15. Transient and dynamic solutions

15.1 Introduction to transient systems: The general linear model equation is extended into the transient domain by including the first time derivative of the solution:

\[ -D(x, t) \frac{\partial^2 u(x, t)}{\partial x^2} + A(x, t) \frac{\partial u(x, t)}{\partial x} + C(x, t) u(x, t) + F(x, t) = G(x, t) \frac{\partial u(x, t)}{\partial t} \]  

(15.1-1)

Time dependent scalar field problems solved by using finite elements in space usually lead to a matrix ordinary differential equation in time of the form

\[ [K][T(t)] + [M][\dot{T}(t)] = \{c(t)\}, \quad \dot{\{c\}} = \frac{\partial}{\partial t} \{c\} \]  

(15.1-2)

where the first three terms in the PDE contribute a square conduction matrix to \([K]\), the last term in the PDF contributes the square capacity matrix \([M]\), and the source term \(F\) contributes to the source vector \(\{c(t)\}\). This matrix ODE is subject to the essential boundary conditions and the initial condition (IC), \([T(0)] = T_{IC}\). After the matrix system has been modified to include the EBCs, by adding \(c_{EBC}(t)\) to \(c(t)\), only the time history values of the remaining free degrees of freedom remain to be computed.

Many engineering applications involve sharp transients or nonlinear responses, \([K] = [K([T(t)])]\). Such applications require a direct time integration to obtain the solution time history. Some direct methods require an estimate of the initial time rate of change of the solution. It is often approximated as zero, but it can be found from the initial condition as

\[ \{T(0)\} = [M]^{-1}(\{c(0)\} - [K][T(0)]) \]  

(15.1-3)

where the capacitance matrix, \([M]\), can be approximated by its scaled diagonal.

After any essential boundary conditions are applied the original system will be reduced in size to, say
[\mathbf{K}]\{\mathbf{T}(t)\} + [\mathbf{M}]{\dot{\mathbf{T}}(t)} = \{\mathbf{c}(t)\} - \{\mathbf{c}_{EBC}(t)\} \equiv \{p(t)\} \tag{15.1-4}

where \{\mathbf{c}_{EBC}(t)\} comes from any nonzero EBC multiplying the columns of the larger original matrices. There are hundreds of algorithms for integrating this system in time. These include using finite differences in time, finite elements in time, or even space-time finite elements. Since many engineering applications involve discontinuous changes at certain points in time single step methods are more likely to capture and respond accurately to the discontinuity. If it is known that the driving source terms \{\mathbf{c}(t)\}, and the EBC, \{\mathbf{c}_{EBC}(t)\}, are continuous in time then multi-step methods, like the Runge-Kutta four step method, become more efficient.

Most temporal integration schemes use finite differences in time or finite elements in time to derive the algorithm using constant increments of time, say \(\Delta t\), to define time as \(t_n = t_0 + n \Delta t\) at time step number \(n\). Most of the algorithms form a static linear system to be solved at each time step as

\[
[S(\Delta t, \beta, \mathbf{K}, \mathbf{M})]\{\mathbf{T}\}_n = \{\mathbf{F}(\beta, t_n, t_{n-1}, \mathbf{K}, \mathbf{M}, \mathbf{c}, \{\mathbf{T}\}_{n-1})\} \tag{15.1-5}
\]

where \(\beta\) is an algorithm constant. In this form, the left square matrix is constant if it is not non-linear and if the time step, \(\Delta t\), is constant. Assembling the linear system matrix \([S]\) is computationally expensive and obtaining its inverse or factorization is very expensive. Thus, it is desirable to do those operations only once or twice. The right hand column changes with every time step and the result at time step \(n\) is

\[
\{\mathbf{T}\}_n = [S(\Delta t, \beta, \mathbf{K}, \mathbf{M})]^{-1}\{\mathbf{F}\}_n. \tag{15.1-6}
\]

15.2 Generalized trapezoidal algorithms: The popular generalized trapezoidal one-step method in time leads to an updated square matrix defined as

\[
[S] = ([\mathbf{M}]/\Delta t + \beta[\mathbf{K}]) \tag{15.2-1}
\]

and the source vector updated at every time step is

\[
\{\mathbf{f}\}_n = ([\mathbf{M}]/\Delta t + (\beta - 1)[\mathbf{K}])\{\mathbf{T}\}_{n-1} + (1 - \beta)\{\mathbf{p}\}_{n-1} + \beta\{\mathbf{p}\}_n \tag{15.2-2}
\]

These combine to give a linear matrix system to be solved at every time step:

\[
[S]\{\mathbf{T}\}_n = \{\mathbf{f}\}_n \tag{15.2-3}
\]

Note that so long as the time step \(\Delta t\) is kept constant the computationally expensive assembly and factorization (or inversion) of the linear system square matrix \([S]\) only needs to be done once. This contrasts to the source vector \(\{\mathbf{f}\}_n\) which must be updated at every time step. The forward- and backward-substitution of \(\{\mathbf{f}\}_n\) into a factorization of \([S]\) is relatively cheap and fast. The time step is usually smallest at the start of the transient solution. As a steady state solution is approached a large time step could be used, but often is not used. If the time step is changed, then the last solution from the prior small time step is used as the initial condition for the larger time step.
The common choices for the algorithm constant $\beta$ are:

- $\beta = 0$ the conditionally stable forward difference (Euler) method
- $\beta = \frac{1}{2}$ the unconditionally stable trapezoidal (Crank-Nicolson) method
- $\beta = \frac{2}{3}$ the unconditionally stable Galerkin-in-time method
- $\beta = 1$ the unconditionally stable backward difference method.

The above four algorithms are all single step methods since they only require storing the results from the last time step. They minimize the memory requirements and were developed when computers had extremely small amounts of memory, but their main asset is that they deal well with sudden changes in the source term or the essential boundary conditions. There are many other algorithms that lead to a linear system to be solved at each time step.

When the solution in time is known to be smooth then multiple-step methods can lead to more efficient transient solutions. Of course, they require more memory in order to save the solution from each of the time steps needed to formulate the multi-step algorithm.

### 15.3 One-dimensional Transient Responses

The transient one-dimensional heat conduction equation for the temperature, $T(x,t)$, is

$$ K A \frac{\partial^2 T}{\partial x^2} = \rho c_p A \frac{\partial T}{\partial t} \quad (15.3-1) $$

which is subject to the initial temperature distribution $T(x,0) = T_0(x)$ and essential boundary conditions like $T(0,t) = V_0(t)$ and/or $T(L,t) = V_L(t)$ at the ends of the domain. For an element with constant properties the conduction matrix is linear

$$ S^e = \int_{L^e} dH(r)^T K^e(x) \frac{dH(r)}{dx} A(x) dx = \frac{K^e A^e L^e}{2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \quad (15.3-2) $$

and the element heat capacity matrix is

$$ M^e = \int_{L^e} H(r)^T \rho c_p^e(x) H(r) dx = \frac{\rho c_p^e L^e}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \quad (15.3-3) $$

For a constant volumetric rate of heat generation the element source vector is

$$ c_Q^e = \int_{L^e} H(r)^T Q^e(x) A(x) dx = \frac{Q^e A^e L^e}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad (15.3-4) $$

Consider a system with three equal elements in the domain $L = 3L^e$. Then the assembled matrix system for the three elements, with a constant source, $Q$, gives

$$ [S] \{T(t)\} + [M] \{\dot{T}(t)\} = \{c_Q(t)\} $$

where
\[
[S] = \frac{K^e A^e}{L^e} \begin{bmatrix}
1 & -1 & 0 & 0 \\
-1 & 2 & -1 & 0 \\
0 & -1 & 2 & -1 \\
0 & 0 & -1 & 1 \\
\end{bmatrix}
\]

\[
[M] = \frac{\rho^e c_p^e L^e}{6} \begin{bmatrix}
2 & 1 & 0 & 0 \\
1 & 4 & 1 & 0 \\
0 & 1 & 4 & 1 \\
0 & 0 & 1 & 2 \\
\end{bmatrix}
\]

\[
c_Q = \frac{Q^e A^e L^e}{2} \begin{bmatrix}
1 & 2 & 2 & 1 \\
\end{bmatrix}^T
\]

For the conditionally stable Euler method, \( \beta = 0 \), the finite difference approximation in time is

\[
\{T(t)\} = \frac{\partial T}{\partial t} \approx \frac{T_{t + \Delta t} - T_t}{\Delta t} = \frac{T_{n+1} - T_n}{\Delta t}
\]

so at time \((t + \Delta t)\) the four equations become

\[
[S] \{T_{n+1}\} + [M] \begin{bmatrix}
\frac{T_{n+1} - T_n}{\Delta t} \\
\end{bmatrix} = \{c_Q(t_{n+1})\},
\]

or

\[
[S] \{T_{n+1}\} + [M] \begin{bmatrix}
\frac{T_{n+1} - T_n}{\Delta t} \\
\end{bmatrix} = \{c_Q(t_{n+1})\} + [M] \begin{bmatrix}
\frac{T_n}{\Delta t} \\
\end{bmatrix}
\]

\[
= \frac{Q^e A^e L^e}{2} \begin{bmatrix}
1 \\
2 \\
2 \\
(t + \Delta t) \\
\end{bmatrix} + \frac{\rho^e c_p^e L^e}{6\Delta t} \begin{bmatrix}
2 & 1 & 0 & 0 \\
1 & 4 & 1 & 0 \\
0 & 1 & 4 & 1 \\
0 & 0 & 1 & 2 \\
\end{bmatrix} \begin{bmatrix}
T_1 \\
T_2 \\
T_3 \\
T_4 \\
\end{bmatrix}_{(t + \Delta t)}
\]

(15.3-6)

At the first time step \((n = 0, t = 0)\) the initial condition, say \( T = T_0 \), will be applied to start the time history. Thereafter, the essential boundary conditions, \( T_{EBC}(t + \Delta t) \) will be applied at every time step to the first and last node. Thus, the restrained degree of freedom numbers are \( \text{Fixed} = [1, 4] \), and the other equation numbers, \( \text{Free} = [2, 3] \), are to be solved for at each time step.

If the essential boundary conditions vary with time then \( \frac{\partial T_1}{\partial t} = \frac{\partial T_0}{\partial t} \), and \( \frac{\partial T_4}{\partial t} = \frac{\partial T_L}{\partial t} \). Assume that the Initial Condition is that all nodes are at a temperature of \( T_0 \). Then the free rows and columns at \( t = 0, (n = 0) \), and the effects of the essential boundary conditions (moved to right) give the equations for first step in the time history for the free DOFs:

\[
\begin{bmatrix}
K^e A^e \\
L^e \\
\end{bmatrix} \begin{bmatrix}
2 & -1 \\
-1 & 2 \\
\end{bmatrix} + \frac{\rho^e c_p^e L^e}{6\Delta t} \begin{bmatrix}
4 & 1 \\
1 & 4 \\
\end{bmatrix} \begin{bmatrix}
T_2 \\
T_3 \\
\end{bmatrix}_{(1)}
\]
\[ \frac{\dd V}{\dd t} = \frac{Q^e A^e L^e}{2} \left\{ \begin{array}{c} 2 \\ 2 \end{array} \right\}_{(1)} + \frac{\rho^e c_p^e L^e}{6\Delta t} \left\{ \begin{array}{c} 4 \\ 1 \end{array} \right\}_{(1)} \{ T_0 \}, \quad \{ T_0 \}_{(0)} \] (15.3-7)

\[
-V_0(0) \frac{K^e A^e}{L^e} \left\{ \begin{array}{c} -1 \\ 0 \end{array} \right\} + \frac{\partial V_0(0)}{\partial t} \frac{\rho^e c_p^e L^e}{6\Delta t} \left\{ \begin{array}{c} 1 \\ 0 \end{array} \right\} - V_L(0) \frac{K^e A^e}{L^e} \left\{ \begin{array}{c} 0 \\ -1 \end{array} \right\} + \frac{\partial V_L(0)}{\partial t} \frac{\rho^e c_p^e L^e}{6\Delta t} \left\{ \begin{array}{c} 1 \\ 0 \end{array} \right\}
\]

Multiplying the RHS by the inverse of the square matrix gives the next free values, \( T \) (\textit{Free}).

Note that the initial conditions define a steady state condition so that the initial reactions at nodes 1 and 4 would be recovered as described previously. Now that all four temperatures are known at the end of step \( n = 1 \) the Fixed rows can be optionally used to recover the new reactions at this step. In other words, the reactions are also transient and can be recovered.

The next set of free equations, including the updated EBCs, \( V_0(\Delta t) \) and \( V_L(\Delta t) \), become

\[
\left[ \frac{K^e A^e}{L^e} \left\{ \begin{array}{c} 2 \\ -1 \end{array} \right\} + \frac{\rho^e c_p^e L^e}{6\Delta t} \left\{ \begin{array}{c} 4 \\ 1 \end{array} \right\} \right] \{ T_2 \} = \frac{\dd V}{\dd t} \left\{ \begin{array}{c} 2 \\ 2 \end{array} \right\}_{(2)} + \frac{\rho^e c_p^e L^e}{6\Delta t} \left\{ \begin{array}{c} 4 \\ 1 \end{array} \right\} \{ T_3 \}_{(1)} \quad (15.3-8)
\]

\[
-V_0(\Delta t) \frac{K^e A^e}{L^e} \left\{ \begin{array}{c} -1 \\ 0 \end{array} \right\} + \frac{1}{6} \frac{\partial V_0(\Delta t)}{\partial t} \frac{\rho^e c_p^e L^e}{\Delta t} \left\{ \begin{array}{c} 1 \\ 0 \end{array} \right\} - V_L(\Delta t) \frac{K^e A^e}{L^e} \left\{ \begin{array}{c} 0 \\ -1 \end{array} \right\} + \frac{1}{6} \frac{\partial V_L(\Delta t)}{\partial t} \frac{\rho^e c_p^e L^e}{\Delta t} \left\{ \begin{array}{c} 1 \\ 0 \end{array} \right\}
\]

and so forth until \( n_{\text{max}} \) has been reached.

To present a numerical example of the time stepping process let the initial temperature at all nodes be \( T_0 = 0 \), and thereafter the essential boundary conditions are constant with values of \( V_0 = 10 \) and \( V_L = 20 \), and the material is heat source free (\( Q^e = 0 \)). Then dividing the remaining matrices by \( K^e A^e / L^e \) to define a new constant

\[ \alpha = \frac{\rho^e c_p^e (L^e)^2}{\Delta t K^e A^e} \]

Note that as the time step gets smaller this constant gets bigger. The backward difference method is unconditionally stable but it loses accuracy with each additional time step. You should rerun a study with a smaller time step until the two results are in agreement over the time interval of interest.

The starting step for \( \{ T_2 \}_{(1)} \) for no heat source and constant essential boundary conditions simplifies to

\[
\left[ \begin{array}{c} 2 \\ -1 \end{array} \right] + \alpha \left\{ \begin{array}{c} 4 \\ 1 \end{array} \right\} \{ T_0 \}_{(0)} \} = \{ 0 \} + \alpha \left\{ \begin{array}{c} 4 \\ 1 \end{array} \right\} \{ T_0 \}_{(0)} - V_0 \{ -1 \} + \alpha \{ 0 \} - V_L \{ 0 \} + \alpha \{ 0 \}
\]

and the remaining steps are
\[
\begin{bmatrix}
2 & -1 \\
-1 & 2
\end{bmatrix} + \frac{\alpha}{6} \begin{bmatrix}
4 & 1 \\
1 & 4
\end{bmatrix} \begin{bmatrix}
T_2^{(n+1)} \\
T_3^{(n+1)}
\end{bmatrix}
\]

\[
= \begin{bmatrix}
0 \\
0
\end{bmatrix} + \frac{\alpha}{6} \begin{bmatrix}
4 & 1 \\
1 & 4
\end{bmatrix} \begin{bmatrix}
T_2^{(n)} \\
T_3^{(n)}
\end{bmatrix} - V_0 \begin{bmatrix}
-1 \\
0
\end{bmatrix} - V_L \begin{bmatrix}
0 \\
-1
\end{bmatrix} + \frac{\alpha}{6} \begin{bmatrix}
0 \\
0
\end{bmatrix} + \frac{\alpha}{6} \begin{bmatrix}
0 \\
0
\end{bmatrix}.
\]

Since \( Q_e = 0 \) the previous solutions showed that the steady state solution will be a straight line between the constant essential boundary values of \( V_0 \) and \( V_L \). The results of such a study with \( a = 8 \) and ten time steps are shown in Fig.15.3-1 and the portion of the Matlab script that computed the time histories is given in Fig.15.3-2.

Figure 15.3-Node temperatures for early steps after the Initial Conditions
15.4 Two-dimensional Transient Responses: To illustrate two-dimensional transient responses consider the one-eighth symmetry model of the cross-section of a square electrical conductor. It is initially at a constant temperature. When the current is turned on the rate of heat generation per unit volume becomes a constant \( Q = i R^2 \) where \( i \) is the current flow and \( R \) is the electrical resistance of the material. As shown in Fig. 15.4-1 it is modelled with four linear triangles, T3, and six nodal points. The temperature of the interior three nodes will vary with time until their steady state values are reached. The steady state solution was given as Example 12.14-1.

A typical Matlab script for running the generalized trapezoidal transient integration, `gen_trap_history.m`, and portions of it are given in Fig. 15.4-2a and 2b. The first part of the illustrated calculation allocates the needed memory and extracts the independent DOF. Those steps could actually be done in the calling program. Here they serve as reminder that the presence of essential boundary conditions also influences the transient solution as well. The second part of the calculations is quite compact when done in matrix notation. If the EBC were time dependent, then a minor extension would allow for that requirement. The presence of any EBC over time means that the reactions at those DOF will change with time. That optional calculation is included in the illustrated script.
The output from a transient solution can be listed but that is not very practical; so the time history is usually saved to a file (binary is best) to be post-processed by graphing the time history of selected degrees of freedom, contouring the mesh results, or making mesh carpet plots at selected times, etc. A typical listing of transient temperatures and reaction heat flows is given in Fig. 15.4-3; while time history graphs of the temperature at three DOFs is seen in Fig. 15.4-4. From those time history graphs it is seen that for most of the time steps there are only small changes at a point as the steady state solution is approached.

For two-dimensional solutions carpet plots of the temperatures at one or more time steps can be informative, especially when the number of degrees of freedom reach practical levels. Figure 15.4-5 shows a group of temperature surfaces at different times. There the mesh has been increased to 64 elements. The script, hidden_result_time_1248.m, accepts a beginning time step number (here first = 1) and an integer constant (here mult = 4) that defines the next time step as a multiple of the prior one (here time step numbers 1, 4, 16, 64) to view how the solution changes in the beginning and how it approaches the steady state values.
function [R_u, React, time] = gen_trap_history (S, M, ...  
    C_Q, beta, n_steps, delta_t, ...  
    ic_value, Fixed, bc_value, Free) % == ---------  
% Define generalized trapezoidal time integration data  
% for solution results R(t) from Initial Condition state  
% [beta * S + M / delta_t] * R_new = C_rhs  
% C_rhs = [(beta - 1) S + M / delta_t] * R_old  
% + (1 - beta) C_old + beta C_new, here C_old=C_new  
% beta = 1/2 Crank-Nicolson method, beta = 2/3 Galerkin in time  
% beta = 0 Forward difference, beta = 1 Backward difference  
% n_steps = number of time steps, delta_t = size of time step  
% time = actual time values, R_u = solution history  

n_bc = size (Fixed, 2) ; % number of essential BC  
n_d = size (C_Q, 1) ; % number of system DOF  
n_unkn = n_d - n_bc ; % number of unknowns  
time = [0:1:n_steps] * delta_t ; % actual time values  

% Allocate time history storage, inset initial condition  
S_uu = zeros (n_unkn, n_unkn) ; % allocate conduction  
M_uu = zeros (n_unkn, n_unkn) ; % allocate capacity  
C_Qu = zeros (n_unkn, 1) ; % allocate source  
C_bc= zeros (n_unkn, 1) ; % allocate EBC source  
C_old= zeros (n_unkn, time) ; % allocate time source  
R_u = zeros (n_unkn, n_steps+1) ; % allocate time history  
R_bc = ones (n_bc, 1) * bc_value ; % known BC temperatures  
React= zeros (n_bc, n_steps) ; % optional reaction history  
R_u = zeros (n_unkn, n_steps+1) ; % allocate time history  
R_u(:, 1) = ic_value ; % insert initial condition  

% Extract the unknown equations active in the time history  
% M_uu * R_u_dot + S_uu * R_u = C_Qu(t) + C_bc(t)  
M_uu = M (Free, Free) ; % free capacity  
S_uu = S (Free, Free) ; % free conduction  
S_ue = S (Free, Fixed) ; % coupled to EBC  
S_bb = S (Fixed, Fixed) ; % optional  
C_Qu = C_Q (Free) ; % free source  

% Form matrices for linear system S_hat * R_u(time) = F(time)  
% that will be solved at each time step for R_u  
% [beta * S + M / delta_t] * R_new = C_rhs  
% C_rhs = [(beta - 1) S + M / delta_t] * R_old + C_Qu + C_bc  
S_hat = beta * S_uu + M_uu / delta_t ; % constant in time  
S_inv = inv (S_hat) ; % invert once  

% Square matrix terms for RHS:  
% F = C_Qu + C_bc + S RHS*R_u old  
S_RHS = S_hat - S_uu ; % constant in time  

% Time marching solution, for constant data and delta_t  
R_u(:, 1) = ic_value ; % insert initial condition  
for k = 2:1:n_steps+1 ; % march to next time ---> --- --- --- ---  
% source from prior time step  
C_old = S_RHS * R_u(:, k-1) ; % prior time  

% update the effective source terms  
C_bc = -S_ue * R_bc ; % if EBC time dependent  
F = C_Qu + C_bc + C_old ; % at this time  
% Backsubstitute current effective source for current solution  
R_u(:, k) = S_inv * F ; % answer  
% Recover optional reaction heat flow at EBC equations  
React(:, k) = S_bb * R_bc - C_Q (Fixed) ;  
end % for all time steps <--- <--- <--- <--- <--- <--- <---  
% end gen_trap_history  

Figure 15.4-2a Preparing for a transient integration of the finite element matrices

% Form matrices for linear system S_hat * R_u(time) = F(time)  
% that will be solved at each time step for R_u  
% [beta * S + M / delta_t] * R_new = C_rhs  
% C_rhs = [(beta - 1) S + M / delta_t] * R_old + C_Qu + C_bc  
S_hat = beta * S_uu + M_uu / delta_t ; % constant in time  
S_inv = inv (S_hat) ; % invert once  

% Square matrix terms for RHS:  
% F = C_Qu + C_bc + S RHS*R_u old  
S_RHS = S_hat - S_uu ; % constant in time  

% Time marching solution, for constant data and delta_t  
R_u(:, 1) = ic_value ; % insert initial condition  
for k = 2:1:n_steps+1 ; % march to next time ---> --- --- --- ---  
% source from prior time step  
C_old = S_RHS * R_u(:, k-1) ; % prior time  

% update the effective source terms  
C_bc = -S_ue * R_bc ; % if EBC time dependent  
F = C_Qu + C_bc + C_old ; % at this time  
% Backsubstitute current effective source for current solution  
R_u(:, k) = S_inv * F ; % answer  
% Recover optional reaction heat flow at EBC equations  
React(:, k) = S_bb * R_bc - C_Q (Fixed) ;  
end % for all time steps <--- <--- <--- <--- <--- <--- <---  
% end gen_trap_history  

Figure 15.4-2b Time stepping the independent DOF and recovering the reactions
% Time history solution for 1/8 symmetry square with
% constant heat generation Q=6 to steady state
% connections: (1) 1 2 3  Kr=Kc=8
% (2) 2 4 5  right \ KBC=5
% (3) 3 5 6  right L=4
% (4) 5 3 2
% ic_value = 5, bc_value = 5, beta = 0.5

Fixed - [ 4 5  6]; Free - [ 1  2  3]

Current time step = 2.50e-02

<table>
<thead>
<tr>
<th>Step</th>
<th>Time</th>
<th>Solution at DOF 1 2 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.000e+00</td>
<td>5.000e+00 5.000e+00 5.000e+00</td>
</tr>
<tr>
<td>2</td>
<td>2.500e-02</td>
<td>5.115e+00 5.209e+00 5.195e+00</td>
</tr>
<tr>
<td>3</td>
<td>5.000e-02</td>
<td>5.273e+00 5.379e+00 5.356e+00</td>
</tr>
<tr>
<td>4</td>
<td>7.500e-02</td>
<td>5.429e+00 5.540e+00 5.494e+00</td>
</tr>
<tr>
<td>5</td>
<td>1.000e-01</td>
<td>5.639e+00 5.682e+00 5.615e+00</td>
</tr>
<tr>
<td>6</td>
<td>1.250e-01</td>
<td>5.821e+00 5.826e+00 5.722e+00</td>
</tr>
<tr>
<td>7</td>
<td>1.500e-01</td>
<td>6.005e+00 5.953e+00 5.819e+00</td>
</tr>
<tr>
<td>8</td>
<td>1.750e-01</td>
<td>6.177e+00 6.072e+00 5.908e+00</td>
</tr>
<tr>
<td>9</td>
<td>2.000e-01</td>
<td>6.342e+00 6.182e+00 5.990e+00</td>
</tr>
<tr>
<td>10</td>
<td>2.250e-01</td>
<td>6.497e+00 6.285e+00 6.065e+00</td>
</tr>
</tbody>
</table>

... 76, 1.875e+00 | 8.724e+00 7.733e+00 7.119e+00 7.119e+00 7.119e+00
77, 1.900e+00 | 8.725e+00 7.734e+00 7.119e+00 7.119e+00 7.119e+00
78, 1.925e+00 | 8.727e+00 7.735e+00 7.119e+00 7.119e+00 7.119e+00
79, 1.950e+00 | 8.729e+00 7.736e+00 7.119e+00 7.119e+00 7.119e+00
80, 1.975e+00 | 8.733e+00 7.737e+00 7.119e+00 7.119e+00 7.119e+00

NOTE: steady state 8.7500 7.7500 7.1250

<table>
<thead>
<tr>
<th>Step</th>
<th>Time</th>
<th>Reactions at DOF 4 5 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.000e+00</td>
<td>0.000e+00 0.000e+00 0.000e+00</td>
</tr>
<tr>
<td>2</td>
<td>2.500e-02</td>
<td>-4.803e+00 -1.380e+00 -4.000e+00</td>
</tr>
<tr>
<td>3</td>
<td>5.000e-02</td>
<td>-5.516e+00 -1.485e+00 -4.000e+00</td>
</tr>
<tr>
<td>4</td>
<td>7.500e-02</td>
<td>-6.163e+00 -1.595e+00 -4.000e+00</td>
</tr>
<tr>
<td>5</td>
<td>1.000e-01</td>
<td>-6.756e+00 -1.692e+00 -4.000e+00</td>
</tr>
<tr>
<td>6</td>
<td>1.250e-01</td>
<td>-7.305e+00 -1.778e+00 -4.000e+00</td>
</tr>
<tr>
<td>7</td>
<td>1.500e-01</td>
<td>-7.814e+00 -1.855e+00 -4.000e+00</td>
</tr>
<tr>
<td>8</td>
<td>1.750e-01</td>
<td>-8.288e+00 -1.926e+00 -4.000e+00</td>
</tr>
<tr>
<td>9</td>
<td>2.000e-01</td>
<td>-8.729e+00 -1.992e+00 -4.000e+00</td>
</tr>
<tr>
<td>10</td>
<td>2.250e-01</td>
<td>-9.141e+00 -2.052e+00 -4.000e+00</td>
</tr>
</tbody>
</table>

... 76, 1.875e+00 | -1.493e+01 -2.890e+01 -4.000e+00
77, 1.900e+00 | -1.493e+01 -2.890e+01 -4.000e+00
78, 1.925e+00 | -1.491e+01 -2.891e+01 -4.000e+00
79, 1.950e+00 | -1.494e+01 -2.892e+01 -4.000e+00
80, 1.975e+00 | -1.494e+01 -2.892e+01 -4.000e+00

Steady state -15.000 -29.000 -4.0000

Figure 15.4-3 Nodal and reaction time histories for square with heat generation
Figure 15.4-4 Transient history of a square with heat generation

Figure 15.4-5 Temperature surfaces at specified time steps
Example 15.4-1 Given: Prior example 12.13-1 gave the steady state solution for a conducting square area with a uniform rate of internal heat generation. Use the Crank-Nicolson method to compute the early transient part of that solution. Solution: The \(6 \times 6\) conduction matrix, \([S]\), and the \(6 \times 1\) source vector, \([c]\), were developed in that example. The transient solution requires the assembled element capacity matrix (from Table 12-1):

\[
\mathbf{m}^e = \int_{A^e} \mathbf{H}^T \rho c_p \mathbf{H} \, dA = \frac{\rho^e c_p^e A^e}{12} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}
\]

Setting the material properties \(\rho c_p = 1\), setting the initial condition at all nodes at \(T = 5\), and using the same essential boundary conditions as the steady state case the time stepping algorithm in Fig. 15.4-1 gives the time history for the three free dof shown in Fig. 15.4-3. Clearly, all of the free dof have nearly reached their steady state values after 2 seconds.

The reactions at the EBC also vary with time and since they depend on the temperature gradient they more quickly approach their steady state values. The early details for the time histories of the temperatures and the reactions are listed in Fig. 15.4-4.

15.5 Accuracy and control: The much older finite difference method for spatial approximations leads to the capacitance, \([M]\), being diagonal while the consistent finite element spatial form of \([M]\) is a full element matrix. The finite element system matrix is still sparse and diagonally dominant, but not diagonal. More recent studies recommend using the average of \([M]\) and its scaled diagonal.

For a given time step the temperature response depends on the system non-dimensional Fourier number, \(Fo \equiv k \Delta t / \rho c L^2\), which represents the ratio of heat transfer by conduction through a distance of \(L\), in a volume of \(L^3\), to the rate of energy storage in the same volume. The larger the Fourier number the larger the distance of penetration of a temperature change into a solid in a given time due to a surface thermal shock (sudden change in surface temperature). A large value of the Fourier number indicates a faster propagation of heat through a body.

Transient finite element solutions are often validated by comparing the time history to analytical solutions where a surface temperature changes instantly from its initial condition (IC) value to a different value set by an essential boundary condition (EBC), or where a sudden heat flow into the surface occurs. There are physical problems of that sort, like a surface hit with a powerful laser beam, but if the solution at \(t = 0^+\) is very important the problem should be formulated with non-Fourier heat transfer which includes the second time derivative of the temperature.

Improper meshing and/or improper time step selection can lead to impossible physical temperatures in the first few time steps at locations just inside a surface subject to a thermal shock. Some authors say such a response is due to the consistent formulation of the capacitance matrix and the diagonal (finite difference) form should be used instead. If those early near surface temperatures are important, then build the proper model and solve it in the consistent way. The diagonal matrix results always look physically possible and most analysts use it Consider a thermal shock where a temperature is suddenly raised on the surface of a body. The temperature along a line normal to that surface is sketched in Fig. 15.5-1. There are analytic solutions for the transient temperature along a line normal to the surface. They start with a very
sharp temperature gradient that decreases as time proceeds. In any element, the element temperature value is like a least square fit over the element domain to the exact value. In the direction normal to the surface there is a very narrow region of changed temperature. If the finite element lengths at and near the boundary is too large in comparison to the temperature change region then the calculation likely will give a physically impossible temperature decrease when heat is added to the surface. This response is caused by the time step being too small in relation to the mesh size and the material properties. The non-dimensional Fourier number for the system is too small and the remedy is to decrease the size of the elements in the direction normal to the shocked surface and or to increase the time step size. In other words, the user has not built a consistent mesh and/or time step for the physical problem being modelled. Diagonal capacitance matrices are in part popular because they can hide poor and/or inexperienced modelling skills and assure what always looks like a physically correct solution.

Choosing the time step size affects the stability of the time history and the presence of oscillations in the solution. The Crank-Nicolson method is unconditionally stable (the solution will not blow up) but it still can oscillate wildly about the true solution. There are mathematical limits on the time step size needed to be stable and to avoid oscillations. When the parameter \( \beta < 0.5 \), the time step must be less than the critical value of \( \Delta t_{\text{critical}} = 2 / [(1 - 2\beta) \lambda_{\text{max}}] \) in order to obtain a stable time history. There the eigenvalue is the maximum found from \( [[S] - \lambda[M]] = 0 \) which can be bounded by the maximum of the element eigenvalues \( [[S]^e - \lambda^e[M]^e] = 0 \) since it has been shown by Iron’s Theorem that \( \lambda_{\text{min}} \leq \lambda_{\text{min}} \leq \lambda_{\text{max}} \leq \lambda_{\text{max}}^e \).

**Figure 15.5-1 Thermal shock requires both mesh and time step control**

### Example 15.5-1 Given:
A transient heat transfer analysis is to be conducted using linear two-node line elements in a uniform mesh. Estimate the critical time step using the eigenvalue from a single element. **Solution:** The consistent element matrices give \( [[S]^e - \lambda^e[M]^e] = 0 \), or

\[
\begin{bmatrix}
\frac{\rho^e c^e L^e}{6} & 2 & 1 \\
0 & 1 & 2
\end{bmatrix} - \lambda^e \begin{bmatrix}
\rho^e c^e L^e \\
3 & 0 & 3 \\
0 & 3
\end{bmatrix} = 0
\]

The capacitance matrix, \( [M]^e \), can be diagonalized by scaling its original diagonal sum \( (4/6) \) up to match the total sum \( (6/6) \):

\[
\frac{\rho^e c^e L^e}{6} \begin{bmatrix}
2 & 1 \\
1 & 2
\end{bmatrix} \Rightarrow \frac{\rho^e c^e L^e}{6} \begin{bmatrix}
3 & 0 \\
0 & 3
\end{bmatrix} = \frac{\rho^e c^e L^e}{2} \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\]
This form prevents physically impossible answers due to poor element sizes and an inconsistent time step size. Solving this small eigen-value problem gives $\lambda_1^2=0$ and the second value for the consistent form is $\lambda_2^2 = 12 \frac{k^e}{\rho^e c^e L^2}$ while the diagonal form is one-third smaller. For the Euler ($\beta = 0$) algorithm of time marching using the diagonal capacitance is

$$\Delta t \leq \Delta t_{\text{critical}} = \frac{\rho^e c^e L^2}{2 k^e}$$

That same time step size will avoid the onset of oscillations in the Crank-Nicolson method.

### 15.6 Introduction to dynamic solutions

In the sections on elasticity and stress analysis in Chapter 13 it was shown that the element stiffness and consistent mass matrices

$$K^e = \int_{\Omega^e} B^e^T E^e B^e \, d\Omega, \quad M^e = \int_{\Omega^e} N^e^T \rho^e N^e \, d\Omega$$

are assembled in exactly the same way to define the system stiffness and mass matrices, $[K]$ and $[M]$. Dynamic structures and wave propagation problems solved by finite elements usually lead to a matrix ordinary hyperbolic differential equation in time of the form

$$[K]\{\delta(t)\} + [D]\{\delta(t)\} + [M]\{\ddot{\delta}(t)\} = \{f(t)\}, \quad \{\} = \partial\{\}/\partial t \quad (15.6-1)$$

where $[D]$ is the system damping matrix. These equations of motion for the system are subject to essential boundary conditions and the initial conditions (IC), on the displacements $\delta(0) = \delta_{IC}$, and the velocities $\dot{\delta}(0) = \dot{\delta}_{IC}$. Many engineering applications involve sharp transients or nonlinear stiffness responses, $[K] = K\{\delta(t)\}$. Such applications require a direct time integration to obtain the solution time history. Some direct methods require an estimate of the initial acceleration of the solution. It is often approximated as zero, but it can be found from the initial condition as

$$\{\ddot{\delta}(0)\} = [M]^{-1}(\{f(0)\} - [K]\{\delta_{IC}\} - [D]\{\delta_{IC}\}) \quad (15.6-2)$$

where the mass matrix, $[M]$, can be approximated by its scaled diagonal $[M] \approx [M]_{\text{Diagonal}}$.

The damping matrix, $[D]$, definition varies with the application but by far the most common form is a combination of structural damping and mass damping:

$$[D] = \alpha[K] + \beta[M] \quad (15.6-3)$$

where the two proportions are usually related to the critical dampening, $\xi$, and a frequency range, $\omega_1 < \omega_2$ of interest:

$$\alpha = 2(\xi_2 \omega_2 - \xi_1 \omega_1)/(\omega_2^2 - \omega_1^2)$$

$$\beta = 2\omega_1\omega_2(\xi_1 \omega_2 - \xi_2 \omega_1)/(\omega_2^2 - \omega_1^2). \quad (15.6-5)$$

After any essential boundary conditions are applied the original system will be reduced in size to, say

$$[K]\{\delta(t)\} + [D]\{\dot{\delta}(t)\} + [M]\{\ddot{\delta}(t)\} = \{p(t)\} \quad (15.6-6)$$
where \( \{p(t)\} \) includes the original time dependent loads, \( f(t) \), plus the effects of the (constant or) time dependent essential boundary conditions.

At this point there are many algorithms for integrating this system in time. Most temporal integration schemes use finite differences in time or finite elements in time to derive the algorithm using constant increments of time, say \( \Delta t \), to define time as \( t_n = (t_0 + n \Delta t) \) at time step number \( n \). Most of the algorithms form a linear system to be solved at each time step as

\[
[S(\Delta t, a_j, K, D, M)]\{\delta\}_n = \left\{f_n(a_j, t_n, t_{n-1}, K, D, M, p, \{\delta\}_{n-1})\right\}
\]

where the \( a_j \) are a set of algorithm constants (usually \( 1 \leq j \leq 8 \)). In this form, the left square matrix is constant if the time step, \( \Delta t \), is constant. Assembling \([K]\) is computationally expensive and obtaining its inverse or factorization is very expensive. Thus, it is desirable to do those operations only once or twice. The right hand column changes with every time step and the result at time step \( n \) is

\[
\{\delta\}_n = [S(\Delta t, a_j, K, D, M)]^{-1} \{f\}_n.
\]

For example, the popular Newmark Beta method has two constants, \( \beta \) and \( \gamma \), that combine with the time step to define six other constants

\[
a_1 = \frac{1}{\beta \Delta t^2}, \quad a_2 = \frac{1}{\beta \Delta t}, \quad a_3 = \frac{1}{2\beta} - 1
\]

\[
a_4 = \frac{\gamma}{\beta \Delta t}, \quad a_5 = \frac{\gamma}{\beta} - 1, \quad a_6 = \left(\frac{\gamma}{2\beta} - 1\right)\Delta t
\]

\[
[S] = ([M] + a_4[D] + a_1[K])
\]

\[
\{f\}_n = \{f\}_{n-1} + [K] \left( a_1\{\delta\}_{n-1} + a_2\{\dot{\delta}\}_{n-1} + a_3\{\ddot{\delta}\}_{n-1} \right)
\]

\[
+ [D] \left( a_4\{\delta\}_{n-1} + a_5\{\dot{\delta}\}_{n-1} + a_6\{\ddot{\delta}\}_{n-1} \right)
\]

then the velocity and acceleration are updated as

\[
\{\delta\}_{n+1} = a_4(\{\delta\}_{n+1} - \{\delta\}_n) - a_5\{\dot{\delta}\}_n - a_6\{\ddot{\delta}\}_n
\]

\[
\{\dot{\delta}\}_{n+1} = a_1(\{\delta\}_{n+1} - \{\delta\}_n - \Delta t\{\dot{\delta}\}_n) - a_3\{\ddot{\delta}\}_n
\]

Common four choices for \( \beta \) and \( \gamma \) are:

1. The Newmark average acceleration method; \( \beta = 1/4, \gamma = 1/2 \) is unconditionally stable. The accuracy is \( O(\Delta t^2) \)
2. The linear acceleration method; \( \beta = 1/6, \gamma = 1/2 \) is conditionally stable. The accuracy is \( O(\Delta t^2) \)
3. The Fox–Goodwin method; \( \beta = 1/12, \gamma = 1/2 \) is conditionally stable. The accuracy is \( O(\Delta t^4) \)
4. The Hilber-Hughes-Taylor $\alpha$-method; $\beta = (1 - \alpha)^2/4, \gamma = (1 - 2\alpha)/2$ with the main constant range $-1/3 \leq \alpha \leq 0$ is unconditionally stable. The accuracy is $O(\Delta t^2)$.

15.7 Wilson method: Another direct time history integration algorithm (used here) is the Wilson method. The Wilson method is an extension of the linear acceleration method with $\gamma = 1/2$, as illustrated in Fig. 15.7-1, in a manner that makes it stable. It assumes that the acceleration is linear from time $t$ to $(t + \theta \Delta t)$ with $\theta \geq 1$, and calculates the displacement, velocity, and acceleration there. Then the method interpolates back to find those values at $(t + \Delta t)$. Theoretical studies show it to be unconditionally stable. The accuracy is $O(\Delta t^2)$.

Integrating the linear acceleration assumption gives the predicted velocity and displacement:

\[
\ddot{\delta}(t + \theta \Delta t) = \ddot{\delta}(t) + \left[ \ddot{\delta}(t + \theta \Delta t) + \ddot{\delta}(t) \right]/2
\]

\[
\ddot{\delta}(t + \theta \Delta t) = \ddot{\delta}(t) + \theta \Delta t \ddot{\delta}(t) + (\theta \Delta t)^2 [\ddot{\delta}(t + \theta \Delta t) + 2\ddot{\delta}(t)]/6
\]

and these are re-written in terms of the displacement at the extended time:

\[
\ddot{\delta}(t + \theta \Delta t) = 6[\ddot{\delta}(t + \theta \Delta t) - \ddot{\delta}(t)]/(\theta \Delta t)^2 - \ddot{\delta}(t)/\theta \Delta t - 2\ddot{\delta}(t)
\]

\[
\ddot{\delta}(t + \theta \Delta t) = 3[\ddot{\delta}(t + \theta \Delta t) - \ddot{\delta}(t)]/\theta \Delta t - \theta \Delta t \ddot{\delta}(t)/2
\] (15.7-1)

These estimates are substituted into the equation of motion (15.6-6) to define the matrix equations for solving one time step. In other words, during one step the Wilson method calculates the solution at $(t + \theta \Delta t)$ using the linear system

\[
S(t + \theta \Delta t) \delta(t + \theta \Delta t) = c(t + \theta \Delta t)
\] (15.7-2)

where the effective matrices are

\[
S(t + \theta \Delta t) = 3G/\theta \Delta t + K, \text{ with } G \equiv 2M/\theta \Delta t + D
\]

\[
c(t + \theta \Delta t) = p(t + \theta \Delta t)
\]

\[
+ G \left\{ 3 \ddot{\delta}(t)/\theta \Delta t + 3 \ddot{\delta}(t) + \theta \Delta t \ddot{\delta}(t) \right\}
\]

\[
- D \left\{ \ddot{\delta}(t) + \theta \Delta t \ddot{\delta}(t)/2 \right\}
\] (15.7-3)

Equation (15.7-2) is solved once per time step for the new displacement, $\delta(t + \theta \Delta t)$, and then the method interpolates backward to find the values at $(t + \Delta t)$ as

\[
\ddot{\delta}(t + \Delta t) = \ddot{\delta}(t) + \{\ddot{\delta}(t + \theta \Delta t) - \ddot{\delta}(t)\}/\theta
\]

\[
\ddot{\delta}(t + \Delta t) = \ddot{\delta}(t) + \theta \Delta t \ddot{\delta}(t) + \theta \Delta t \left\{ \ddot{\delta}(t + \theta \Delta t) - \ddot{\delta}(t) \right\}/2\theta
\]

\[
\delta(t + \Delta t) = \delta(t) + \theta \Delta t \ddot{\delta}(t)
\] (15.7-4)

which are then used to start the next integration step. The Wilson algorithm is implemented in the Matlab script Wilson_time_history.m.
The above algorithms are all single step methods since they only require storing the results from the last time step. They minimize the memory requirements and were developed when computers had extremely small memory, but their main asset is that they deal well with sudden changes in the solution such as earthquakes imposing rapid changes in the displacements, or external impact forces. There are many other algorithms that lead to a linear system to be solved at each time step.

![Image](image1.png)

**Figure 15.7-1 The Wilson method extends the linear acceleration step**

It is difficult to debug and gain experience with these, and other, direct time history integration methods. Exact solutions are rare, except for single DOF models. Biggs, in *Introduction to Structural Dynamics*, gave the exact analytic time history for a three DOF spring-mass system with time varying loads shown in Fig. 15.7-2. The exact time history solution given by Biggs consists of two segments. The first segment covers the time when each load is decreasing to zero. The values of the displacement and velocities at the end of that time are used as initial conditions to continue the solution as a free vibration response. That example solution for the displacements, velocities, and accelerations was also computed by the Wilson method included in the provided script. Using a smaller time step leads to a more accurate solution, but requires a proportional longer execution time.

![Image](image2.png)

**Figure 15.7-2 The Biggs’ analytic time history spring-mass system**

For example, using the Wilson method to solve the Biggs model with a time step of 0.00125 sec. gives the displacement and velocity curves compared to the exact results (dashed) in Fig.
15.7-3. Those results are fairly accurate, but doubling the time step, in Fig. 15.7-4, to cut the run time in half significantly increases the errors in the results.

Figure 15.7-3 Biggs’ model displacement (top) and velocity from $\Delta t = 0.00125 \text{ sec}$
Figure 15.7-4 Biggs' model displacement (top) and velocity from $\Delta t = 0.0025$ sec
15.8 Summary:

15.8.1 Transient Summary: Scalar field model ordinary differential equation:

\[
\frac{\partial}{\partial x} \left( k_x \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left( k_y \frac{\partial u}{\partial y} \right) + T u + Q(x, y) = \rho c_p \frac{\partial u}{\partial \tau}
\]

Equivalent integral form:

\[
\int_G u \left( k_{nn} \frac{\partial u}{\partial n} \right) d\Gamma - \int_G \left( \frac{\partial u}{\partial x} (k_x \frac{\partial u}{\partial x}) + \frac{\partial u}{\partial y} (k_y \frac{\partial u}{\partial y}) \right) d\Omega - \int_G u \frac{\partial u}{\partial \tau} d\Omega - \int_G u \frac{\partial u}{\partial \tau} d\Omega = 0
\]

Conduction, convection, advection, and source matrices:

\[
S_k^e = \int_{\Omega^e} \vec{\nabla}H^T \kappa \vec{\nabla}H d\Omega, \quad A_v^e = \int_{\Omega^e} H^{eT} m^e v^e B^e d\Omega
\]

\[
M_h^e = \int_{\Omega^e} H^{eT} h^e H^e d\Omega, \quad M_p^e = \int_{\Omega^e} H^{eT} \rho^e c_p^e H^e d\Omega,
\]

\[
c_Q^e = \int_{\Omega^e} H^{eT} Q^e d\Omega, \quad c_{NBC}^b = \int_{\Gamma^b} H^b T \left( k_{nn} \frac{\partial u}{\partial n} \right) d\Gamma, \quad c_q^b = \int_{\Gamma^b} H^b T q_n d\Gamma
\]

Transient matrix system:

\[
[S_k^e + A_v^e + M_h^e]u(t) + [M_p^e] \frac{\partial u}{\partial \tau} = c(t)
\]

Steady state matrix system, \( \frac{\partial u}{\partial \tau} = 0 \): \( [S_k^e + A_v^e + M_h^e]u = c \)

Linear triangle conduction and volumetric source matrices:

\[
S_k^e = \frac{t^e}{4 A^e} \begin{bmatrix}
    k_x^e \begin{bmatrix}
        b_1 b_1 & b_1 b_2 & b_1 b_3 \\
        b_2 b_1 & b_2 b_2 & b_2 b_3 \\
        b_3 b_1 & b_3 b_2 & b_3 b_3
    \end{bmatrix} + k_y^e \begin{bmatrix}
        c_1 c_1 & c_1 c_2 & c_1 c_3 \\
        c_2 c_1 & c_2 c_2 & c_2 c_3 \\
        c_3 c_1 & c_3 c_2 & c_3 c_3
    \end{bmatrix}
\end{bmatrix}, \quad c_Q^e = \frac{Q^e A^e t^e}{3} \begin{bmatrix}
    1 \\
    1 \\
    1
\end{bmatrix}
\]

Linear triangle face and edge convection matrices (and normal flux matrices, \( h^e \rightarrow q_n^e \)):

\[
M_h^e = \frac{h^e A^e}{12} \begin{bmatrix}
    2 & 1 & 1 \\
    1 & 2 & 1 \\
    1 & 1 & 2
\end{bmatrix}, \quad c_h^b = \frac{h^b t^b L^b}{2} \begin{bmatrix}
    1 \\
    1 \\
    1
\end{bmatrix}
\]
Linear rectangle conduction and volumetric source matrices:

\[
S^e = \frac{k^e t^e l_x^e}{6 l_y^e} \begin{bmatrix}
2 & -2 & -1 & 1 \\
-2 & 2 & 1 & -1 \\
-1 & 1 & 2 & -2 \\
1 & -1 & -2 & 2
\end{bmatrix} + \frac{k^e t^e l_y^e}{6 l_x^e} \begin{bmatrix}
2 & 1 & -1 & -2 \\
1 & 2 & -2 & -1 \\
-1 & -2 & 2 & 1 \\
-2 & -1 & 1 & 2
\end{bmatrix}, \quad c_Q^e = \frac{Q^e A^e t^e}{4} \begin{bmatrix} 1 \\
1 \\
\end{bmatrix}
\]

Linear rectangle face and edge convection matrices (and normal flux matrices, \( h^e \rightarrow q^e_h \)):

\[
M^e_h = \frac{h^e l_x^e l_y^e}{36} \begin{bmatrix}
4 & 2 & 1 & 2 \\
2 & 4 & 2 & 1 \\
1 & 2 & 4 & 2 \\
2 & 1 & 2 & 4
\end{bmatrix}, \quad c_h^e = \frac{h^e b^e l_x^e}{2} \begin{bmatrix} 1 \\
\end{bmatrix}
\]

15.8.2 Dynamic Summary

System equation of motion: \([K]\{\delta(t)\} + [D]\{\ddot{\delta}(t)\} + [M]\{\dddot{\delta}(t)\} = \{f(t)\}, \quad \{\} = \partial\{\} / \partial t

\([K] = \) system stiffness matrix, \([M] = \) system mass matrix (consistent, diagonal, or averaged)

\([D] = \alpha[K] + \beta[M] = \) damping matrix for critical dampening, \(\xi\), and frequency range,
\(\omega_1 < \omega_2 \) with \(\alpha = 2(\xi_2 \omega_2 - \xi_1 \omega_1) / (\omega_2^2 - \omega_1^2)\), \(\beta = 2 \omega_1 \omega_2 (\xi_1 \omega_2 - \xi_2 \omega_1) / (\omega_2^2 - \omega_1^2)\)

Element stiffness matrix:

\(K^e = \int_{\Omega^e} B^e T^e B^e d\Omega\)

Element consistent mass matrix:

\(M^e = \int_{\Omega^e} N^e T^e \rho^e N^e d\Omega\)

Displacement interpolation: \(u(x) = N(r, s) \delta^e\)

Linear system solved at each time step: with algorithm constants \(a_j\)

\([S(\Delta t, a_j, K, D, M)]\{\delta\}_n = \{f\}_n (a_j, t_n, t_{n-1}, K, D, M, p, \{\delta\}_{n-1})\]

\(\{\delta\}_n = [S(\Delta t, a_j, K, D, M)]^{-1} \{f\}_n\)

Commonly

\(S = ([M] + a_4 [D] + a_1 [K])\)

\(\{f\}_n = \{f\}_{n-1} + [K] (a_1 \{\delta\}_{n-1} + a_2 \{\ddot{\delta}\}_{n-1} + a_3 \{\dddot{\delta}\}_{n-1}) + [D] (a_4 \{\delta\}_{n-1} + a_5 \{\dot{\delta}\}_{n-1} + a_6 \{\ddot{\delta}\}_{n-1})\)

\(\{\dot{\delta}\}_{n+1} = a_4 (\{\delta\}_{n+1} - \{\delta\}_n) - a_5 \{\dot{\delta}\}_n - a_6 \{\ddot{\delta}\}_n\)

\(\{\ddot{\delta}\}_{n+1} = a_1 (\{\delta\}_{n+1} - \{\delta\}_n - \Delta t \{\dot{\delta}\}_n) - a_3 \{\dddot{\delta}\}_n\)

Linear bar stiffness and mass (or convection) matrices:

\(K^e = \frac{E^e A^e}{L^e} \begin{bmatrix} 1 & -1 \\
-1 & 1
\end{bmatrix}, \quad m^e = \frac{\rho^e A^e L^e}{6} \begin{bmatrix} 2 \\
1
\end{bmatrix}\)
Linear bar linear line load resultant:
\[ c^e_f = \frac{1}{6L^e} \begin{bmatrix} 2 \\ 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \]

Quadratic bar stiffness, and mass (or convection) matrices:
\[
K^e = \frac{E^e A^e}{3L^e} \begin{bmatrix}
7 & -8 & 1 \\
-8 & 16 & -8 \\
1 & -8 & 7
\end{bmatrix},
\]
\[
m^e = \frac{\rho^e A^e L^e}{30} \begin{bmatrix}
4 & 2 & -1 \\
2 & 16 & 2 \\
-1 & 2 & 4
\end{bmatrix}
\]

Cubic bar stiffness and mass matrices:
\[
K^e = \frac{E^e A^e}{40L^e} \begin{bmatrix}
148 & -189 & 54 & -13 \\
-189 & 432 & -297 & 54 \\
54 & -297 & 432 & -189 \\
-13 & 54 & -189 & 148
\end{bmatrix},
\]
\[
m^e = \frac{\rho^e A^e L^e}{1680} \begin{bmatrix}
128 & 99 & -36 & -19 \\
99 & 648 & -81 & -36 \\
-36 & -81 & 648 & 99 \\
19 & -36 & 99 & 128
\end{bmatrix}
\]

Cubic beam stiffness, and mass matrices:
\[
K^e = \frac{E I^e}{L^3} \begin{bmatrix}
12 & 6L & -12 & 6L \\
6L & 4L^2 & -6L & 2L^2 \\
-12 & -6L & 12 & -6L \\
6L & 2L^2 & -6L & 4L^2
\end{bmatrix},
\]
\[
m^e = \frac{\rho^e A^e L^e}{420} \begin{bmatrix}
156 & 22L & 54 & -13L \\
22L & 4L^2 & 13L & -3L^2 \\
54 & 13L & 156 & -22L \\
-13L & -3L^2 & -22L & 4L^2
\end{bmatrix}
\]
Cubic beam linear line load resultant:  
\[ c_f^e = \frac{L}{60}  \begin{bmatrix} 21 & 9 \\ 3L & 2L \\ 9 & 21 \\ -2L & -3L \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \]

Cubic beam geometric stiffness matrix, for axial tension load N:
\[ K_e^g = \frac{N}{30L}  \begin{bmatrix} 36 & 3L & -36 & 3L \\ 3L & 4L^2 & -3L & -L^2 \\ -36 & -3L & 36 & -3L \\ 3L & -L^2 & -3L & 4L^2 \end{bmatrix} \]

Quintic beam stiffness, and mass matrices:
\[ m^e = \frac{\rho A e L}{13860}  \begin{bmatrix} 2092 & 114 L & 880 & -160 L & 262 & -29 L \\ 114 L & 8 L^2 & 88 L & -12 L^2 & 29 L & -3 L^2 \\ 880 & 88 L & 5632 & 0 & 880 & -88 L \\ -160 L & -12 L^2 & 0 & 128 L^2 & 160 L & -12 L^2 \\ 262 & 29 L & 880 & 160 L & 2092 & -114 L \\ -29 L & -3 L^2 & -88 L & 12 L^2 & -114 L & 8 L^2 \end{bmatrix} \]
\[ m_d^e = \frac{\rho A e L}{1133748}  \begin{bmatrix} 241626 & 0 & 0 & 0 & 0 & 0 \\ 0 & 409 L^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 650496 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6544 L^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 241626 & 0 \\ 0 & 0 & 0 & 0 & 0 & 409 L^2 \end{bmatrix} \]

Quintic beam quadratic line load resultant:  
\[ c_f^a = \frac{L}{420}  \begin{bmatrix} 57 & 44 & -3 \\ 3L & 4L & 0 \\ -8L & 0 & 8L \\ -3 & 44 & 57 \\ 0 & -4L & -3L \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \]
Quintic beam geometric stiffness matrix, for axial tension load N:

\[
K_G^e = \frac{N}{630L} \begin{bmatrix}
1,668 & 39L & -1,536 & 240L & -132 & -9L \\
39L & 28L^2 & -48L & -8L^2 & 9L & -5L^2 \\
-1,536 & -48L & 3,072 & 0 & -1,536 & 48L \\
240L & -8L^2 & 0 & 256L^2 & -240L & -8L^2 \\
-132 & 9L & -1,536 & -240L & 1,668 & -39L \\
\end{bmatrix}
\]

15.7 Exercises:

Index
acceleration update, 15
algorithm constant, 2
average acceleration method, 15
averaged mass matrix, 12
backward difference method, 3
backward-substitution, 2
beam line load resultant, 23
Biggs exact time history, 17
binary file, 8
capacity matrix, 1, 12
carpet plot, 8
conditionally stable, 3
conduction matrix, 1
consistent mass matrix, 12, 14
Crank-Nicolson method, 3, 12, 13
critical dampening, 14, 21
critical time step, 13
cubic bar, 22
cubic beam, 22
damping matrix, 14
decrease element size, 13
diagonal mass matrix, 12
direct time integration, 1, 14
dynamic solutions, 14
dearthquake, 17
eigenvalue, 13
electrical conductor, 7
electrical resistance, 7
external impact force, 17
factorization, 2
finite differences in time, 2
finite elements in time, 2
forward difference method, 3
forward-substitution, 2
Fourier number, 12
Fox-Goodwin method, 15
frequency range, 14, 21
Galerkin-in-time method, 3
gen_trap_history.m, 7
generalized trapezoidal integration, 2, 7
geometric stiffness matrix, 23
heat generation, 7
Hilber-Hughes-Taylor method, 16
impossible temperatures, 12
improper mesh, 12
improper time step, 12
initial condition, 1, 12
internal heat generation, 12
Iron’s Theorem, 13
laser beam, 12
least square fit, 13
linear acceleration method, 15
linear bar, 21
linear matrix system, 2
mass damping, 14
matrix factorization, 2
mesh at shock surface, 13
multiple-step method, 3
Newmark Beta method, 15
non-Fourier heat transfer, 12
one-step method, 2
ordinary differential equation, 1
quadratic bar, 22
quintic beam, 23
Runge-Kutta integration, 2
scalar field problem, 1
scaled diagonal, 1
scaled diagonal mass, 14
sharp transients, 1
single step methods, 3, 17
space-time finite elements, 2
spring-mass system, 17
steady state, 2, 8, 20
structural damping, 14
summary, 20  
temporal integration, 2  
thermal shock, 12  
time dependent EBC, 7  
time dependent loads, 15  
time dependent reactions, 7  
time history, 1, 8  
time history graph, 8  
time oscillations, 13  
time step, 2  
transient analysis, 1  
transient history, 11  
transient matrix system, 20  
velocity update, 15  
wave propagation, 14  
Wilson method, 16