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:

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

\[
\begin{align*}
M\ddot{u}(t) + C\dot{u}(t) + Ku(t) &= f(t) \\
u(t_0) &= u_0, \quad \dot{u}(t_0) = \dot{u}_0
\end{align*}
\]

where:

- **M**: Mass Matrix
- **K**: Stiffness Matrix
- **u**: Displacements
- **\dot{u}**: Velocities
- **\ddot{u}**: Accelerations
- **f**: external force vector

or alternatively:

\[
\begin{align*}
F_I(t) + F_D(t) + F_E(t) &= f(t)
\end{align*}
\]

where:

- **F_I(t)**: Inertial Force
- **F_D(t)**: Damping Force
- **F_E(t)**: Internal Force
Introduction to Dynamic Analysis

Example: 3-dof system

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

\[ u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}, \quad f(t) = \begin{bmatrix} f_1(t) \\ f_2(t) \\ f_3(t) \end{bmatrix} \]
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, 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.
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 $\Delta t$
2. A particular variation of displacements, velocities and accelerations within each time interval is assumed.
Direct Integration Methods

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

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

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 $\theta$ 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 $\theta$)
## Direct Integration Methods

### Most Commonly used Direct Integration Methods

<table>
<thead>
<tr>
<th>Central Difference Method</th>
<th>Acceleration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Velocity</strong></td>
<td><strong>Acceleration</strong></td>
</tr>
<tr>
<td>$U(t) = \frac{U(t+\Delta t) - U(t-\Delta t)}{2\Delta t}$</td>
<td>$\ddot{U}(t) = \frac{U(t+\Delta t) - 2U(t) + U(t-\Delta t)}{\Delta t}$</td>
</tr>
</tbody>
</table>

![Graph showing velocity and acceleration using central difference method](image)

- Velocity equation:
  
  \[ U(t) = \frac{U(t+\Delta t) - U(t-\Delta t)}{2\Delta t} \]

- Acceleration equation:
  
  \[ \ddot{U}(t) = \frac{U(t+\Delta t) - 2U(t) + U(t-\Delta t)}{\Delta t} \]
### The Houbolt Method

#### Displacement

\[
U_i = U_{i+\Delta t} - \Delta t \dot{U}_{i+\Delta t} + \frac{\Delta t^2}{2} \ddot{U}_{i+\Delta t} - \frac{\Delta t^3}{6} \dddot{U}_{i+\Delta t}
\]

\[
U_{i-\Delta t} = U_{i+\Delta t} - 2\Delta t \dot{U}_{i+\Delta t} + \frac{(2\Delta t)^2}{2} \ddot{U}_{i+\Delta t} - \frac{(2\Delta t)^3}{6} \dddot{U}_{i+\Delta t}
\]

#### Velocity

\[
\dot{U}_{i+\Delta t} = \frac{1}{6\Delta t} \left( 11U_{i+\Delta t} - 18U_i + 9U_{i-\Delta t} - 2U_{i-2\Delta t} \right)
\]

#### Acceleration

\[
\dddot{U}_{i+\Delta t} = \frac{1}{\Delta t^2} \left( 2U_{i+\Delta t} - 5U_i + 4U_{i-\Delta t} - U_{i-2\Delta t} \right)
\]

These approximations are derived via the displacement approximation.

Houbolt’s method uses a third-order interpolation of displacements extending two steps back in time.
### The Newmark Method

<table>
<thead>
<tr>
<th>Displacement</th>
<th>Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Newmark method uses a second order Taylor expansion for approximating Velocities and Accelerations:</td>
<td></td>
</tr>
<tr>
<td>$U_{t+\Delta t} = U_t + \Delta t \dot{U}_v + \frac{\Delta t^2}{2} \ddot{U}_D$, where $\dot{U}_v, \ddot{U}_D$ are approximations of the Acceleration</td>
<td></td>
</tr>
<tr>
<td>$\dot{U}_v = (1-\delta) \dot{U}<em>t + \delta \dot{U}</em>{t+\Delta t}$, if $\delta = 0.5$</td>
<td></td>
</tr>
<tr>
<td>$\ddot{U}_D = (1-2\alpha) \ddot{U}<em>t + 2\alpha \ddot{U}</em>{t+\Delta t}$, if $\alpha = 1/4$</td>
<td></td>
</tr>
</tbody>
</table>

$\gamma$ and $\beta$ are used in place of $\delta$ and $\alpha$, respectively, in the following slides.
### Stability/Accuracy of DIMs

<table>
<thead>
<tr>
<th>Integration Method</th>
<th>Type of Method</th>
<th>Critical Step Size ((\Delta t_{cr}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central Different</td>
<td>Explicit</td>
<td>(\frac{2}{\omega})</td>
</tr>
<tr>
<td>Newmark Method</td>
<td>Implicit</td>
<td>(\frac{3.464}{\omega})</td>
</tr>
<tr>
<td>(\gamma = \frac{1}{2}, \beta = \frac{1}{6}) (Linear Acceleration)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Newmark Method</td>
<td>Implicit</td>
<td>Unconditionally Stable</td>
</tr>
<tr>
<td>(\gamma = \frac{1}{2}, \beta = \frac{1}{4}) (Constant-Average-Acceleration)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Newmark Method</td>
<td>Explicit</td>
<td>(\frac{2}{\omega})</td>
</tr>
<tr>
<td>(\gamma = \frac{1}{2}, \beta = 0) (Central Difference)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wilson-(\theta)</td>
<td>Implicit</td>
<td>Unconditionally Stable when (\theta \geq 1.37)</td>
</tr>
</tbody>
</table>
The Newmark method is the most widely used multi-step time integration algorithm for structural analysis:

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

$$\begin{cases} M\ddot{u}_{k+1} + C\dot{u}_{k+1} + K u_{k+1} = f(t_{k+1}) \\ u(t_0) = u_0, \dot{u}(t_0) = \dot{u}_0 \end{cases}$$

Interpolation equations (Newmark, 1959):

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

where $\beta$ and $\gamma$ are the parameters of the time integration algorithm.

Remark:

\( \gamma \) and \( \beta \) 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 \( \theta \) method with \( \theta = 1 \)
A linear problem is solved at each time step:

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

Predictors depend on previous time step solutions:

\[
\begin{align*}
\ddot{u}_{k+1} &= u_k + \dot{u}_k\Delta t + \ddot{u}_k\left(\frac{1}{2} - \beta\right)\Delta t^2 \\
\dot{\ddot{u}}_{k+1} &= \dot{u}_k + \ddot{u}_k(1 - \gamma)\Delta t
\end{align*}
\]

Correctors determine the current time step solution:

\[
\begin{align*}
u_{k+1} &= \ddot{u}_{k+1} + \dddot{u}_{k+1}\beta\Delta t^2 \\
\dot{u}_{k+1} &= \ddot{u}_{k+1} + \dddot{u}_{k+1}\gamma\Delta t
\end{align*}
\]
Implicit/explicit algorithm:

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

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

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

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

5: \( u_{k+1} \leftarrow \tilde{u}_{k+1} + \ddot{\tilde{u}}_{k+1} \beta \Delta t^2 \)

- \( \beta = 0 \): explicit algorithm
- \( \beta \neq 0 \): implicit algorithm \( \rightarrow \) inversion of \( K \)
The Newmark Algorithm

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

\[
\begin{align*}
\tilde{u}_{k+1} &= u_k + \dot{u}_k \Delta t + \ddot{u}_k \left( \frac{1}{2} - \beta \right) \Delta t^2 \\
\hat{u}_{k+1} &= \dot{u}_k + \ddot{u}_k (1 - \gamma) \Delta t
\end{align*}
\]
The Newmark Algorithm

Step #1: calculation of predictors:
\[
\begin{align*}
\ddot{u}_{k+1} &= u_k + \dot{u}_k \Delta t + \ddot{u}_k \left( \frac{1}{2} - \beta \right) \Delta t^2 \\
\ddot{\tilde{u}}_{k+1} &= \dot{u}_k + \ddot{u}_k (1 - \gamma) \Delta t
\end{align*}
\]

Step #2: solution of the linear problem:
\[
\ddot{u}_{k+1} = \left( M + C \gamma \Delta t + K \beta \Delta t^2 \right)^{-1} \left( f(t_{k+1}) - K \ddot{u}_{k+1} - C \dot{\ddot{u}}_{k+1} \right)
\]
The Newmark Algorithm

Step #1: calculation of predictors:

\[
\begin{align*}
\tilde{u}_{k+1} &= u_k + \dot{u}_k \Delta t + \ddot{u}_k \left( \frac{1}{2} - \beta \right) \Delta t^2 \\
\hat{u}_{k+1} &= \hat{u}_k + \ddot{u}_k (1 - \gamma) \Delta t
\end{align*}
\]

Step #2: solution of the linear problem:

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

Step #3: calculation of correctors:

\[
\begin{align*}
u_{k+1} &= \tilde{u}_{k+1} + \ddot{u}_{k+1} \beta \Delta t^2 \\
\dot{u}_{k+1} &= \hat{u}_{k+1} + \ddot{u}_{k+1} \gamma \Delta t
\end{align*}
\]
The Newmark Algorithm

Step #1: calculation of predictors:

\[
\begin{align*}
\ddot{\tilde{u}}_{k+1} &= u_k + \dot{u}_k \Delta t + \ddot{u}_k \left( \frac{1}{2} - \beta \right) \Delta t^2 \\
\dot{\tilde{u}}_{k+1} &= \dot{u}_k + \ddot{u}_k (1 - \gamma) \Delta t
\end{align*}
\]

Step #2: solution of the linear problem:

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

Step #3: calculation of correctors:

\[
\begin{align*}
u_{k+1} &= \ddot{\tilde{u}}_{k+1} + \ddot{u}_{k+1} \beta \Delta t^2 \\
\dot{u}_{k+1} &= \dot{\tilde{u}}_{k+1} + \ddot{u}_{k+1} \gamma \Delta t
\end{align*}
\]

- $\beta = 0$: explicit algorithm
- $\beta \neq 0$: implicit algorithm $\rightarrow$ inversion of $K$
The Central Difference Method

Example - 2 DOF system

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

2 dof system example $\Delta t = 0.28s$
The Newmark Method

2 dof system example $\Delta t = 1 \text{s}$

Discrete Time Step, $\Delta t=1 \text{ sec}$
We introduce the augmented state vector: \( x = \begin{bmatrix} u_1 & u_2 & v_1 & v_2 \end{bmatrix}^T \) (controllable form equivalent). Then,

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

\[
\dot{x} = A x + B p(t)
\]

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

\[ \dot{x} = Ax + Bp(t) \]

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

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4
\end{bmatrix}
+ \Phi_{4 \times 2} p(t)
\]

\[ y = Cx + Dp(t) \]
State Space Equation Formulation

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

\[
M \ddot{x} + R(x) = F(t)
\]

\[
\Downarrow
\]

\[
\dot{x} = M^{-1} (F(t) - R(x)) = H(x, t)
\]

with,

\[
M = \begin{bmatrix}
1 & 0 & 0 \\
0 & m & 0 \\
0 & 0 & 1
\end{bmatrix},
\quad x = \begin{bmatrix}
u \\
v \\
s
\end{bmatrix},
\quad R = \begin{bmatrix}
-v \\
r(u, v, s) \\
g(u, v, s)
\end{bmatrix},
\quad F(t) = \begin{bmatrix}
0 \\
f(t)
\end{bmatrix}
\]

and,

- \( r \) is the nonlinear restoring force vector that depends on displacement \( u \), velocity \( v \) and additional state variables \( s \).
- \( g \) is an additional nonlinear function that determines the evolution of the additional state variables \( s \).
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 \( \frac{dx}{dt} = H(x(t), t) \), \( x(t_0) = 0 \)

- **Forward Euler Method**
  \[
  x_{k+1} = x_k + H(x_k, t_k) \Delta t
  \]
  where \( \Delta t \) is the integration time step. This explicit expression is obtained from the truncated Taylor Expansion of \( x(t_k + \Delta t) \)

- **Backward Euler Method**
  \[
  x_{k+1} = x_k + H(x_{k+1}, t_{k+1}) \Delta t
  \]
  This implicit expression (since \( x_{k+1} \) is on the right hand side) is obtained from the truncated Taylor Expansion of \( x(t_{k+1} - \Delta t) \)
State Space Equation Formulation

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

\[
\begin{align*}
    k_1 &= H(x_k, t_k)\Delta t \\
    k_2 &= H(x_k + \frac{1}{2}k_1, t_k + \frac{1}{2}\Delta t)\Delta t \\
    x_{k+1} &= x_k + k_2 + O(\Delta t^3)
\end{align*}
\]

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

\[
\begin{align*}
    k_1 &= H(x_k, t_k)\Delta t \\
    k_2 &= H(x_k + \frac{1}{2}k_1, t_k + \frac{1}{2}\Delta t)\Delta t \\
    k_3 &= H(x_k + \frac{1}{2}k_2, t_k + \frac{1}{2}\Delta t)\Delta t \\
    k_4 &= H(x_k + k_3, t_k + \Delta t)\Delta t \\
    x_{k+1} &= x_k + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 + O(\Delta t^5)
\end{align*}
\]
Let’s now revisit Time Stepping Algorithms

Linear Single-Degree-of-Freedom (S-DoF) system of frequency $\omega_c$:

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

State-space representation:

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

Analytical solution for a generic instant $t$:

$$\begin{bmatrix} u \\ \dot{u} \end{bmatrix} = e^{\begin{bmatrix} 0 & 1 \\ -\omega_c^2 & 0 \end{bmatrix} t} \begin{bmatrix} u_0 \\ \dot{u}_0 \end{bmatrix}$$
Let’s now revisit Time Stepping Algorithms

Linear Single-Degree-of-Freedom (S-DoF) system of frequency $\omega_c$:

\[ \ddot{u} + \omega_c^2 u = 0 \]

State-space representation:

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

Analytical solution for one-time-step transition $\Delta t$:

\[
\begin{bmatrix} u_k \\ \dot{u}_k \end{bmatrix} = e^{ \begin{bmatrix} 0 & 1 \\ -\omega_c^2 & 0 \end{bmatrix} \Delta t} \begin{bmatrix} u_0 \\ \dot{u}_0 \end{bmatrix} = A_c^k \begin{bmatrix} u_0 \\ \dot{u}_0 \end{bmatrix}
\]
Analysis of Time Stepping Algorithms

Differential operator:

\[
A = \begin{bmatrix}
0 & 1 \\
-\omega_c^2 & 0
\end{bmatrix}
\]

\[
A\Phi = \lambda\Phi
\]

\[
\Phi^{-1}A\Phi = \Omega
\]

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

Transition matrix (exact):

\[
A_c = e^{\left(\begin{bmatrix}
0 & 1 \\
-\omega_c^2 & 0
\end{bmatrix}\Delta t\right)}
\]

\[
A_c\Phi_c = \lambda_c\Phi_c
\]

\[
\Phi_c^{-1}A_c\Phi_c = \Omega_c
\]

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

The following relation holds:

\[
\begin{cases}
\Omega_c = e^{\Omega\Delta t} \\
\Phi_c = \Phi
\end{cases}
\]
This is the proof:

\[ A_c = e^{A \Delta t} = \sum_{n=0}^{\infty} \frac{(A \Delta t)^n}{n!} \]

\[ = \sum_{n=0}^{\infty} \frac{(\Phi \Omega \Phi^{-1} \Delta t)^n}{n!} \]

\[ = \Phi \left( \sum_{n=0}^{\infty} \frac{(\Omega \Delta t)^n}{n!} \right) \Phi^{-1} \]

\[ = \Phi e^{\Omega \Delta t} \Phi^{-1} \]

\[ = \Phi \Omega_c \Phi^{-1} \]

\[ = \Phi_c \Omega_c \Phi^{-1} \]
Analysis of the Newmark Algorithm

Calculation of the transition matrix $A_d$ for the Newmark algorithm:

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

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

Calculation of the transition matrix $A_d$ for the Newmark algorithm:

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

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

\[
\begin{bmatrix}
\dot{u}_{k+1} \\
\ddot{u}_{k+1} \\
\dddot{u}_{k+1}
\end{bmatrix} = A_d \begin{bmatrix}
\dot{u}_k \\
\ddot{u}_k \\
\dddot{u}_k
\end{bmatrix}
\]
The Newmark Algorithm: Stability

The spectral radius $\rho$ of the transition matrix $A_d$ is defined as,

$$A_d \Phi_d = \lambda_d \Phi_d$$

$$\begin{bmatrix}
    u_{k+1} \\
    \dot{u}_{k+1} \\
    \ddot{u}_{k+1}
\end{bmatrix} = A_d^{k+1} \begin{bmatrix}
    u_0 \\
    \dot{u}_0 \\
    \ddot{u}_0
\end{bmatrix} = \Phi_d \begin{bmatrix}
    \lambda_{d,1}^{k+1} & 0 & 0 \\
    0 & \lambda_{d,2}^{k+1} & 0 \\
    0 & 0 & 0
\end{bmatrix} \Phi_d^{-1} \begin{bmatrix}
    u_0 \\
    \dot{u}_0 \\
    \ddot{u}_0
\end{bmatrix}$$

$$\rho = \max_j |\lambda_{d,j}| : \begin{cases}
    \rho \leq 1 \text{ stable} \\
    \rho > 1 \text{ unstable}
\end{cases}$$

and it is expressed as function of the dimensionless frequency $\omega_c \Delta t$. 
\[ \ddot{u} + \omega_c^2 u = 0 \]
Differential operator:

\[ A\Phi = \lambda \Phi \]
\[ \downarrow \]
\[ \Phi^{-1} A \Phi = \]
\[ \lambda = \pm i\omega_c \]

Transition matrix (approx):

\[ A_d \Phi_d = \lambda_d \Phi_d \]
\[ \downarrow \]
\[ \Phi_d^{-1} A_d \Phi_d = \]
\[ \lambda_d = e^{(-\xi_d \omega_d \pm i\omega_d \sqrt{1-\xi_d^2})\Delta t} \]

The following relation holds:

\[
\begin{align*}
\omega_d &\approx \arg \left( \frac{\lambda_d}{\Delta t} \right) \neq \omega_c \\
\xi_d &\approx -\frac{\ln|\lambda_d|}{\omega_d \Delta t} \neq 0
\end{align*}
\]

\(\omega_d\) and \(\xi_d\) are frequency and damping of the discretized system.
\[ \ddot{u} + \omega_c^2 u = 0 \]
The Newmark Algorithm: Damping Distortion

\[ \ddot{u} + \omega_c^2 u = 0 \]
The order of accuracy $p$ can be evaluated numerically:

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

$$\omega_c = 2\pi, \quad u_0 = 1 \text{m}, \quad v_0 = 0 \frac{\text{m}}{s}, \quad t_k = 0.1 \text{s}$$
Irons and Treharne’ Theorem (1972):

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

where.

- \(\omega_c\) is the \(c\)-th frequency of the model
- \(\omega_e\) is the frequency of the \(e\)-th element
Stability of the Newmark Method

For zero damping the Newmark method is conditionally stable if

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

where \( \omega_{\text{max}} \) 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:

\[
\begin{align*}
\mathbf{M} \ddot{\mathbf{u}}_n + \mathbf{r}(\mathbf{u}_n, \dot{\mathbf{u}}_n) &= \mathbf{f}(t_n) \\
\mathbf{u}(t_0) &= \mathbf{u}_0, \dot{\mathbf{u}}(t_0) &= \dot{\mathbf{u}}_0
\end{align*}
\]

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 $\Delta t = t_{k+1} - t_k$:

\[
\begin{cases}
M \ddot{u}_{k+1} + r \left( u_{k+1}, \dot{u}_{k+1} \right) = f \left( t_{k+1} \right) \\
u(t_0) = u_0, \dot{u}(t_0) = \dot{u}_0
\end{cases}
\]

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

\[
M \ddot{u}_{k+1} + r \left( \ddot{u}_{k+1} + \ddot{u}_{k+1} \beta \Delta t^2, \dot{u}_{k+1} + \ddot{u}_{k+1} \gamma \Delta t \right) = f \left( t_{k+1} \right)
\]

**Predictors** depend on previous time step solutions:

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

**Correctors** determine the current time step solution:

\[
\begin{cases}
u_{k+1} = \dddot{u}_{k+1} + \dddot{u}_{k+1} \beta \Delta t^2 \\
\dot{u}_{k+1} = \ddot{u}_{k+1} + \ddot{u}_{k+1} \gamma \Delta t
\end{cases}
\]
Implicit implementation based on Newton-Raphson iterations:

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

Assembly of restoring force and stiffness matrix loop over elements:

1: \( \text{for } i = 1 \text{ to } I \text{ do} \)
2: \( r_{i,k+1} \leftarrow \text{elementForce}(Z_i u_{k+1}) \)
3: \( r_{k+1} \leftarrow r_{k+1} + Z_i^T r_{i,k+1} \)
4: \( \text{end for} \)
Nonlinear Implementations of the Newmark Algorithm

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

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

In the above:

- The matrix to be inverted is $(M + C_\gamma \Delta t)$
- This matrix remains constant during the iterative procedure
- It can be factorized once and then only backward substitutions have to be performed
In the explicit version of the method $\beta = 0$ and the update step in each iteration becomes:

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

If further:

- $M$ is lumped (diagonal)
- $C$ 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: \( \tilde{u}_{k+1} \leftarrow u_k + \ddot{u}_k \Delta t + \dddot{u}_k \left( \frac{1}{2} - \beta \right) \Delta t^2 \)
2: \( \ddot{u}_{k+1} \leftarrow \dot{u}_k + \ddot{u}_k \left( 1 - \gamma \right) \Delta t \)
3: \( \dot{u}_{k+1} \leftarrow \left( M + C \gamma \Delta t + K \beta \Delta t^2 \right)^{-1} \left( f(t_{k+1}) - r \left( \tilde{u}_{k+1}, \ddot{u}_{k+1} \right) \right) \)
4: \( \dot{u}_{k+1} \leftarrow \dot{u}_{k+1} + \ddot{u}_{k+1} \gamma \Delta t \)
5: \( u_{k+1} \leftarrow \tilde{u}_{k+1} + \ddot{u}_{k+1} \beta \Delta t^2 \)

- \( K \) 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 \( K \)) truncated at one iteration.

Assembly of restoring force and stiffness matrix loop over elements:

1: \textbf{for } i = 1 \textbf{ to } I \textbf{ do} \\
2: \hspace{1cm} r_{i,k+1} \leftarrow \text{elementForce} \left( Z_i u_{k+1} \right) \\
3: \hspace{1cm} r_{k+1} \leftarrow r_{k+1} + Z_i^T r_{i,k+1} \\
4: \hspace{1cm} \textbf{end for}

1: \textbf{for } i = 1 \textbf{ to } I \textbf{ do} \\
2: \hspace{1cm} K_{i,k+1} \leftarrow \text{elementStiff} \left( Z_i u_{k+1} \right) \\
3: \hspace{1cm} K_{k+1} \leftarrow K_{k+1} + Z_i^T K_{i,k+1} Z_i \\
4: \hspace{1cm} \textbf{end for}
Nonlinear Implementation of the Newmark Algorithm

Single modified Newton-Raphson (constant $K$) iteration

In this case a residual force balance $\varepsilon_{k+1}^{(2)}$ occurs that tends to zero as $\Delta t$ tends to zero.
The HHT-\(\alpha\) Algorithm

\(\alpha\)-shifted equation of motion:

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

Interpolation equations and integration parameters:

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

\[\beta = (1 - \alpha)^2 / 4, \quad \gamma = (1 - 2\alpha)/2\]

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

Implicit implementation based on Newton-Raphson iterations:

1. \( \dot{u}_{k+1} \leftarrow 0 \)
2. \( u_{k+1} \leftarrow u_k + \dot{u}_k \Delta t + \ddot{u}_k (\frac{1}{2} - \beta) \Delta t^2 + \ddot{u}_{k+1} \beta \Delta t^2 \)
3. \( \dot{u}_{k+1} \leftarrow \dot{u}_k + \ddot{u}_k (1 - \gamma) \Delta t + \ddot{u}_{k+1} \gamma \Delta t \)
4. \( \epsilon \leftarrow \hat{f}_{k+1} \)
5. \( \textbf{while} \quad \| \epsilon \| >= Tol \quad \textbf{do} \)
6. \( \Delta \ddot{u}_{k+1} \leftarrow \hat{M}^{-1} \epsilon \)
7. \( \ddot{u}_{k+1} \leftarrow \ddot{u}_{k+1} + \Delta \ddot{u}_{k+1} \)
8. \( \dot{u}_{k+1} \leftarrow \dot{u}_{k+1} + \Delta \ddot{u}_{k+1} \gamma \Delta t \)
9. \( u_{k+1} \leftarrow u_{k+1} + \Delta \ddot{u}_{k+1} \beta \Delta t^2 \)
10. \( \epsilon \leftarrow \hat{f}_{k+1} \)
11. \( \textbf{end while} \)

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

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

\[
\hat{f}_{k+1} = (1 + \alpha) f_{k+1} - \alpha f_k - (1 + \alpha) \ddot{r}_{k+1} + \alpha \ddot{r}_k - (1 + \alpha) C \ddot{u}_{k+1} + \alpha C \ddot{u}_k + \alpha (C \gamma \Delta t + K \beta \Delta t^2) \ddot{u}_k
\]
Nonlinear Implementation of the HHT-α Algorithm

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

1: \( \tilde{u}_{k+1} \leftarrow u_k + \dot{u}_k \Delta t + \ddot{u}_k \left( \frac{1}{2} - \beta \right) \Delta t^2 \)
2: \( \hat{u}_{k+1} \leftarrow \dot{u}_k + \ddot{u}_k (1 - \gamma) \Delta t \)
3: \( \ddot{u}_{k+1} \leftarrow \hat{M}^{-1} \hat{f}_{k+1} \)
4: \( \dot{u}_{k+1} \leftarrow \hat{u}_{k+1} + \ddot{u}_{k+1} \gamma \Delta t \)
5: \( u_{k+1} \leftarrow \tilde{u}_{k+1} + \ddot{u}_{k+1} \beta \Delta t^2 \)

- \( K \) 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 \( K \)) truncated at one iteration.

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

\[
\hat{M} = M + C \gamma \Delta t (1 + \alpha) + K \beta \Delta t^2 (1 + \alpha) \\
\hat{f}_{k+1} = (1 + \alpha) f_{k+1} - \alpha f_k - (1 + \alpha) \ddot{r}_{k+1} + \alpha \ddot{r}_k - \\
(1 + \alpha) C \dot{u}_{k+1} + \alpha C \dot{u}_k + \alpha (C \gamma \Delta t + K \beta \Delta t^2) \ddot{u}_k
\]
Analysis of the HHT-α Algorithm: Spectral Radius

\[ \ddot{u} + \omega_c^2 u = 0 \]

**Graph:**

- Parameters: \( \alpha = -0.33, -0.10, 0.00 \)
- Variables: \( \omega_c, \Delta t \)
- Graph shows the spectral radius \( \rho \) as a function of \( \omega_c \Delta t \) for different values of \( \alpha \). The lines indicate the decay of the spectral radius with increasing \( \omega_c \Delta t \).
\[ \ddot{u} + \omega_c^2 u = 0 \]
Analysis of the HHT-$\alpha$ Algorithm: Damping Bias

\[ \ddot{u} + \omega_c^2 u = 0 \]
Free-decay response of a S-DoF system:

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

$$\omega_c = \frac{2\pi}{5}, \quad u_0 = 1 m, \quad v_0 = 0 \frac{m}{s}, \quad \Delta t = 2 e - 4 s$$
Free-decay response of a S-DoF system:

\[ \ddot{u} + \omega_c^2 u = 0 \]

\[ \omega_c = 2\pi 5 \text{ rad/s}, \quad u_0 = 1 \text{ m}, \quad v_0 = 0 \frac{m}{s}, \quad \Delta t = 2e - 2s \]
Recalling the Eigenvalue Analysis

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

\[ \mathbf{M} \ddot{\mathbf{u}} + \mathbf{K} \mathbf{u} = 0 \]

Defining a matrix \( \Phi \) whose columns are the eigen-vectors \( \{ \phi_i, \ i = 1 \ldots n \} \) and a diagonal matrix \( \Omega^2 \) storing the eigenvalues \( \{ \omega_i^2, \ i = 1 \ldots n \} \), i.e.:

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

we can write the eigen-problem as:

\[
[\mathbf{K} - \mathbf{M}\Omega^2] \Phi = 0
\]
Mode Superposition Method

\( M - Orthonormality \)

We may scale \( \Phi \), as they form a basis, and we can chose these such that:

\[
\Phi^T M \Phi = I \quad \rightarrow \quad \Phi^T K \Phi = \Omega^2
\]

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

\[
\ddot{q}(t) + \Phi^T C \Phi \dot{q}(t) + \Omega^2 q(t) = \Phi^T f(t)
\]

\[
q_0 = \Phi^T M u_0; \quad \dot{q}_0 = \Phi^T M \dot{u}_0
\]

under a Proportional Damping Assumption: \( \phi_i^T C \phi_j = 2 \omega_i \xi_i \delta_{ij} \), where \( \xi_i \) is a modal damping parameter and \( \delta_{ij} \) 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 \( C \) 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:

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

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.