9

Algorithms for Hyperbolic and Parabolic-Hyperbolic Problems

9.1 ONE-STEP ALGORITHMS FOR THE SEMIDISCRETE EQUATION OF MOTION

9.1.1 The Newmark Method

Recall from Chapter 7 that the semidiscrete equation of motion is written as

$$\boldsymbol{M}\ddot{\boldsymbol{d}} + \boldsymbol{C}\dot{\boldsymbol{d}} + \boldsymbol{K}\boldsymbol{d} = \boldsymbol{F} \tag{9.1.1}$$

where $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{C}$ is the viscous damping matrix, $\boldsymbol{K}$ is the stiffness matrix, $\boldsymbol{F}$ is the vector of applied forces, and $\boldsymbol{d}$, $\dot{\boldsymbol{d}}$, and $\ddot{\boldsymbol{d}}$ are the displacement, velocity and acceleration vectors, respectively. We take $\boldsymbol{M}$, $\boldsymbol{C}$, and $\boldsymbol{K}$ to be symmetric; $\boldsymbol{M}$ is positive-definite, and $\boldsymbol{C}$ and $\boldsymbol{K}$ are positive-semidefinite.

The initial-value problem for (9.1.1) consists of finding a displacement, $\boldsymbol{d} = \boldsymbol{d}(t)$, satisfying (9.1.1) and the given initial data:

$$\boldsymbol{d}(0) = \boldsymbol{d}_0 \tag{9.1.2}$$

$$\dot{\boldsymbol{d}}(0) = \boldsymbol{v}_0 \tag{9.1.3}$$

Perhaps the most widely used family of direct methods for solving (9.1.1) to (9.1.3) is the *Newmark family* [1], which consists of the following equations:

$$\boldsymbol{M}\boldsymbol{a}_{n+1} + \boldsymbol{C}\boldsymbol{v}_{n+1} + \boldsymbol{K}\boldsymbol{d}_{n+1} = \boldsymbol{F}_{n+1} \tag{9.1.4}$$

$$\boldsymbol{d}_{n+1} = \boldsymbol{d}_n + \Delta t \boldsymbol{v}_n + \frac{\Delta t^2}{2}[(1 - 2\beta)\boldsymbol{a}_n + 2\beta \boldsymbol{a}_{n+1}] \tag{9.1.5}$$

$$v_{n+1} = v_n + \Delta t[(1 - \gamma)a_n + \gamma a_{n+1}] \tag{9.1.6}$$

where d_n, v_n, and a_n are the approximations of $d(t_n)$, $\dot{d}(t_n)$, and $\ddot{d}(t_n)$, respectively. Equation (9.1.4) is simply the equation of motion in terms of the approximate solution, and (9.1.5) and (9.1.6) are finite difference formulas describing the evolution of the approximate solution. The parameters β and γ determine the stability and accuracy characteristics of the algorithm under consideration. Equations (9.1.4) to (9.1.6) may be thought of as three equations for determining the three unknowns d_{n+1}, v_{n+1}, and a_{n+1}, it being assumed that d_n, v_n, and a_n are known from the previous step's calculations. The Newmark family contains as special cases many well-known and widely used methods.

Implementation: a-form (acceleration-based form)

In our case beta = 0 (purely explicit scheme), gamma = 1/2.

There are several possible implementations. We will sketch one, but we leave further details until Sec. 9.4, which deals with operator and mesh partitions. The results in Sec. 9.4 include the Newmark method as a special case. Define *predictors*:

$$\tilde{d}_{n+1} = d_n + \Delta t v_n + \frac{\Delta t^2}{2}(1 - \beta)a_n \tag{9.1.7}$$

$$\tilde{v}_{n+1} = v_n + (1 - \gamma)\Delta t a_n \tag{9.1.8}$$

Equations (9.1.5) and (9.1.6) may then be written as

$$d_{n+1} = \tilde{d}_{n+1} + \beta \Delta t^2 a_{n+1} \tag{9.1.9}$$

$$v_{n+1} = \tilde{v}_{n+1} + \gamma \Delta t a_{n+1} \tag{9.1.10}$$

Matrix C is zero in the elastic case (no viscoelasticity) with no absorbing conditions. Matrix C is not zero when Stacey or PML absorbing conditions are turned on; there is then a non zero contribution to the left-hand side even in the purely explicit case.

The recursion relation determines a_{n+1}:

$$(M + \gamma \Delta t C + \beta \Delta t^2 K)a_{n+1} = F_{n+1} - C\tilde{v}_{n+1} - K\tilde{d}_{n+1} \tag{9.1.12}$$

Equations (9.1.9) and (9.1.10) may then be used to calculate d_{n+1} and v_{n+1}, respectively.

This form of implementation is convenient for generalization to algorithms that employ "mesh partitions" (see Sec. 9.4) but is not the most efficient implementation.

$$v_{n+1} = v_n + \Delta t[(1 - \gamma)a_n + \gamma a_{n+1}] \tag{9.1.6}$$

where d_n, v_n, and a_n are the approximations of $d(t_n)$, $\dot{d}(t_n)$, and $\ddot{d}(t_n)$, respectively. Equation (9.1.4) is simply the equation of motion in terms of the approximate solution, and (9.1.5) and (9.1.6) are finite difference formulas describing the evolution of the approximate solution. The parameters β and γ determine the stability and accuracy characteristics of the algorithm under consideration. Equations (9.1.4) to (9.1.6) may be thought of as three equations for determining the three unknowns d_{n+1}, v_{n+1}, and a_{n+1}, it being assumed that d_n, v_n, and a_n are known from the previous step's calculations. The Newmark family contains as special cases many well-known and widely used methods.

This is the original version of the previous page.

Implementation: a-form

There are several possible implementations. We will sketch one, but we leave further details until Sec. 9.4, which deals with operator and mesh partitions. The results in Sec. 9.4 include the Newmark method as a special case. Define *predictors*:

$$\tilde{d}_{n+1} = d_n + \Delta t v_n + \frac{\Delta t^2}{2}(1 - 2\beta)a_n \tag{9.1.7}$$

$$\tilde{v}_{n+1} = v_n + (1 - \gamma)\Delta t a_n \tag{9.1.8}$$

Equations (9.1.5) and (9.1.6) may then be written as

$$d_{n+1} = \tilde{d}_{n+1} + \beta\Delta t^2 a_{n+1} \tag{9.1.9}$$

$$v_{n+1} = \tilde{v}_{n+1} + \gamma\Delta t a_{n+1} \tag{9.1.10}$$

To start the process, a_0 may be calculated from

$$Ma_0 = F - Cv_0 - Kd_0 \tag{9.1.11}$$

or specified directly. The recursion relation determines a_{n+1}:

$$(M + \gamma\Delta t C + \beta\Delta t^2 K)a_{n+1} = F_{n+1} - C\tilde{v}_{n+1} - K\tilde{d}_{n+1} \tag{9.1.12}$$

Equations (9.1.9) and (9.1.10) may then be used to calculate d_{n+1} and v_{n+1}, respectively.

This form of implementation is convenient for generalization to algorithms that employ "mesh partitions" (see Sec. 9.4) but is not the most efficient implementation.