The time dimension – discrete approximation in time

18.1 Introduction

In the last chapter we have shown how semi-discretization of dynamic or transient field problems leads in linear cases to sets of ordinary differential equations of the form

$$\begin{align*}
M\dot{a} + C\dot{a} + K\ddot{a} + f &= 0 \\
\text{where } \frac{da}{dt} &= \dot{a}, \text{ etc.}
\end{align*}$$

(18.1)

subject to initial conditions

$$a(0) = a_0 \quad \text{and} \quad \dot{a}(0) = \dot{a}_0$$

for dynamics or

$$C\dot{a} + Ka + f = 0$$

(18.2)

subject to the initial condition

$$a(0) = a_0$$

for heat transfer or similar problems.

In many practical situations non-linearities exist, typically altering the above equations by making

$$M = M(a) \quad C = C(a) \quad Ka = P(a)$$

(18.3)

The analytical solutions previously discussed, while providing much insight into the behaviour patterns (and indispensable in establishing such properties as natural system frequencies), are in general not economical for the solution of transient problems in linear cases and not applicable when non-linearity exists. In this chapter we shall therefore revert to discretization processes applicable directly to the time domain.

For such discretization the finite element method, including in its definition the finite difference approximation, is of course widely applicable and provides the greatest possibilities, though much of the classical literature on the subject uses
only the latter.\textsuperscript{1–6} We shall demonstrate here how the finite element method provides a useful generalization unifying many existing algorithms and providing a variety of new ones.

As the time domain is infinite we shall inevitably curtail it to a finite time increment $\Delta t$ and relate the initial conditions at $t_n$ (and sometimes before) to those at time $t_{n+1} = t_n + \Delta t$, obtaining so-called recurrence relations. In all of this chapter, the starting point will be that of the semi-discrete equations (18.1) or (18.2), though, of course, the full space-time domain discretization could be considered simultaneously. This, however, usually offers no advantage, for, with the regularity of the time domain, irregular space-time elements are not required. Indeed, if product-type shape functions are chosen, the process will be identical to that obtained by using first semi-discretization in space followed by time discretization. An exception here is provided in convection dominated problems where simultaneous discretization may be desirable, as we shall discuss in the Volume 3.

The first concepts of space-time elements were introduced in 1969–70\textsuperscript{7–10} and the development of processes involving semi-discretization is presented in references 11–20. Full space-time elements are described for convection-type equations in references 21, 22 and 23 and for elastodynamics in references 24, 25 and 26.

The presentation of this chapter will be divided into four parts. In the first we shall derive a set of single-step recurrence relations for the linear first- and second-order problems of Eqs (18.2) and (18.1). Such schemes have a very general applicability and are preferable to multistep schemes described in the second part as the time step can be easily and adaptively varied. In the third part we briefly describe a discontinuous Galerkin scheme and show its application in some simple problems. In the final part we shall deal with generalizations necessary for non-linear problems.

When discussing stability problems we shall often revert to the concept of modally uncoupled equations introduced in the previous chapter. Here we recall that the equation systems (18.1) and (18.2) can be written as a set of scalar equations:

\begin{equation}
 m_i \ddot{y}_i + c_i \dot{y}_i + k_i y_i + f_i = 0 \quad (18.4)
\end{equation}

or

\begin{equation}
 c_i \dot{y}_i + k_i y_i + f_i = 0 \quad (18.5)
\end{equation}

in the respective eigenvalue participation factors $y_i$. We shall find that the stability requirements here are dependent on the eigenvalues associated with such equations, $\omega_i$. It turns out, however, fortunately, that it is never necessary to obtain the system eigenvalues or eigenvectors due to a powerful theorem first stated for finite element problems by Irons and Treharne.\textsuperscript{27}

The theorem states simply that the system eigenvalues can be bounded by the eigenvalues of individual elements $\omega^e$. Thus

\begin{equation}
 \begin{align*}
 \min_j (\omega_j)^2 & \geq \min_e (\omega^e)^2 \\
 \max_j (\omega_j)^2 & \leq \max_e (\omega^e)^2
\end{align*} \quad (18.6)
\end{equation}

The stability limits can thus (as will be shown later) be related to Eqs (18.4) or (18.5) written for a single element.
18.2 Simple time-step algorithms for the first-order equation

18.2.1 Weighted residual finite element approach

We shall now consider Eq. (18.2) which may represent a semi-discrete approximation to a particular physical problem or simply be itself a discrete system. The objective is to obtain an approximation for $a_{n+1}$ given the value of $a_n$ and the forcing vector $f$ acting in the interval of time $\Delta t$. It is clear that in the first interval $a_n$ is the initial condition $a_0$, thus we have an initial value problem. In subsequent time intervals $a_n$ will always be a known quantity determined from the previous step.

In each interval, in the manner used in all finite element approximations, we assume that $a$ varies as a polynomial and take here the lowest (linear) expansion as shown in Fig. 18.1 writing

$$a \approx \hat{a}(t) = a_n + \frac{\tau}{\Delta t}(a_{n+1} - a_n)$$

(18.7)

with $\tau = t - t_n$.

This can be translated to the standard finite element expansion giving

$$\hat{a}(t) = \sum N_i a_i = \left(1 - \frac{\tau}{\Delta t}\right)a_n + \left(\frac{\tau}{\Delta t}\right)a_{n+1}$$

(18.8)

in which the unknown parameter is $a_{n+1}$.

The equation by which this unknown parameter is provided will be a weighted residual approximation to Eq. (18.2). Accordingly, we write the variational problem

$$\int_0^{\Delta t} w(\tau)^T [Ca + Ka + f] \, d\tau = 0$$

(18.9)

in which $w(\tau)$ is an arbitrary weighting function. We write the approximate form

$$w(\tau) = W(\tau) \delta a_{n+1}$$

(18.10)

![Fig. 18.1 Approximation to $a$ in the time domain.](image-url)
in which $\delta a_{n+1}$ is an arbitrary parameter. With this approximation the weighted residual equation to be solved is given by

$$
\int_0^{\Delta t} W(\tau)|C\dot{a} + Ka + f| \, d\tau = 0
$$

(18.11)

Introducing $\theta$ as a weighting parameter given by

$$
\theta = \frac{1}{\Delta t} \int_0^{\Delta t} W(\tau) d\tau
$$

(18.12)

we can immediately write

$$
\frac{1}{\Delta t} C(a_{n+1} - a_n) + K[a_n + \theta(a_{n+1} - a_n)] + \bar{f} = 0
$$

(18.13)

where $\bar{f}$ represents an average value of $f$ given by

$$
\bar{f} = \frac{\int_0^{\Delta t} Wf \, d\tau}{\int_0^{\Delta t} W \, d\tau}
$$

(18.14)

or

$$
\bar{f} = f_n + \theta(f_{n+1} - f_n)
$$

(18.15)

if a linear variation of $f$ is assumed within the time increment.

Equation (18.13) is in fact almost identical to a finite difference approximation to the governing equation (18.2) at time $t_n + \theta \Delta t$, and in this example little advantage is gained by introducing the finite element approximation. However, the averaging of the forcing term is important, as shown in Fig. 18.2, where a constant $W$ (that is $\theta = 1/2$) is used and a finite difference approximation presents difficulties.

Figure 18.3 shows how different weight functions can yield alternate values of the parameter $\theta$. The solution of Eq. (18.13) yields

$$
a_{n+1} = (C + \theta \Delta t K)^{-1}[(C - (1 - \theta) \Delta t K)a_n - \Delta t f]
$$

(18.16)

Fig. 18.2 'Averaging' of the forcing term in the finite-element-time approach.
Simple time-step algorithms for the first-order equation

and it is evident that in general at each step of the computation a full equation system needs to be solved though of course a single inversion is sufficient for linear problems in which the time increment $\Delta t$ is held constant. Methods requiring such an inversion are called explicit. However, when $\theta = 0$ and the matrix $C$ is approximated by its lumped equivalent $C_L$ the solution is called explicit and is exceedingly cheap for each time interval. We shall show later that explicit algorithms are conditionally stable (requiring the $\Delta t$ to be less than some critical value $\Delta t_{\text{crit}}$) whereas implicit methods may be made unconditionally stable for some choices of the parameters.

18.2.2 Taylor series collocation

A frequently used alternative to the algorithm presented above is obtained by approximating separately $a_{n+1}$ and $\dot{a}_{n+1}$ by truncated Taylor series. We can write, assuming
that \( a_n \) and \( \dot{a}_n \) are known:

\[
a_{n+1} \approx a_n + \Delta t \dot{a}_n + \beta \Delta t (\dot{a}_{n+1} - \dot{a}_n)
\]

(18.17)

and use collocation to satisfy the governing equation at \( t_{n+1} \) [or alternatively using the weight function shown in Fig. 18.3(c)]

\[
Ca_{n+1} + Ka_{n+1} + f_{n+1} = 0
\]

(18.18)

In the above \( \beta \) is a parameter, \( 0 \leq \beta \leq 1 \), such that the last term of Eq. (18.17) represents a suitable difference approximation to the truncated expansion.

Substitution of Eq. (18.17) into Eq. (18.18) yields a recurrence relation for \( \dot{a}_{n+1} \):

\[
\dot{a}_{n+1} = -(C + \beta \Delta t K)^{-1} [K(a_n + (1 - \beta) \Delta t \dot{a}_n) + f_{n+1}]
\]

(18.19)

where \( a_{n+1} \) is now computed by substitution of Eq. (18.19) into Eq. (18.17).

We remark that:

(a) the scheme is not self-starting† and requires the satisfaction of Eq. (18.2) at \( t = 0 \);

(b) the computation requires, with identification of the parameters \( \beta = \theta \), an identical equation-solving problem to that in the finite element scheme of Eq. (18.16) and, finally, as we shall see later, stability considerations are identical.

The procedure is introduced here as it has some advantages in non-linear computations which will be shown later.

### 18.2.3 Other single-step procedures

As an alternative to the weighted residual process other possibilities of deriving finite element approximations exist, as discussed in Chapter 3. For instance, variational principles in time could be established and used for the purpose. This was indeed done in the early approaches to finite element approximation using Hamilton’s or Gurtin’s variational principle. However, as expected, the final algorithms turn out to be identical. A variant on the above procedures is the use of a least square approximation for minimization of the equation residual. This is obtained by insertion of the approximation (18.7) into Eq. (18.2). The reader can verify that the recurrence relation becomes

\[
\left( \frac{1}{\Delta t} C^T C + \frac{1}{2} (K^T C + C^T K) + \frac{1}{3} \Delta t K^T K \right) a_{n+1} - \left( \frac{1}{\Delta t} C^T C + \frac{1}{2} (K^T C - C^T K) - \frac{1}{6} \Delta t K^T K \right) a_n
\]

(18.20)

\[
+ \frac{1}{\Delta t^2} C^T \int_0^{\Delta t} f d\tau + \frac{1}{\Delta t} K^T \int_0^{\Delta t} f d\tau
\]

requiring a more complex equation solution and always remaining ‘implicit’. For this reason the algorithm is largely of purely theoretical interest, though as expected its

† By ‘self-starting’ we mean an algorithm is directly applicable without solving any subsidiary equations. Other definitions are also in use.
accuracy is good, as shown in Fig. 18.4, in which a single degree of freedom equation (18.2) is used with

\[
\mathbf{K} \rightarrow \mathbf{K} = 1 \quad \mathbf{C} \rightarrow \mathbf{C} = 1 \quad \mathbf{f} \rightarrow f = 0
\]

with initial condition \(a_0 = 1\). Here, the various algorithms previously discussed are compared. Now we see from this example that the \(\theta = 1/2\) algorithm performs almost as well as the least squares one. It is popular for this reason and is known as the Crank–Nicolson scheme after its originator.\(^{32}\)

### 18.2.4 Consistency and approximation error

For the convergence of any finite element approximation, it is necessary and sufficient that it be consistent and stable. We have discussed these two conditions in Chapter 10 and introduced appropriate requirements for boundary value problems. In the temporal approximation similar conditions apply though the stability problem is more delicate.

Clearly the function \(a\) itself and its derivatives occurring in the equation have to be approximated with a truncation error of \(O(\Delta t^\alpha)\), where \(\alpha \geq 1\) is needed for consistency to be satisfied. For the first-order equation (18.2) it is thus necessary to use an approximating polynomial of order \(p \geq 1\) which is capable of approximating \(a\) to at least \(O(\Delta t)\).

The truncation error in the local approximation of \(a\) with such an approximation is \(O(\Delta t^2)\) and all the algorithms we have presented here using the \(p = 1\) approximation of Eq. (18.7) will have at least that local accuracy,\(^{33}\) as at a given time, \(t = n\Delta t\), the
The time dimension – discrete approximation in time

total error can be magnified \( n \) times and the final accuracy at a given time for schemes discussed here is of order \( O(\Delta t) \) in general.

We shall see later that the arguments used here lead to \( p \geq 2 \) for the second-order equation (18.1) and that an increase of accuracy can generally be achieved by use of higher order approximating polynomials.

It would of course be possible to apply such a polynomial increase to the approximating function (18.7) by adding higher order degrees of freedom. For instance, we could write in place of the original approximation a quadratic expansion:

\[
a \approx \tilde{a}(r) = a_n + \frac{\tau}{\Delta t}(a_{n+1} - a_n) + \frac{\tau}{\Delta t} \left( 1 - \frac{\tau}{\Delta t} \right) \tilde{a}_{n+1}
\]

(18.21)

where \( \tilde{a} \) is a hierarchic internal variable. Obviously now both \( a_{n+1} \) and \( \tilde{a}_{n+1} \) are unknowns and will have to be solved for simultaneously. This is accomplished by using the weighting function

\[
w = W(\tau)\delta a_{n+1} + \tilde{W}(\tau)\delta \tilde{a}_{n+1}
\]

(18.22)

where \( W(\tau) \) and \( \tilde{W}(\tau) \) are two independent weighting functions. This will obviously result in an increased size of the problem.

It is of interest to consider the first of these obtained by using the weighting \( W \) alone in the manner of Eq. (18.11). The reader will easily verify that we now have to add to Eq. (18.13) a term involving \( \tilde{a}_{n+1} \) which is

\[
\left[ \frac{1}{\Delta t} (1 - 2\theta)C + (\theta - \bar{\theta})K \right] \tilde{a}_{n+1}
\]

(18.23)

where

\[
\bar{\theta} = \frac{1}{\Delta t^2} \int_0^{\Delta t} W \tau^2 \, d\tau \int_0^{\Delta t} W \, d\tau
\]

It is clear that the choice of \( \theta = \bar{\theta} = 1/2 \) eliminates the quadratic term and regains the previous scheme, thus showing that the values so obtained have a local truncation error of \( O(\Delta t^3) \). This explains why the Crank–Nicolson scheme possesses higher accuracy.

In general the addition of higher order internal variables makes recurrence schemes too expensive and we shall later show how an increase of accuracy can be more economically achieved.

In a later section of this chapter we shall refer to some currently popular schemes in which often sets of \( a \)'s have to be solved for simultaneously. In such schemes a discontinuity is assumed at the initial condition and additional parameters (\( \tilde{a} \)) can be introduced to keep the same linear conditions we assumed previously. In this case an additional equation appears as a weighted satisfaction of continuity in time.

The procedure is therefore known as the discontinuous Galerkin process and was introduced initially by Lesaint and Raviart \( ^{34} \) to solve neutron transport problems. It has subsequently been applied to solve problems in fluid mechanics and heat transfer \( ^{22,35,36} \) and to problems in structural dynamics. \( ^{24–26} \) As we have already stated, the introduction of additional variables is expensive, so somewhat limited use of the concept has so far been made. However, one interesting application is in error estimation and adaptive time stepping. \( ^{37} \)
18.2.5 Stability

If we consider any of the recurrence algorithms so far derived, we note that for the homogeneous form (i.e., with \( f = 0 \)) all can be written in the form

\[ a_{n+1} = A a_n \]  

(18.24)

where \( A \) is known as the amplification matrix.

The form of this matrix for the first algorithm derived is, for instance, evident from Eq. (18.16) as

\[ A = (C + \theta \Delta tK)^{-1}(C - (1 - \theta)\Delta tK) \]  

(18.25)

Any errors present in the solution will of course be subject to amplification by precisely the same factor.

A general solution of any recurrence scheme can be written as

\[ a_{n+1} = \mu a_n \]  

(18.26)

and by insertion into Eq. (18.24) we observe that \( \mu \) is given by eigenvalues of the matrix as

\[ (A - \mu I)a_n = 0 \]  

(18.27)

Clearly if any eigenvalue \( \mu \) is such that

\[ |\mu| > 1 \]  

(18.28)

all initially small errors will increase without limit and the solution will be unstable. In the case of complex eigenvalues the above is modified to the requirement that the modulus of \( \mu \) satisfies Eq. (18.28).

As the determination of system eigenvalues is a large undertaking it is useful to consider only a scalar equation of the form (18.5) (representing, say, one-element performance). The bounding theorems of Irons and Trehan\textsuperscript{27} will show why we do so and the results will provide general stability bounds if maximums are used. Thus for the case of the algorithm discussed in Eq. (18.27) we have a scalar \( A \), i.e.

\[ A = \frac{c - (1 - \theta)\Delta tK}{c + \theta \Delta tK} = \frac{1 - (1 - \theta)\omega \Delta t}{1 + \theta \omega \Delta t} = \mu \]  

(18.29)

where \( \omega = k/c \) and \( \mu \) is evaluated from Eq. (18.27) simply as \( \mu = A \) to allow non-trivial \( a_n \). (This is equivalent to making the determinant of \( A - \mu I \) zero in the more general case.)

In Fig. 18.5 we show how \( \mu \) (or \( A \)) varies with \( \omega \Delta t \) for various \( \theta \) values. We observe immediately that:

(a) for \( \theta \geq 1/2 \)

\[ |\mu| \leq 1 \]  

(18.30)

and such algorithms are unconditionally stable;

(b) for \( \theta < 1/2 \) we require

\[ \omega \Delta t \leq \frac{2}{1 - 2\theta} \]  

(18.31)

for stability. Such algorithms are therefore only conditionally stable. Here of course the explicit form with \( \theta = 0 \) is typical.
The critical value of $\Delta t$ below which the scheme is stable with $\theta < 1/2$ needs the determination of the maximum value of $\mu$ from a typical element. For instance, in the case of the thermal conduction problem in which we have the coefficients $c_{ii}$ and $k_{ii}$ defined by expressions

$$c_{ii} = \int_{\Omega} \tilde{c} N_i^2 \, d\Omega \quad \text{and} \quad k_{ii} = \int_{\Omega} \nabla N_i \tilde{k} \nabla N_i \, d\Omega \quad (18.32)$$

we can presuppose uniaxial behaviour with a single degree of freedom and write for a linear element

$$N = \frac{h - x}{h} \quad c = \int_0^h \tilde{c} N^2 \, dx = \frac{1}{3} \tilde{c} h \quad k = \int_0^h \tilde{k} \left( \frac{dN}{dx} \right)^2 \, dx = \frac{\tilde{k}}{h}$$

Now

$$\omega = \frac{k}{c} = \frac{3 \tilde{k}}{\tilde{c} h^2}$$

This gives

$$\Delta t \leq \frac{2}{1 - 2\theta} \frac{\tilde{c} h^2}{3 \tilde{k}} = \Delta t_{\text{crit}} \quad (18.33)$$

which of course means that the smallest element size, $h_{\text{min}}$, dictates overall stability. We note from the above that:

(a) in first-order problems the critical time step is proportional to $h^2$ and thus decreases rapidly with element size making explicit computations difficult;

(b) if mass lumping is assumed and therefore $c = \tilde{c} h/2$ the critical time step is larger.

In Fig. 18.6 we show the performance of the scheme described in Sec. 18.2.1 for various values of $\theta$ and $\Delta t$ in the example we have already illustrated in Fig. 18.4, but now using larger values of $\Delta t$. We note now that the conditionally stable scheme with $\theta = 0$ and a stability limit of $\Delta t = 2$ shows oscillations as this limit is approached ($\Delta t = 1.5$) and diverges when exceeded.
Stability computations which were presented for the algorithm of Sec. 18.2.1 can of course be repeated for the other algorithms which we have discussed.

If identical procedures are used, for instance on the algorithm of Sec. 18.2.2, we shall find that the stability conditions, based on the determinant of the amplification matrix \((A - \mu I)\), are identical with the previous one providing we set \(\theta = \beta\). Algorithms that give such identical determinants will be called similar in the following presentations.
In general, it is possible for different amplification matrices $A$ to have identical determinants of $(A - \mu I)$ and hence identical stability conditions, but differ otherwise. If in addition the amplification matrices are the same, the schemes are known as identical. In the two cases described here such an identity can be shown to exist despite different derivations.

18.2.6 Some further remarks. Initial conditions and examples

The question of choosing an optimal value of $\theta$ is not always obvious from theoretical accuracy considerations. In particular with $\theta = 1/2$ oscillations are sometimes present, as we observe in Fig. 18.6 ($\Delta t = 2.5$), and for this reason some prefer to use $\theta = 2/3$, which is considerably ‘smoother’ (and which incidentally corresponds to a standard Galerkin approximation). In Table 18.1 we show the results for a one-dimensional finite element problem where a bar at uniform initial temperature is subject to zero temperatures applied suddenly at the ends. Here 10 linear elements are used in the space dimension with $L = 1$. The oscillation errors occurring with $\theta = 1/2$ are much reduced for $\theta = 2/3$. The time step used here is much longer than that corresponding to the lowest eigenvalue period, but the main cause of the oscillation is in the abrupt discontinuity of the temperature change.

For similar reasons Liniger derives $\theta$ which minimizes the error in the whole time domain and gives $\theta = 0.878$ for the simple one-dimensional case. We observe in Fig. 18.5 how well the amplification factor fits the exact solution with these values. Again this value will smooth out many oscillations. However, most oscillations are introduced by simply using a physically unrealistic initial condition.

In part at least, the oscillations which for instance occur with $\theta = 1/2$ and $\Delta t = 2.5$ (see Fig. 18.6) in the previous example are due to a sudden jump in the forcing term introduced at the start of the computation. This jump is evident if we consider this simple problem posed in the context of the whole time domain. We can take the problem as implying

$$f(t) = -1 \quad \text{for} \quad t < 0$$

<table>
<thead>
<tr>
<th>$\theta$</th>
<th>$x = 0.1$</th>
<th>$x = 0.2$</th>
<th>$x = 0.3$</th>
<th>$x = 0.4$</th>
<th>$x = 0.5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta t = 0.01$</td>
<td>2/3</td>
<td>1/2</td>
<td>2/3</td>
<td>1/2</td>
<td>2/3</td>
</tr>
<tr>
<td>0.01</td>
<td>10.8</td>
<td>28.2</td>
<td>1.6</td>
<td>3.2</td>
<td>0.5</td>
</tr>
<tr>
<td>0.02</td>
<td>0.5</td>
<td>3.5</td>
<td>2.1</td>
<td>9.5</td>
<td>0.1</td>
</tr>
<tr>
<td>0.03</td>
<td>1.3</td>
<td>9.9</td>
<td>0.5</td>
<td>0.7</td>
<td>0.8</td>
</tr>
<tr>
<td>0.05</td>
<td>0.5</td>
<td>4.5</td>
<td>0.4</td>
<td>0.2</td>
<td>0.5</td>
</tr>
<tr>
<td>0.10</td>
<td>0.1</td>
<td>1.4</td>
<td>0.1</td>
<td>2.0</td>
<td>0.1</td>
</tr>
<tr>
<td>0.15</td>
<td>0.3</td>
<td>2.2</td>
<td>0.3</td>
<td>2.1</td>
<td>0.3</td>
</tr>
<tr>
<td>0.20</td>
<td>0.6</td>
<td>2.6</td>
<td>0.6</td>
<td>2.6</td>
<td>0.6</td>
</tr>
<tr>
<td>0.25</td>
<td>1.4</td>
<td>3.5</td>
<td>1.4</td>
<td>3.5</td>
<td>1.4</td>
</tr>
</tbody>
</table>
Simple time-step algorithms for the first-order equation

Fig. 18.7 Importance of 'smoothing' the force term in elimination of oscillations in the solution. $\Delta t = 2.5$.

giving the solution $u = 1$ with a sudden change at $t = 0$, resulting in

$$f(t) = 0 \quad \text{for} \quad t \geq 0$$

As shown in Fig. 18.7 this represents a discontinuity of the loading function at $t = 0$.

Although load discontinuities are permitted by the algorithm they lead to a sudden discontinuity of $\dot{u}$ and hence induce undesirable oscillations. If in place of this discontinuity we assume that $f$ varies linearly in the first time step $\Delta t$ ($-\Delta t/2 \leq t \leq \Delta t/2$) then smooth results are obtained with a much improved physical representation of the true solution, even for such a long time step as $t = 2.5$, as shown in Fig. 18.7.

Similar use of smoothing is illustrated in a multidegree of freedom system (the representation of heat conduction in a wall) which is solved using two-dimensional finite elements$^{40}$ (Fig. 18.8).

Here the problem corresponds to an instantaneous application of prescribed temperature ($T = 1$) at the wall sides with zero initial conditions. Now again troublesome
The time dimension – discrete approximation in time

Fig. 18.8 Transient heating of a bar; comparison of discontinuous and interpolated (smoothed) initial conditions for single-step schemes.

Oscillations are almost eliminated for $\theta = 1/2$ and improved results are obtained for other values of $\theta$ (2/3, 0.878) by assuming the step change to be replaced by a continuous one. Such smoothing is always advisable and a continuous representation of the forcing term is important.

We conclude this section by showing a typical example of temperature distribution in a practical example in which high-order elements are used (Fig. 18.9).
Simple time-step algorithms for the first-order equation

Fig. 18.9 Temperature distribution in a cooled rotor blade, initially at zero temperature.

(a) $t = 0.5\,\text{s}$

(b) $t = 1.0\,\text{s}$

(c) Steady-state solution

Specific heat $c = 0.11\,\text{cal/gm} \, ^\circ\text{C}$
Density $\rho = 7.99\,\text{gm/cm}^3$
Conductivity $k = 0.05\,\text{cal/s cm} \, ^\circ\text{C}$
Gas temperature around blade $= 1145\,^\circ\text{C}$
Heat transfer coefficient $\alpha$ varies from 0.390 to 0.056 on the outside surfaces of the blade (A–B)

<table>
<thead>
<tr>
<th>Hole number</th>
<th>Cooling hole temperature $\alpha$ around perimeter of each hole</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>545 $^\circ\text{C}$</td>
</tr>
<tr>
<td>2</td>
<td>587 $^\circ\text{C}$</td>
</tr>
</tbody>
</table>
18.3 General single-step algorithms for first- and second-order equations

18.3.1 Introduction

We shall introduce in this section two general single-step algorithms applicable to Eq. (18.1):

\[ \mathbf{M} \ddot{\mathbf{a}} + \mathbf{C} \dot{\mathbf{a}} + \mathbf{K} \mathbf{a} + \mathbf{f} = \mathbf{0} \]

These algorithms will of course be applicable to the first-order problem of Eq. (18.2) simply by putting \( \mathbf{M} = \mathbf{0} \).

An arbitrary degree polynomial \( p \) for approximating the unknown function \( \mathbf{a} \) will be used and we must note immediately that for the second-order equations \( p \geq 2 \) is required for consistency as second-order derivatives have to be approximated.

The first algorithm \( \text{SSp}_j \) (single step with approximation of degree \( p \) for equations of order \( j = 1,2 \)) will be derived by use of the weighted residual process and we shall find that the algorithm of Sec. 18.2.1 is but a special case. The second algorithm \( \text{GNp}_j \) (generalized Newmark\(^{41} \) with degree \( p \) and order \( j \)) will follow the procedures using a truncated Taylor series approximation in a manner similar to that described in Sec. 18.2.2.

In what follows we shall assume that at the start of the interval, i.e., at \( t = t_n \), we know the values of the unknown function \( \mathbf{a} \) and its derivatives, that is \( a_n, \dot{a}_n, \ddot{a}_n \) up to \( a_{n-1} \), and our objective will be to determine \( \mathbf{a}_{n+1}, \dot{a}_{n+1}, \ddot{a}_{n+1} \) up to \( a_{n+1} \), where \( p \) is the order of the expansion used in the interval.

This is indeed a rather strong presumption as for first-order problems we have already stated that only a single initial condition, \( \mathbf{a}(0) \), is given and for second-order problems two conditions, \( \mathbf{a}(0) \) and \( \mathbf{a}(0) \), are available (i.e., the initial displacement and velocity of the system). We can, however, argue that if the system starts from rest we could take \( \mathbf{a}(0) \) to \( a_{n-1} \) of \( 0 \) as equal to zero and, providing that suitably continuous forcing of the system occurs, the solution will remain smooth in the higher derivatives. Alternatively, we can differentiate the differential equation to obtain the necessary starting values.

18.3.2 The weighted residual finite element form \( \text{SSp}_j^{18,19} \)

The expansion of the unknown vector \( \mathbf{a} \) will be taken as a polynomial of degree \( p \). With the known values of \( a_n, \dot{a}_n, \ddot{a}_n \) up to \( a_{n-1} \) at the beginning of the time step \( \Delta t \), we write, as in Sec. 18.2.1,

\[ \tau = t - t_n \quad \Delta t = t_{n+1} - t_n \quad (18.34) \]

and using a polynomial expansion of degree \( p \),

\[ \mathbf{a} \approx \hat{\mathbf{a}} = a_n + \tau \dot{a}_n + \tau^2 \ddot{a}_n + \cdots + \frac{1}{(p-1)!} \tau^{p-1} a_{n-1} + \frac{1}{p!} \tau^p a^p_n \quad (18.35) \]
where the only unknown is the vector $a^p_n$,

$$a^p_n \equiv \dot{a} = \frac{d^p}{dt^p} a$$  \hspace{1cm} (18.36)

which represents some average value of the $p$th derivative occurring in the interval $\Delta t$. The approximation to $a$ for the case of $p = 2$ is shown in Fig. 18.10.

We recall that in order to obtain a consistent approximation to all the derivatives that occur in the differential equations (18.1) and (18.2), $p \geq 2$ is necessary for the full dynamic equation and $p \geq 1$ is necessary for the first-order equation. Indeed the lowest approximation, that is $p = 1$, is the basis of the algorithm derived in the previous section.

The recurrence algorithm will now be obtained by inserting $a$, $\dot{a}$ and $\ddot{a}$ obtained by differentiating Eq. (18.35) into Eq. (18.1) and satisfying the weighted residual equation with a single weighting function $W(\tau)$. This gives

$$\int_0^{\Delta t} W(\tau) \left[ M \left( \dddot{a}_n + \tau \ddot{a}_n + \cdots + \frac{1}{(p-2)!} \tau^{p-2} \dot{a}^p_n \right) + C \left( \dddot{a}_n + \tau \ddot{a}_n + \cdots + \frac{1}{(p-1)!} \tau^{p-1} \dot{a}^p_n \right) + K \left( a_n + \tau \dot{a}_n + \cdots + \frac{1}{p!} \tau^p a^p_n \right) + f \right] d\tau = 0$$  \hspace{1cm} (18.37)

as the basic equation for determining $a^p_n$. 

---

**Fig. 18.10** A second-order time approximation.
Without specifying the weighting function used we can, as in Sec. 18.2.1, generalize its effects by writing

\[ \theta_k = \frac{\int_0^{\Delta t} W \tau^k \, d\tau}{\int_0^{\Delta t} W \, d\tau} \quad k = 0, 1, \ldots, p \]  
(18.38)

\[ \bar{f} = \frac{\int_0^{\Delta t} W f \, d\tau}{\int_0^{\Delta t} W \, d\tau} \]

where we note \( \theta_0 \) is always unity. Equation (18.37) can now be written more compactly as

\[ A \alpha_n^p + M \bar{a}_{n+1} + C \hat{a}_{n+1} + K \bar{a}_{n+1} + \bar{f} = 0 \]  
(18.39)

where

\[ A = \frac{\Delta t^{p-2}}{(p-2)!} M + \frac{\Delta t^{p-1}}{(p-1)!} C + \frac{\Delta t^p}{p!} K \]

\[ \bar{a}_{n+1} = \sum_{q=0}^{p-1} \frac{\theta_q \Delta t^q}{q!} \bar{a}_n \]

\[ \hat{a}_{n+1} = \sum_{q=1}^{p-1} \frac{\theta_q \Delta t^{q-1}}{(q-1)!} \hat{a}_n \]  
(18.40)

\[ \bar{a}_{n+1} = \sum_{q=2}^{p-1} \frac{\theta_q \Delta t^{q-2}}{(q-2)!} \bar{a}_n \]

As \( \bar{a}_{n+1}, \hat{a}_{n+1} \) and \( \bar{a}_{n+1} \) can be computed directly from the initial values we can solve Eq. (18.39) to obtain

\[ \alpha_n^p = -A^{-1} [M \bar{a}_{n+1} + C \hat{a}_{n+1} + K \bar{a}_{n+1} + \bar{f}] \]  
(18.41)

It is important to observe that \( \bar{a}_{n+1}, \hat{a}_{n+1} \) and \( \bar{a}_{n+1} \) here represent some mean predicted values of \( a_{n+1}, \hat{a}_{n+1} \) and \( \bar{a}_{n+1} \) in the interval and satisfy the governing Eq. (18.1) in a weighted sense if \( \alpha_n^p \) is chosen as zero.

The procedure is now complete as knowledge of the vector \( \alpha_n^p \) permits the evaluation of \( a_{n+1} \) to \( \alpha_n^p a_{n+1} \) from the expansion originally used in Eq. (18.35) by putting \( \tau = \Delta t \). This gives

\[ a_{n+1} = a_n + \Delta t \hat{a}_n + \cdots + \frac{\Delta t^p}{p!} \alpha_n^p = \hat{a}_{n+1} + \frac{\Delta t^p}{p!} \alpha_n^p \]

\[ \hat{a}_{n+1} = \hat{a}_n + \Delta t \bar{a}_n + \cdots + \frac{\Delta t^{p-1}}{(p-1)!} \alpha_n^p = \bar{a}_{n+1} + \frac{\Delta t^{p-1}}{(p-1)!} \alpha_n^p \]  
(18.42)

\[ \vdots \]

\[ \alpha_n^p = \alpha_n + \Delta t \alpha_n^p \]

In the above \( \hat{a}, \bar{a}, \) etc., are again quantities that can be written down \( a \) priori (before solving for \( \alpha_n^p \)). These represent predicted values at the end of the interval with \( \alpha_n^p = 0 \).
To summarize, the general algorithm necessitates the choice of values for $\theta_1$ to $\theta_p$ and requires

(a) computation of $\bar{a}$, $\dot{a}$ and $\ddot{a}$ using the definitions of Eqs (18.40);
(b) computation of $a_n^p$ by solution of Eq. (18.41);
(c) computation of $a_{n+1}^p$ to $a_{n+1}$ by Eqs (18.42).

After completion of stage (c) a new time step can be started. In first-order problems the computation of $\ddot{a}$ can obviously be omitted.

If matrices $C$ and $M$ are diagonal the solution of Eq. (18.41) is trivial providing we choose

$$\theta_p = 0$$  \hspace{1cm} (18.43)

With this choice the algorithms are explicit but, as we shall find later, only sometimes conditionally stable.

When $\theta_p \neq 0$, implicit algorithms of various kinds will be available and some of these will be found to be unconditionally stable. Indeed, it is such algorithms that are of great practical use.

Important special cases of the general algorithm are the SS11 and SS22 forms given below.

**The SS11 algorithm**

If we consider the first-order equation (that is $j = 1$) it is evident that only the value of $a_n$ is necessarily specified as the initial value for any computation. For this reason the choice of a linear expansion in the time interval is natural ($p = 1$) and the SS11 algorithm is for that reason most widely used.

Now the approximation of Eq. (18.35) is simply

$$a = a_n + \tau a \quad (a_n^1 = a = \dot{a})$$  \hspace{1cm} (18.44)

and the approximation to the average satisfaction of Eq. (18.2) is simply

$$Ca + K(a_{n+1} + \theta \Delta t a) + \bar{f} = 0$$  \hspace{1cm} (18.45)

with $a_{n+1} = a_n$. Solution of Eq. (18.45) determines $a$ as

$$a = -(C + \theta \Delta t K)^{-1}(\bar{f} + Ka_n)$$  \hspace{1cm} (18.46)

and finally

$$a_{n+1} = a_n + \Delta t a$$  \hspace{1cm} (18.47)

The reader will verify that this process is identical to that developed in Eqs (18.7)–(18.13) and hence will not be further discussed except perhaps for noting the more elegant computation form above.

**The SS22 algorithm**

With Eq. (18.1) we considered a second-order system ($j = 2$) in which the necessary initial conditions require the specification of two quantities, $a_n$ and $\dot{a}_n$. The simplest and most natural choice here is to specify the minimum value of $p$, that is $p = 2$, as this does not require computation of additional derivatives at the start. This algorithm, SS22, is thus basic for dynamic equations and we present it here in full.
The time dimension – discrete approximation in time

From Eq. (18.35) the approximation is a quadratic

\[ a = a_n + \tau \dot{a}_n + \frac{1}{2} \tau^2 a \quad (\alpha_n^2 = a = \ddot{a}) \]  \hspace{1cm} (18.48)

The approximate form of the 'average' dynamic equation is now

\[ M\ddot{a} + C(\dot{a}_{n+1} + \theta_1 \Delta t \ddot{a}) + K(\ddot{a}_{n+1} + \frac{1}{2} \theta_2 \Delta t \dot{a}) + \ddot{f} = 0 \]  \hspace{1cm} (18.49)

with predicted 'mean' values

\[ \ddot{a}_{n+1} = a_n + \theta_1 \Delta t \dot{a}_n \]  \hspace{1cm} (18.50)
\[ \dot{a}_{n+1} = \dot{a}_n \]

After evaluation of \( a \) from Eq. (18.49), the values of \( a_{n+1} \) are found by Eqs (18.42) which become simply

\[ a_{n+1} = a_n + \Delta t \dot{a}_n + \frac{1}{2} \Delta t^2 a \]  \hspace{1cm} (18.51)
\[ \dot{a}_{n+1} = \dot{a}_n + \Delta t \ddot{a} \]

This completes the algorithm which is of much practical value in the solution of dynamics problems.

In many respects it resembles the Newmark algorithm\(^{41}\) which we shall discuss in the next section and which is widely used in practice. Indeed, its stability properties turn out to be identical with the Newmark algorithm, i.e.,

\[ \theta_1 = \gamma \]
\[ \theta_2 = 2\beta \] \hspace{1cm} (18.52)
\[ \theta_1 \geq \theta_2 \geq \frac{1}{2} \]

for unconditional stability. In the above \( \gamma \) and \( \beta \) are conventionally used Newmark parameters.

For \( \theta_2 = 0 \) the algorithm is 'explicit' (assuming both \( M \) and \( C \) to be diagonal) and can be made conditionally stable if \( \theta_1 \geq 1/2 \).

The algorithm is clearly applicable to first-order equations described as \( SS21 \) and we shall find that the stability conditions are identical. In this case, however, it is necessary to identify an initial condition for \( \dot{a}_0 \) and

\[ \dot{a}_0 = -C^{-1}(K\dot{a}_0 + \ddot{f}_0) \]

is one possibility.

### 18.3.3 Truncated Taylor series collocation algorithm GNpj

It will be shown that again as in Sec. 18.2.2 a non-self-starting process is obtained, which in most cases, however, gives an algorithm similar to the SSpj one we have derived. The classical Newmark method\(^{41}\) will be recognized as a particular case together with its derivation process in a form presented generally in existing texts.\(^{42}\) Because of this similarity we shall term the new algorithm generalized Newmark (GNpj).
In the derivation, we shall now consider the satisfaction of the governing equation (18.1) only at the end points of the interval $\Delta t$ [collocation which results from the weighting function shown in Fig. 18.3(c)] and write

$$M\ddot{a}_{n+1} + C\dot{a}_{n+1} + K a_{n+1} + f_{n+1} = 0 \quad (18.53)$$

with appropriate approximations for the values of $a_{n+1}$, $\dot{a}_{n+1}$ and $\ddot{a}_{n+1}$.

If we consider a truncated Taylor series expansion similar to Eq. (18.17) for the function $a$ and its derivatives, we can write

$$a_{n+1} = a_n + \Delta t \dot{a}_n + \cdots + \frac{\Delta t^p}{p!} a_n + \beta_p \frac{\Delta t^p}{p!} \left( \ddot{a}_{n+1} - \ddot{a}_n \right)$$

$$\ddot{a}_{n+1} = \ddot{a}_n + \Delta t \dddot{a}_n + \cdots + \frac{\Delta t^{p-1}}{(p-1)!} \dddot{a}_n + \beta_{p-1} \frac{\Delta t^{p-1}}{(p-1)!} \left( \dddot{a}_{n+1} - \dddot{a}_n \right) \quad (18.54)$$

In Eqs (18.44) we have effectively allowed for a polynomial of degree $p$ (i.e., by including terms up to $\Delta t^p$) plus a Taylor series remainder term in each of the expansions for the function and its derivatives with a parameter $\beta_j$, $j = 1, 2, \ldots, p$, which can be chosen to give good approximation properties to the algorithm.

Insertion of the first three expressions of (18.54) into Eq. (18.53) gives a single equation from which $a_{n+1}$ can be found. When this is determined, $a_{n+1}$ to $a_{n+p-1}$ can be evaluated using Eqs (18.54). Satisfying Eq. (18.53) is almost a ‘collocation’ which could be obtained by inserting the expressions (18.54) into a weighted residual form (18.37) with $W = \delta(t_{n+1})$ (the Dirac delta function). However, the expansion does not correspond to a unique function $a$.

In detail we can write the first three expansions of Eqs (18.54) as

$$a_{n+1} = \dddot{a}_n + \beta_p \frac{\Delta t^p}{p!} \dddot{a}_n$$

$$\ddot{a}_{n+1} = \ddot{a}_n + \beta_{p-1} \frac{\Delta t^{p-1}}{(p-1)!} \dddot{a}_n \quad (18.55)$$

where

$$\dddot{a}_{n+1} = a_n + \Delta t \dddot{a}_n + \cdots + (1 - \beta_p) \frac{\Delta t^p}{p!} \dddot{a}_n + \cdots$$

$$\dddot{a}_{n+1} = a_n + \Delta t \dddot{a}_n + \cdots + (1 - \beta_{p-1}) \frac{\Delta t^{p-1}}{(p-1)!} \dddot{a}_n + \cdots \quad (18.56)$$

$$\dddot{a}_{n+1} = a_n + \Delta t \dddot{a}_n + \cdots + (1 - \beta_{p-2}) \frac{\Delta t^{p-2}}{(p-2)!} \dddot{a}_n + \cdots$$
Inserting the above into Eq. (18.53) gives

\[ \ddot{a}_{n+1} = -A^{-1}\{M\dddot{a}_{n+1} + C\dot{a}_{n+1} + K\ddot{a}_{n+1} + f_{n+1}\} \quad (18.57) \]

where

\[ A = \frac{\beta_p^2 \Delta t^p - 2}{(p-2)!} M + \frac{\beta_{p-1} \Delta t^{p-1}}{(p-1)!} C + \frac{\beta_p \Delta t^p}{p!} K \]

Solving the above equation for \( \ddot{a}_{n+1} \), we have

\[ \ddot{a}_{n+1} = -A[M\dddot{a}_{n+1} + C\dot{a}_{n+1} + K\ddot{a}_{n+1} + f_{n+1}] \quad (18.58) \]

We note immediately that the above expression is formally identical to that of the SSpj algorithm, Eq. (18.41), if we make the substitutions

\[ \beta_p = \theta_p, \quad \beta_{p-1} = \theta_{p-1}, \quad \beta_{p-2} = \theta_{p-2} \quad (18.59) \]

However, \( \dddot{a}_{n+1}, \dddot{a}_{n+1}, \) etc., in the generalized Newmark, GNpj, are not identical to \( \dddot{a}_{n+1}, \dddot{a}_{n+1}, \) etc., in the SSpj algorithms. In the SSpj algorithm these represent predicted mean values in the interval \( \Delta t \) while in the GNpj algorithms they represent predicted values at \( t_{n+1} \).

The computation procedure for the GN algorithms is very similar to that for the SS algorithms, starting now with known values of \( a_n \) to \( \dot{a}_n \). As before we have the given initial conditions and we can usually arrange to use the differential equation and its derivatives to generate higher derivatives for \( a \) at \( t = 0 \). However, the GN algorithm requires more storage because of the necessity of retaining and using \( a_n \) in the computation of the next time step.

An important member of this family is the GN22 algorithm. However, before presenting this in detail we consider another form of the truncated Taylor series expansion which has found considerable use recently, especially in non-linear applications.

An alternative is to use a weighted residual approach with a collocation weight function placed at \( t = t_{n+\theta} \) on the governing equation. This gives a generalization to Eq. (18.13) of

\[ \frac{1}{\Delta t} M(\dddot{a}_{n+1} - \dddot{a}_n) + C\dot{a}_{n+\theta} + K\ddot{a}_{n+\theta} + f_{n+\theta} = 0 \quad (18.60) \]

where an interpolated value for \( a_{n+\theta} \) and \( \dddot{a}_{n+\theta} \) may be written as

\[ a_{n+\theta} = a_n + \theta(a_{n+1} - a_n) \]
\[ \dddot{a}_{n+\theta} = \dddot{a}_n + \theta(\dddot{a}_{n+1} - \dddot{a}_n) \quad (18.61) \]

This form may be combined with a weighted residual approach as described in reference 16. A collocation algorithm for this form is generalized in references 43–46. An advantage of this latter form is an option which permits the generation of energy and momentum conserving properties in the discrete dynamic problem. These generalizations are similar to the GNpj algorithm described in this section although the optimal parameters are usually different.
**The Newmark algorithm (GN22)**

We have already mentioned the classical Newmark algorithm as it is one of the most popular for dynamic analysis. It is indeed a special case of the general algorithm of the preceding section in which a quadratic \((p = 2)\) expansion is used, this being the minimum required for second-order problems. We describe here the details in view of its widespread use.

The expansion of Eq. (18.54) for \(p = 2\) gives

\[
\begin{align*}
\mathbf{a}_{n+1} & = \mathbf{a}_n + \Delta t \dot{\mathbf{a}}_n + \frac{1}{2} (1 - \beta_2) \Delta t^2 \ddot{\mathbf{a}}_n + \frac{1}{2} \beta_2 \Delta t^2 \dddot{\mathbf{a}}_{n+1} = \dot{\mathbf{a}}_{n+1} + \frac{1}{2} \beta_2 \Delta t^2 \dddot{\mathbf{a}}_{n+1} \\
\dot{\mathbf{a}}_{n+1} & = \dot{\mathbf{a}}_n + (1 - \beta_1) \Delta t \ddot{\mathbf{a}}_n + \beta_1 \Delta t \dddot{\mathbf{a}}_{n+1} = \ddot{\mathbf{a}}_{n+1} + \beta_1 \Delta t \dddot{\mathbf{a}}_{n+1}
\end{align*}
\]  

(18.62)

and this together with the dynamic equation (18.53),

\[
\mathbf{M} \dddot{\mathbf{a}}_{n+1} + \mathbf{C} \ddot{\mathbf{a}}_{n+1} + \mathbf{K} \mathbf{a}_{n+1} + \mathbf{f}_{n+1} = \mathbf{0}
\]  

(18.63)

allows the three unknowns \(\mathbf{a}_{n+1}, \dot{\mathbf{a}}_{n+1},\) and \(\ddot{\mathbf{a}}_{n+1}\) to be determined.

We now proceed as we have already indicated and solve first for \(\ddot{\mathbf{a}}_{n+1}\) by substituting (18.62) into (18.63). This yields as the first step

\[
\dddot{\mathbf{a}}_{n+1} = -\mathbf{A}^{-1} \{\mathbf{f}_{n+1} + \mathbf{C} \ddot{\mathbf{a}}_{n+1} + \mathbf{K} \mathbf{a}_{n+1}\}
\]  

(18.64)

where

\[
\mathbf{A} = \mathbf{M} + \beta_1 \Delta t \mathbf{C} + \frac{1}{2} \beta_2 \Delta t^2 \mathbf{K}
\]  

(18.65)

After this step the values of \(\mathbf{a}_{n+1}\) and \(\dot{\mathbf{a}}_{n+1}\) can be found using Eqs (18.62).

As in the general case, \(\beta_2 = 0\) produces an explicit algorithm whose solution is very simple if \(\mathbf{M}\) and \(\mathbf{C}\) are assumed diagonal.

It is of interest to remark that the accuracy can be slightly improved and yet the advantages of the explicit form preserved for SS/GN algorithms by a simple iterative process within each time increment. In this, for the GN algorithm, we predict \(\dddot{\mathbf{a}}_{n+1}^i, \dot{\mathbf{a}}_{n+1}^i,\) and \(\mathbf{a}_{n+1}^i\) using expressions (18.55) with

\[
\left(\dddot{\mathbf{a}}_{n+1}^i\right)^{-1}
\]

setting for \(i = 1\)

\[
\left(\dot{\mathbf{a}}_{n+1}^i\right)^0 = 0
\]

This is followed by rewriting the governing equation (18.57) as

\[
\mathbf{M} \left[ \dddot{\mathbf{a}}_{n+1}^{-1} + \frac{\beta_2 \Delta t^{p-2}}{(p-2)!} \dddot{\mathbf{a}}_{n+1} \right] + \mathbf{C} \ddot{\mathbf{a}}_{n+1}^{-1} + \mathbf{K} \mathbf{a}_{n+1}^{-1} + \mathbf{f}_{n+1} = \mathbf{0}
\]

(18.66)

and solving for \(\dddot{\mathbf{a}}_{n+1}^i, \dot{\mathbf{a}}_{n+1}^i,\) and \(\mathbf{a}_{n+1}^i\).

This predictor-corrector iteration has been successfully used for various algorithms, though of course the stability conditions remain unaltered from those of a simple explicit scheme.\(^7\)

For implicit schemes we note that in the general case, Eqs (18.62) have scalar coefficients while Eq. (18.63) has matrix coefficients. Thus, for the implicit case some users prefer a slightly more complicated procedure than indicated above in which the first unknown determined is \(\mathbf{a}_{n+1}\). This may be achieved by expressing
The time dimension – discrete approximation in time

Eqs (18.62) in terms of the $a_{n+1}$ to obtain

$$\ddot{a}_{n+1} = \ddot{a}_{n+1} + \frac{2}{\beta_2 \Delta t^2} a_{n+1}$$
$$\dot{a}_{n+1} = \dot{a}_{n+1} + \frac{2 \beta_1}{\beta_2 \Delta t} a_{n+1}$$  \hspace{1cm} (18.67)

where

$$\ddot{a}_{n+1} = -\frac{2}{\beta_2 \Delta t^2} a_n - \frac{2}{\beta_2 \Delta t} \dot{a}_n - \frac{1 - \beta_2}{\beta_2} \ddot{a}_n$$
$$\dot{a}_{n+1} = -\frac{2 \beta_1}{\beta_2 \Delta t} a_n + \left(1 - \frac{2 \beta_1}{\beta_2} \right) \dot{a}_n + \left(1 - \frac{\beta_1}{\beta_2} \right) \Delta t \ddot{a}_n$$  \hspace{1cm} (18.68)

These are now substituted into Eq. (18.63) to give the result

$$a_{n+1} = -A^{-1} \left( f_{n+1} + CA_{n+1} + MA_{n+1} \right)$$  \hspace{1cm} (18.69)

where now

$$A = \frac{2}{\beta_2 \Delta t^2} M + \frac{2 \beta_1}{\beta_2 \Delta t} C + K$$

which again on using Eqs (18.67) and (18.68) gives $\dot{a}$ and $\ddot{a}$. The inversion is here identical to within a scalar multiplier but as mentioned before precludes use of the explicit form where $\beta_2$ is zero.

18.3.4 Stability of general algorithms

Consistency of the general algorithms of SS and GN type is self-evident and assured by their formulation.

In a similar manner to that used in Sec. 18.2.5 we can conclude from this that the local truncation error is $O(\Delta t^{p+1})$ as the expansion contains all terms up to $\tau^p$. However, the total truncation error after $n$ steps is only $O(\Delta t^p)$ for first-order equation system and $O(\Delta t^{p-1})$ for the second-order one. Details of accuracy discussions and reasons for this can be found in reference 6.

The question of stability is paramount and in this section we shall discuss it in detail for the SS type of algorithms. The establishment of similar conditions for the GN algorithms follows precisely the same pattern and is left as an exercise to the reader. It is, however, important to remark here that it can be shown that

(a) the SS and GN algorithms are generally similar in performance;
(b) their stability conditions are identical when $\theta_p = \beta_p$.

The proof of the last statement requires some elaborate algebra and is given in reference 6.

The determination of stability requirements follows precisely the pattern outlined in Sec. 18.2.5. However for practical reasons we shall

(a) avoid writing explicitly the amplification matrix $A$;
(b) immediately consider the scalar equation system implying modal decomposition
and no forcing, i.e.,

\[ m\ddot{a} + c\dot{a} + ka = 0 \]  \hspace{1cm} (18.70)

Equations (18.39), (18.40) and (18.42) written in scalar terms define the recurrence
algorithms. For the homogeneous case the general solution can be written down as

\[ a_{n+1} = \mu a_n \]
\[ \dot{a}_{n+1} = \mu \ddot{a}_n \]
\[ \vdots \]
\[ a_{n+1} = a_n \]  \hspace{1cm} (18.71)

and substitution of the above into the equations governing the recurrence can be
written quite generally as

\[ SX_n = 0 \]  \hspace{1cm} (18.72)

where

\[ X_n = \begin{bmatrix} a_n \\ \Delta t\ddot{a}_n \\ \vdots \\ \Delta t^p a_n \end{bmatrix} \]  \hspace{1cm} (18.73)

The matrix \( S \) is given below in a compact form which can be verified by the reader:

\[ S = \begin{bmatrix} b_0 & b_1 & b_2 & \cdots & b_{p-1} & b_p \\ 1 - \mu & 1 & \frac{1}{2!} & \cdots & \frac{1}{(p-1)!} & \frac{1}{p!} \\ 0 & 1 - \mu & 1 & \cdots & \frac{1}{(p-2)!} & \frac{1}{(p-1)!} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & \frac{1}{2!} \\ 0 & 0 & 0 & \cdots & 1 - \mu & 1 \end{bmatrix} \]  \hspace{1cm} (18.74)

where

\[ b_0 = \theta_0 \Delta t^2 k, \quad \theta_0 = 1 \]
\[ b_1 = \theta_0 \Delta tc + \theta_1 \Delta t^2 k \]
\[ b_q = \frac{\theta_{q-2}}{(q-2)!} m + \frac{\theta_{q-1} \Delta t}{(q-1)!} c + \frac{\theta_q \Delta t^2}{q!} k, \quad q = 2, 3, \ldots, p \]

For non-trivial solutions for the vector \( X_n \) to exist it is necessary for the deter-
minant of \( S \) to be zero:

\[ \det S = 0 \]  \hspace{1cm} (18.75)
Fig. 18.11 The $\mu = (1 + z)/(1 - z)$ transformation.

This provides a characteristic polynomial of order $p$ for $\mu$ which yields the eigenvalues of the amplification matrix. For stability it is sufficient and necessary that the moduli of all eigenvalues [see Eq. (18.28)] satisfy

$$|\mu| \leq 1 \quad (18.76)$$

We remark that in the case of repeated roots the equality sign does not apply. The reader will have noticed that the direct derivation of the determinant of $S$ is much simpler than writing down matrix $A$ and finding the eigenvalues. The results are, of course, identical.

The calculation of stability limits, even with the scalar (modal) equation system, is non-trivial. For this reason in what follows we shall only do it for $p = 2$ and $p = 3$. However, two general procedures will be introduced here.

The first of these is the so-called $z$ transformation. In this we use a change of variables in the polynomial putting

$$1 + z = 1 - z \quad (18.77)$$

where $z$ as well as $\mu$ are in general complex numbers. It is easy to show that the requirement of Eq. (18.76) is identical to that demanding the real part of $z$ to be negative (see Fig. 18.11).

The second procedure introduced is the well-known Routh–Hurwitz condition which states that for a polynomial with $c_0 > 0$

$$c_0 z^n + c_1 z^{n-1} + \cdots + c_{n-1} z + c_n = 0 \quad (18.78)$$

the real part of all roots will be negative if, for $c_1 > 0$,

$$\det \begin{bmatrix} c_1 & c_3 \\ c_0 & c_2 \end{bmatrix} > 0 \quad \text{and} \quad \det \begin{bmatrix} c_1 & c_3 & c_5 \\ c_0 & c_2 & c_4 \\ 0 & c_1 & c_3 \end{bmatrix} > 0 \quad (18.79)$$
and generally

\[
\begin{vmatrix}
  c_1 & c_3 & c_5 & c_7 & \cdots \\
  c_2 & c_4 & c_6 & \cdots \\
  0 & c_1 & c_3 & c_5 & \cdots \\
  0 & 0 & c_2 & c_4 & \cdots \\
  \vdots & \vdots & \vdots & \vdots & \ddots \\
  0 & 0 & 0 & 0 & \cdots & c_n \\
\end{vmatrix}
\]

\[> 0 \quad (18.80)\]

With these tools in hand we can discuss in detail the stability of specific algorithms.

### 18.3.5 Stability of SS22/SS21 algorithms

The recurrence relations for the algorithm given in Eqs (18.49) and (18.51) can be written after inserting

\[
a_{n+1} = \mu a_n \quad \dot{a}_{n+1} = \mu \dot{a}_n \quad f = 0
\]

as

\[
m \dot{\phi} + c(\dot{\phi} + \theta_1 \Delta t \alpha) + k(a_n + \Delta t \dot{a}_n + \frac{1}{2} \theta_2 \Delta t^2 \alpha) = 0
\]

\[
-\mu a_n + a_n + \Delta t a_n + \frac{1}{2} \theta_2 \Delta t^2 \alpha = 0
\]

\[
-\mu \dot{a}_n + \dot{a}_n + \Delta t \dot{a}_n + \theta_1 \Delta t \alpha = 0
\]

Changing the variable according to Eq. (18.77) results in the characteristic polynomial

\[
c_0 z^2 + c_1 z + c_2 = 0
\]

with

\[
c_0 = 4m + (4\theta_1 - 2)\Delta t c + 2(\theta_2 - \theta_1)\Delta t^2 k
\]

\[
c_1 = 2\Delta t c + (2\theta_1 - 1)\Delta t^2 k
\]

\[
c_2 = \Delta t^2 k
\]

The Routh–Hurwitz requirements for stability is simply that

\[
c_0 > 0 \quad c_1 \geq 0 \quad \text{det} \begin{bmatrix} c_1 & 0 \\ c_0 & c_2 \end{bmatrix} > 0
\]

or simply

\[
c_0 > 0 \quad c_1 \geq 0 \quad c_2 > 0
\]

These inequalities give for unconditional stability the condition that

\[
\theta_2 \geq \theta_1 \geq \frac{1}{2}
\]

This condition is also generally valid when \(m = 0\), i.e., for the SS21 algorithm (the first-order equation) though now \(\theta_2 = \theta_1 = \frac{1}{2}\) must be excluded.
It is possible to satisfy the inequalities (18.85) only at some values of $\Delta t$ yielding conditional stability. For the explicit process $\theta_2 = 0$ with SS22/SS21 algorithms the inequalities (18.85) demand that

\[ 2m + (2\theta_1 - 1)\Delta tc - \theta_1 \Delta t^2 k \geq 0 \]
\[ 2c + (2\theta_1 - 1)\Delta tk \geq 0 \]  
(18.87)

The second one is satisfied whenever

\[ \theta_1 \geq \frac{1}{2} \]  
(18.88)

and for $\theta_1 = 1/2$ the first supplies the requirement that

\[ \Delta t^2 \leq \frac{4m}{k} \]  
(18.89)

The last condition does not permit an explicit scheme for SS21, i.e., when $m = 0$. Here, however, if we take $\theta_1 > 1/2$ we have from the first equation of Eq. (18.87)

\[ \Delta t < \frac{2\theta_1 - 1}{\theta_1} \frac{c}{k} \]  
(18.90)

It is of interest for problems of structural dynamics to consider the nature of the bounds in an elastic situation. Here we can use the same process as that described in Sec. 18.2.5 for first-order problems of heat conduction. Looking at a single element with a single degree of freedom and consistent mass yields in place of condition (18.89)

\[ \Delta t \leq \frac{2}{\sqrt{3}} \frac{h}{C} = \Delta t_{\text{crit}} \]

where $h$ is the element size and

\[ C = \sqrt{\frac{E}{\rho}} \]

is the speed of elastic wave propagation. For lumped mass matrices the factor becomes $\sqrt{2}$.

Once again the ratio of the smallest element size over wave speed governs the stability but it is interesting to note that in problems of dynamics the critical time step is proportional to $h$ while, as shown in Eq. (18.33), for first-order problems it is proportional to $h^2$. Clearly for decreasing mesh size explicit schemes in dynamics are more efficient than in thermal analysis and are exceedingly popular in certain classes of problems.

### 18.3.6 Stability of various higher order schemes and equivalence with some known alternatives

Identical stability considerations as those described in previous sections can be applied to SS32/SS31 and higher order approximations. We omit here the algebra and simply quote some results.\(^6\)
Table 18.2 SS21 equivalents

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Theta values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zlamal</td>
<td>$\theta_1 = \frac{9}{4}, \theta_2 = 2$</td>
</tr>
<tr>
<td>Gear</td>
<td>$\theta_1 = \frac{5}{2}, \theta_2 = 2$</td>
</tr>
<tr>
<td>Liniger</td>
<td>$\theta_1 = 1.0848, \theta_2 = 1$</td>
</tr>
<tr>
<td>Liniger</td>
<td>$\theta_1 = 1.2184, \theta_2 = 1.292$</td>
</tr>
</tbody>
</table>

SS32/31. Here for zero damping ($c = 0$) in SS32 we require for unconditional stability that

\[ \theta_1 > \frac{1}{2}, \quad \theta_2 \geq \theta_1 + \frac{1}{6}, \quad \theta_2 \geq \frac{1}{4}, \quad \theta_3 \geq \frac{3}{2}, \quad 3\theta_1\theta_2 - 3\theta_1^2 + \theta_1 \geq \theta_3 \tag{18.91} \]

For first-order problems ($m = 0$), i.e., SS31, the first requirements are as in dynamics but the last one becomes

\[ 3\theta_1\theta_2 - 3\theta_1^2 + \theta_1 \geq \theta_3 - \frac{[6\theta_1(\theta_1 - 1) + 1]^2}{9(2\theta_1 - 1)} \tag{18.92} \]

With $\theta_3 = 0$, i.e., an explicit scheme when $c = 0$,

\[ \Delta t^2 \leq \frac{12(2\theta_1 - 1)}{6\theta_2 - 1} \frac{m}{k} \tag{18.93} \]

and when $m = 0$,

\[ \Delta t \leq \frac{\theta_2 - \theta_1}{6\theta_2 - 1} \frac{c}{k} \tag{18.94} \]

SS42/41. For this (and indeed higher orders) unconditional stability in dynamics problems $m \neq 0$ does not exist. This is a consequence of a theorem by Dahlquist.\[51\]

The SS41 scheme can have unconditional stability but the general expressions for this are cumbersome. We quote one example that is unconditionally stable:

\[ \theta_1 = \frac{5}{2}, \quad \theta_2 = \frac{35}{6}, \quad \theta_3 = \frac{25}{2}, \quad \theta_4 = 24 \]

This set of values corresponds to a backward difference four-step algorithm of Gear.\[52\]

It is of general interest to remark that certain members of the SS (or GN) families of algorithms are similar in performance and identical in the stability (and hence recurrence) properties to others published in the large literature on the subject. Each algorithm claims particular advantages and properties. In Tables 18.2–18.4 we show some members of this family.\[38–57\] Clearly many more algorithms that are

Table 18.3 SS31 equivalents

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Theta values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gear</td>
<td>$\theta_1 = 2, \theta_2 = \frac{11}{4}, \theta_3 = 6$</td>
</tr>
<tr>
<td>Liniger</td>
<td>$\theta_1 = 1.84, \theta_2 = 3.07, \theta_3 = 4.5$</td>
</tr>
<tr>
<td>Liniger</td>
<td>$\theta_1 = 0.8, \theta_2 = 1.03, \theta_3 = 1.29$</td>
</tr>
</tbody>
</table>
Table 18.4 SS32 equivalents

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Theta values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Houmbolt$^{54}$</td>
<td>$\theta_1 = 2$, $\theta_2 = \frac{1}{2}$, $\theta_3 = 6$</td>
</tr>
<tr>
<td>Wilson $^{55}$</td>
<td>$\theta_1 = \Theta$, $\theta_2 = \Theta^2$, $\theta_3 = \Theta^3$ (\Theta = 1.4 common)</td>
</tr>
<tr>
<td>Bossak–Newmark$^{56}$</td>
<td>$\theta_1 = 1 + \alpha_B$, $\theta_2 = \frac{1}{2} + \alpha_B + 2\beta_B$, $\theta_3 = 6\beta_B$</td>
</tr>
<tr>
<td>Bossak–Newmark$^{56}$</td>
<td>$\gamma_B = \frac{1}{2} - \alpha_B$, $\gamma_H = \frac{1}{2} - \alpha_H$, $\beta_B = \frac{1}{2} - \frac{1}{2} \alpha_B$</td>
</tr>
<tr>
<td>Hilber–Hughes–Taylor$^{57}$</td>
<td>$\gamma_B = 1 - \alpha_B$, $\gamma_H = \frac{1}{2} - \alpha_H$, $\beta_B = \frac{1}{2} - \frac{1}{2} \alpha_B$</td>
</tr>
</tbody>
</table>

Applicable are present in the general formulae and a study of their optimal parameters is yet to be performed.

We remark here that identity of stability and recurrence always occurs with multistep algorithms, which we shall discuss in the next section.

**Multistep methods**

**18.4 Multistep recurrence algorithms**

**18.4.1 Introduction**

In the previous sections we have been concerned with recurrence algorithms valid within a single time step and relating the values of $a_{n+1}$, $a_{n+1}$, $a_{n+2}$ to $a_n$, $\dot{a}_n$, $\ddot{a}_n$, etc. It is possible to derive, using very similar procedures to those previously introduced, multistep algorithms in which we relate $a_{n+1}$ to the values $a_n$, $a_{n-1}$, $a_{n-2}$, etc., without explicitly introducing the derivatives. Much classical work on stability and accuracy has been introduced on such multistep algorithms and hence they deserve mention here.

We shall show in this section that a series of such algorithms may be simply derived using the weighted residual process and that, for constant time increments $\Delta t$, this set possesses identical stability and accuracy properties to the SS$p_j$ procedures.

**18.4.2 The approximation procedure for a general multistep algorithm**

As in Sec. 18.3.2 we shall approximate the function $a$ of the second-order equation

$$M\ddot{a} + C\dot{a} + Ka + f = 0$$

(18.95)
by a polynomial expansion of the order \( p \), now containing a single unknown \( a_{n+1} \). This polynomial assumes knowledge of the value of \( a_n, a_{n-1}, \ldots, a_{n-p+1} \) at appropriate times \( t_n, t_{n-1}, \ldots, t_{n-p+1} \) (Fig. 18.12).

We can write this polynomial as

\[
a(t) = \sum_{j=1-p}^{1} N_j(t) a_{n+j} \tag{18.96}
\]

where Lagrange interpolation in time is given by (see Chapter 8)

\[
N_j(t) = \prod_{k=1-p, k \neq j}^{1} \frac{t - t_{n+k}}{t_{n+j} - t_{n+k}} \tag{18.97}
\]

The derivatives of the shape functions may be constructed by writing

\[
N_j(t) = \frac{n_j(t)}{n_j(t_{n+j})} \tag{18.98}
\]

and differentiating the numerator. Accordingly writing

\[
n_j(t) = \prod_{k=1-p, k \neq j}^{1} (t - t_{n+k}) \tag{18.99}
\]

the derivative becomes

\[
\frac{dn_j}{dt} = \sum_{m=p-1}^{1} \prod_{k=p-1, k \neq j}^{1} \frac{(t - t_{n+k})}{t_{n+m} - t_{n+k}} \tag{18.100}
\]

Now

\[
\frac{dN_j}{dt} = \frac{1}{n_j(t_{n+j})} \frac{dn_j}{dt} = \tilde{N}_j \tag{18.101}
\]

These expressions can be substituted into Eq. (18.84) giving

\[
\dot{a} = \sum_{j=1-p}^{1} \tilde{N}_j(t) a_{n+j} \tag{18.102}
\]

\[
\ddot{a} = \sum_{j=1-p}^{1} \tilde{N}_j(t) a_{n+j} \tag{18.103}
\]

Insertion of \( a, \dot{a} \) and \( \ddot{a} \) into the weighted residual equation form of Eq. (18.83) yields

\[
\int_{t_n}^{t_{n+1}} W(t) \sum_{j=1-p}^{1} [(\tilde{N}_j M + \tilde{N}_j C + N_j K) a_{n+j} + N_j f_{n+j}] \, dt = 0 \tag{18.104}
\]

with the forcing functions interpolated similarly from its nodal values.
Fig. 18.12 Multistep polynomial approximation.
**Two point interpolation: \( p = 1 \)**

Evaluating Eq. (18.85) we obtain

\[
N_1 = \frac{t - t_{n+1}}{t_{n+1} - t_n} = \frac{1}{\Delta t} (t - t_{n+1}) = \frac{\tau}{\Delta t}
\]

\[
N_0 = \frac{t - t_{n+1}}{t_n - t_{n+1}} = \frac{1}{\Delta t} (t_{n+1} - t) = 1 - \frac{\tau}{\Delta t}
\]

where \( \Delta t = t_{n+1} - t_n \) and \( \tau = t - t_n \). Here the derivative is computed directly as

\[
\frac{dN_1}{dt} = - \frac{dN_0}{dt} = \frac{1}{\Delta t}
\]

Second derivatives are obviously zero, hence this form may only be used for first-order equations.

**Three point interpolation: \( p = 2 \)**

Evaluating Eq. (18.85)

\[
N_1 = \frac{(t - t_{n-1})(t - t_n)}{(t_{n+1} - t_{n-1})(t_{n+1} - t_n)}
\]

\[
N_0 = \frac{(t - t_{n-1})(t - t_{n+1})}{(t_n - t_{n-1})(t_n - t_{n+1})}
\]

\[
N_{-1} = \frac{(t - t_n)(t - t_{n+1})}{(t_{n-1} - t_n)(t_{n-1} - t_{n+1})}
\]

The derivatives follow immediately from Eqs (18.87) and (18.88) as

\[
\frac{dN_1}{dt} = \frac{(t - t_n) + (t - t_{n-1})}{(t_{n+1} - t_{n-1})(t_{n+1} - t_n)}
\]

\[
\frac{dN_0}{dt} = \frac{(t - t_{n+1}) + (t - t_{n-1})}{(t_n - t_{n-1})(t_n - t_{n+1})}
\]

\[
\frac{dN_{-1}}{dt} = \frac{(t - t_{n+1}) + (t - t_n)}{(t_{n-1} - t_n)(t_{n-1} - t_{n+1})}
\]

This is the lowest order which can be used for second-order equations and has second derivatives

\[
\frac{d^2N_1}{dt^2} = \frac{2}{(t_{n+1} - t_{n-1})(t_{n+1} - t_n)}
\]

\[
\frac{d^2N_0}{dt^2} = \frac{2}{(t_n - t_{n-1})(t_n - t_{n+1})}
\]

\[
\frac{d^2N_{-1}}{dt^2} = \frac{2}{(t_{n-1} - t_n)(t_{n-1} - t_{n+1})}
\]
18.4.3 Constant $\Delta t$ form

For the remainder of our discussion here we shall assume a constant time increment for all steps is given by $\Delta t$. To develop the constant increment form we introduce the natural coordinate $\xi$ defined as

$$\xi = \frac{t - t_n}{\Delta t}, \quad 1 - p \leq \xi \leq 1$$

$$j = 1 - p, 2 - p, \ldots, 0, 1$$

and now assume the shape functions $N_j$ in Eq. (18.84) are functions of the natural coordinates and given by

$$N_j(\xi) = \frac{n_j(\xi)}{n_j(j)}$$

(18.105)

where

$$n_j(\xi) = \prod_{k=1-p}^{1} (\xi - k)$$

$$n_j(j) = \prod_{k=1-p}^{1} (j - k)$$

Derivatives with respect to $\xi$ are given by

$$n_j'(\xi) = \sum_{k=1-p}^{1} \prod_{l=1-p}^{1} (\xi - l)$$

(18.106)

$$n_j''(\xi) = \sum_{k=1-p}^{1} \sum_{l=1-p}^{1} \prod_{m=1-p}^{1} (\xi - m)$$

(18.107)

Using the chain rule these derivatives give the time derivatives

$$\dot{a} = \frac{1}{\Delta t} \sum_{j=1-p}^{1} \frac{n_j'}{n_j(j)} a_{n+k}$$

(18.108)

$$\ddot{a} = \frac{1}{\Delta t^2} \sum_{j=1-p}^{1} \frac{n_j''}{n_j(j)} a_{n+k}$$

(18.109)

The weighted residual equation may now be written as

$$\int_0^1 W(\xi) \sum_{j=1-p}^{1} [(N_j''M + \Delta t N_j' C + \Delta t^2 N_j K) a_{n+j} + \Delta t^2 N_j f_{n+j}] d\xi = 0$$

(18.111)
Using the parameters
\[ \phi_q = \frac{\int_0^1 W \xi^q \, d\xi}{\int_0^1 W \, d\xi}, \quad q = 1, 2, \ldots, p; \quad \phi_0 = 1 \] (18.112)
we now have an algorithm that enables us to compute \( a_{n+1} \) from known values \( a_{n-p+1}, a_{n-p+2}, \ldots, a_n \). [Note: so long as the limits of integration are the same in Eqs (18.111) and (18.112) it makes no difference what we choose them to be.]

**Four-point interpolation: \( p = 3 \)**

For \( p = 3 \), Eq. (18.93) gives
\[ N_{-2}(\xi) = -\frac{1}{6} (\xi^3 - \xi) \] (18.113)
\[ N_{-1}(\xi) = \frac{1}{2} (\xi^3 + \xi^2 - 2\xi) \]
\[ N_0(\xi) = -\frac{1}{2} (\xi^3 + 2\xi^2 - \xi - 2) \] (18.114)
\[ N_1(\xi) = \frac{1}{6} (\xi^3 + 3\xi^2 + 2\xi) \]

Similarly, from Eqs (18.94) and (18.95),
\[ N_{-2}'(\xi) = -\frac{1}{6} (3\xi^2 - 1) \] (18.115)
\[ N_{-1}'(\xi) = \frac{1}{2} (3\xi^2 + 2\xi - 2) \]
\[ N_0'(\xi) = -\frac{1}{2} (3\xi^2 + 4\xi - 1) \]
\[ N_1'(\xi) = \frac{1}{6} (3\xi^2 + 6\xi + 2) \] (18.116)

and
\[ N_{-2}''(\xi) = -\xi \] (18.117)
\[ N_{-1}''(\xi) = 3\xi + 1 \]
\[ N_0''(\xi) = -3\xi - 2 \] (18.118)
\[ N_1''(\xi) = \xi + 1 \]

We now have a three-step algorithm for the solution of Eq. (18.83) of the form (taking \( f = 0 \))
\[ \sum_{j=-2}^{1} [\alpha_{j+2} M + \gamma_{j+2} \Delta t C + \beta_{j+2} \Delta t^2 K] a_{n+j} = 0 \] (18.119)

where
\[ \alpha_{j+2} = \int_0^1 W(\xi) N_j'' \, d\xi \]
\[ \gamma_{j+2} = \int_0^1 W(\xi) N_j' \, d\xi \] (18.120)
\[ \beta_{j+2} = \int_0^1 W(\xi) N_j \, d\xi \]
After integration the above gives

\[ \begin{align*}
\alpha_0 &= -\phi_1 & \gamma_0 &= -\frac{1}{\delta}(3\phi_2 - 1) & \beta_0 &= -\frac{1}{\delta}(\phi_3 - \phi_1) \\
\alpha_1 &= 3\phi_1 + 1 & \gamma_1 &= \frac{1}{\delta}(3\phi_2 + 2\phi_1 - 2) & \beta_1 &= \frac{1}{\delta}(\phi_3 + \phi_2 - 2\phi_1) \\
\alpha_2 &= -3\phi_1 - 2 & \gamma_2 &= -\frac{1}{\delta}(3\phi_2 + 2\phi_1 - 1) & \beta_2 &= -\frac{1}{\delta}(\phi_3 + 2\phi_2 - \phi_1 - 2) \\
\alpha_3 &= \phi_1 + 1 & \gamma_3 &= \frac{1}{\delta}(3\phi_2 + 6\phi_1 + 2) & \beta_3 &= \frac{1}{\delta}(\phi_3 + 3\phi_2 + 2\phi_1)
\end{align*} \] (18.121)

An algorithm of the form given in Eq. (18.119) is called a linear three-step method. The general \( p \)-step form is

\[ \sum_{j=1-p}^{1} [\alpha_{j+p-1} + \gamma_{j+p-1} \Delta t C + \beta_{j+p-1} \Delta t^2 K] a_{n+j} = 0 \] (18.122)

This is the form generally given in mathematics texts; it is an extension of the form given by Lambert\(^2\) for \( C = 0 \). The weighted residual approach described here derives the \( \alpha \)'s, \( \beta \)'s and \( \gamma \)'s in terms of the parameters \( \phi_i, i = 0, 1, \ldots, p \) and thus ensures consistency.

From Eq. (18.122) the unknown \( a_{n+1} \) is obtained in the form

\[ a_{n+1} = [\alpha_3 M + \gamma_3 \Delta t C + \beta_3 \Delta t^2 K]^{-1} F \] (18.123)

where \( F \) is expressed in terms of known values. For example, for \( p = 3 \) the matrix to be inverted is

\[ [(\phi_1 + 1) M + \left( \frac{1}{\delta} \phi_2 + \phi_1 + \frac{1}{\delta} \right) \Delta t C + \left( \frac{1}{\delta} \phi_3 + \frac{1}{\delta} \phi_2 + \frac{1}{\delta} \phi_1 \right) \Delta t^2 K] \]

Comparing this with the matrix to be inverted in the \( \text{SSpj} \) algorithm given in Eq. (18.41) suggests a correspondence between \( \text{SSpj} \) and the \( p \)-step algorithm above which we explore further in the next section.

### 18.4.4 The relationship between \( \text{SSpj} \) and the weighted residual \( p \)-step algorithm

For simplicity we now consider the \( p \)-step algorithm described in the previous section applied to the homogeneous scalar equation

\[ m \ddot{a} + c \dot{a} + ka = 0 \] (18.124)

As in previous stability considerations we can obtain the general solution of the recurrence relation

\[ \sum_{j=1-p}^{1} [\alpha_{j+p-1} + \gamma_{j+p-1} \Delta t c + \beta_{j+p-1} \Delta t^2 k] a_{n+j} = 0 \] (18.125)

by putting \( a_{n+j} = \mu^{p-1+j} \) where the values of \( \mu \) are the roots \( \mu_k \) of the stability polynomial of the \( p \)-step algorithm:

\[ \sum_{j=1-p}^{1} [\alpha_{j+p-1} + \gamma_{j+p-1} \Delta t c + \beta_{j+p-1} \Delta t^2 k] \mu^{p-1+j} = 0 \] (18.126)
This stability polynomial can be quite generally identified with the one resulting from the determinant of Eq. \((18.74)\) as shown in reference 6, by using a suitable set of relations linking \(\theta_i\) and \(\phi_j\). Thus, for instance, in the case of \(p = 3\) discussed we shall have the identity of stability and indeed of the algorithm when

\[
\theta_1 = \phi_1 + 1 \\
\theta_2 = \phi_2 + 2\phi_1 + \frac{2}{3} \\
\theta_3 = \phi_3 + 3\phi_2 + 2\phi_1
\]


(18.127)

Table 18.5 summarizes the identities of \(p = 2, 3\) and 4.

Many results obtained previously with \(p\)-step methods\(^{15,58}\) can be used to give the accuracy and stability properties of the solution produced by the SS\(p\) algorithms. Tables 18.6 and 18.7 give the accuracy of stable algorithms from the SS\(p\)1 and SS\(p\)2 families respectively for \(p = 2, 3, 4\). Algorithms that are only conditionally stable (i.e., only stable for values of the time step less than some critical value) are marked CS. Details are given in reference 2.

We conclude this section by writing in full the second degree (two-step) algorithm that corresponds precisely to SS22 and GN22 methods. Indeed, it is written below in the form originally derived by Newmark\(^{41}\):

\[
[M + \gamma \Delta t C + \beta \Delta t^2 K] a_{n+1} \\
- 2M + (1 - 2\gamma)\Delta t C + (\frac{1}{2} - 2\beta + \gamma) \Delta t^2 K] a_n \\
+ [M - (1 - \gamma)\Delta t C + (\frac{1}{2} + \beta - \gamma) \Delta t^2 K] a_{n-1} + \Delta t^2 \bar{f} = 0
\]

(18.128)
Here of course, we have the original Newmark parameters $\beta, \gamma$, which can be changed to correspond with the SS22/GN22 form as follows:

$$\gamma = \theta_1 = \beta_1 \quad \beta = \frac{1}{2} \theta_2 = \frac{1}{2} \beta_2$$

The explicit form of this algorithm ($\beta = \theta_2 = \beta_2 = 0$) is frequently used as an alternative to the single-step explicit form. It is then known as the central difference approximation obtained by direct differencing. The reader can easily verify that the simplest finite difference approximation of Eq. (18.1) in fact corresponds to the above with $\beta = 0$ and $\gamma = 1/2$.

### 18.5 Some remarks on general performance of numerical algorithms

In Secs 18.2.5 and 18.3.3 we have considered the exact solution of the approximate recurrence algorithm given in the form

$$a_{n+1} = \mu a_n, \quad \text{etc.} \quad (18.129)$$

for the modally decomposed, single degree of freedom systems typical of Eqs (18.4) and (18.5). The evaluation of $\mu$ was important to ensure that its modulus does not exceed unity and stability is preserved.

However, analytical solution of the linear homogeneous differential equations is also easy to obtain in the form

$$a = \bar{a} e^{\lambda t} \quad (18.130)$$

$$a_{n+1} = a_n e^{\lambda \Delta t} \quad (18.131)$$

and comparison of $\mu$ with such a solution is always instructive to provide information on the performance of algorithms in the particular range of eigenvalues.

In Fig. 18.5 we have plotted the exact solution $e^{-\omega \Delta t}$ and compared it with the values of $\mu$ for various $\theta$ algorithms approximating the first-order equation, noting that here

$$\lambda = -\omega = -\frac{k}{c} \quad (18.132)$$

and is real.
Immediately we see that there the performance error is very different for various values of $\Delta t$ and obviously deteriorates at large values. Such values in a real multivariable problem correspond of course to the ‘high-frequency’ responses which are often less important, and for smooth solutions we favour algorithms where $\mu$ tends to values much less than unity for such problems. However, response through the whole time range is important and attempts to choose an optimal value of $\theta$ for various time ranges has been performed by Liniger. Table 18.1 of Sec. 18.2.6 illustrates how an algorithm with $\theta = 2/3$ and a higher truncation error than that.
Fig. 18.14 SS23, GN23 or their two-step equivalent.
Some remarks on general performance of numerical algorithms

Fig. 18.15 Comparison of the SS22 and GN22 (Newmark) algorithms: a single DOF dynamic equation with periodic forcing term, $\theta_1 = \beta_1 = 1/2$, $\theta_2 = \beta_2 = 0$. 
Fig. 18.15 Continued.
Some remarks on general performance of numerical algorithms

Fig. 18.15 Continued.
of $\theta = 1/2$ can perform better in a multidimensional system because of such properties.

Similar analysis can be applied to the second-order equation. Here, to simplify matters, we consider only the homogeneous undamped equation in the form

$$ma + ka = 0$$  \hspace{1cm} (18.133)

in which the value of $\lambda$ is purely imaginary and corresponds to a simple oscillator. By examining $\mu$ we can find not only the amplitude ratio (which for high accuracy should be unity) but also the phase error.

In Fig. 18.13(a) we show both the variation of the modulus $\mu$ (which is called the spectral radius) and in Fig. 18.13(b) that of the relative period for the SS22/GN22 schemes, which of course are also applicable to the two-step equivalent. The results are plotted against $\Delta t / T$ where $T = \frac{2\pi}{\omega}$; $\omega^2 = \frac{k}{m}$ \hspace{1cm} (18.134)

In Fig. 18.14(a) and (b) similar curves are given for the SS23 and GN23 schemes frequently used in practice and discussed previously.

Here as in the first-order problem we often wish to suppress (or damp out) the response to frequencies in which $\Delta t / T$ is large (say greater than 0.1) in multidegree of freedom systems, as such a response will invariably be inaccurate. At the same time below this limit it is desirable to have amplitude ratios as close to unity as possible. It is clear that the stability limit with $\theta_1 = \theta_2 = 1/2$ giving unit response everywhere is often undesirable (unless physical damping is sufficient to damp high frequency modes) and that some algorithmic damping is necessary in these cases. The various schemes shown in Figs 18.13 and 18.14 can be judged accordingly and provide the reason for a search for an optimum algorithm.

We have remarked frequently that although schemes can be identical with regard to stability their performances may differ slightly. In Fig. 18.15 we illustrate the application of SS22 and GN22 to a single degree of freedom system showing results and errors in each scheme.

18.6 Time discontinuous Galerkin approximation

A time discontinuous Galerkin formulation may be deduced from the finite element in time approximation procedure considered in this chapter. This is achieved by assuming the weight function $W$ and solution variables $a$ are approximated within each time interval $\Delta t$ as

$$a = a_n^+ + \Delta a(t) \quad t_n^- \leq t < t_{n+1}^-$$

$$W = W_n^+ + \Delta W(t) \quad t_n^- \leq t < t_{n+1}^-$$  \hspace{1cm} (18.135)

where the time $t_n^-$ is the limit from times smaller than $t_n$ and $t_n^+$ is the limit from times larger than $t_n$ and, thus, admit a discontinuity in the approximation to occur at each discrete time location. The functions $\Delta a$ and $\Delta W$ are defined to be zero at $t_n$ and continuous up to the time $t_{n+1}^-$ where again a discontinuity can occur during the next time interval.
The discrete form of the governing equations may be deduced starting from the time dependent partial differential equations where standard finite elements in space are combined with the time discontinuous Galerkin approximation and defining a weak form in a space-time slab. Alternatively, we may begin with the semi-discrete form as done previously in this chapter for other finite element in time methods. In this second form, for the first-order case, we write

$$I = \int_{t_n}^{t_{n+1}} W^T (C\dot{a} + Ka + f) \, d\tau = 0$$

(18.136)

Due to the discontinuity at $t_n$, it is necessary to split the integral into

$$I = \int_{t_n}^{t_{n+1}} W^T (C\dot{a} + Ka + f) \, d\tau + \int_{t_n}^{t_{n+1}} W^T (C\dot{a} + Ka + f) \, d\tau = 0$$

(18.137)

which gives

$$I = (W_n^+)^T [C(a^+ - a^-)] + (W_n^+)^T \int_{t_n}^{t_{n+1}} (C\dot{a} + Ka + f) \, d\tau$$

$$+ \int_{t_n}^{t_{n+1}} (\Delta W)^T (C\dot{a} + Ka + f) \, d\tau = 0$$

(18.138)

in which now all integrals involve approximations to functions which are continuous.

To apply the above process to a second-order equation it is necessary first to reduce the equation to a pair of first-order equations. This may be achieved by defining the momenta

$$p = M\dot{a}$$

(18.139)

and then writing the pair

$$M\dot{a} - p = 0$$

(18.140)

$$\dot{p} + C\dot{a} + Ka + f = 0$$

(18.141)

The time discrete process may now be applied by introducing two weighting functions as described in reference 37.

**Example: Solution of the scalar equation** To illustrate the process we consider the simple first-order scalar equation

$$c\dot{u} + ku + f = 0$$

(18.142)

We consider the specific approximations

$$u(t) = u_n^+ + \tau\Delta u_{n+1}^-$$

$$W(t) = W_n^+ + \tau\Delta W_{n+1}^-$$

(18.143)

where $\Delta u_{n+1}^- = u_{n+1}^- - u_n^+$, etc., and

$$\tau = \frac{t - t_n}{t_{n+1} - t_n} = \frac{t - t_n}{\Delta t}$$
The time dimension – discrete approximation in time

defines the time interval $0 < \tau < \Delta t$. This approximation gives the integral form

$$I = W_n^+ [c(u_n^+ - u_n^-)] + W_n^+ \int_0^{\Delta t} \left[ \frac{1}{\Delta t} c \Delta u_{n+1}^- + k(u_n^+ + \tau \Delta u_{n+1}^-) + f \right] d\tau$$

$$+ \int_0^{\Delta t} \Delta W_{n+1}^- \left[ \frac{1}{\Delta t} c \Delta u_{n+1}^- + k(u_n^+ + \tau \Delta u_{n+1}^-) + f \right] d\tau \quad (18.144)$$

Evaluation of the integrals gives the pair of equations

$$\begin{bmatrix}
(c + \Delta tk) & \frac{1}{2} \Delta tk \\
\frac{1}{2} \Delta tk & (c + \frac{1}{2} \Delta tk)
\end{bmatrix}
\begin{bmatrix}
\Delta u_n^- \\
\Delta u_{n+1}^-
\end{bmatrix}
+ \begin{bmatrix}
\Delta tf^- \\
\Delta t \Delta f^-
\end{bmatrix}
= \begin{bmatrix}
cu_n^- \\
0
\end{bmatrix} \quad (18.145)$$

where

$$\begin{bmatrix}
f^- \\
\Delta f^-
\end{bmatrix}
= \int_0^{\Delta t} \begin{bmatrix}
f \\
\tau f
\end{bmatrix} d\tau \quad (18.146)$$

Thus, with linear approximation of the variables the time discontinuous Galerkin method gives two equations to be solved for the two unknowns $u_n^+$ and $u_{n+1}^-$. It is possible to also perform a solution with constant approximation. Based on the above this is achieved by setting $\Delta u_{n+1}^-$ and $\Delta W_{n+1}^-$ to zero yielding the single equation

$$(c + \Delta tk)u_n^+ + \Delta tf^- = cu_n^- \quad (18.147)$$

and now since the approximation is constant over the entire time the $u_n^+$ also define exactly the $u_{n+1}^-$ value. This form will now be recognized as identical to the backward difference implicit scheme defined in Fig. 18.4 for $\theta = 1$.

### 18.7 Concluding remarks

The derivation and examples presented in this chapter cover, we believe, the necessary tool-kit for efficient solution of many transient problems governed by Eqs (18.1) and (18.2). In the next chapter we shall elaborate further on the application of the procedures discussed here and show that they can be extended to solve coupled problems which frequently arise in practice and where simultaneous solution by time stepping is often needed.

Finally, as we have indicated in Eq. (18.3), many problems have coefficient matrices or other variations which render the problem non-linear. This topic will be addressed further in the second volume where we note also that the issue of stability after many time steps is more involved than the procedures introduced here to investigate local stability.

### References


