

DUKE UNIVERSITY
Department of Civil & Environmental Engineering

CE 283. Structural Dynamics

Spring 2010

Henri P. Gavin

Numerical Integration in Structural Dynamics

A damped structural system subjected to dynamic forces and possibly experiencing nonlinear material behavior is modeled by

$$M\ddot{x}(t) + C\dot{x}(t) + Kx(t) + R(x(t), \dot{x}(t)) = f^{\text{ext}}(t), \quad (1)$$

where M is a positive definite mass matrix, C is a non-negative definite damping matrix, and K is a non-negative definite stiffness matrix. The nonlinear restoring forces are given in $R(x, \dot{x})$ and $f^{\text{ext}}(t)$ is a vector of external dynamic loads. At any point in time, $t = t_{i+1} = (i+1)h$, we may solve for the accelerations in terms of the displacements, velocities, and the applied forces.

$$\ddot{x}(t_{i+1}) = -M^{-1}[C\dot{x}(t_{i+1}) + Kx(t_{i+1}) + R(x(t_{i+1}), \dot{x}(t_{i+1})) - f^{\text{ext}}(t_{i+1})]. \quad (2)$$

Given values for accelerations, velocities, displacements, and applied forces at time t_i , if we can extrapolate the velocities and displacements forward in time by a time step h to time $t_{i+1} = t_i + h$, then we may compute the acceleration $\ddot{x}(t_{i+1})$ using equation 2. The various numerical integration algorithms described in this document^{1 2} differ primarily in the manner in which $x(t_{i+1})$ and $\dot{x}(t_{i+1})$ are computed from $x(t_i)$, $\dot{x}(t_i)$, $\ddot{x}(t_i)$, $f^{\text{ext}}(t_i)$, and $f^{\text{ext}}(t_{i+1})$. Note that since differentiation amplifies high frequencies, changes in x and \dot{x} from t_i to t_{i+1} will be much smoother than the corresponding changes in \ddot{x} .

¹Clough, R. and Penzien, J., *Dynamics of Structures*, 2nd ed., McGraw-Hill, 1993.

²Hanson, R.D., CE 611: Structural Dynamics, class notes, The University of Michigan, 1988

The Central Difference Method

The central difference approximations for the first and second derivatives are

$$\dot{x}(t_i) = \dot{x}_i \approx \frac{1}{2h} (x_{i+1} - x_{i-1}), \quad (3)$$

$$\ddot{x}(t_i) = \ddot{x}_i \approx \frac{1}{h^2} (x_{i+1} - 2x_i + x_{i-1}). \quad (4)$$

Substituting approximations (3) and (4) into equation (1), and re-arranging terms,

$$\left[\frac{1}{h^2}M + \frac{1}{2h}C \right] x_{i+1} = \left[\frac{2}{h^2}M - K \right] x_i + \left[-\frac{1}{h^2}M + \frac{1}{2h}C \right] x_{i-1} + f_i^{\text{ext}}, \quad (5)$$

which may be written $A_0 x_{i+1} = A_1 x_i + A_2 x_{i-1} + f_i^{\text{ext}}$. The matrix A_0 is positive definite and may be factorized prior to the iterative solution for x_i at each time step.

The numerical stability of the central difference method depends on the choice of the time step interval, h . To obtain a stable solution, $h < T_i/\pi$, where T_i is the shortest natural period of the structural system. If the central difference method is used with a time step larger than T_i/π the solution will increase exponentially.

A step-by-step procedure for the central difference method may be written as³:

1. Input the mass, M , damping, C , stiffness, K , matrices and the time step interval h .
2. Initialize x_0 , \dot{x}_0 , \ddot{x}_0 and compute $x_{-1} = x_0 - h\dot{x}_0 + \frac{h^2}{2}\ddot{x}_0$.
3. Form the matrices A_0 , A_1 and A_2 , and triangularize A_0 using LDL^T factorization.
4. for $i = 1$ to N
 - solve $A_0 x_i = A_1 x_{i-1} + A_2 x_{i-2} + f_{i-1}^{\text{ext}}$ for x_i using LDL^T back-substitution.
 - calculate \dot{x}_{i-1} and \ddot{x}_{i-1} using equations (3) and (4), if required.
 - write $t_i = ih$, x_i , \dot{x}_i and \ddot{x}_i to a data file.

³K.J. Bathe, *Finite Element Procedures in Engineering Analysis*, Prentice-Hall, 1982, pp. 439–449, 499–506.

The Implicit Linear Acceleration Method

Consider the Taylor series expansions for displacement, velocity, and acceleration:

$$x(t_{i+1}) = x_{i+1} = x_i + h\dot{x}_i + \frac{h^2}{2!}\ddot{x}_i + \frac{h^3}{3!}x_i^{iii} + \frac{h^4}{4!}x_i^{iv} + \dots \quad (6)$$

$$\dot{x}(t_{i+1}) = \dot{x}_{i+1} = \dot{x}_i + h\ddot{x}_i + \frac{h^2}{2!}x_i^{iii} + \frac{h^3}{3!}x_i^{iv} + \dots \quad (7)$$

$$\ddot{x}(t_{i+1}) = \ddot{x}_{i+1} = \ddot{x}_i + hx_i^{iii} + \frac{h^2}{2!}x_i^{iv} + \dots \quad (8)$$

Rearranging equation (8),

$$hx_i^{iii} = \ddot{x}_{i+1} - \ddot{x}_i - \frac{h^2}{2!}x_i^{iv} - \dots, \quad (9)$$

and substituting equation (9) into equations (6) and (7) results in

$$x_{i+1} = x_i + h\dot{x}_i + \frac{h^2}{2}\ddot{x}_i + \frac{h^2}{6}\left(\ddot{x}_{i+1} - \ddot{x}_i - \frac{h^2}{2!}x_i^{iv} - \dots\right) + \frac{h^4}{4!}x_i^{iv} + \dots, \quad (10)$$

$$\dot{x}_{i+1} = \dot{x}_i + h\ddot{x}_i + \frac{h}{2}\left(\ddot{x}_{i+1} - \ddot{x}_i - \frac{h^2}{2!}x_i^{iv} - \dots\right) + \frac{h^3}{3!}x_i^{iv} + \dots \quad (11)$$

Truncating the fourth time-derivative and higher from these expansion, the resulting finite difference approximations are

$$x_{i+1} \approx x_i + h\dot{x}_i + \frac{h^2}{6}(\ddot{x}_{i+1} + 2\ddot{x}_i), \quad (12)$$

and

$$\dot{x}_{i+1} \approx \dot{x}_i + \frac{h}{2}(\ddot{x}_{i+1} + \ddot{x}_i). \quad (13)$$

These relationships are *implicit* because \ddot{x}_{i+1} needs to be determined in order to find x_{i+1} and \dot{x}_{i+1} . Note that through the substitutions above, we eliminate the third time-derivative of x , and that the method is accurate to within the fourth time-derivative of x .

This method is called 'linear acceleration' because the third time derivative of x has been eliminated, and the acceleration can be considered to change linearly with time.

The Implicit Linear Acceleration Method, Made Explicit for Linear Structural Dynamics

Recall that at time t_{i+1} we can satisfy the equations of motion, by calculating the acceleration with equation (2). Substituting equations (12) and (13) for the displacements and velocities at time t_{i+1} into equation (2),

$$M\ddot{x}_{i+1} + C\{\dot{x}_i + (h/2)(\ddot{x}_i + \ddot{x}_{i+1})\} + K\{x_i + h\dot{x}_i + (h^2/6)(\ddot{x}_{i+1} + 2\ddot{x}_i)\} = f_{i+1}^{\text{ext}}, \quad (14)$$

where the nonlinearities, $R(x, \dot{x})$, are negligible. Collecting similar derivatives of x ,

$$\left[M + \frac{h}{2}C + \frac{h^2}{6}K \right] \ddot{x}_{i+1} = f_{i+1}^{\text{ext}} - Kx_i - [C + hK]\dot{x}_i - \left[\frac{h}{2}C + \frac{h^2}{3}K \right] \ddot{x}_i. \quad (15)$$

and re-arranging,

$$\left[M + \frac{h}{2}C + \frac{h^2}{6}K \right] \ddot{x}_{i+1} = \left[-\frac{h}{2}C - \frac{h^2}{3}K \right] \ddot{x}_i - Kx_i - C\dot{x}_i - hK\dot{x}_i + f_{i+1}^{\text{ext}}. \quad (16)$$

Recall from the equations of motion, that

$$M\ddot{x}_i - f_i^{\text{ext}} = -Kx_i - C\dot{x}_i. \quad (17)$$

Substituting equation (17) into (16), we obtain the closed-form linear acceleration recurrence relations for structural dynamics simulation.

$$\left[M + \frac{h}{2}C + \frac{h^2}{6}K \right] \ddot{x}_{i+1} = \left[M - \frac{h}{2}C - \frac{h^2}{3}K \right] \ddot{x}_i - hK\dot{x}_i + f_{i+1}^{\text{ext}} - f_i^{\text{ext}}. \quad (18)$$

$$x_{i+1} = x_i + h\dot{x}_i + \frac{h^2}{6} [\ddot{x}_{i+1} + 2\ddot{x}_i]. \quad (19)$$

and

$$\dot{x}_{i+1} = \dot{x}_i + \frac{h}{2} [\ddot{x}_{i+1} + \ddot{x}_i]. \quad (20)$$

These relationships are now *explicit* because \ddot{x}_{i+1} can be determined from the current response values (x_i and \dot{x}_i), the current dynamic load f_i^{ext} , and the next dynamic load f_{i+1}^{ext} . Note that within each time step, the dynamic equations of equilibrium are satisfied both at time t_i and at time t_{i+1} .

The Newmark- β method — incremental formulation

The finite difference approximations for the Newmark- β method are

$$x_{i+1} \approx x_i + h\dot{x}_i + h^2 \left[\left(\frac{1}{2} - \beta \right) \ddot{x}_i + \beta \ddot{x}_{i+1} \right], \quad (21)$$

and

$$\dot{x}_{i+1} \approx \dot{x}_i + h [(1 - \gamma)\ddot{x}_i + \gamma\ddot{x}_{i+1}]. \quad (22)$$

If $\beta = 1/4$ and $\gamma = 1/2$ the Newmark- β method is implicit and unconditionally stable. In this case the acceleration within the time interval $t \in [t_i, t_{i+1})$ is presumed to be constant. If $\beta = 1/6$ and $\gamma = 1/2$ the Newmark- β method is identical to the linear acceleration method. If $\beta = 0$ and $\gamma = 1/2$ the Newmark- β method is identical to the central difference method. For linear structural dynamics, if $2\beta \geq \gamma \geq 1/2$, then the Newmark- β method is stable regardless of the size of the time-step, h . In general, however, the stability of the algorithm depends on the size of the time step, and the Newmark- β method is conditionally stable if $\gamma < 1/2$. For $\gamma = 1/2$ the Newmark- β method is second-order accurate; it is first order accurate for all other values of γ .

Formulating the finite difference relationships in terms of the increments of displacement ($\delta x_i = x_{i+1} - x_i$), velocity ($\delta \dot{x}_i = \dot{x}_{i+1} - \dot{x}_i$), acceleration ($\delta \ddot{x}_i = \ddot{x}_{i+1} - \ddot{x}_i$), and nonlinear restoring force ($\delta R_i = R(x_{i+1}, \dot{x}_{i+1}) - R(x_i, \dot{x}_i)$)

$$\delta \ddot{x}_i = \frac{1}{\beta h^2} \delta x_i - \frac{1}{\beta h} \dot{x}_i - \frac{1}{2\beta} \ddot{x}_i, \quad (23)$$

and

$$\delta \dot{x}_i = \frac{\gamma}{\beta h} \delta x_i - \frac{\gamma}{\beta} \dot{x}_i + h \left(1 - \frac{\gamma}{2\beta} \right) \ddot{x}_i. \quad (24)$$

Now, satisfying incremental equilibrium over the time step, h ,

$$M\delta \ddot{x}_i + C\delta \dot{x}_i + K\delta x_i + \delta R_i = f_{i+1}^{\text{ext}} - f_i^{\text{ext}} = \delta f_i^{\text{ext}}, \quad (25)$$

re-grouping terms, and solving for the increment in displacements,

$$\left[\frac{6}{h^2}M + \frac{3}{h}C + K \right] \delta x_i = \delta f_i^{\text{ext}} - \delta R_i + \left[3M + \frac{h}{2}C \right] \ddot{x}_i + \left[\frac{6}{h}M + 3C \right] \dot{x}_i, \quad (26)$$

where $\beta = 1/6$ and $\gamma = 1/2$. Solving this linear system of equations⁴ for δx_i , the displacements are updated with,

$$x_{i+1} = x_i + \delta x_i, \quad (27)$$

the velocities are updated with

$$\dot{x}_{i+1} = -2\dot{x}_i - \frac{h}{2}\ddot{x}_i + \frac{3}{h}\delta x_i, \quad (28)$$

⁴If the incremental nonlinear restoring forces δR_i are not negligible, equation (26) is solved using the Newton-Raphson method, equation (46).

and the accelerations satisfy the equations of motion,

$$\ddot{x}_{i+1} = -M^{-1}[C\dot{x}_{i+1} + Kx_{i+1} + R(x_{i+1}, \dot{x}_{i+1}) - f_{i+1}^{\text{ext}}]. \quad (29)$$

Similar relations may be found for other values of β and γ . Note that in this incremental formulation one needs to obtain the inverse of the mass matrix, M , and the inverse of the matrix $[6M/h^2 + 3C/h + K]$.

The HHT- α method

The HHT- α method⁵ is a generalization of the Newmark- β method and reduces to the Newmark- β method for $\alpha = 0$. The finite difference equations are identical to those of the Newmark- β method, (equations (21) and (22)). The equations of motion are modified, however, using a parameter α .

$$M\ddot{x}_{i+1} + (1 - \alpha)C\dot{x}_{i+1} + \alpha C\dot{x}_i + (1 - \alpha)Kx_{i+1} + \alpha Kx_i = (1 - \alpha)f_{i+1}^{\text{ext}} + \alpha f_i^{\text{ext}}. \quad (30)$$

Equation (30) is analogous to equation (25) with the added condition that the acceleration is constant within the interval $t \in [t_i, t_{i+1})$. If

$$0 \leq \alpha \leq 1/3 \quad (31)$$

$$\beta = (1 + \alpha)^2/4, \quad \text{and} \quad (32)$$

$$\gamma = 1/2 + \alpha, \quad (33)$$

the HHT- α method is second-order accurate and unconditionally stable. The HHT- α method is useful in structural dynamics simulations incorporating many degrees of freedom, and in which it is desirable to numerically attenuate (or dampen-out) the response at high frequencies. Increasing α decreases the response at frequencies above $1/(2h)$, provided that β and γ are defined as above.

Substituting the Newmark- β finite difference relationships into equation (30),

$$\begin{aligned} M\ddot{x}_{i+1} &+ (1 - \alpha)C\{\dot{x}_i + h[(1 - \gamma)\ddot{x}_i + \gamma\ddot{x}_{i+1}]\} + \alpha C\dot{x}_i \\ &+ (1 - \alpha)K\{x_i + h\dot{x}_i + h^2[(1/2 - \beta)\ddot{x}_i + \beta\ddot{x}_{i+1}]\} + \alpha Kx_i \\ &= (1 - \alpha)f_{i+1}^{\text{ext}} + \alpha f_i^{\text{ext}}, \end{aligned} \quad (34)$$

and grouping terms, we obtain

$$\begin{aligned} [M + h(1 - \alpha)\gamma C + h^2(1 - \alpha)\beta K]\ddot{x}_{i+1} &+ [h(1 - \alpha)(1 - \gamma)C + h^2(1 - \alpha)(1/2 - \beta)K]\ddot{x}_i \\ &+ [C + h(1 - \alpha)K]\dot{x}_i \\ &+ Kx_i \\ &= (1 - \alpha)f_{i+1}^{\text{ext}} + \alpha f_i^{\text{ext}}. \end{aligned} \quad (35)$$

These are a set of linear equations for \ddot{x}_{i+1} in terms of \ddot{x}_i , \dot{x}_i , x_i , the external dynamic load, and the structure's mass, damping, and stiffness. Equations (21), (22), and (35) provide the recurrence relationship for the HHT- α method.

⁵Hughes, T.J.R., Analysis of Transient Algorithms with Particular Reference to Stability Behavior. in *Computational Methods for Transient Analysis*, North-Holland, 1983, pp. 67–155.

The HHT- α method — incremental formulation

The HHT- α method may be formulated in an incremental form, as was done with the Newmark- β method in the previous section. Again, formulating the finite difference relationships in terms of the increments of displacement ($\delta x_i = x_{i+1} - x_i$), velocity ($\delta \dot{x}_i = \dot{x}_{i+1} - \dot{x}_i$), acceleration ($\delta \ddot{x}_i = \ddot{x}_{i+1} - \ddot{x}_i$), and nonlinear restoring force, and difference between equation (30) (evaluated at time t_{i+1}) and the same expression evaluated at time t_i , results in

$$M\delta\ddot{x}_i + (1-\alpha)C\delta\dot{x}_i + \alpha C\delta\dot{x}_{i-1} + (1-\alpha)K\delta x_i + \alpha K\delta x_{i-1} + (1-\alpha)\delta R_i + \alpha\delta R_{i-1} = (1-\alpha)\delta f_i^{\text{ext}} + \alpha\delta f_{i-1}^{\text{ext}}. \quad (36)$$

Now, substituting in expressions for $\delta \dot{x}$ and $\delta \ddot{x}$ at times t_i and t_{i-1} , and grouping terms results in

$$\begin{aligned} \left[\frac{1}{\beta h^2} M + (1-\alpha) \frac{\gamma}{\beta h} C + (1-\alpha) K \right] \delta x_i &= \left[\frac{1}{\beta h} M + (1-\alpha) \frac{\gamma}{\beta} C \right] \dot{x}_i \\ &+ \left[\frac{1}{2\beta} M - (1-\alpha) h \left(1 - \frac{\gamma}{2\beta} \right) C \right] \ddot{x}_i \\ &- \alpha \left[\frac{\gamma}{\beta h} C + K \right] \delta x_{i-1} \\ &+ \alpha \frac{\gamma}{\beta} C \dot{x}_{i-1} - \alpha h \left(1 - \frac{\gamma}{2\beta} \right) C \ddot{x}_{i-1} \\ &- (1-\alpha)\delta R_i - \alpha\delta R_{i-1} \\ &+ (1-\alpha)\delta f_i + \alpha\delta f_{i-1}. \end{aligned} \quad (37)$$

If $R(x, \dot{x}) = 0$, these are a set of linear equations for δx_i in terms of \dot{x}_i , \ddot{x}_i , δx_{i-1} , \dot{x}_{i-1} , \ddot{x}_{i-1} , δx_{i-1} , the external dynamic load, and the structure's mass, damping, and stiffness. Equations (21), (22), and (37) provide the recurrence relationship for the HHT- α method in incremental form. Solving this linear system of equations⁶ for δx_i , the displacements are updated with,

$$x_{i+1} = x_i + \delta x_i, \quad (38)$$

the velocities are updated with

$$\dot{x}_{i+1} = \left(1 - \frac{\gamma}{\beta} \right) \dot{x}_i + \frac{\gamma}{\beta h} \delta x_i + h \left(1 - \frac{\gamma}{2\beta} \right) \ddot{x}_i. \quad (39)$$

and the accelerations satisfy the HHT- α form of the equations of motion,

$$\ddot{x}_{i+1} = -M^{-1} \left[(1-\alpha)C\dot{x}_{i+1} + \alpha C\dot{x}_i + (1-\alpha)Kx_{i+1} + \alpha Kx_i - (1-\alpha)f_{i+1}^{\text{ext}} - \alpha f_i^{\text{ext}} \right]. \quad (40)$$

This algorithm is explicit *regardless* of the values of α , β , or γ .

⁶If the incremental nonlinear restoring forces δR_i are not negligible, equation (37) is solved using the Newton-Raphson method, equation (46).

Simulation of SDOF Bilinear Hysteretic Behavior

Bilinear hysteretic behavior may be simulated using linear dynamic models. The basic idea is that each branch of the hysteresis may be described by an equation of the form $R = K(x - d)$, where the restoring force is zero when $x = d$ and K will take the value of the pre-yield stiffness or the strain hardening stiffness. The procedure is as follows:

1. Initialize K using the pre-yield stiffness K_{hi} . Initialize $x_0, \dot{x}_0, \ddot{x}_0, f_0^{\text{ext}}$, and d to be zero at time $t_0 = 0$. The value d is the equilibrium displacement of the bilinear hysteretic system.
2. At time $t = t_{i+1}$, read the value of the force f_{i+1}^{ext} , solve equation (26) for δx_i , and calculate x_{i+1} and \dot{x}_{i+1} using equations (27) and (28).
3. Compute the yielding force level from

$$f^{\text{yield}} = K_{lo}x_{i+1} + (K_{hi} - K_{lo}) x_y \text{sgn}(\dot{x}_{i+1}) \quad (41)$$

4. Check for yielding:

If $K = K_{hi}$ and $\dot{x}_{i+1} > 0$ and $K(x_{i+1} - d) > f^{\text{yield}}$,
then the system has just exceeded its positive yield force level.
Set $K = K_{lo}$ and $d = x_y(1 - K_{hi}/K_{lo})$.

If $K = K_{hi}$ and $\dot{x}_{i+1} < 0$ and $K(x_{i+1} - d) < f^{\text{yield}}$,
then the system has just exceeded its negative yield force level.
Set $K = K_{lo}$ and $d = x_y(K_{hi}/K_{lo} - 1)$.

5. Check for load reversal:

If $K = K_{lo}$ and $\dot{x}_{i+1}\dot{x}_i < 0$,
then the velocity has changed directions.
Set $K = K_{hi}$ and $d = x_{i+1} - (K_{lo}/K_{hi})(x_{i+1} - d)$

6. Compute \ddot{x}_{i+1} using

$$\ddot{x}_{i+1} = -M^{-1}[C\dot{x}_{i+1} + K(x_{i+1} - d) - f_{i+1}^{\text{ext}}]. \quad (42)$$

7. Increment the time step. $i = i + 1$.
8. Go back to step 2 and repeat until the end of the time history.

Newton-Raphson Iterative Solution for Nonlinear Behavior

Equations (26) or (37) may be written $\tilde{K} \delta x_i = \tilde{f}_i$. If the nonlinear restoring force increment δR_i is not negligible, then \tilde{f} depends on δx and $\delta \dot{x}$. For the case of the linear acceleration method ($\alpha = 0$, $\beta = 1/6$, $\gamma = 1/2$),

$$\tilde{K} = \left[\frac{6}{h^2} M + \frac{3}{h} C + K \right], \quad (43)$$

and

$$\tilde{f}_i(\delta x_i) = \delta f_i^{\text{ext}} - R(x_i + \delta x_i, \dot{x}_i + \delta \dot{x}_i) + R(x_i, \dot{x}_i) + \left[3M + \frac{h}{2} C \right] \ddot{x}_i + \left[\frac{6}{h} M + 3C \right] \dot{x}_i. \quad (44)$$

The Newton-Raphson algorithm⁷ is an efficient method to solve the nonlinear equations $\tilde{K} \delta x_i = \tilde{f}_i(\delta x_i)$ for the incremental displacement δx_i . The algorithm is iterative and proceeds as follows:

1. The initial value for δx_i is denoted $\delta x_i^{(0)}$ and is arbitrarily set to zero. The corresponding value for \tilde{f}_i is

$$\tilde{f}_i(\delta x_i^{(0)}) = \delta f_i^{\text{ext}} + \left[3M + \frac{h}{2} C \right] \ddot{x}_i + \left[\frac{6}{h} M + 3C \right] \dot{x}_i \quad (45)$$

2. The Newton-Raphson recurrence relation is simply

$$\delta x_i^{(n+1)} = \tilde{K}^{-1} \tilde{f}_i(\delta x_i^{(n)}) \quad (46)$$

where

$$\tilde{f}_i(\delta x_i^{(n)}) = \tilde{f}_i(\delta x_i^{(0)}) - R(x_i + \delta x_i^{(n)}, \dot{x}_i + \delta \dot{x}_i^{(n)}) + R(x_i, \dot{x}_i), \quad (47)$$

and

$$\delta \dot{x}_i^{(n)} = \frac{3}{h} \delta x_i^{(n)} - 3\dot{x}_i - \frac{h}{2} \ddot{x}_i. \quad (48)$$

3. Equation (46) is iterated upon until the incremental displacement δx_i converges, i.e., $\|\delta x_i^{(n+1)} - \delta x_i^{(n)}\| < \epsilon$.

The convergence of this form of the Newton-Raphson method depends on the local smoothness of $R(x, \dot{x})$. Convergence will be improved, for a particular time step, by making h smaller.

Note that \tilde{K} is strongly positive definite; it does not depend on δx_i ; and it is inverted or factorized only once at the beginning of the simulation. If convergence is problematic, then a set of \tilde{K} matrices may be formed, some with small time steps, and factorized prior to the simulation.

As an alternative to Newton-Raphson iterations, the approximation $\delta R_i \approx \delta R_{i-1}$ results in an explicit formulation for the simulation of nonlinear behavior, and can significantly reduce computational time, but with some loss of accuracy.

⁷Press, W.H., *et al.*, *Numerical Recipes*, Cambridge Univ. Press, 1993. <http://www.nr.com/>

State Variable Formulations

The second order ordinary differential equations of motion for a structural system are

$$M\ddot{r}(t) + C\dot{r}(t) + Kr(t) + R(r(t), \dot{r}(t)) = f^{\text{ext}}(t), \quad (49)$$

where we have re-named the displacement vector $x(t)$ of equation (1) with $r(t)$. This set of N second order differential equations may be written as a set of $2N$ first order differential equations, as follows:

$$\frac{d}{dt} \begin{bmatrix} r \\ \dot{r} \end{bmatrix} = \begin{bmatrix} 0_{N \times N} & I_{N \times N} \\ -M^{-1}K & -M^{-1}C \end{bmatrix} \begin{bmatrix} r \\ \dot{r} \end{bmatrix} + \begin{bmatrix} 0_{N \times 1} \\ -M^{-1}R(r, \dot{r}) \end{bmatrix} + \begin{bmatrix} 0_{N \times N} \\ M^{-1} \end{bmatrix} f^{\text{ext}}(t). \quad (50)$$

The vector $[r^T \ \dot{r}^T]^T$ is called the *state vector*; along with the external forces, it can completely describe the state of the system at any point in time. State-space models are conventionally written as

$$\dot{x} = Ax + g(x) + Bu, \quad (51)$$

where x is the *state vector*, A is the *dynamics matrix*, $g(x)$ contains any nonlinear terms in the equations of state, B is the *input matrix*, and u is the *input* to the system. For a linear structure, $g(x) = 0$ and the eigenvalues of A contain the complex *poles* of the system. For under-damped dynamic systems the poles are complex-valued. The imaginary part of the pole is the damped natural frequency, and the real part of the pole is the negative of the natural frequency times the damping ratio.

A convenient feature of state-variable formulations is that first order systems can be easily combined. If components of the structural system have first-order dynamics (for example, visco-elasticity or Bouc-Wen hysteresis), then the first order dynamics of those components can be appended to the state vector, and the simulation of the second order structural system coupled with the first order structural components can proceed in parallel.

The state variable formulation is the most general description of a dynamic system, and many methods exist for the simulation of state-variable models. A set of these methods has been implemented in MATLAB. See, for example, the MATLAB command `ode45`.

For large second-order systems ($N > 1000$), however, the HHT- α method is typically more efficient in terms of memory and speed.

A Fixed-Step Fourth-Order Runge-Kutta Solver

In the computation of transient response analyses of linear or nonlinear systems, it is common for the external forcing to be sampled at uniformly-spaced increments in time, and to desire the response to be computed at these same points in time. If the sample interval $h = \Delta t$ is less than one-tenth of the shortest natural period involved in the system (over the course of the simulation) then a fixed time-step Runge-Kutta solver can compute a solution quickly, and with reasonable accuracy.⁸ The most famous of these solvers is fourth-order accurate. The goal of the solver is to advance the solution, x , of a general system of

⁸Press, W.H., *et al.*, *Numerical Recipes*, Cambridge Univ. Press, 1993. Section 16.1 <http://www.nr.com/>

non-homogeneous first-order ordinary differential equations

$$\dot{x}(t) = f(t, x(t), u(t)) , \quad x(0) = x_o , \quad (52)$$

from time t to time $t+\Delta t$. Given the state vector $x(t)$, we endeavour to find the state $x(t+\Delta t)$. The solution scheme is illustrated in Figure 1. Within each time-step, the algorithm evaluates

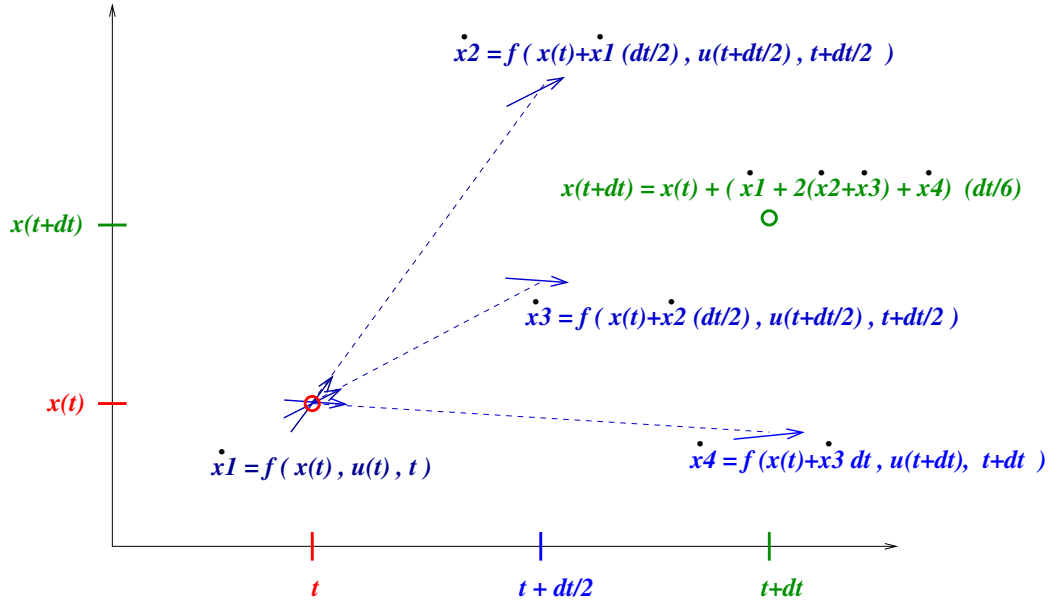


Figure 1. Solution Scheme of the Fourth Order Runge-Kutta Method.

\dot{x} four times.

$$\dot{x}_1 = f(t, x(t), u(t)) \quad (53)$$

$$\dot{x}_2 = f\left(t + \frac{\Delta t}{2}, x(t) + \dot{x}_1 \frac{\Delta t}{2}, u\left(t + \frac{\Delta t}{2}\right)\right) \quad (54)$$

$$\dot{x}_3 = f\left(t + \frac{\Delta t}{2}, x(t) + \dot{x}_2 \frac{\Delta t}{2}, u\left(t + \frac{\Delta t}{2}\right)\right) \quad (55)$$

$$\dot{x}_4 = f(t + \Delta t, x(t) + \dot{x}_3 \Delta t, u(t + \Delta t)) \quad (56)$$

The desired solution, $x(t + \Delta t)$, is computed from the four state derivatives

$$x(t + \Delta t) = x(t) + (\dot{x}_1 + 2(\dot{x}_2 + \dot{x}_3) + \dot{x}_4) \frac{\Delta t}{6} , \quad (57)$$

time is incremented, $t = t + \Delta t$, and the solution marches on. The solution scheme is implemented in the MATLAB function `ode4u.m`, <http://www.duke.edu/~hpgavin/ode4u.m>

ode4u.m

```

1 function [time, x_sol, x_drv] = ode4u( dxdt, time, x0, u, params )
2 % [time, x_sol, x_drv] = ode4u( dxdt, time, x0, u, params )
3 %
4 % Solve a system of nonhomogeneous ordinary differential equations using the
5 % 4th order Runge-Kutta method.
6 %
7 % Input Variable      Description
8 % -----
9 % dxdt                : a function of the form x_dot = dxdt(t,x,u,params)
10 %                    which provides the state derivative given the time, t,
11 %                    the state, x, external forcing, u, and parameters
12 % time                : a vector of points in time at which the solution
13 %                    is computed
14 % x0                  : initial values of the states - a column vector
15 % u                   : m-by-p matrix of external forcing for the ode, where
16 %                    m = number of inputs and p = length(time);
17 % params              : vector of optional parameters for the function dxdt
18 %
19 % Output Variable     Description
20 % -----
21 % time                : is returned, un-changed
22 % x_sol               : the solution to the differential equation at
23 %                    times given in the vector time.
24 % x_drv               : the derivative of the solution to the differential
25 %                    equation at times given in the vector time.
26 %
27 % Henri Gavin, Civil and Environmental Engineering, Duke University, Feb. 2007
28 % WH Press, et al, Numerical Recipes in C, 1992, Section 16.1
29
30 points = length(time);
31 n      = length(x0(:));
32
33 if nargin < 5, params = 0;      end
34 if nargin < 4, u = zeros(1,points);      end
35
36 x_sol = zeros(n,points);      % memory allocation
37 x_drv = zeros(n,points);      % memory allocation
38 x_sol(:,1) = x0(:);          % the initial conditions
39
40 [ru,cu] = size(u);
41 if cu < points
42     u(:,cu+1:points) = 0;      % pad u with zeros if it is not long enough
43 end
44
45 for p = 1:points-1
46
47     t      = time(p);
48     dt     = time(p+1)-t;      % the time step for this interval
49     dt2    = dt/2;            % half of the time step
50
51     dxdt1  = feval( dxdt, t,      x0, u(:,p), params );
52     dxdt2  = feval( dxdt, t + dt2, x0 + dxdt1*dt2, (u(:,p)+u(:,p+1))/2.0, params );
53     dxdt3  = feval( dxdt, t + dt2, x0 + dxdt2*dt2, (u(:,p)+u(:,p+1))/2.0, params );
54     dxdt4  = feval( dxdt, t + dt,  x0 + dxdt3*dt, u(:,p+1), params );
55
56     x0 = x0 + ( dxdt1 + 2*(dxdt2 + dxdt3) + dxdt4 )*dt/6;
57
58     x_sol(:,p+1) = x0(:);
59     x_drv(:,p)   = dxdt1(:);
60
61 end
62 x_drv(:,p+1) = dxdt1(:);
63
64 % ----- ODE4U

```

A Variable-Step Fourth-Fifth-Order Runge-Kutta Solver

For problems in which the time increments are not much shorter than the shortest natural periods involved in the system, and for problems involving hard nonlinearities, such as problems involving impact or friction, fixed-step Runge-Kutta solvers may be numerically unstable. In such cases one may attempt a re-analysis using a shorter time step for every time step, or one may use a solver in which the large fixed time-steps are automatically subdivided only when necessary. A feature of such variable time-step solvers is that a desired level of accuracy may be enforced throughout the simulation, by comparing a fourth-order accurate solution with a fifth-order accurate solution.

The implementation of the fourth-fifth order solver discussed here was proposed by Cash and Karp in 1990.⁹ To advance the solution from a time t to a time $t + \Delta t$, six state derivatives are computed

$$\dot{x}_1 = f(t + a_1\Delta t, x(t), u(t + a_1\Delta t)) \quad (58)$$

$$\dot{x}_2 = f(t + a_2\Delta t, x(t) + \dot{x}_1 b_{21}\Delta t, u(t + a_2\Delta t)) \quad (59)$$

$$\dot{x}_3 = f(t + a_3\Delta t, x(t) + (\dot{x}_1 b_{31} + \dot{x}_2 b_{32})\Delta t, u(t + a_3\Delta t)) \quad (60)$$

$$\dot{x}_4 = f(t + a_4\Delta t, x(t) + (\dot{x}_1 b_{41} + \dot{x}_2 b_{42} + \dot{x}_3 b_{43})\Delta t, u(t + a_4\Delta t)) \quad (61)$$

$$\dot{x}_5 = f(t + a_5\Delta t, x(t) + (\dot{x}_1 b_{51} + \dot{x}_2 b_{52} + \dot{x}_3 b_{53} + \dot{x}_4 b_{54})\Delta t, u(t + a_5\Delta t)) \quad (62)$$

$$\dot{x}_6 = f(t + a_6\Delta t, x(t) + (\dot{x}_1 b_{61} + \dot{x}_2 b_{62} + \dot{x}_3 b_{63} + \dot{x}_4 b_{64} + \dot{x}_5 b_{65})\Delta t, u(t + a_6\Delta t)) \quad (63)$$

The fourth order predictor at time $t + \Delta t$, is $x_4(t + \Delta t)$, and the fifth order predictor at time $t + \Delta t$, is $x_5(t + \Delta t)$.

$$x_4(t + \Delta t) = x(t) + (\dot{x}_1 c_{41} + \dot{x}_3 c_{43} + \dot{x}_4 c_{44} + \dot{x}_5 c_{45} + \dot{x}_6 c_{46})\Delta t \quad (64)$$

$$x_5(t + \Delta t) = x(t) + (\dot{x}_1 c_{51} + \dot{x}_3 c_{53} + \dot{x}_4 c_{54} + \dot{x}_6 c_{56})\Delta t \quad (65)$$

The coefficients a_i , b_{ij} , and c_{ij} are provided in Cash and Karp's paper. If every term of the truncation error vector, $\epsilon = |x_4 - x_5|/|x_5|$ is less than the required tolerance, ϵ_{tol} , then the solution x_5 is considered sufficiently accurate. In this case, $x(t + \Delta t)$ is assigned to x_5 , t is assigned to $t + \Delta t$, and the solution marches on. Otherwise, the time step is divided into N_{ss} sub-steps,¹⁰ where

$$N_{\text{ss}} = \left\lceil N_{\text{ss}} \max \left[\max \left((\epsilon/\epsilon_{\text{tol}})^{1/4}, 1.1 \right), 1 \right] \right\rceil, \quad (66)$$

and N_{ss} is initialized to 1, and $\lceil \cdot \rceil$ is the round-up operator.

These concepts are implemented in the MATLAB function `ode45u.m`¹¹

⁹Cash, J.R. and Karp, A.H., "A Variable Order Runge-Kutta Method for Initial Value Problems with Rapidly Varying Right-Hand Sides," *ACM Transactions on Mathematical Software*, vol 16, no 3, pp 201-222 (1990). <http://www.duke.edu/~hpgavin/ce283/Cash-90.pdf>

¹⁰Press, W.H., *et al.*, *Numerical Recipes*, Cambridge Univ. Press, 1993. Section 16.2 <http://www.nr.com/>

¹¹<http://www.duke.edu/~hpgavin/ode45u.m>

