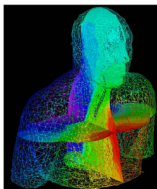
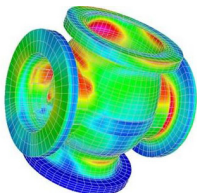


The Finite Element Method for the Analysis of Non-Linear and Dynamic Systems: Non-Linear Dynamics Part I

Prof. Dr. Eleni Chatzi

Dr. Giuseppe Abbiati, Dr. Konstantinos Agathos

Lecture 5/Part A - 23 November, 2017



Learning Goals

- To recall the equation of motion for a linear and elastic system.
- Learn how to use eigenvalue analysis for reducing the dimension of the system to be solved.
- Use Direct Integration Methods for solving the ordinary differential equation of motion.
- Focus Study: The Newmark Method.
- Nonlinear Implementation of The Newmark Method.

Reference:

M.A. Dokainish, K. Subbaraj, A survey of direct time-integration methods in computational structural dynamicsI. Explicit methods, In Computers & Structures, Volume 32 : 6, pp. 1371 – 1386, 1989.

Dynamic Equation of Motion - Initial Value Problem (IVP):

$$\begin{cases} \mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{C}\dot{\mathbf{u}}(t) + \mathbf{K}\mathbf{u}(t) = \mathbf{f}(t) \\ \mathbf{u}(t_0) = \mathbf{u}_0, \dot{\mathbf{u}}(t_0) = \dot{\mathbf{u}}_0 \end{cases}$$

or alternatively:

$$\mathbf{F}_I(t) + \mathbf{F}_D(t) + \mathbf{F}_E(t) = \mathbf{f}(t)$$

where:

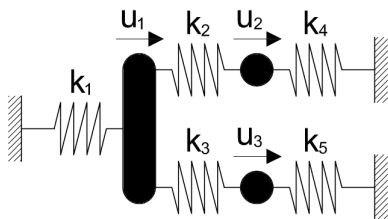
M : Mass Matrix
K : Stiffness Matrix
u : Displacements
u : Velocities
u : Accelerations
f : external force vector

where:

F_I(t) = Mu(t) Inertial Force
F_D(t) = Cu(t) Damping Force
F_E(t) = Ku(t) Internal Force

Introduction to Dynamic Analysis

Example: 3-dof system



$$\mathbf{M} = \begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix}, \mathbf{K} = \begin{bmatrix} k_1 + k_2 + k_3 & -k_2 & -k_3 \\ -k_2 & k_2 + k_4 & 0 \\ -k_3 & 0 & k_3 + k_5 \end{bmatrix}$$

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}, \mathbf{f}(t) = \begin{bmatrix} f_1(t) \\ f_2(t) \\ f_3(t) \end{bmatrix}$$

When is Dynamic Analysis required in Structural Engineering?

- The decision on carrying out a dynamic structural analysis is up to engineering judgment. For a number of problems, despite variations of loads, a static or pseudo-static analysis may be admissible.
- In general, if the loading varies over time with frequencies higher than the eigen-frequencies of the structure, \mapsto a dynamic analysis will be required.

Objective

How to numerically solve the original dynamic Equation of motion or the modal set of equations for non-proportional damping?

$$\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{C}\dot{\mathbf{u}}(t) + \mathbf{K}\mathbf{u}(t) = \mathbf{f}(t)$$

In principle, the equilibrium equations may be solved by any standard numerical integration scheme **BUT!**

Efficient numerical efforts must be considered and it is worthwhile to investigate dedicated techniques of integration, particularly aimed for the analysis of finite element assemblies.

Direct Integration Methods

These methods rely in discretizing the continuous problem. For given initial conditions at time zero, we attempt to satisfy dynamic equilibrium at discrete points in time.

Most methods use equal time intervals. However, this is not mandatory; in some cases a variable time step might be employed. This is most commonly the case for special classes of problems such as Impact Problems.

Direct implies: The equations are solved in their **original** form, with two main ideas utilized:

1. The equilibrium equations are satisfied only at time steps, i.e., at discrete times with intervals Δt
2. A particular variation of displacements, velocities and accelerations within each time interval is assumed.

Discretization of the IVP with a time step $\Delta t = t_{k+1} - t_k$:

$$\begin{cases} \mathbf{M}\ddot{\mathbf{u}}_{k+1} + \mathbf{C}\dot{\mathbf{u}}_{k+1} + \mathbf{K}\mathbf{u}_{k+1} = \mathbf{f}(t_{k+1}) \\ \mathbf{u}(t_0) = \mathbf{u}_0, \dot{\mathbf{u}}(t_0) = \dot{\mathbf{u}}_0 \end{cases}$$

where,

- \mathbf{M} : mass matrix.
- \mathbf{C} : damping.
- \mathbf{K} : stiffness matrix.
- \mathbf{f} : external force vector.
- $\mathbf{u}_{k+1}, \dot{\mathbf{u}}_{k+1}, \ddot{\mathbf{u}}_{k+1}$: displacement, velocity and acceleration vectors at t_{k+1} .

Direct Integration Methods

Accuracy depends on the previously defined assumptions as well as the choice of time intervals!

These methods come in two main categories:

- 1 Explicit methods - the right hand side of the discretized equations of motion exclusively employs variables from the previous time instant k
- 2 Implicit methods - the right hand side of the discretized equations of motion exclusively includes variables from the current time instant $k + 1$

Note that:

Implicit integration is not necessarily more accurate than explicit! The major benefit of implicit integration is stability. Many of these methods are able to run with any arbitrarily large time step, for any input, unless we are lying at the limits of floating point math (unconditionally stable). Obviously a large time step implies throwing away accuracy.

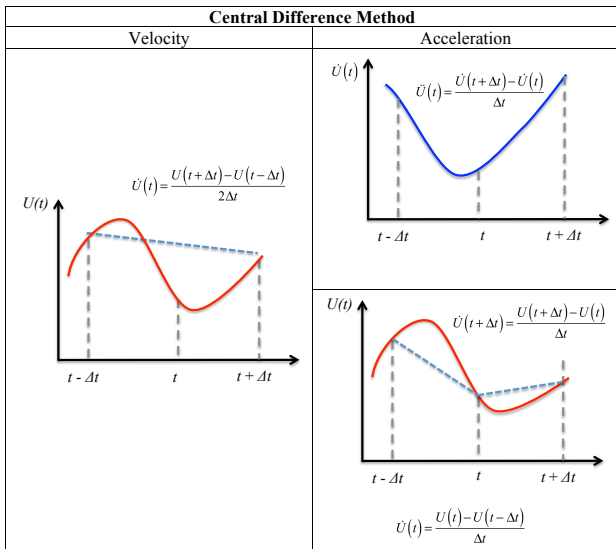
Most Commonly used Direct Integration Methods

(for the case of the Dynamic Equation of Motion)

- 1 The Central Difference Method (CDF)
- 2 The Houbolt method
- 3 The Newmark method
- 4 The Wilson θ method
- 5 Coupling of integration operators

The difference in items 1-4 lies in the way we choose a discretized equivalent of the derivatives. The overall setup of the solution is very much similar for all methods. Additionally depending on the resulting equations some schemes are explicit (CDF) and others implicit (Houbolt, Newmark, Wilson θ)

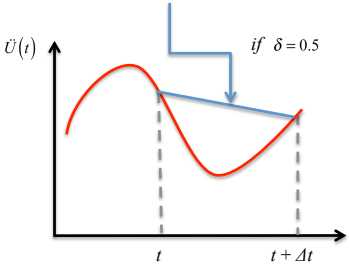
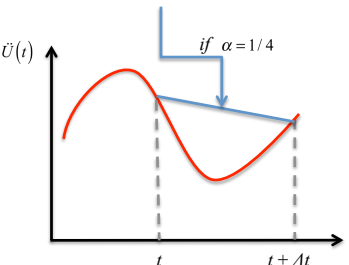
Most Commonly used Direct Integration Methods



Most Commonly used Direct Integration Methods

The Houbolt Method	
Displacement	Velocity
$U_t = U_{t+\Delta t} - \Delta t \dot{U}_{t+\Delta t} + \frac{\Delta t^2}{2} \ddot{U}_{t+\Delta t} - \frac{\Delta t^3}{6} \dddot{U}_{t+\Delta t}$ $U_{t-\Delta t} = U_{t+\Delta t} - 2\Delta t \dot{U}_{t+\Delta t} + \frac{(2\Delta t)^2}{2} \ddot{U}_{t+\Delta t} - \frac{(2\Delta t)^3}{6} \dddot{U}_{t+\Delta t}$	$\dot{U}_{t+\Delta t} = \frac{1}{6\Delta t} (11U_{t+\Delta t} - 18U_t + 9U_{t-\Delta t} - 2U_{t-2\Delta t})$
	Acceleration
	$\ddot{U}_{t+\Delta t} = \frac{1}{\Delta t^2} (2U_{t+\Delta t} - 5U_t + 4U_{t-\Delta t} - U_{t-2\Delta t})$ <p>These approximations are derived via the displacement approximation.</p>
<p>Houbolt's method uses a third-order interpolation of displacements extending two steps back in time.</p>	

Most Commonly used Direct Integration Methods

The Newmark Method	
Displacement	Velocity
The Newmark method uses a second order Taylor expansion for approximating Velocities and Accelerations: $\dot{U}_{t+\Delta t} = \dot{U}_t + \Delta t \ddot{U}_v$ Accelerations: $U_{t+\Delta t} = U_t + \Delta t \dot{U}_v + \frac{\Delta t^2}{2} \ddot{U}_D$, where \ddot{U}_v, \ddot{U}_D are approximations of the Acceleration	
$\ddot{U}_v = (1-\delta)\ddot{U}_t + \delta\ddot{U}_{t+\Delta t}$  <p>The graph shows acceleration $\ddot{U}(t)$ on the vertical axis and time on the horizontal axis. A red curve represents the true acceleration. A blue line segment connects the points (t, \ddot{U}_t) and $(t+\Delta t, \ddot{U}_{t+\Delta t})$. A blue arrow points from the equation above to this line segment with the text "if $\delta = 0.5$".</p>	$\ddot{U}_D = (1-2\alpha)\ddot{U}_t + 2\alpha\ddot{U}_{t+\Delta t}$  <p>The graph shows acceleration $\ddot{U}(t)$ on the vertical axis and time on the horizontal axis. A red curve represents the true acceleration. A blue line segment connects the points (t, \ddot{U}_t) and $(t+\Delta t, \ddot{U}_{t+\Delta t})$. A blue arrow points from the equation above to this line segment with the text "if $\alpha = 1/4$".</p>

γ and β are used in place of δ and α , respectively, in the following slides.

Stability/Accuracy of DIMs

Integration Method	Type of Method	Critical Step Size (Δt_{cr})
Central Different	Explicit	$\frac{2}{\omega}$
Newmark Method $\gamma = \frac{1}{2}, \beta = \frac{1}{6}$ (Linear Acceleration)	Implicit	$\frac{3.464}{\omega}$
Newmark Method $\gamma = \frac{1}{2}, \beta = \frac{1}{4}$ (Constant-Average-Acceleration)	Implicit	Unconditionally Stable
Newmark Method $\gamma = \frac{1}{2}, \beta = 0$ (Central Difference)	Explicit	$\frac{2}{\omega}$
Wilson- θ	Implicit	Unconditionally Stable when $\theta \geq 1.37$

Focus Case: The Newmark Algorithm

The Newmark method is the most widely used multi-step time integration algorithm for structural analysis:

Discretization of the IVP with a time step $\Delta t = t_{k+1} - t_k$:

$$\begin{cases} \mathbf{M}\ddot{\mathbf{u}}_{k+1} + \mathbf{C}\dot{\mathbf{u}}_{k+1} + \mathbf{K}\mathbf{u}_{k+1} = \mathbf{f}(t_{k+1}) \\ \mathbf{u}(t_0) = \mathbf{u}_0, \dot{\mathbf{u}}(t_0) = \dot{\mathbf{u}}_0 \end{cases}$$

Interpolation equations (Newmark, 1959):

$$\begin{cases} \mathbf{u}_{k+1} = \mathbf{u}_k + \dot{\mathbf{u}}_k \Delta t + \ddot{\mathbf{u}}_k \left(\frac{1}{2} - \beta\right) \Delta t^2 + \ddot{\mathbf{u}}_{k+1} \beta \Delta t^2 \\ \dot{\mathbf{u}}_{k+1} = \dot{\mathbf{u}}_k + \ddot{\mathbf{u}}_k (1 - \gamma) \Delta t + \ddot{\mathbf{u}}_{k+1} \gamma \Delta t \end{cases}$$

where β and γ are the parameters of the time integration algorithm.

Newmark, N. M. (1959). A method of computation for structural dynamics. Journal of the engineering mechanics division, 85(3), 67–94.

Remark:

γ and β are parameters, effectively acting as weights for calculating the approximation of the acceleration, and may be adjusted to balance accuracy and stability.

Parameter $\gamma = 1/2$ ensures second order accuracy whilst,

- $\beta = 0$ makes the algorithm explicit and equivalent to the central difference method.
- $\beta = 1/4$ makes the algorithm implicit and equivalent to the trapezoidal rule (unconditionally stable).
- $\gamma = 1/2, \beta = 1/6$ is known as the linear acceleration method, which also correspond to the Wilson θ method with $\theta = 1$

Linear Implementation of the Newmark Algorithm

A **linear problem** is solved at each time step:

$$\mathbf{M}\ddot{\mathbf{u}}_{k+1} + \mathbf{C}\left(\dot{\tilde{\mathbf{u}}}_{k+1} + \ddot{\mathbf{u}}_{k+1}\gamma\Delta t\right) + \mathbf{K}\left(\tilde{\mathbf{u}}_{k+1} + \ddot{\mathbf{u}}_{k+1}\beta\Delta t^2\right) = \mathbf{f}(t_{k+1})$$

Predictors depend on previous time step solutions:

$$\begin{cases} \tilde{\mathbf{u}}_{k+1} = \mathbf{u}_k + \dot{\mathbf{u}}_k\Delta t + \ddot{\mathbf{u}}_k\left(\frac{1}{2} - \beta\right)\Delta t^2 \\ \dot{\tilde{\mathbf{u}}}_{k+1} = \dot{\mathbf{u}}_k + \ddot{\mathbf{u}}_k(1 - \gamma)\Delta t \end{cases}$$

Correctors determine the current time step solution:

$$\begin{cases} \mathbf{u}_{k+1} = \tilde{\mathbf{u}}_{k+1} + \ddot{\mathbf{u}}_{k+1}\beta\Delta t^2 \\ \dot{\mathbf{u}}_{k+1} = \dot{\tilde{\mathbf{u}}}_{k+1} + \ddot{\mathbf{u}}_{k+1}\gamma\Delta t \end{cases}$$

Implicit/explicit algorithm:

$$1: \tilde{\mathbf{u}}_{k+1} \leftarrow \mathbf{u}_k + \dot{\mathbf{u}}_k \Delta t + \ddot{\mathbf{u}}_k \left(\frac{1}{2} - \beta\right) \Delta t^2$$

$$2: \tilde{\dot{\mathbf{u}}}_{k+1} \leftarrow \dot{\mathbf{u}}_k + \ddot{\mathbf{u}}_k (1 - \gamma) \Delta t$$

$$3: \ddot{\mathbf{u}}_{k+1} \leftarrow (\mathbf{M} + \mathbf{C}\gamma\Delta t + \mathbf{K}\beta\Delta t^2)^{-1} \left(\mathbf{f}(t_{k+1}) - \mathbf{C}\tilde{\dot{\mathbf{u}}}_{k+1} - \mathbf{K}\tilde{\mathbf{u}}_{k+1} \right)$$

$$4: \dot{\mathbf{u}}_{k+1} \leftarrow \tilde{\dot{\mathbf{u}}}_{k+1} + \ddot{\mathbf{u}}_{k+1} \gamma \Delta t$$

$$5: \mathbf{u}_{k+1} \leftarrow \tilde{\mathbf{u}}_{k+1} + \ddot{\mathbf{u}}_{k+1} \beta \Delta t^2$$

- $\beta = 0$: explicit algorithm
- $\beta \neq 0$: implicit algorithm \rightarrow inversion of \mathbf{K}

Step #1: calculation of **predictors** :

$$\begin{cases} \tilde{\mathbf{u}}_{k+1} = \mathbf{u}_k + \dot{\mathbf{u}}_k \Delta t + \ddot{\mathbf{u}}_k \left(\frac{1}{2} - \beta \right) \Delta t^2 \\ \dot{\tilde{\mathbf{u}}}_{k+1} = \dot{\mathbf{u}}_k + \ddot{\mathbf{u}}_k (1 - \gamma) \Delta t \end{cases}$$

Step #1: calculation of **predictors** :

$$\begin{cases} \tilde{\mathbf{u}}_{k+1} = \mathbf{u}_k + \dot{\mathbf{u}}_k \Delta t + \ddot{\mathbf{u}}_k \left(\frac{1}{2} - \beta\right) \Delta t^2 \\ \dot{\tilde{\mathbf{u}}}_{k+1} = \dot{\mathbf{u}}_k + \ddot{\mathbf{u}}_k (1 - \gamma) \Delta t \end{cases}$$

Step #2: solution of the **linear problem** :

$$\ddot{\mathbf{u}}_{k+1} = (\mathbf{M} + \mathbf{C}\gamma\Delta t + \mathbf{K}\beta\Delta t^2)^{-1} \left(\mathbf{f}(t_{k+1}) - \mathbf{K}\tilde{\mathbf{u}}_{k+1} - \mathbf{C}\dot{\tilde{\mathbf{u}}}_{k+1} \right)$$

The Newmark Algorithm

Step #1: calculation of **predictors** :

$$\begin{cases} \tilde{\mathbf{u}}_{k+1} = \mathbf{u}_k + \dot{\mathbf{u}}_k \Delta t + \ddot{\mathbf{u}}_k \left(\frac{1}{2} - \beta\right) \Delta t^2 \\ \dot{\tilde{\mathbf{u}}}_{k+1} = \dot{\mathbf{u}}_k + \ddot{\mathbf{u}}_k (1 - \gamma) \Delta t \end{cases}$$

Step #2: solution of the **linear problem** :

$$\ddot{\mathbf{u}}_{k+1} = (\mathbf{M} + \mathbf{C}\gamma\Delta t + \mathbf{K}\beta\Delta t^2)^{-1} \left(\mathbf{f}(t_{k+1}) - \mathbf{K}\tilde{\mathbf{u}}_{k+1} - \mathbf{C}\dot{\tilde{\mathbf{u}}}_{k+1} \right)$$

Step #3: calculation of **correctors** :

$$\begin{cases} \mathbf{u}_{k+1} = \tilde{\mathbf{u}}_{k+1} + \ddot{\mathbf{u}}_{k+1} \beta \Delta t^2 \\ \dot{\mathbf{u}}_{k+1} = \dot{\tilde{\mathbf{u}}}_{k+1} + \ddot{\mathbf{u}}_{k+1} \gamma \Delta t \end{cases}$$

The Newmark Algorithm

Step #1: calculation of **predictors** :

$$\begin{cases} \tilde{\mathbf{u}}_{k+1} = \mathbf{u}_k + \dot{\mathbf{u}}_k \Delta t + \ddot{\mathbf{u}}_k \left(\frac{1}{2} - \beta\right) \Delta t^2 \\ \dot{\tilde{\mathbf{u}}}_{k+1} = \dot{\mathbf{u}}_k + \ddot{\mathbf{u}}_k (1 - \gamma) \Delta t \end{cases}$$

Step #2: solution of the **linear problem** :

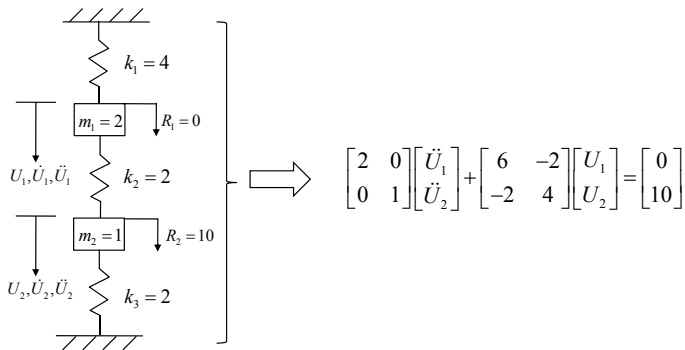
$$\ddot{\mathbf{u}}_{k+1} = (\mathbf{M} + \mathbf{C}\gamma\Delta t + \mathbf{K}\beta\Delta t^2)^{-1} \left(\mathbf{f}(t_{k+1}) - \mathbf{K}\tilde{\mathbf{u}}_{k+1} - \mathbf{C}\dot{\tilde{\mathbf{u}}}_{k+1} \right)$$

Step #3: calculation of **correctors** :

$$\begin{cases} \mathbf{u}_{k+1} = \tilde{\mathbf{u}}_{k+1} + \ddot{\mathbf{u}}_{k+1} \beta \Delta t^2 \\ \dot{\mathbf{u}}_{k+1} = \dot{\tilde{\mathbf{u}}}_{k+1} + \ddot{\mathbf{u}}_{k+1} \gamma \Delta t \end{cases}$$

- $\beta = 0$: explicit algorithm
- $\beta \neq 0$: implicit algorithm \rightarrow inversion of \mathbf{K}

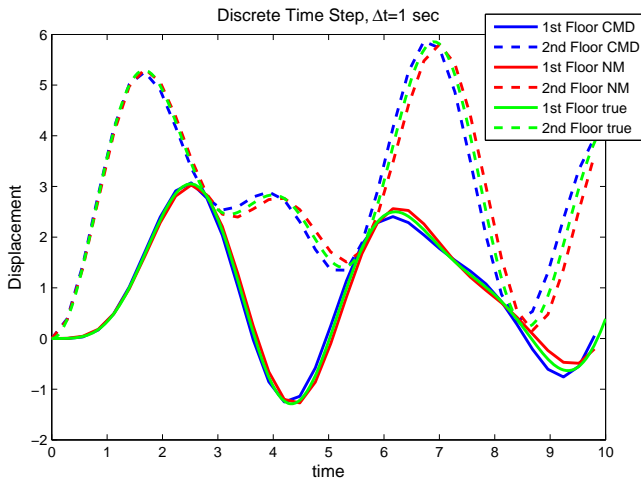
Example - 2 DOF system



For this system the natural periods are $T_1 = 4.45$, $T_2 = 2.8$

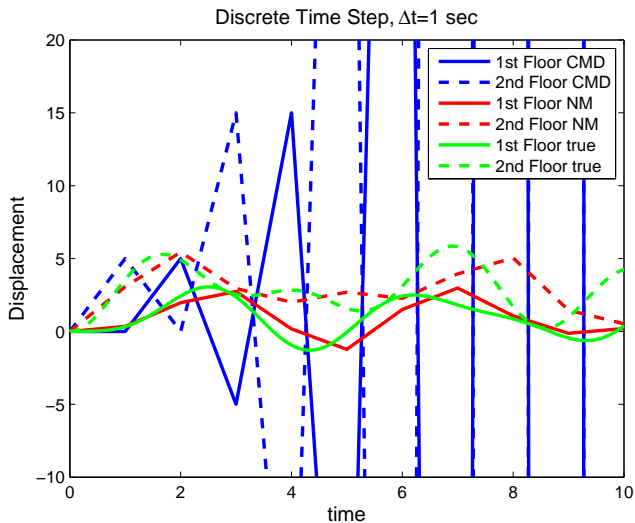
The Newmark Method

2dof system example $\Delta t = 0.28s$



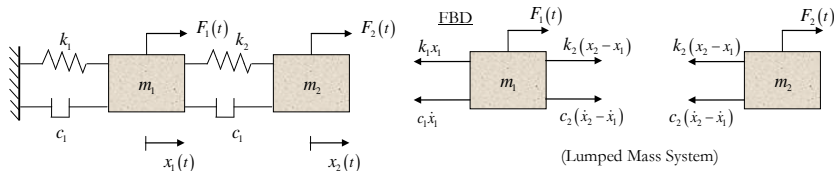
The Newmark Method

2dof system example $\Delta t = 1s$



State Space Equation Formulation

2dof Mass Spring System



We introduce the augmented state vector: $\mathbf{x} = [u_1 \quad u_2 \quad v_1 \quad v_2]^T$
(controllable form equivalent). Then,

$$\begin{bmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \dot{v}_1 \\ \dot{v}_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ [-\mathbf{m}^{-1}\mathbf{k}] & [-\mathbf{m}^{-1}\mathbf{c}] & & \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ v_1 \\ v_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ [\mathbf{m}^{-1}] & \end{bmatrix} \begin{bmatrix} f_1(t) \\ f_2(t) \end{bmatrix}$$
$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bp}(t)$$

where \mathbf{m} , \mathbf{c} and \mathbf{k} are mass, damping and stiffness matrix of the underlying second order mechanical system.

2dof Mass Spring System

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bp}(t)$$

Assume you would like to monitor the displacement x_1, x_2 . Then the "observation vector" is:

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \mathbf{0}_{4 \times 2} \mathbf{p}(t)$$
$$\mathbf{y} = \mathbf{Cx} + \mathbf{Dp}(t)$$

State Space Equation Formulation

From phase-space to state-space (nonlinear):

$$\mathbf{M}\dot{\mathbf{x}} + \mathbf{R}(\mathbf{x}) = \mathbf{F}(t)$$
$$\downarrow$$
$$\dot{\mathbf{x}} = \mathbf{M}^{-1} (\mathbf{F}(t) - \mathbf{R}(\mathbf{x})) = \mathbf{H}(\mathbf{x}, t)$$

with,

$$\mathbf{M} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{m} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}, \mathbf{x} = \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{s} \end{bmatrix}, \mathbf{R} = \begin{bmatrix} -\mathbf{v} \\ \mathbf{r}(\mathbf{u}, \mathbf{v}, \mathbf{s}) \\ \mathbf{g}(\mathbf{u}, \mathbf{v}, \mathbf{s}) \end{bmatrix}, \mathbf{F}(t) = \begin{bmatrix} \mathbf{0} \\ \mathbf{f}(t) \\ \mathbf{0} \end{bmatrix}$$

and,

- \mathbf{r} is the nonlinear restoring force vector that depends on displacement \mathbf{u} , velocity \mathbf{v} and additional state variables \mathbf{s} .
- \mathbf{g} is an additional nonlinear function that determines the evolution of the additional state variables \mathbf{s} .

State Space Equation Formulation

Using the state space representation we have converted a 2nd order ODE into an equivalent 1st order ODE system. We can now use any 1st order ODE integration method to convert the **continuous system** into a **discrete one** and obtain an approximate solution:

1st order ODE Integration Methods

Assume $\frac{d\mathbf{x}}{dt} = \mathbf{H}(\mathbf{x}(t), t)$, $\mathbf{x}(t_0) = \mathbf{0}$

- **Forward Euler Method**

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{H}(\mathbf{x}_k, t_k)\Delta t$$

where Δt is the integration time step. This **explicit** expression is obtained from the truncated Taylor Expansion of $\mathbf{x}(t_k + \Delta t)$

- **Backward Euler Method**

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{H}(\mathbf{x}_{k+1}, t_{k+1})\Delta t$$

This **implicit** expression (since \mathbf{x}_{k+1} is on the right hand side) is obtained from the truncated Taylor Expansion of $\mathbf{x}(t_{k+1} - \Delta t)$

- **2nd Order Runge Kutta (RK2)**

$$\begin{cases} \mathbf{k}_1 = \mathbf{H}(\mathbf{x}_k, t_k)\Delta t \\ \mathbf{k}_2 = \mathbf{H}(\mathbf{x}_k + \frac{1}{2}\mathbf{k}_1, t_k + \frac{1}{2}\Delta t)\Delta t \\ \mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{k}_2 + O(\Delta t^3) \end{cases}$$

- **4th Order Runge Kutta (RK4)**

$$\begin{cases} k_1 = \mathbf{H}(\mathbf{x}_k, t_k)\Delta t \\ k_2 = \mathbf{H}(\mathbf{x}_k + \frac{1}{2}\mathbf{k}_1, t_k + \frac{1}{2}\Delta t)\Delta t \\ k_3 = \mathbf{H}(\mathbf{x}_k + \frac{1}{2}\mathbf{k}_2, t_k + \frac{1}{2}\Delta t)\Delta t \\ k_4 = \mathbf{H}(\mathbf{x}_k + \mathbf{k}_3, t_k + \Delta t)\Delta t \end{cases}$$
$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{1}{6}\mathbf{k}_1 + \frac{1}{3}\mathbf{k}_2 + \frac{1}{3}\mathbf{k}_3 + \frac{1}{6}\mathbf{k}_4 + O(\Delta t^5)$$

Let's now revisit Time Stepping Algorithms

Linear Single-Degree-of-Freedom (S-DoF) system of frequency ω_c :

$$\ddot{u} + \omega_c^2 u = 0$$

State-space representation:

$$\frac{d}{dt} \begin{bmatrix} u \\ \dot{u} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_c^2 & 0 \end{bmatrix} \begin{bmatrix} u \\ \dot{u} \end{bmatrix}$$

Analytical solution for a generic instant t :

$$\begin{bmatrix} u \\ \dot{u} \end{bmatrix} = e \left(\begin{bmatrix} 0 & 1 \\ -\omega_c^2 & 0 \end{bmatrix} t \right) \begin{bmatrix} u_0 \\ \dot{u}_0 \end{bmatrix}$$

Let's now revisit Time Stepping Algorithms

Linear Single-Degree-of-Freedom (S-DoF) system of frequency ω_c :

$$\ddot{u} + \omega_c^2 u = 0$$

State-space representation:

$$\frac{d}{dt} \begin{bmatrix} u \\ \dot{u} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_c^2 & 0 \end{bmatrix} \begin{bmatrix} u \\ \dot{u} \end{bmatrix} = \mathbf{A} \begin{bmatrix} u \\ \dot{u} \end{bmatrix}$$

Analytical solution for one-time-step transition Δt :

$$\begin{bmatrix} u_k \\ \dot{u}_k \end{bmatrix} = e^{\left(\begin{bmatrix} 0 & 1 \\ -\omega_c^2 & 0 \end{bmatrix} k\Delta t \right)} \begin{bmatrix} u_0 \\ \dot{u}_0 \end{bmatrix} = \mathbf{A}_c^k \begin{bmatrix} u_0 \\ \dot{u}_0 \end{bmatrix}$$

Analysis of Time Stepping Algorithms

Differential operator:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -\omega_c^2 & 0 \end{bmatrix}$$

↓

$$\mathbf{A}\Phi = \lambda\Phi$$

↓

$$\Phi^{-1}\mathbf{A}\Phi =$$

$$\Omega = \begin{bmatrix} +i\omega_c & 0 \\ 0 & -i\omega_c \end{bmatrix}$$

Transition matrix (exact):

$$\mathbf{A}_c = e\left(\begin{bmatrix} 0 & 1 \\ -\omega_c^2 & 0 \end{bmatrix}\Delta t\right)$$

↓

$$\mathbf{A}_c\Phi_c = \lambda_c\Phi_c$$

↓

$$\Phi_c^{-1}\mathbf{A}_c\Phi_c =$$

$$\Omega_c = \begin{bmatrix} e^{+i\omega_c\Delta t} & 0 \\ 0 & e^{-i\omega_c\Delta t} \end{bmatrix}$$

The following relation holds:

$$\begin{cases} \Omega_c = e^{\Omega\Delta t} \\ \Phi_c = \Phi \end{cases}$$

This is the proof:

$$\begin{aligned}\mathbf{A}_c &= e^{\mathbf{A}\Delta t} = \sum_{n=0}^{\infty} \frac{(\mathbf{A}\Delta t)^n}{n!} \\ &= \sum_{n=0}^{\infty} \frac{(\Phi\Omega\Phi^{-1}\Delta t)^n}{n!} \\ &= \Phi \left(\sum_{n=0}^{\infty} \frac{(\Omega\Delta t)^n}{n!} \right) \Phi^{-1} \\ &= \Phi e^{\Omega\Delta t} \Phi^{-1} \\ &= \Phi\Omega_c\Phi^{-1} \\ &= \Phi_c\Omega_c\Phi_c^{-1}\end{aligned}$$

Analysis of the Newmark Algorithm

Calculation of the transition matrix \mathbf{A}_d for the Newmark algorithm:

$$\begin{cases} \ddot{u}_{k+1} + \omega_c^2 u_{k+1} = 0 \\ u_{k+1} = u_k + \dot{u}_k \Delta t + \ddot{u}_k \left(\frac{1}{2} - \beta\right) \Delta t^2 + \ddot{u}_{k+1} \beta \Delta t^2 \\ \dot{u}_{k+1} = \dot{u}_k + \ddot{u}_k (1 - \gamma) \Delta t + \ddot{u}_{k+1} \gamma \Delta t \end{cases}$$

$$\begin{bmatrix} 1 & 0 & -\beta \Delta t^2 \\ 0 & 1 & -\gamma \Delta t \\ \omega_c^2 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_{k+1} \\ \dot{u}_{k+1} \\ \ddot{u}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & \left(\frac{1}{2} - \beta\right) \Delta t^2 \\ 0 & 1 & (1 - \gamma) \Delta t \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_k \\ \dot{u}_k \\ \ddot{u}_k \end{bmatrix}$$

Analysis of the Newmark Algorithm

Calculation of the transition matrix \mathbf{A}_d for the Newmark algorithm:

$$\begin{cases} \ddot{u}_{k+1} + \omega_c^2 u_{k+1} = 0 \\ u_{k+1} = u_k + \dot{u}_k \Delta t + \ddot{u}_k \left(\frac{1}{2} - \beta\right) \Delta t^2 + \ddot{u}_{k+1} \beta \Delta t^2 \\ \dot{u}_{k+1} = \dot{u}_k + \ddot{u}_k (1 - \gamma) \Delta t + \ddot{u}_{k+1} \gamma \Delta t \end{cases}$$

$$\begin{bmatrix} u_{k+1} \\ \dot{u}_{k+1} \\ \ddot{u}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\beta \Delta t^2 \\ 0 & 1 & -\gamma \Delta t \\ \omega_c^2 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & \Delta t & \left(\frac{1}{2} - \beta\right) \Delta t^2 \\ 0 & 1 & (1 - \gamma) \Delta t \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_k \\ \dot{u}_k \\ \ddot{u}_k \end{bmatrix}$$

$$\begin{bmatrix} u_{k+1} \\ \dot{u}_{k+1} \\ \ddot{u}_{k+1} \end{bmatrix} = \mathbf{A}_d \begin{bmatrix} u_k \\ \dot{u}_k \\ \ddot{u}_k \end{bmatrix}$$

The Newmark Algorithm: Stability

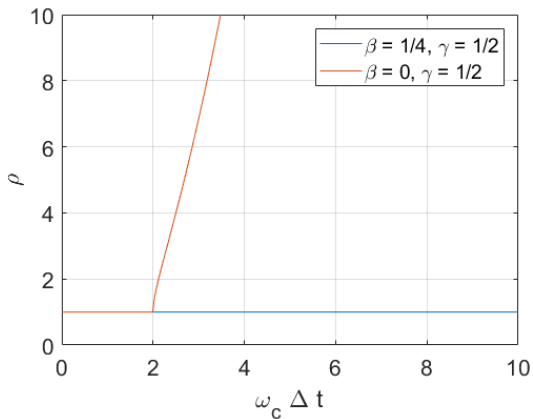
The spectral radius ρ of the transition matrix \mathbf{A}_d is defined as,

$$\begin{aligned}\mathbf{A}_d \Phi_d &= \lambda_d \Phi_d \\ \downarrow \\ \begin{bmatrix} u_{k+1} \\ \dot{u}_{k+1} \\ \ddot{u}_{k+1} \end{bmatrix} &= \mathbf{A}_d^{k+1} \begin{bmatrix} u_0 \\ \dot{u}_0 \\ \ddot{u}_0 \end{bmatrix} = \Phi_d \begin{bmatrix} \lambda_{d,1}^{k+1} & 0 & 0 \\ 0 & \lambda_{d,2}^{k+1} & 0 \\ 0 & 0 & 0 \end{bmatrix} \Phi_d^{-1} \begin{bmatrix} u_0 \\ \dot{u}_0 \\ \ddot{u}_0 \end{bmatrix} \\ \downarrow \\ \rho &= \max_j |\lambda_{d,j}| : \begin{cases} \rho \leq 1 & \text{stable} \\ \rho > 1 & \text{unstable} \end{cases}\end{aligned}$$

and it is expressed as function of the dimensionless frequency $\omega_c \Delta t$.

The Newmark Algorithm: Spectral Radius

$$\ddot{u} + \omega_c^2 u = 0$$



The Newmark Algorithm: Distortions

Differential operator:

$$\mathbf{A}\Phi = \lambda\Phi$$

↓

$$\Phi^{-1}\mathbf{A}\Phi =$$

$$\lambda = \pm i\omega_c$$

Transition matrix (approx):

$$\mathbf{A}_d\Phi_d = \lambda_d\Phi_d$$

↓

$$\Phi_d^{-1}\mathbf{A}_d\Phi_d =$$

$$\lambda_d = e^{(-\xi_d\omega_d \pm i\omega_d\sqrt{1-\xi_d^2})\Delta t}$$

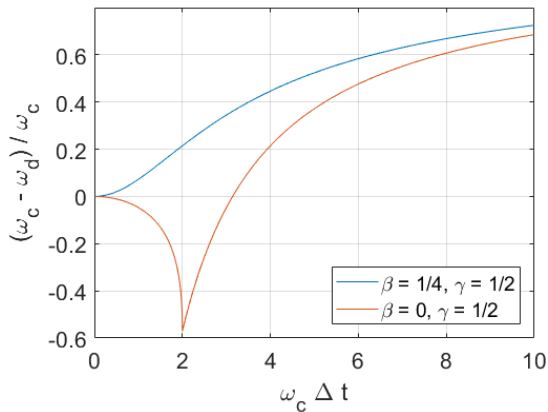
The following relation holds:

$$\begin{cases} \omega_d \approx \arg \frac{\lambda_d}{\Delta t} \neq \omega_c \\ \xi_d \approx -\frac{\ln|\lambda_d|}{\omega_d\Delta t} \neq 0 \end{cases}$$

ω_d and ξ_d are frequency and damping of the discretized system.

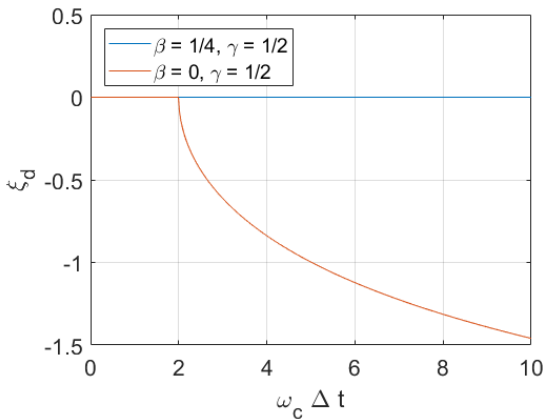
The Newmark Algorithm: Frequency Distortion

$$\ddot{u} + \omega_c^2 u = 0$$



The Newmark Algorithm: Damping Distortion

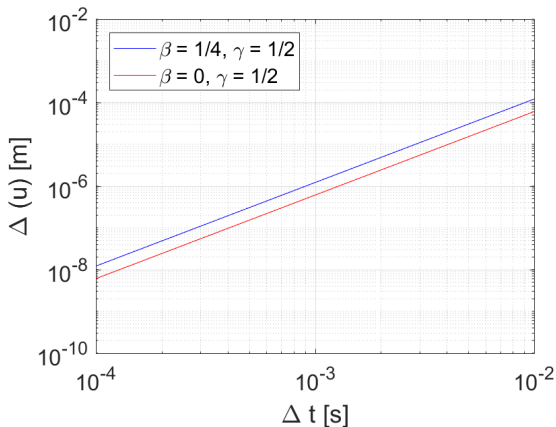
$$\ddot{u} + \omega_c^2 u = 0$$



Analysis of the Newmark Algorithm: Order of Accuracy

The order of accuracy p can be evaluated numerically:

$$\Delta(u) = |u_k - u(t_k)| \propto \Delta t^p$$



$$\omega_c = 2\pi, u_0 = 1m, v_0 = 0 \frac{m}{s}, t_k = 0.1s$$

Irons and Treharne' Theorem (1972):

$$\max_c (\omega_c)^2 \leq \max_e (\omega_e)^2$$

where.

- ω_c is the c -th frequency of the model
- ω_e is the frequency of the e -th element

Stability of the Newmark Method

For zero damping the Newmark method is conditionally stable if

$$\gamma \geq \frac{1}{2}, \quad \beta \leq \frac{1}{2} \quad \text{and} \quad \Delta t \leq \frac{1}{\omega_{max} \sqrt{\frac{\gamma}{2} - \beta}}$$

where ω_{max} is the maximum natural frequency.

The Newmark method is unconditionally stable if

$$2\beta \geq \gamma \geq \frac{1}{2}$$

Stability of the Newmark Method

However, if $\gamma \geq \frac{1}{2}$, errors are introduced. These errors are associated with “numerical damping” and “period elongation”, i.e. a seemingly larger damping and period of oscillation than in reality.

Because of the unconditional stability of the average acceleration method, it is the most robust method to be used for the step-by-step dynamic analysis of large complex structural systems in which a large number of high frequencies, short periods, are present.

The only problem with the method is that the short periods, which are smaller than the time step, oscillate indefinitely after they are excited. The higher mode oscillation can however be reduced by the addition of stiffness proportional (artificial) damping.

source: csiberkeley.com

Nonlinear Problem Formulation:

$$\begin{cases} \mathbf{M}\ddot{\mathbf{u}}_n + \mathbf{r}(\mathbf{u}_n, \dot{\mathbf{u}}_n) = \mathbf{f}(t_n) \\ \mathbf{u}(t_0) = \mathbf{u}_0, \dot{\mathbf{u}}(t_0) = \dot{\mathbf{u}}_0 \end{cases}$$

where,

- \mathbf{M} : mass matrix.
- \mathbf{r} : restoring force.
- \mathbf{f} : external force vector.
- $\mathbf{u}_n, \dot{\mathbf{u}}_n, \ddot{\mathbf{u}}_n$: displacement, velocity and acceleration vectors at t_n .

Nonlinear Implementation of the Newmark Algorithm

Discretization of the IVP with a time step $\Delta t = t_{k+1} - t_k$:

$$\begin{cases} \mathbf{M}\ddot{\mathbf{u}}_{k+1} + \mathbf{r}(\mathbf{u}_{k+1}, \dot{\mathbf{u}}_{k+1}) = \mathbf{f}(t_{k+1}) \\ \mathbf{u}(t_0) = \mathbf{u}_0, \dot{\mathbf{u}}(t_0) = \dot{\mathbf{u}}_0 \end{cases}$$

A **nonlinear problem** is solved at each time step:

$$\mathbf{M}\ddot{\mathbf{u}}_{k+1} + \mathbf{r}(\tilde{\mathbf{u}}_{k+1} + \ddot{\mathbf{u}}_{k+1}\beta\Delta t^2, \dot{\tilde{\mathbf{u}}}_{k+1} + \ddot{\mathbf{u}}_{k+1}\gamma\Delta t) = \mathbf{f}(t_{k+1})$$

Predictors depend on previous time step solutions:

$$\begin{cases} \tilde{\mathbf{u}}_{k+1} = \mathbf{u}_k + \dot{\mathbf{u}}_k\Delta t + \ddot{\mathbf{u}}_k(\frac{1}{2} - \beta)\Delta t^2 \\ \dot{\tilde{\mathbf{u}}}_{k+1} = \dot{\mathbf{u}}_k + \ddot{\mathbf{u}}_k(1 - \gamma)\Delta t \end{cases}$$

Correctors determine the current time step solution:

$$\begin{cases} \mathbf{u}_{k+1} = \tilde{\mathbf{u}}_{k+1} + \ddot{\mathbf{u}}_{k+1}\beta\Delta t^2 \\ \dot{\mathbf{u}}_{k+1} = \dot{\tilde{\mathbf{u}}}_{k+1} + \ddot{\mathbf{u}}_{k+1}\gamma\Delta t \end{cases}$$

Nonlinear Implementations of the Newmark Algorithm

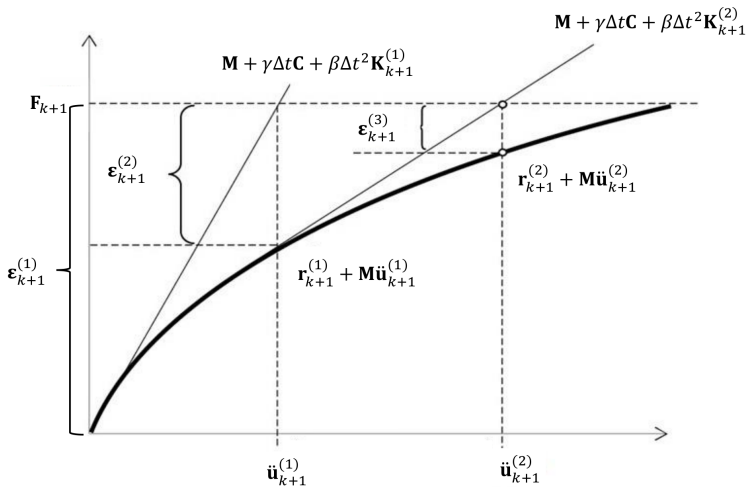
Implicit implementation based on Newton-Raphson iterations:

- 1: $\ddot{\mathbf{u}}_{k+1} \leftarrow 0$
- 2: $\mathbf{u}_{k+1} \leftarrow \mathbf{u}_k + \dot{\mathbf{u}}_k \Delta t + \ddot{\mathbf{u}}_k \left(\frac{1}{2} - \beta\right) \Delta t^2 + \ddot{\mathbf{u}}_{k+1} \beta \Delta t^2$
- 3: $\dot{\mathbf{u}}_{k+1} \leftarrow \dot{\mathbf{u}}_k + \ddot{\mathbf{u}}_k (1 - \gamma) \Delta t + \ddot{\mathbf{u}}_{k+1} \gamma \Delta t$
- 4: $\boldsymbol{\varepsilon} \leftarrow \mathbf{f}(t_{k+1}) - \mathbf{r}(\mathbf{u}_{k+1}, \dot{\mathbf{u}}_{k+1}) - \mathbf{M}\ddot{\mathbf{u}}_{k+1}$
- 5: **while** $\|\boldsymbol{\varepsilon}\| \geq Tol$ **do**
- 6: $\Delta\ddot{\mathbf{u}}_{k+1} \leftarrow (\mathbf{M} + \mathbf{C}\gamma\Delta t + \mathbf{K}\beta\Delta t^2)^{-1} \boldsymbol{\varepsilon}$
- 7: $\ddot{\mathbf{u}}_{k+1} \leftarrow \ddot{\mathbf{u}}_{k+1} + \Delta\ddot{\mathbf{u}}_{k+1}$
- 8: $\dot{\mathbf{u}}_{k+1} \leftarrow \dot{\mathbf{u}}_{k+1} + \Delta\ddot{\mathbf{u}}_{k+1} \gamma \Delta t$
- 9: $\mathbf{u}_{k+1} \leftarrow \mathbf{u}_{k+1} + \Delta\ddot{\mathbf{u}}_{k+1} \beta \Delta t^2$
- 10: $\boldsymbol{\varepsilon} \leftarrow \mathbf{f}(t_{k+1}) - \mathbf{r}(\mathbf{u}_{k+1}, \dot{\mathbf{u}}_{k+1}) - \mathbf{M}\ddot{\mathbf{u}}_{k+1}$
- 11: **end while**

Assembly of restoring force and stiffness matrix loop over elements:

- | | |
|---------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------|
| 1: for $i = 1$ to l do | 1: for $i = 1$ to l do |
| 2: $\mathbf{r}_{i,k+1} \leftarrow \text{elementForce}(\mathbf{Z}_i \mathbf{u}_{k+1})$ | 2: $\mathbf{K}_{i,k+1} \leftarrow \text{elementStiff}(\mathbf{Z}_i \mathbf{u}_{k+1})$ |
| 3: $\mathbf{r}_{k+1} \leftarrow \mathbf{r}_{k+1} + \mathbf{Z}_i^T \mathbf{r}_{i,k+1}$ | 3: $\mathbf{K}_{k+1} \leftarrow \mathbf{K}_{k+1} + \mathbf{Z}_i^T \mathbf{K}_{i,k+1} \mathbf{Z}_i$ |
| 4: end for | 4: end for |

Nonlinear Implementations of the Newmark Algorithm



Full Newton-Raphson iteration

Explicit Integration for Nonlinear Problems

In the explicit version of the method $\beta = 0$ and the update step in each iteration becomes:

$$\Delta \ddot{\mathbf{u}}_{k+1} \leftarrow (\mathbf{M} + \mathbf{C}\gamma\Delta t)^{-1} \boldsymbol{\varepsilon}$$

In the above:

- The matrix to be inverted is $(\mathbf{M} + \mathbf{C}\gamma\Delta t)$
- This matrix remains constant during the iterative procedure
- It can be factorized once and then only backward substitutions have to be performed

Explicit Integration for Nonlinear Problems

In the explicit version of the method $\beta = 0$ and the update step in each iteration becomes:

$$\Delta \ddot{\mathbf{u}}_{k+1} \leftarrow (\mathbf{M} + \mathbf{C}\gamma\Delta t)^{-1} \boldsymbol{\varepsilon}$$

If further:

- \mathbf{M} is lumped (diagonal)
- \mathbf{C} is either lumped or zero

Then solution of the system is trivial!

The same is true for all explicit schemes!

- The above feature can be exploited to solve nonlinear static problems
- Damping of the system is increased to remove dynamic effects
- If damping is set correctly the solution quickly converges to the static one

Implicit Vs Explicit integration

Explicit integration:

- Does not require inversion of a tangent stiffness matrix
- Conditionally stable - limited by small timesteps
- Usually preferred for problems of small duration involving high frequencies such as wave propagation

Implicit integration:

- Requires inversion of a tangent stiffness matrix
- Unconditionally stable - allows for a larger timestep
- Usually preferred for structural dynamics problems of larger duration involving lower frequencies

Nonlinear Implementation of the Newmark Algorithm

Linearly-implicit implementation (a.k.a. operator splitting):

$$1: \tilde{\mathbf{u}}_{k+1} \leftarrow \mathbf{u}_k + \dot{\mathbf{u}}_k \Delta t + \ddot{\mathbf{u}}_k \left(\frac{1}{2} - \beta \right) \Delta t^2$$

$$2: \dot{\tilde{\mathbf{u}}}_{k+1} \leftarrow \dot{\mathbf{u}}_k + \ddot{\mathbf{u}}_k (1 - \gamma) \Delta t$$

$$3: \ddot{\mathbf{u}}_{k+1} \leftarrow (\mathbf{M} + \mathbf{C}\gamma\Delta t + \mathbf{K}\beta\Delta t^2)^{-1} \left(\mathbf{f}(t_{k+1}) - \mathbf{r}(\tilde{\mathbf{u}}_{k+1}, \dot{\tilde{\mathbf{u}}}_{k+1}) \right)$$

$$4: \dot{\mathbf{u}}_{k+1} \leftarrow \dot{\tilde{\mathbf{u}}}_{k+1} + \ddot{\mathbf{u}}_{k+1} \gamma \Delta t$$

$$5: \mathbf{u}_{k+1} \leftarrow \tilde{\mathbf{u}}_{k+1} + \ddot{\mathbf{u}}_{k+1} \beta \Delta t^2$$

- \mathbf{K} is assembled once at the beginning of the simulation
- the linearly-implicit implementation is equivalent to the implicit implementation based on the modified Newton-Raphson method (constant \mathbf{K}) truncated at one iteration.

Assembly of restoring force and stiffness matrix loop over elements:

1: **for** $i = 1$ to l **do**

2: $\mathbf{r}_{i,k+1} \leftarrow \text{elementForce}(\mathbf{Z}_i \mathbf{u}_{k+1})$

3: $\mathbf{r}_{k+1} \leftarrow \mathbf{r}_{k+1} + \mathbf{Z}_i^T \mathbf{r}_{i,k+1}$

4: **end for**

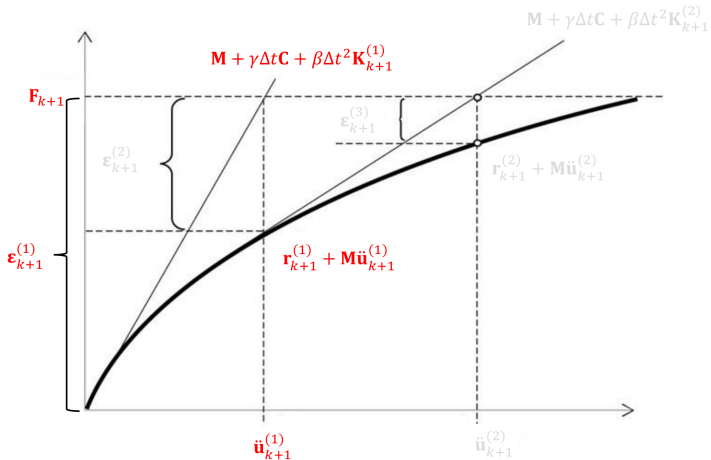
1: **for** $i = 1$ to l **do**

2: $\mathbf{K}_{i,k+1} \leftarrow \text{elementStiff}(\mathbf{Z}_i \mathbf{u}_{k+1})$

3: $\mathbf{K}_{k+1} \leftarrow \mathbf{K}_{k+1} + \mathbf{Z}_i^T \mathbf{K}_{i,k+1} \mathbf{Z}_i$

4: **end for**

Nonlinear Implementation of the Newmark Algorithm



Single modified Newton-Raphson (constant \mathbf{K}) iteration

In this case a residual force balance $\epsilon_{k+1}^{(2)}$ occurs that tends to zero as Δt tends to zero.

The HHT- α Algorithm

α -shifted equation of motion:

$$\begin{cases} \mathbf{M}\ddot{\mathbf{u}}_{k+1} + (1 + \alpha)\mathbf{r}(\mathbf{u}_{k+1}, \dot{\mathbf{u}}_{k+1}) - \alpha\mathbf{r}(\mathbf{u}_k, \dot{\mathbf{u}}_k) = (1 + \alpha)\mathbf{f}(t_{k+1}) - \alpha\mathbf{f}(t_k) \\ \mathbf{u}(t_0) = \mathbf{u}_0, \dot{\mathbf{u}}(t_0) = \dot{\mathbf{u}}_0 \end{cases}$$

Interpolation equations and integration parameters:

$$\begin{cases} \mathbf{u}_{k+1} = \mathbf{u}_k + \dot{\mathbf{u}}_k \Delta t + \ddot{\mathbf{u}}_k \left(\frac{1}{2} - \beta\right) \Delta t^2 + \ddot{\mathbf{u}}_{k+1} \beta \Delta t^2 \\ \dot{\mathbf{u}}_{k+1} = \dot{\mathbf{u}}_k + \ddot{\mathbf{u}}_k (1 - \gamma) \Delta t + \ddot{\mathbf{u}}_{k+1} \gamma \Delta t \end{cases}$$

$$\beta = (1 - \alpha)^2/4, \gamma = (1 - 2\alpha)/2$$

where $\alpha \in [-1/3, 0]$ modulates algorithmic damping.

Hilber, H. M., Hughes, T. J., Taylor, R. L. (1977). Improved numerical dissipation for time integration algorithms in structural dynamics. Earthquake Engineering & Structural Dynamics, 5(3), 283–292.

Nonlinear Implementation of the HHT- α Algorithm

Implicit implementation based on Newton-Raphson iterations:

- 1: $\ddot{\mathbf{u}}_{k+1} \leftarrow 0$
- 2: $\mathbf{u}_{k+1} \leftarrow \mathbf{u}_k + \dot{\mathbf{u}}_k \Delta t + \ddot{\mathbf{u}}_k \left(\frac{1}{2} - \beta\right) \Delta t^2 + \ddot{\mathbf{u}}_{k+1} \beta \Delta t^2$
- 3: $\dot{\mathbf{u}}_{k+1} \leftarrow \dot{\mathbf{u}}_k + \ddot{\mathbf{u}}_k (1 - \gamma) \Delta t + \ddot{\mathbf{u}}_{k+1} \gamma \Delta t$
- 4: $\boldsymbol{\varepsilon} \leftarrow \hat{\mathbf{f}}_{k+1}$
- 5: **while** $\|\boldsymbol{\varepsilon}\| \geq Tol$ **do**
- 6: $\Delta \ddot{\mathbf{u}}_{k+1} \leftarrow \hat{\mathbf{M}}^{-1} \boldsymbol{\varepsilon}$
- 7: $\ddot{\mathbf{u}}_{k+1} \leftarrow \ddot{\mathbf{u}}_{k+1} + \Delta \ddot{\mathbf{u}}_{k+1}$
- 8: $\dot{\mathbf{u}}_{k+1} \leftarrow \dot{\mathbf{u}}_{k+1} + \Delta \ddot{\mathbf{u}}_{k+1} \gamma \Delta t$
- 9: $\mathbf{u}_{k+1} \leftarrow \mathbf{u}_{k+1} + \Delta \ddot{\mathbf{u}}_{k+1} \beta \Delta t^2$
- 10: $\boldsymbol{\varepsilon} \leftarrow \hat{\mathbf{f}}_{k+1}$
- 11: **end while**

The same assembly of restoring force and stiffness matrix loop of the Newmark method is performed and,

$$\hat{\mathbf{M}} = \mathbf{M} + \mathbf{C} \gamma \Delta t (1 + \alpha) + \mathbf{K} \beta \Delta t^2 (1 + \alpha)$$

$$\hat{\mathbf{f}}_{k+1} = (1 + \alpha) \mathbf{f}_{k+1} - \alpha \mathbf{f}_k - (1 + \alpha) \tilde{\mathbf{r}}_{k+1} + \alpha \tilde{\mathbf{r}}_k -$$

$$(1 + \alpha) \mathbf{C} \dot{\mathbf{u}}_{k+1} + \alpha \mathbf{C} \dot{\mathbf{u}}_k + \alpha (\mathbf{C} \gamma \Delta t + \mathbf{K} \beta \Delta t^2) \ddot{\mathbf{u}}_k$$

Nonlinear Implementation of the HHT- α Algorithm

Linearly-implicit implementation (a.k.a. operator splitting):

$$1: \tilde{\mathbf{u}}_{k+1} \leftarrow \mathbf{u}_k + \dot{\mathbf{u}}_k \Delta t + \ddot{\mathbf{u}}_k \left(\frac{1}{2} - \beta \right) \Delta t^2$$

$$2: \dot{\tilde{\mathbf{u}}}_{k+1} \leftarrow \dot{\mathbf{u}}_k + \ddot{\mathbf{u}}_k (1 - \gamma) \Delta t$$

$$3: \ddot{\mathbf{u}}_{k+1} \leftarrow \hat{\mathbf{M}}^{-1} \hat{\mathbf{f}}_{k+1}$$

$$4: \dot{\mathbf{u}}_{k+1} \leftarrow \dot{\tilde{\mathbf{u}}}_{k+1} + \ddot{\mathbf{u}}_{k+1} \gamma \Delta t$$

$$5: \mathbf{u}_{k+1} \leftarrow \tilde{\mathbf{u}}_{k+1} + \ddot{\mathbf{u}}_{k+1} \beta \Delta t^2$$

- \mathbf{K} is assembled once at the beginning of the simulation
- the linearly-implicit implementation is equivalent to the implicit implementation based on the modified Newton-Raphson method (constant \mathbf{K}) truncated at one iteration.

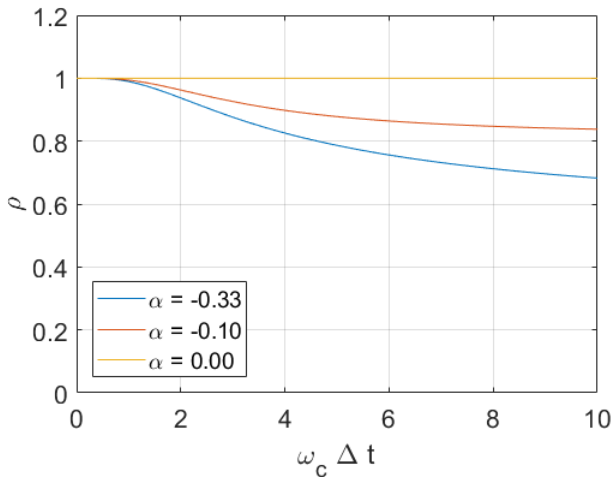
The same assembly of restoring force and stiffness matrix loop of the Newmark method is performed and,

$$\hat{\mathbf{M}} = \mathbf{M} + \mathbf{C} \gamma \Delta t (1 + \alpha) + \mathbf{K} \beta \Delta t^2 (1 + \alpha)$$

$$\begin{aligned} \hat{\mathbf{f}}_{k+1} = & (1 + \alpha) \mathbf{f}_{k+1} - \alpha \mathbf{f}_k - (1 + \alpha) \tilde{\mathbf{r}}_{k+1} + \alpha \tilde{\mathbf{r}}_k - \\ & (1 + \alpha) \mathbf{C} \dot{\tilde{\mathbf{u}}}_{k+1} + \alpha \mathbf{C} \dot{\tilde{\mathbf{u}}}_k + \alpha (\mathbf{C} \gamma \Delta t + \mathbf{K} \beta \Delta t^2) \ddot{\mathbf{u}}_k \end{aligned}$$

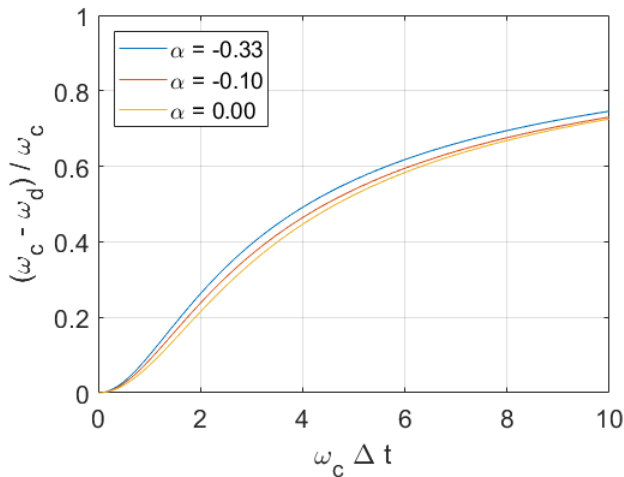
Analysis of the HHT- α Algorithm: Spectral Radius

$$\ddot{u} + \omega_c^2 u = 0$$



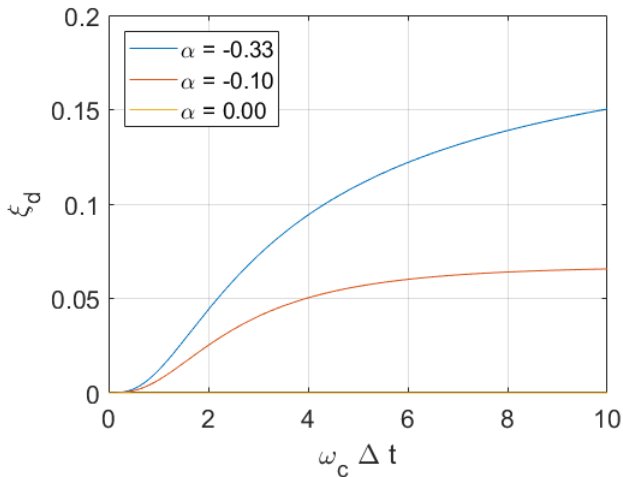
Analysis of the HHT- α Algorithm: Frequency Bias

$$\ddot{u} + \omega_c^2 u = 0$$



Analysis of the HHT- α Algorithm: Damping Bias

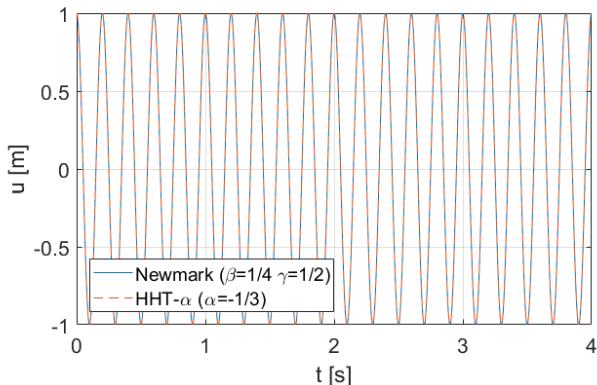
$$\ddot{u} + \omega_c^2 u = 0$$



Algorithmic Damping Comparison

Free-decay response of a S-DoF system:

$$\ddot{u} + \omega_c^2 u = 0$$

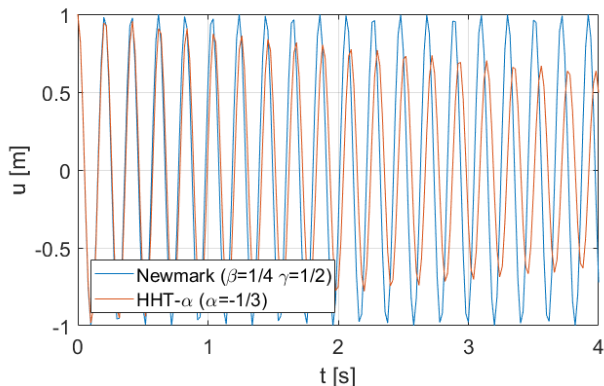


$$\omega_c = 2\pi 5, u_0 = 1m, v_0 = 0 \frac{m}{s}, \Delta t = 2e - 4s$$

Algorithmic Damping Comparison

Free-decay response of a S-DoF system:

$$\ddot{u} + \omega_c^2 u = 0$$



$$\omega_c = 2\pi 5, u_0 = 1m, v_0 = 0 \frac{m}{s}, \Delta t = 2e - 2s$$

Recalling the Eigenvalue Analysis

The eigen-problem is obtained as the solution to the undamped, free vibration equation:

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{0}$$

Defining a matrix Φ whose columns are the eigen-vectors $\{\phi_i, i = 1 \dots n\}$ and a diagonal matrix Ω^2 storing the eigenvalues $\{\omega_i^2, i = 1 \dots n\}$, i.e.:

$$\Phi = [\phi_1, \phi_2, \dots, \phi_n] \quad \Omega^2 = \begin{bmatrix} \omega_1 & & & \\ & \omega_2 & & \\ & & \dots & \\ & & & \omega_n \end{bmatrix}$$

we can write the eigen-problem as:

$$[\mathbf{K} - \mathbf{M}\Omega^2] \Phi = \mathbf{0}$$

Mode Superposition Method

M – Orthonormality

We may scale Φ , as they form a basis, and we can chose these such that:

$$\Phi^T \mathbf{M} \Phi = \mathbf{I} \rightarrow \Phi^T \mathbf{K} \Phi = \Omega^2$$

Using the transformation $\mathbf{u}(t) = \Phi \mathbf{q}(t)$, the IVP is transformed to its modal counterpart:

$$\begin{aligned} \ddot{\mathbf{q}}(t) + \Phi^T \mathbf{C} \Phi \dot{\mathbf{q}}(t) + \Omega^2 \mathbf{q}(t) &= \Phi^T \mathbf{f}(t) \\ \mathbf{q}_0 &= \Phi^T \mathbf{M} \mathbf{u}_0; \quad \dot{\mathbf{q}}_0 = \Phi^T \mathbf{M} \dot{\mathbf{u}}_0 \end{aligned}$$

under a Proportional Damping Assumption: $\phi_i^T \mathbf{C} \phi_j = 2\omega_i \xi_i \delta_{ij}$, where ξ_i is a modal damping parameter and δ_{ij} is the Kronecker delta

Therefore, we end up with n **decoupled SDOF** equations, one for each q_i :

$$\ddot{q}_i(t) + 2\omega_i \xi_i \dot{q}_i(t) + \omega_i^2 q_i(t) = r_i(t)$$

This even admits an analytical solution (Duhamel's Integral). For systems with non-classical damping \mathbf{C} is not diagonal \rightarrow the equations are coupled & solved numerically.

Complete Response

The solution of all n SDOF equations are calculated and the finite element nodal point displacements are obtained by superposition of the response in each mode:

$$\mathbf{u}(t) = \Phi \mathbf{q}(t) \Rightarrow \hat{\mathbf{u}}(t) = \sum_{i=1}^n \phi_i q_i(t)$$

In case only the first m modes are contributing to the solution, we retain the first m eigenvectors with the dimension of the system of equations now reducing to the dimension of $\Phi \in \mathbb{R}^{n \times m}$.

Usually $m \ll n$, which implies that the modal system is significantly reduced.

Note: The analysis only holds for linear systems.