(4);
  xii = zeros(ndime,1);
  dxdxi = zeros(ndime,ndime);
  M = numIntegPt(ndime,nelnd);
  xi = IntegPt(ndime,nelnd,M);
  w = integWt(ndime,nelnd,M);
  for a = 1:nelnd
    for i = 1:ndime
      coorie(i,a) = coor(i,conn(a,iel));
    end
  end
  for im = 1:M
    for i = 1:ndime
      xii(i) = xi(i,im);
    end
    N = ShpFunc(nelnd,ndime,xii);
    dNdxi = ShpFuncDeri(nelnd,ndime,xii);
    dxdxi(:) = 0;
    for i = 1:ndime
      for j = 1:ndime
        for a = 1:nelnd
          dxdxi(i,j) = dxdxi(i,j)+coorie(i,a)*dNdxi(a,j);
        end
      end
    end
    jcb = det(dxdxi);
    for a = 1:nelnd
      for b = 1:nelnd
        for i = 1:ndime
          ir = ndime*(a-1)+i;
          ic = ndime*(b-1)+i;
          mel(ir,ic)=mel(ir,ic)+rho*N(b)*N(a)*w(im)*jcb;
        end
      end
    end
  end
end
```

③ *Main* program

The main program developed in a script mainly deals with

a. Modifying the global mass and global stiffness matrices

b. Newmark method for time-stepping analysis

ⓐ β_1 and β_2 are specified in the 5-th and 6-th numbers in the `mate` array.

ⓑ `dt`, `nstp`, and `nprt` are the time increment, total time steps, and the number of time steps for one *.vtk file and specified in the 7-th, 8-th and 9-th numbers in the `mate` array.

```
.
.
.
[ndime,nnode,nelem,nelnd,npres,ntrac,mate,coor,conn,pres,trac] = ReadInput(infile);
mglob = GlobMass(ndime,nnode,nelem,nelnd,mate,coor,conn);
kglob = GlobStif(ndime,nnode,nelem,nelnd,mate,coor,conn);
rglob = GlobTrac(ndime,nnode,nelem,nelnd,ntrac,mate,coor,conn,trac);
mpres = mglob;
kpres = kglob;
rpres = rglob;
for i = 1:npres
  ir = ndime*(pres(1,i)-1)+pres(2,i);
  for ic = 1:ndime*nnode
    mpres(ir,ic) = 0;
    mpres(ic,ir) = 0;
    kpres(ir,ic) = 0;
    kpres(ic,ir) = 0;
  end
  mpres(ir,ir) = 1.;
  rpres(ir) = pres(3,i);
end
beta1 = mate(5);
beta2 = mate(6);
dt = mate(7);
nstp = mate(8);
nprt = mate(9);
vc = zeros(nnode*ndime,1);
uc = zeros(nnode*ndime,1);
mkpres = mpres+0.5*beta2*dt*dt*kpres;
ac = mkpres\(-kpres*uc+rpres);
iprt = 0;
for i = 1:nstep
  an = mkpres\(rpres-kpres*(uc+dt*vc+0.5*(1.-beta2)*dt*dt*ac));
  vn = (vc+dt*(1.-beta1)*ac+dt*beta1*an);
  un = (uc+dt*vc+(1.-beta2)*0.5*dt*dt*ac+0.5*beta2*dt*dt*an);
  ac = an;
  vc = vn;
  uc = un;
  if(iprt == nprt)
    iprt = 0;
    .
    .
    .
  end
  iprt = iprt+1;
end
.
.
.
```

(2) Natural frequencies and mode shapes

The main program developed in a script deals with natural frequencies and corresponding mode shapes. `nmod` is specified in the 10-th number in the `mate` array.

```
.
```

```
.
.
[ndime,nnode,nelem,nelnd,npres,ntrac,mate,coor,conn,pres,trac] = ReadInput(infile);
mglob = GlobMass(ndime,nnode,nelem,nelnd,mate,coor,conn);
kglob = GlobStif(ndime,nnode,nelem,nelnd,mate,coor,conn);
rglob = GlobTrac(ndime,nnode,nelem,nelnd,ntrac,mate,coor,conn,trac);
mpres = mglob;
kpres = kglob;
rpres = rglob;
for i = 1:npres
  ir = ndime*(pres(1,i)-1)+pres(2,i);
  for ic = 1:ndime*nnode
    mpres(ir,ic) = 0;
    mpres(ic,ir) = 0;
    kpres(ir,ic) = 0;
    kpres(ic,ir) = 0;
  end
  mpres(ir,ir) = 1.;
  rpres(ir) = pres(3,n);
end
mpresroot = sqrtm(mpres);
mpresrootinv = inv(mpresroot);
hpres = mpresrootinv*(kpres*mpresrootinv);
[q,lambda,~] = svd(hpres);
lambdasort = zeros(ndime*nnode,1);
for i = 1:ndime*nnode
  lambdasort(i) = lambda(i,i);
end
lambdasort = sort(lambdasort);
nmod = mate(10);
if(ndime == 2)
  nmodrbm = 3;
else
  nmodrbm = 6;
end
u = zeros(nnode*ndime,nmod);
for k = 1:nmod
  for i = 1:nnode*ndime
    if((lambda(i,i)-lambdasort(nmodrbm+k))^2 < 1e-8)
      ipick = i;
      break;
    end
  end
  u(:,k) = mpresrootinv*q(:,ipick);
end
.
.
.
```