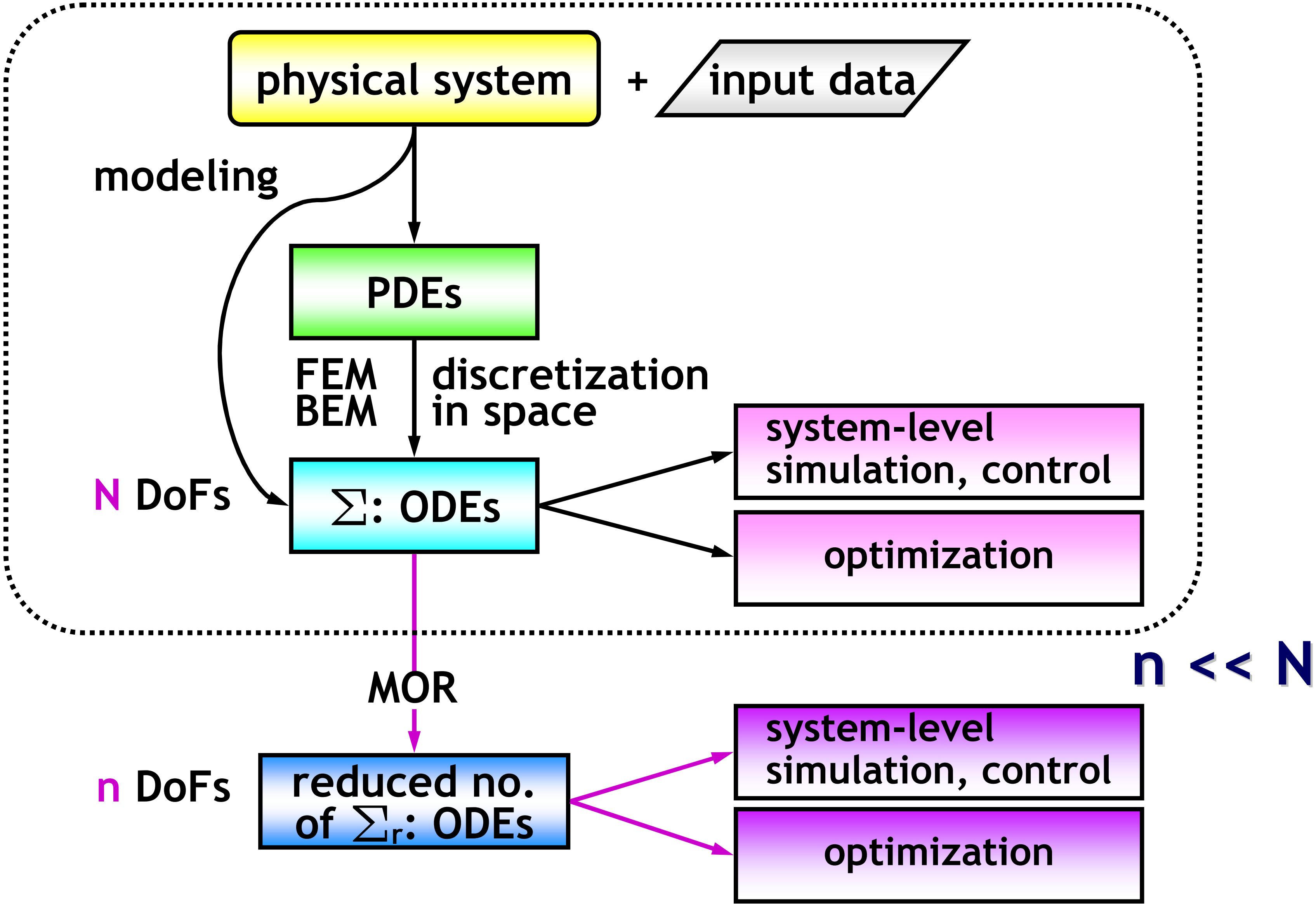
MATLAB을 이용한 동역학해석모델에 대한 모델차수축소기법

February 6, 2009

Prof. Jeong Sam Han
School of Mechanical Engineering
Andong National University

E-mail: jshan@andong.ac.kr

Introduction



Part I

- **Theory on Krylov-based MOR**
  - **Concept**
  - **Moment-matching method**
  - **Krylov subspace**
  - **Arnoldi process**
  - **MOR for frequency response problems**

Concept of Model Order Reduction

- Original 2nd order system (N×N)

$$\underset{(N\times N)}{\mathbf{M}}\,\underset{(N)}{\ddot{\mathbf{x}}} + \underset{(N\times N)}{\mathbf{C}}\,\underset{(N)}{\dot{\mathbf{x}}} + \underset{(N\times N)}{\mathbf{K}}\,\underset{(N)}{\mathbf{x}} = \underset{(N\times 1)}{\mathbf{b}}\,\underset{(1)}{u}$$

$$\mathbf{x}(t) \cong \mathbf{V}\mathbf{z}(t)$$

MOR

$$\underset{(N)}{\mathbf{x}} = \underset{(N\times n)}{\mathbf{V}}\,\underset{(n)}{\mathbf{z}}$$

Projection matrix

- Reduced 2nd order system (n×n)

$$\underset{(n\times n)}{\mathbf{M}_r}\,\underset{(n)}{\ddot{\mathbf{z}}} + \underset{(n\times n)}{\mathbf{C}_r}\,\underset{(n)}{\dot{\mathbf{z}}} + \underset{(n\times n)}{\mathbf{K}_r}\,\underset{(n)}{\mathbf{z}} = \underset{(n\times 1)}{\mathbf{b}_r}\,\underset{(1)}{u}$$

⇨ **Because n<<N, very efficient simulations are possible!**

Moment-Matching Method (Damped)

- SIMO 2nd order system

$$\mathbf{M}\ddot{\mathbf{x}}(t) + \mathbf{C}\dot{\mathbf{x}}(t) + \mathbf{K}\mathbf{x}(t) = \mathbf{b}u(t)$$
$$\mathbf{y}(t) = \mathbf{L}\mathbf{x}(t)$$

$\mathbf{x}(t) \cong \mathbf{V}\mathbf{z}(t)$

Projection

$$\mathbf{M}_r\ddot{\mathbf{z}}(t) + \mathbf{C}_r\dot{\mathbf{z}}(t) + \mathbf{K}_r\mathbf{z}(t) = \mathbf{b}_r u(t)$$
$$\mathbf{y}(t) = \mathbf{L}_r\mathbf{z}(t)$$

$$\mathbf{M}_r = \mathbf{V}^T\mathbf{M}\mathbf{V} \quad \mathbf{K}_r = \mathbf{V}^T\mathbf{K}\mathbf{V}$$
$$\mathbf{b}_r = \mathbf{V}^T\mathbf{b} \quad \mathbf{L}_r = \mathbf{L}\mathbf{V}$$

$$\mathbf{C}_r = \mathbf{V}^T\mathbf{C}\mathbf{V} = \mathbf{V}^T(\alpha\mathbf{M} + \beta\mathbf{K})\mathbf{V} = \alpha\mathbf{M}_r + \beta\mathbf{K}_r$$

proportional damping or constant damping ratio

- Transfer function (for an undamped system)

$$\mathbf{H}(s) = \mathbf{L}(s^2\mathbf{M} + \mathbf{K})^{-1}\mathbf{b} = \sum_{i=0}^{\infty} \mathbf{L}(-\mathbf{K}^{-1}\mathbf{M})^i\mathbf{K}^{-1}\mathbf{b}s^{2i} = \sum_{i=0}^{\infty} \mathbf{m}_i s^{2i}$$

Series expansion

moment of H(s)

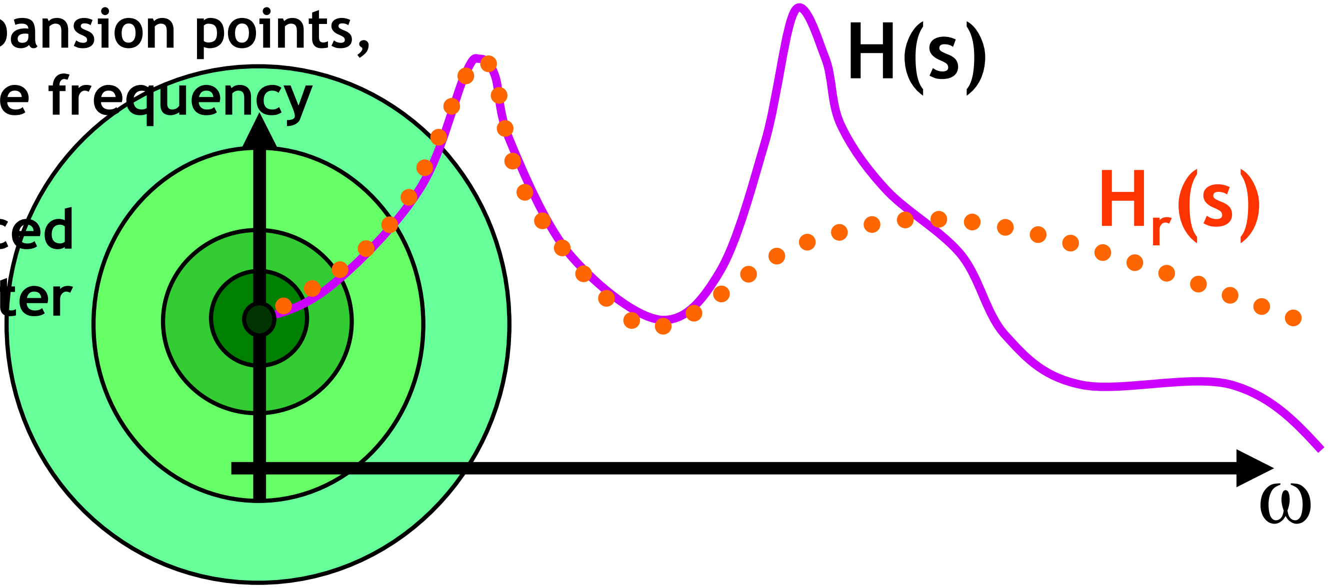
$$\mathbf{H}(s) = \mathbf{m}_0 + \mathbf{m}_1(s - s_0) + \cdots + \mathbf{m}_q(s - s_0)^q + \cdots = \sum_{i=0}^{\infty} \mathbf{m}_i(s - s_0)^i$$
$$\hat{\mathbf{H}}(s) = \hat{\mathbf{m}}_0 + \hat{\mathbf{m}}_1(s - s_0) + \cdots + \hat{\mathbf{m}}_q(s - s_0)^q + \cdots = \sum_{i=0}^{\infty} \hat{\mathbf{m}}_i(s - s_0)^i$$
$$\Rightarrow \mathbf{m}_i = \hat{\mathbf{m}}_i, \quad i = 1, 2, \cdots, n$$

- If the columns of $\mathbf{V}$ form a basis for the Krylov subspace $\mathcal{K}_n(-\mathbf{K}^{-1}\mathbf{M}, \mathbf{K}^{-1}\mathbf{b}) = \mathrm{span}\{\mathbf{K}^{-1}\mathbf{b}, \ldots, (-\mathbf{K}^{-1}\mathbf{M})^{n-1}\cdot\mathbf{K}^{-1}\mathbf{b}\}$, then the first n moments of the original and reduced models match!

Property of Moment-Matching

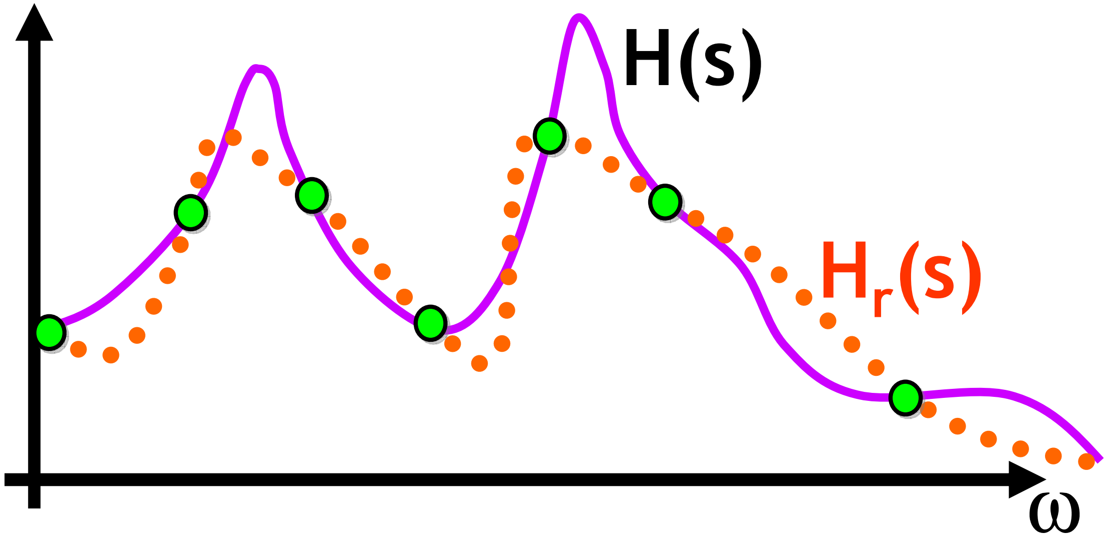
- **Moment-matching:**

- Accurate around expansion points, but inaccurate on wide frequency band
- A higher order reduced model results in a better approximation



- **Point-matching:**

- Can be very inaccurate in between points



Krylov Subspace

Krylov subspace of n-th dimension of $\mathbf{A} \in \Re^{N \times N}$ and $\mathbf{v} \in \Re^{N}$

$$\mathcal{K}_n(\mathbf{A}, \mathbf{v}) = \text{span}\{\mathbf{v}, \mathbf{A}\cdot\mathbf{v}, \mathbf{A}^2\cdot\mathbf{v}, \ldots, \mathbf{A}^{n-1}\cdot\mathbf{v}\}$$

- A subspace spanned by the original $\mathbf{v}$ and the vectors produced by consecutive multiplication of the matrix $\mathbf{A}$ to this vector up to n-1 times
- The resulting vectors form a basis of n-dimensional subspace
- Direct computation is numerically unstable because of rounding errors → Arnoldi process is used to construct an orthonormal basis
- Included in 10 top algorithms of the 20th century
- Named after Russian applied mathematician and naval engineer Alexei Krylov

Orthonormalization

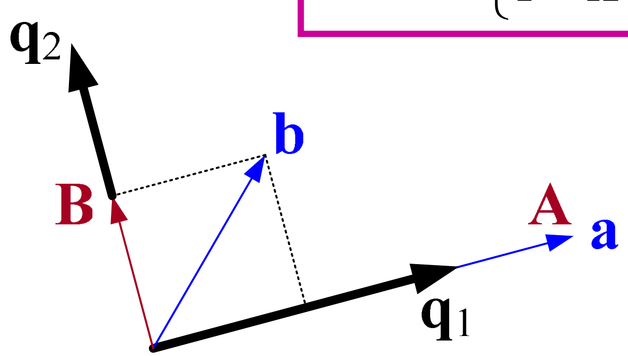
- Gram-Schmidt process
  - To subtract from every new vector its components in the directions that are already settled

Independent vectors a, b, c

⇨ Orthogonal A, B, C

⇨ Orthonormal q_1, q_2, q_3

$$q_i^T q_j = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$$



$$\mathbf{q}_1 = \mathbf{A}/\|\mathbf{A}\|;$$

$$\mathbf{B} = \mathbf{b} - \frac{\mathbf{A}^T\mathbf{b}}{\|\mathbf{A}\|}\frac{\mathbf{A}}{\|\mathbf{A}\|} = \mathbf{b} - (\mathbf{q}_1^T\mathbf{b})\mathbf{q}_1$$

$$\mathbf{q}_2 = \mathbf{B}/\|\mathbf{B}\|;$$

$$\mathbf{C} = \mathbf{c} - \frac{\mathbf{A}^T\mathbf{c}}{\|\mathbf{A}\|}\frac{\mathbf{A}}{\|\mathbf{A}\|} - \frac{\mathbf{B}^T\mathbf{c}}{\|\mathbf{B}\|}\frac{\mathbf{B}}{\|\mathbf{B}\|} = \mathbf{c} - (\mathbf{q}_1^T\mathbf{c})\mathbf{q}_1 - (\mathbf{q}_2^T\mathbf{c})\mathbf{q}_2$$

$$\mathbf{q}_3 = \mathbf{C}/\|\mathbf{C}\|;$$

$$\frac{\mathbf{A}^T\mathbf{b}}{\|\mathbf{A}\|}\frac{\mathbf{A}}{\|\mathbf{A}\|} = \frac{\mathbf{A}^T\mathbf{b}}{\mathbf{A}^T\mathbf{A}}\mathbf{A}$$

$$\frac{\mathbf{B}^T\mathbf{c}}{\|\mathbf{B}\|}\frac{\mathbf{B}}{\|\mathbf{B}\|} = \frac{\mathbf{B}^T\mathbf{c}}{\mathbf{B}^T\mathbf{B}}\mathbf{B}$$

Arnoldi Process

- Given a nonzero starting vector $\mathbf{r}(=\mathbf{K}^{-1}\mathbf{b})$ and a matrix $\mathbf{A}(=\mathbf{K}^{-1}\mathbf{M})$, this algorithm produces orthonormal $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n$ such that

$$\mathrm{span}\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k\} = \mathrm{span}\{\mathbf{r}, \mathbf{Ar}, \dots, \mathbf{A}^{k-1}\mathbf{r}\} \text{ for } k = 1,2,\dots,n$$

```
v_1 = r/‖r‖_2
for k = 1,2,…,n-1
    v_{k+1} ← A v_k                 (new vector generation)
    for j = 1,2,…,k                 (orthogonalization)
        h_{jk} ← v_j^T v_{k+1}      (Gram-Schmidt coefficient)
        v_{k+1} ← v_{k+1} - h_{jk} v_j
    h_{k+1,k} ← ‖v_{k+1}‖_2
    if h_{k+1,k} = 0
        set flag(span{v_1,v_2,…,v_k} is invariant under A)
        exit
    v_{k+1} ← v_{k+1} / h_{k+1,k}   (normalization)
```

⇨ $\mathbf{V}^T\mathbf{V}=\mathbf{I}_n$ where colspan$\{\mathbf{V}\}=\mathcal{K}_n(\mathbf{K}^{-1}\mathbf{M}, \mathbf{K}^{-1}\mathbf{b})$

Generation of New Vector (v_{k+1})

1) Let $\mathbf{v}_{k+1} = \mathbf{A}\mathbf{v}_k$ where $\mathbf{A} = -\mathbf{K}^{-1}\mathbf{M}$
2) Multiply $\mathbf{K}$ on both sides, $\mathbf{K}\mathbf{v}_{k+1} = -\mathbf{M}\mathbf{v}_k$
3) Decompose $\mathbf{K}=\mathbf{L}\mathbf{U}$, then $\mathbf{L}\mathbf{U}\mathbf{v}_{k+1} = -\mathbf{M}\mathbf{v}_k$
4) Solve $\mathbf{L}\mathbf{w} = -\mathbf{M}\mathbf{v}_k$ for $\mathbf{w}$, where $\mathbf{U}\mathbf{v}_{k+1} = \mathbf{w}$
5) Solve $\mathbf{U}\mathbf{v}_{k+1} = \mathbf{w}$ for $\mathbf{v}_{k+1}$

- Decomposition according to the matrix $\mathbf{A}$,
  - Cholesky decomposition: symmetric and positive definite
  - LU decomposition: not symmetric
  - LDL^T decomposition: symmetric but indefinite

MATLAB Implementation for Arnoldi

```
%%% perform model order reduction by arnoldi algorithm (order of n)

[L, U] = lu(K);     % LU matrix factorization (K = L*U)

v = U\(L\B);        % the starting vector by left division
v = (1/norm(full(v)))*v;     % normalizing the starting vector


% generate krylov vectors up to n
for j = 2:n
    v(:,j) = U\(L\(M*v(:,j-1)));
    for k = 1:j-1
        hv = v(:,k)'*v(:,j);
        v(:,j) = v(:,j) - hv*v(:,k);
    end
    v(:,j) = v(:,j)/norm(v(:,j));
end


diff_v = norm(v'*v - eye(n));     % check orthonormality
```

```
r=K^-1B
Kr=B
LUr=B
Ur=L\B
r=U\(L\B)
```

```
w=K^-1Mv
Kw=Mv
LUw=Mv
Uw=L\(Mv)
w=U\(L\(Mv))
```

MOR for Frequency Response Problems

Original system (N×N)

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

$$\mathbf{x}=\mathbf{V}\mathbf{z}$$

$$\mathbf{M}_r=\mathbf{V}^T\mathbf{M}\mathbf{V}$$
$$\mathbf{C}_r=\mathbf{V}^T\mathbf{C}\mathbf{V}$$
$$\mathbf{K}_r=\mathbf{V}^T\mathbf{K}\mathbf{V}$$
$$\mathbf{b}_r=\mathbf{V}^T\mathbf{b}$$

Reduced system (n×n)

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

$$\text{colspan}\{\mathbf{V}\}=\mathcal{K}_n(\mathbf{K}^{-1}\mathbf{M},\ \mathbf{K}^{-1}\mathbf{b})\ \text{where}\ \mathbf{b}\in\Re^N$$

- When a damping matrix $\mathbf{C}$ is modeled as a proportional damping ($\mathbf{C}=\alpha\mathbf{M}+\beta\mathbf{K}$) or a damping with a constant damping ratio ($\mathbf{C}=\beta_c\mathbf{K}$), the above projection matrix $\mathbf{V}$ is enough to generate a reduced system because the damping matrix is a linear combination of $\mathbf{M}$ and/or $\mathbf{K}$ therefore it does not contribute to the projection matrix.

MATLAB Implementation for Projection

```
%%% generate reduced system matrices by projection

Mr = full(v'*M*v);
Kr = full(v'*K*v);
Br = full(v'*B);
Cr = full(C*v);

NAMESr = NAMES;

% damping with a proportional damping
alpha = 2E-6; beta = 2E-6;

Er = alpha*Mr + beta*Kr;        % E = αM + βK
```

Process of Model Order Reduction

ANSYS

FE model generation

- FULL files at Ω_1 and Ω_2
- List of output DOFs

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

$$\underbrace{(\mathbf{K} - \Omega_1^2\mathbf{M} + i\Omega_1\mathbf{C})}_{\mathbf{K}_c}\underbrace{\mathbf{x}_{1c}}_{\mathbf{x}_c} = \underbrace{\mathbf{b}_c}_{\mathbf{F}_c}$$

$$\mathbf{K}_c \mathbf{x}_c = \mathbf{F}_c$$

$$\overbrace{(\mathbf{K} - \Omega_2^2\mathbf{M} + i\Omega_2\mathbf{C})}^{\mathbf{K}_c}\overbrace{\mathbf{x}_{2c}}^{\mathbf{x}_c} = \overbrace{\mathbf{b}_c}^{\mathbf{F}_c}$$

M, C, K, b

MATLAB

Model order reduction

- Reduced-order models

$$\text{colspan}\{\mathbf{V}\} = \mathcal{K}_n\{(s_0\mathbf{M} + \mathbf{K})^{-1}\mathbf{M}, (s_0\mathbf{M} + \mathbf{K})^{-1}\mathbf{b}\}$$

where $s_0 = -(2\pi f)^2$

$$\mathbf{M}_r = \mathbf{V}^T\mathbf{M}\mathbf{V}$$
$$\mathbf{C}_r = \mathbf{V}^T\mathbf{C}\mathbf{V}$$
$$\mathbf{K}_r = \mathbf{V}^T\mathbf{K}\mathbf{V}$$
$$\mathbf{b}_r = \mathbf{V}^T\mathbf{b}$$

$\mathbf{M}_r$, $\mathbf{C}_r$, $\mathbf{K}_r$, $\mathbf{b}_r$

MATLAB

Post-processing

- Frequency response

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

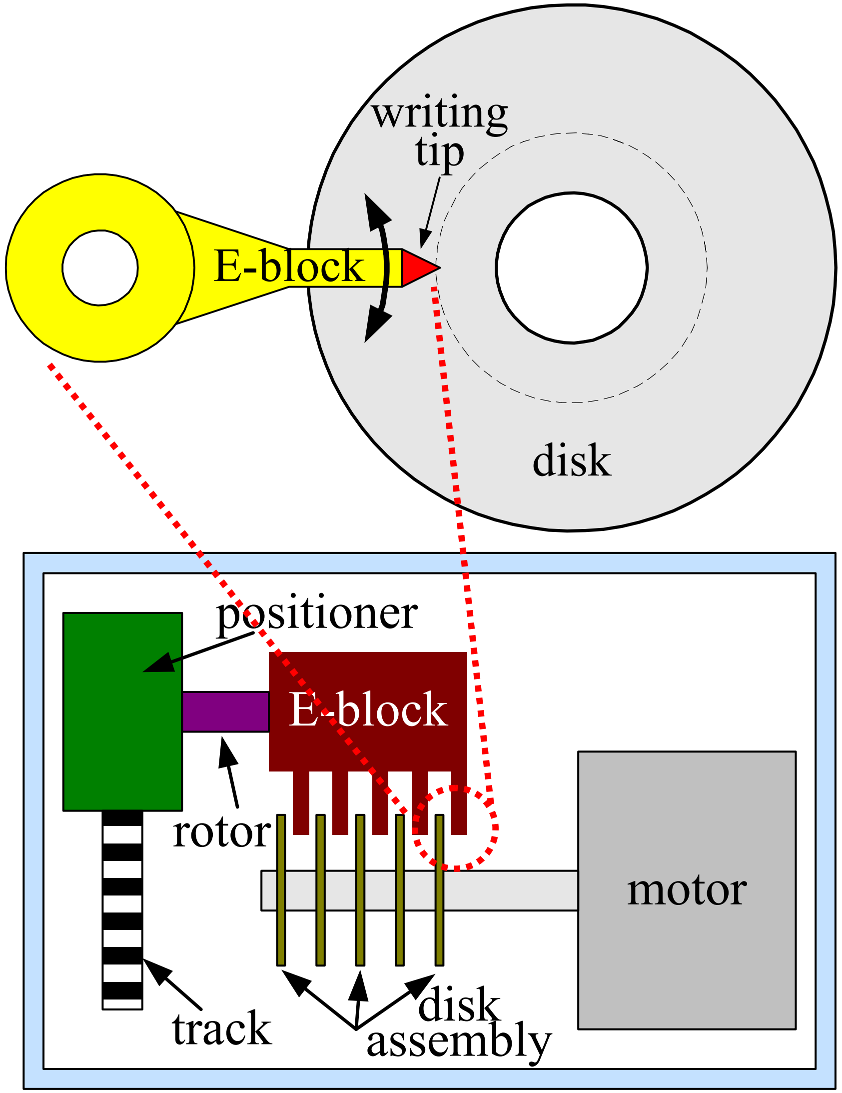
$$(\mathbf{K}_r - \Omega^2\mathbf{M}_r + i\Omega\mathbf{C}_r)\mathbf{z}_c = \mathbf{b}_{rc}$$

MATLAB Implementation for FRF

```
%%% perform frequency responses with the reduced system

nstep = (80+1); fstart = 0.; fend = 80.;  % 0~80 Hz @81 frequencies
fdel = (fend - fstart)/(nstep - 1);

for k = 1:nstep
    kfrq = fstart + (k-1)*fdel;
    komg = 2*pi*kfrq;                     % calculation of omgega

    Kc = Kr - (komg^2)*Mr + i*komg*Er;    % {Kc} generation

    kXc = Kc\Br;                          % solve {Kc}{xc}={Fc}
    kYc = Cr*kXc;                         % y=Cx

    Yc(k,:) = [kfrq, kYc'];               % frq, responses
end
```

Structural Examples

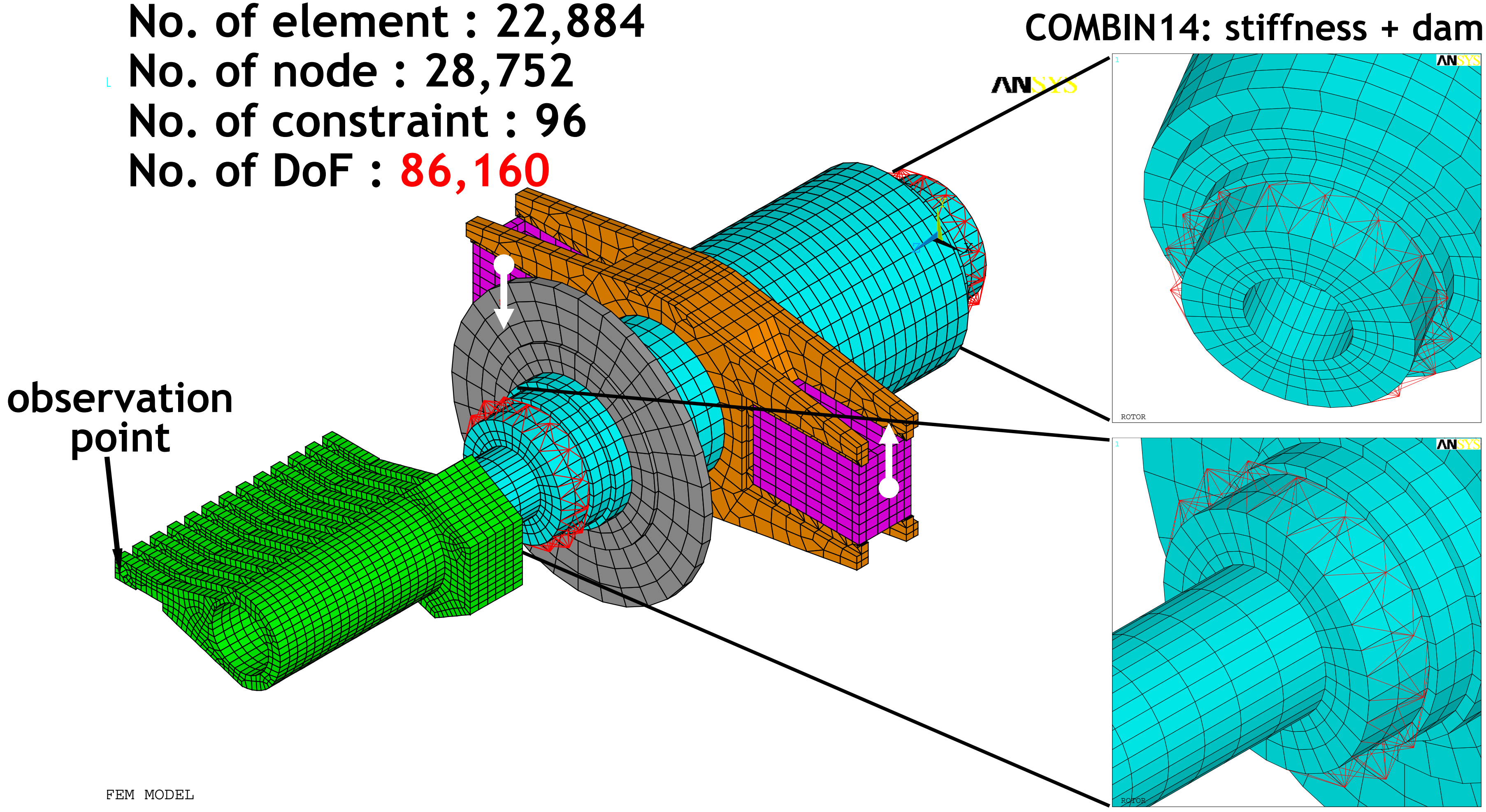
- HDD Track Writer
  - Control to position E-block tips to write data tracks on an HDD disk
  - Frequency response until some kHz is important for control design and for structural improvement
  - Therefore, harmonic simulation is necessary



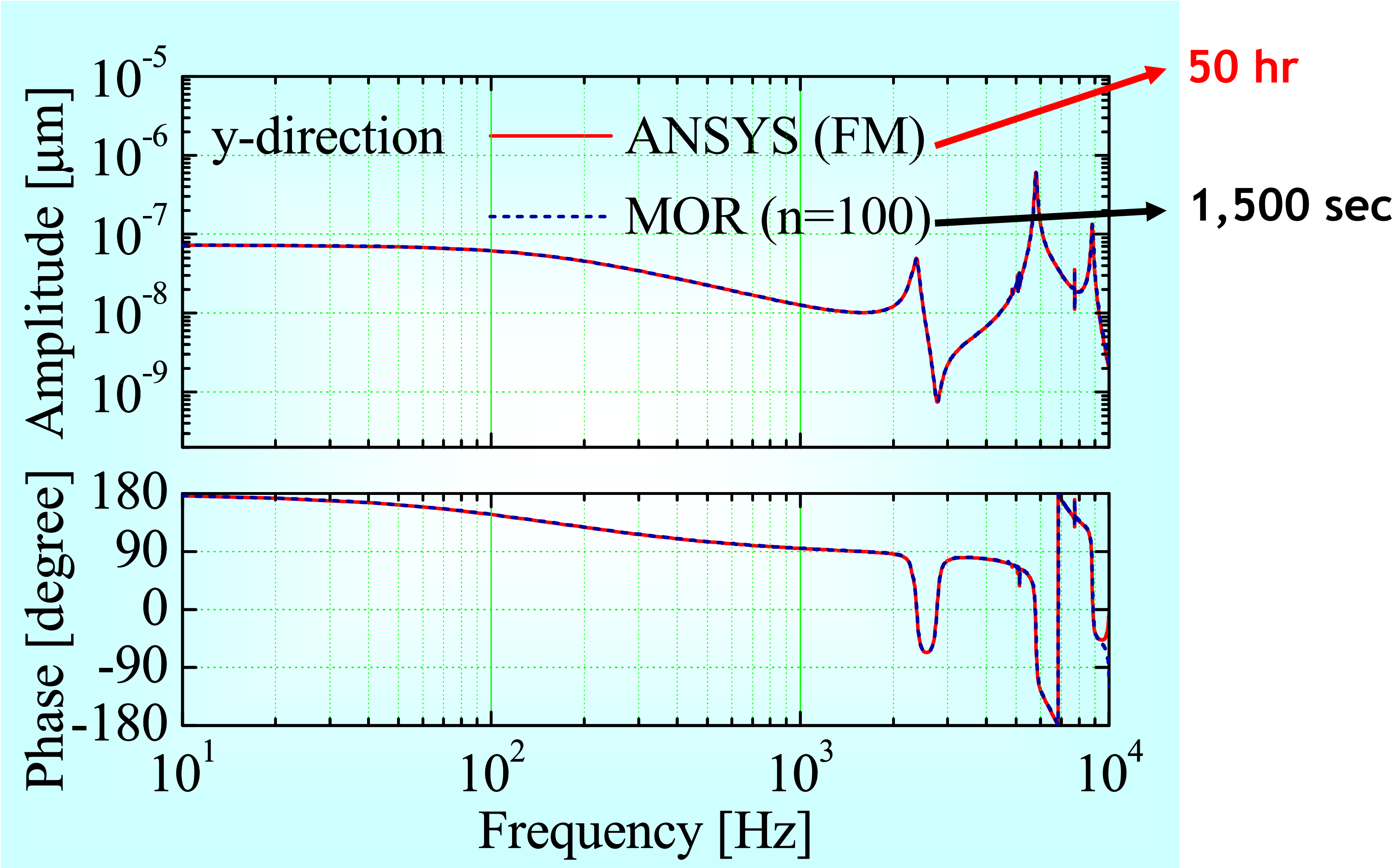
HDD Track Writer

- ANSYS FE model

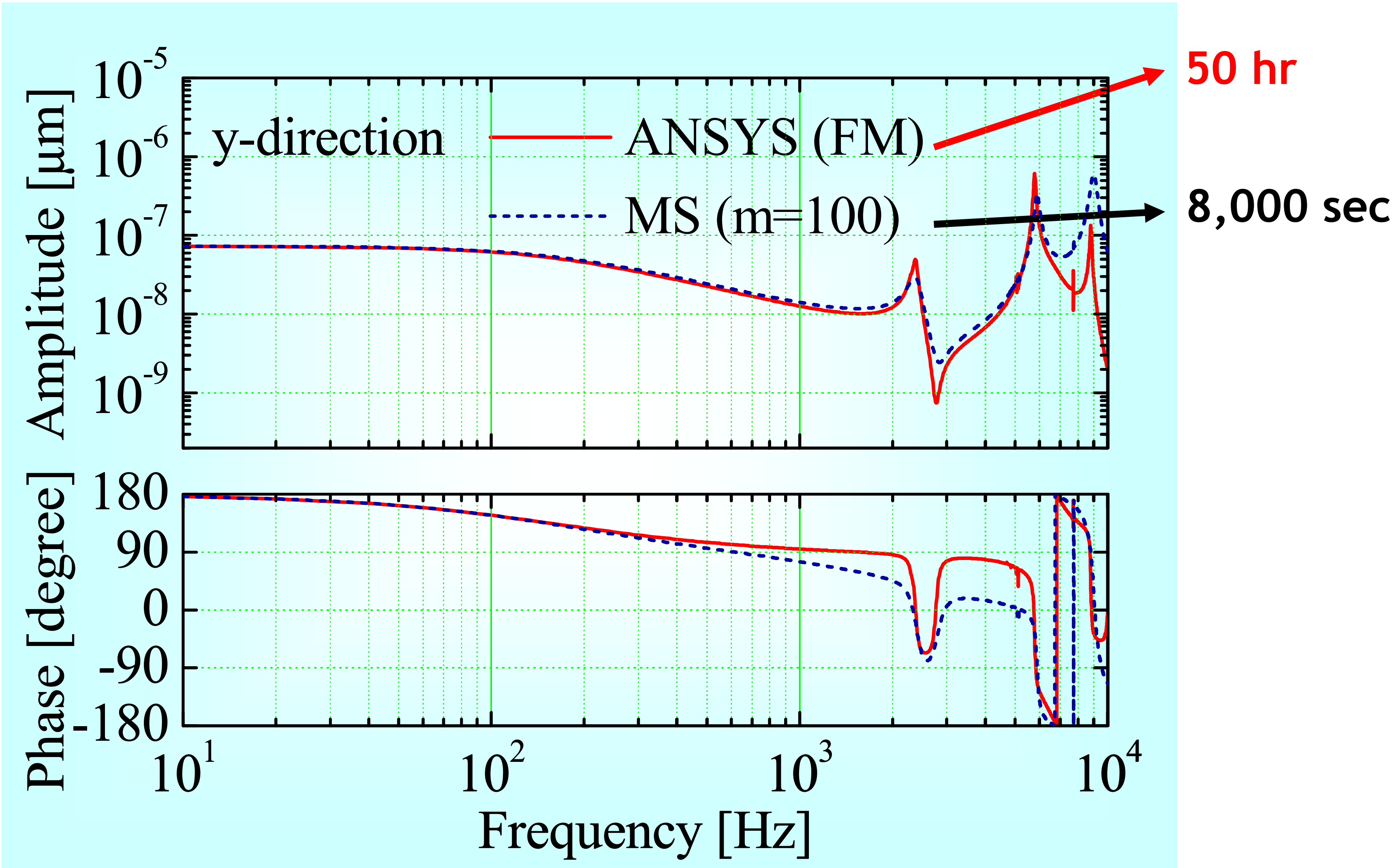
No. of element : 22,884
No. of node : 28,752
No. of constraint : 96
No. of DoF : 86,160



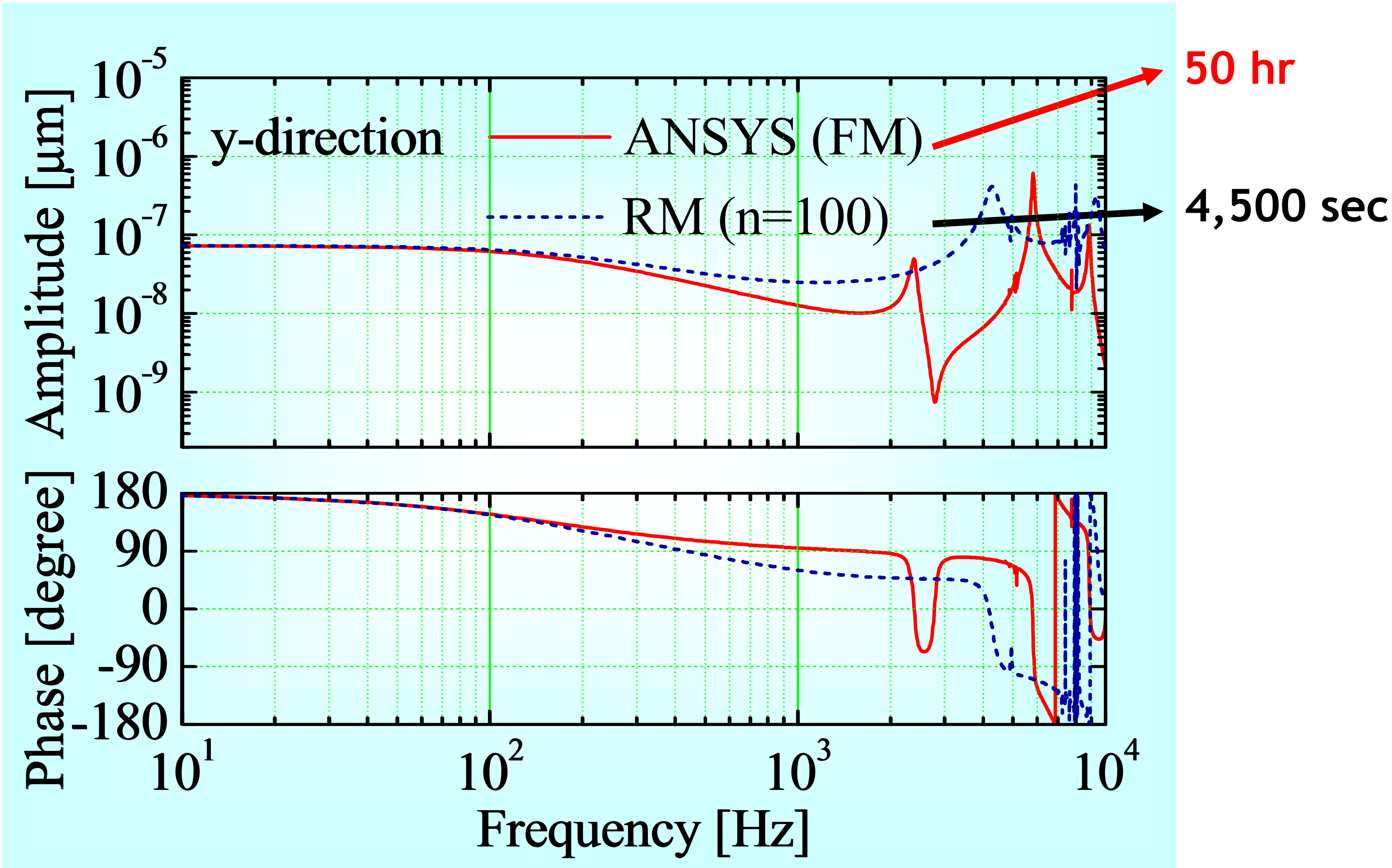
Harmonic Analysis : FM vs. MOR



Harmonic Analysis : FM vs. MS

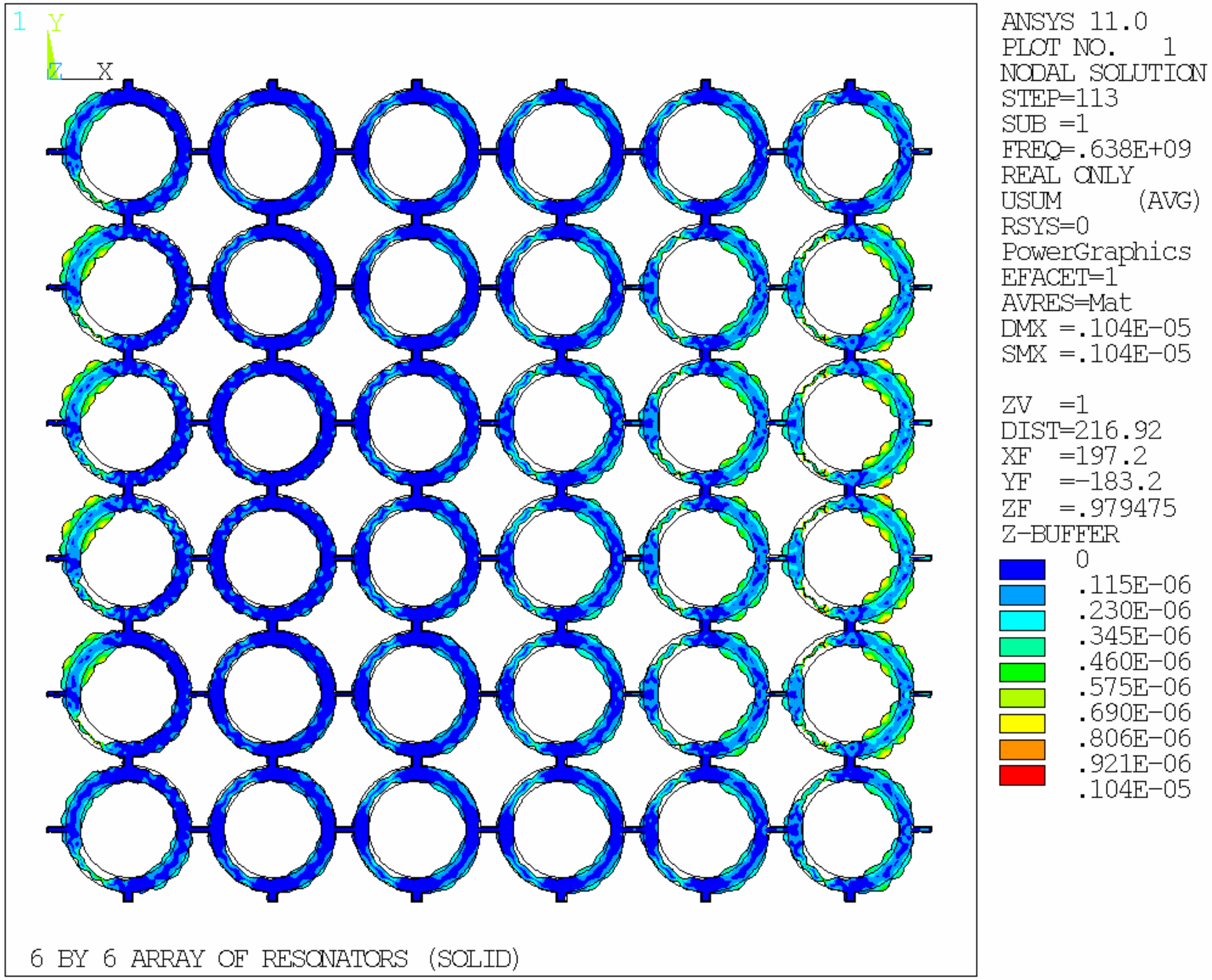


Harmonic Analysis : FM vs. RM



Array-type MEMS resonators

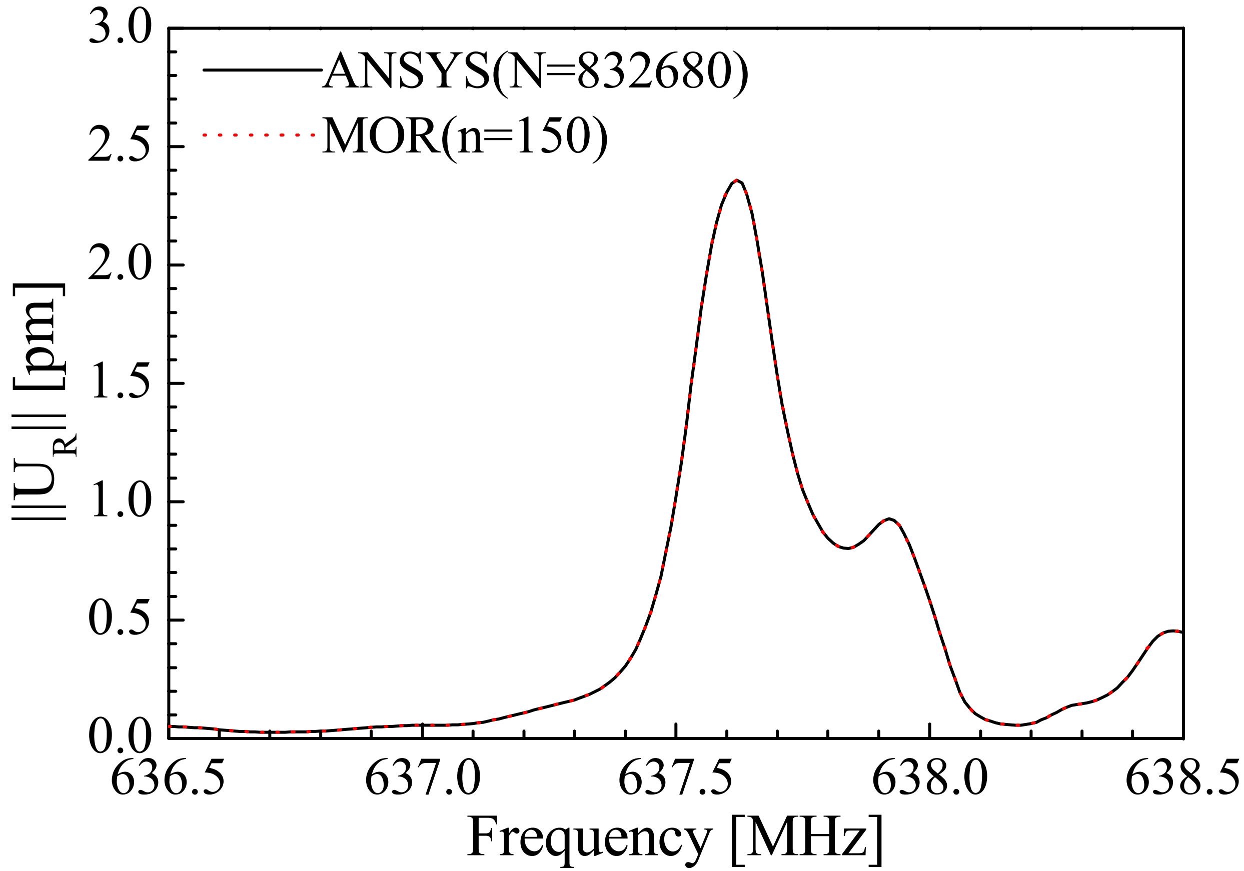
- FRF calculation for 6x6 MEMS resonators
  - Size (μm) : ϕ 40~50, t 2



- No. of elements: 194,880 (SOLID45)
- No. of nodes: 278,160
- No. of DOFs: 832,680
- Frequency range: 636.5~638.5 MHz
- Elapsed time: 89,696 sec. (1 day)
- No. of modes required: 8,000~9,000

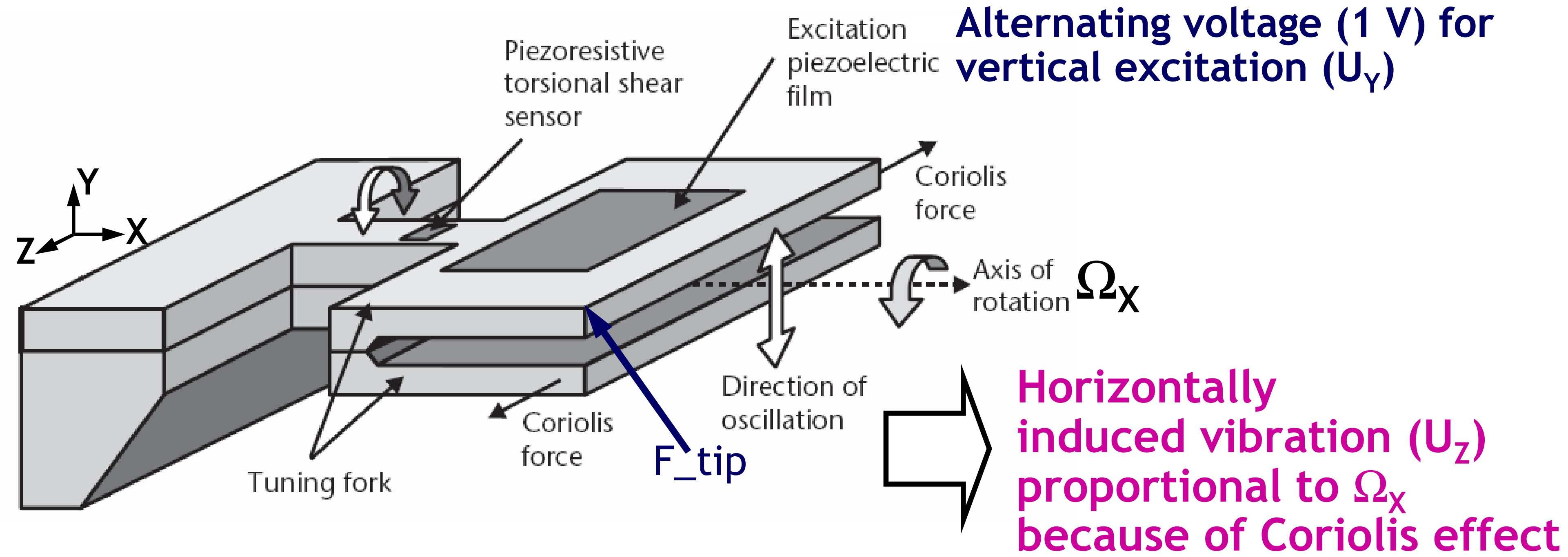
Array-type MEMS resonators

- FRF of a 6x6 resonator



Piezoelectric-Structural Examples

- **Angular velocity μ-sensor using Coriolis effect**

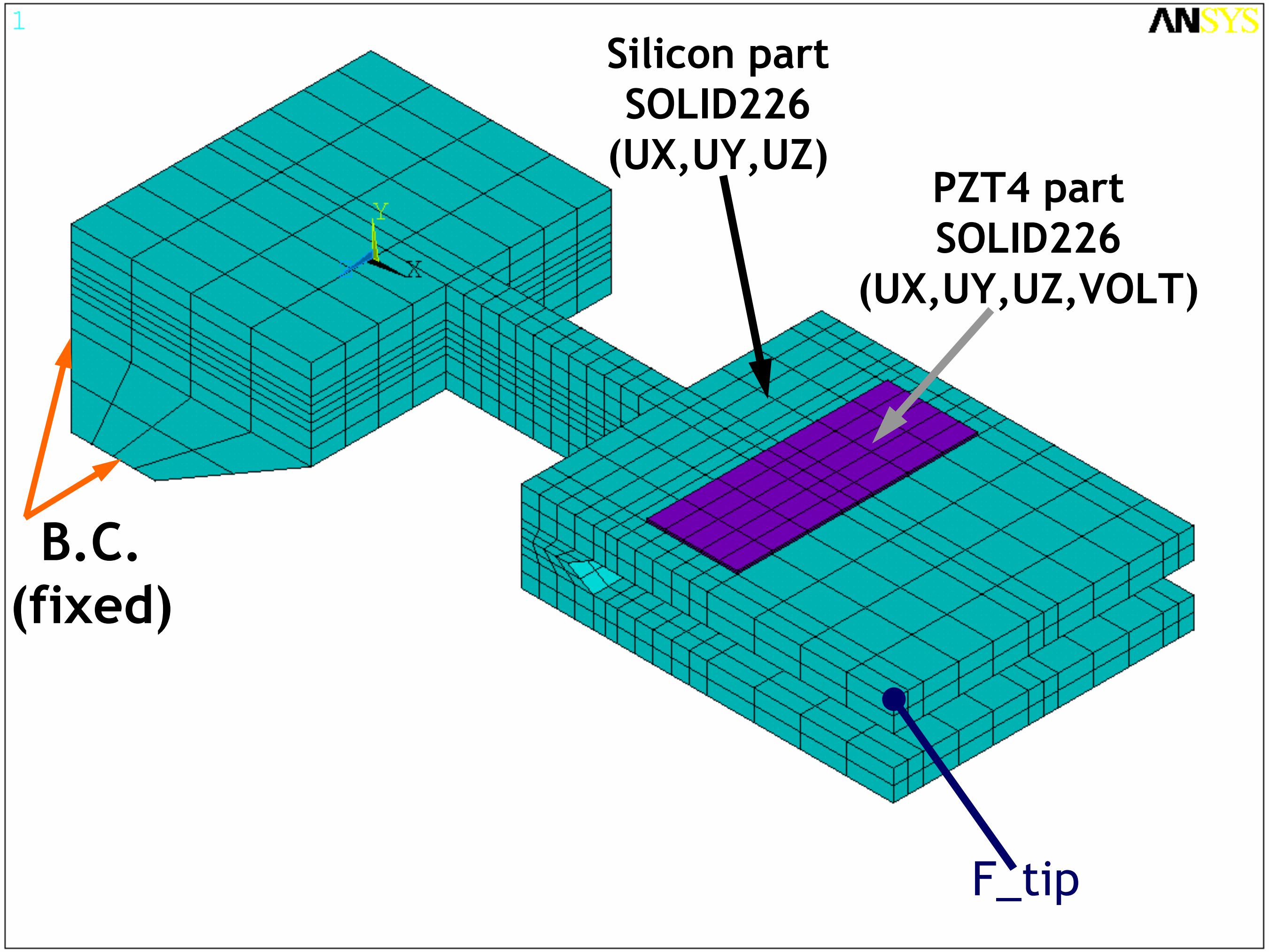


- **1,584 SOLID226 elements**
- **8,867 nodes (UX,UY,UZ,VOLT)**

μ-Gyroscope

- Finite element model

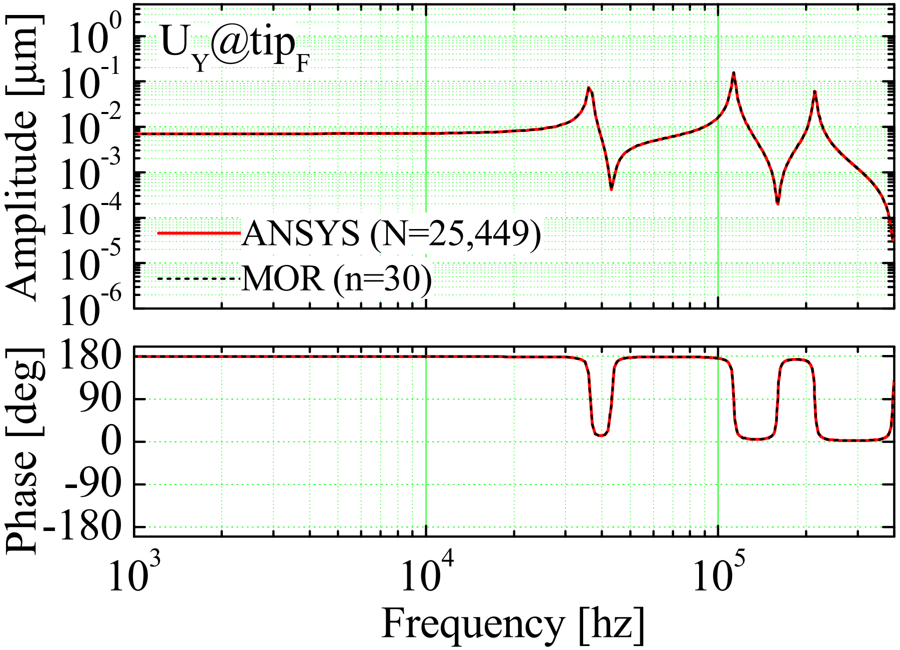
- No. of elements:
  1,584 (SOLID226)
- No. of nodes:
  8,867
- No. of DOFs:
  25,449

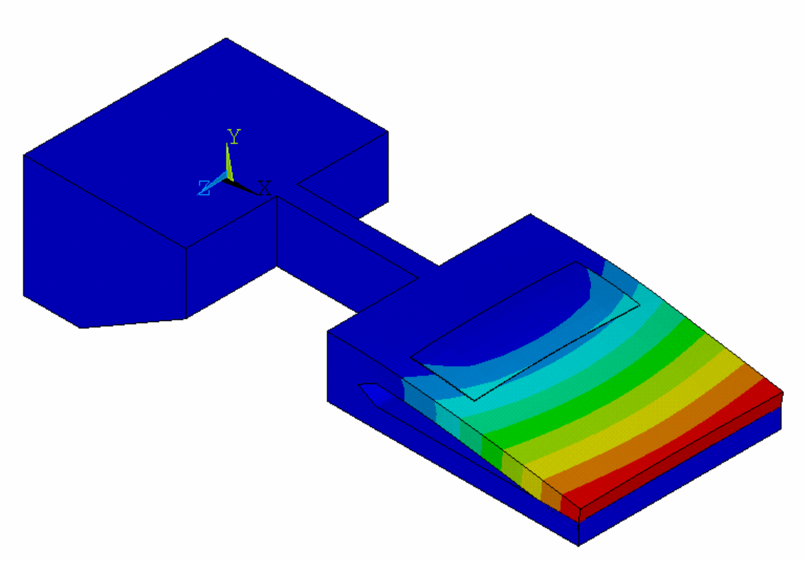


μ-Gyroscope

- Frequency response (Ω_X=0)
  - U_Y@tip_F

vertical excitation





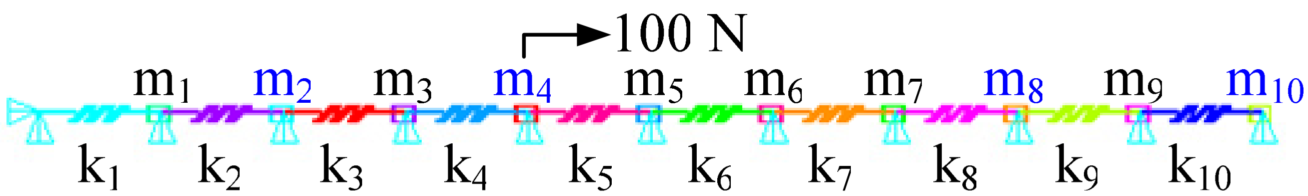
- No. of DOFs: 25,449

Part II

- Tutorials
  - Tutorial 1 : 10 DoF mass-spring system without damping
  - Tutorial 2 : 10 DoF mass-spring system with proportional damping
  - Tutorial 3 : HDD actuator/suspension [Hatch, 2001]

Tutorial (1)

- 10 DoF mass-spring system (no damping)



  - m_i = 0.1 kg (mass21 elements in ANSYS)
  - k_i = 50 kN/m (combin14 elements in ANSYS)

- Calculate frequency response functions using Krylov-based model order reduction
- Compare the results with those by an ANSYS original finite element model

Tutorial (1)

- System matrices (from ANSYS)

$$\mathbf{M}\ddot{\mathbf{x}}(t)+\mathbf{K}\mathbf{x}(t)=\mathbf{b}u(t)$$
$$\mathbf{y}(t)=\mathbf{C}\mathbf{x}(t)$$

$$\mathbf{M}=\begin{pmatrix} 0.1&0&0&0&0&0&0&0&0&0\\ 0&0.1&0&0&0&0&0&0&0&0\\ 0&0&0.1&0&0&0&0&0&0&0\\ 0&0&0&0.1&0&0&0&0&0&0\\ 0&0&0&0&0.1&0&0&0&0&0\\ 0&0&0&0&0&0.1&0&0&0&0\\ 0&0&0&0&0&0&0.1&0&0&0\\ 0&0&0&0&0&0&0&0.1&0&0\\ 0&0&0&0&0&0&0&0&0.1&0\\ 0&0&0&0&0&0&0&0&0&0.1 \end{pmatrix}$$

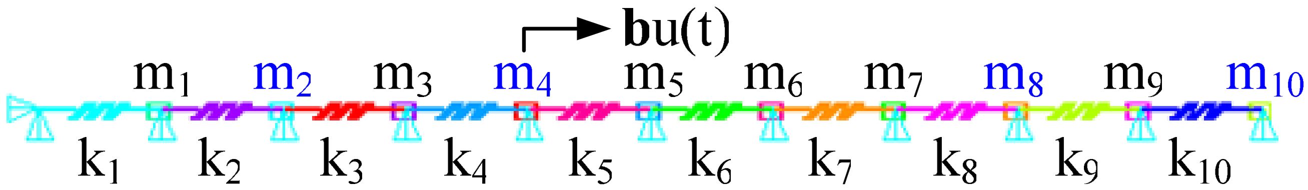
(10×10)

$$\mathbf{K}=\begin{pmatrix} 100000.&-50000.&0&0&0&0&0&0&0&0\\ -50000.&100000.&-50000.&0&0&0&0&0&0&0\\ 0&-50000.&100000.&-50000.&0&0&0&0&0&0\\ 0&0&-50000.&100000.&-50000.&0&0&0&0&0\\ 0&0&0&-50000.&100000.&-50000.&0&0&0&0\\ 0&0&0&0&-50000.&100000.&-50000.&0&0&0\\ 0&0&0&0&0&-50000.&100000.&-50000.&0&0\\ 0&0&0&0&0&0&-50000.&100000.&-50000.&0\\ 0&0&0&0&0&0&0&-50000.&100000.&-50000.\\ 0&0&0&0&0&0&0&0&-50000.&50000. \end{pmatrix}$$

Tutorial (1)

- System matrices (from ANSYS)

$$\mathbf{M}\ddot{\mathbf{x}}(t) + \mathbf{K}\mathbf{x}(t) = \mathbf{b}u(t)$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t)$$



$$\mathbf{b}=\begin{pmatrix} 0. \\ 0. \\ 0. \\ 100. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \end{pmatrix}$$

(10×1)

$$\mathbf{C}=\begin{pmatrix} 0 & 1. & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1. & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1. & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1. \end{pmatrix}$$

(4×10)

$$\mathbf{x}(t)=\begin{Bmatrix} x_{m_1}(t) \\ x_{m_2}(t) \\ x_{m_3}(t) \\ x_{m_4}(t) \\ x_{m_5}(t) \\ x_{m_6}(t) \\ x_{m_7}(t) \\ x_{m_8}(t) \\ x_{m_9}(t) \\ x_{m_{10}}(t) \end{Bmatrix}$$

(10×1)

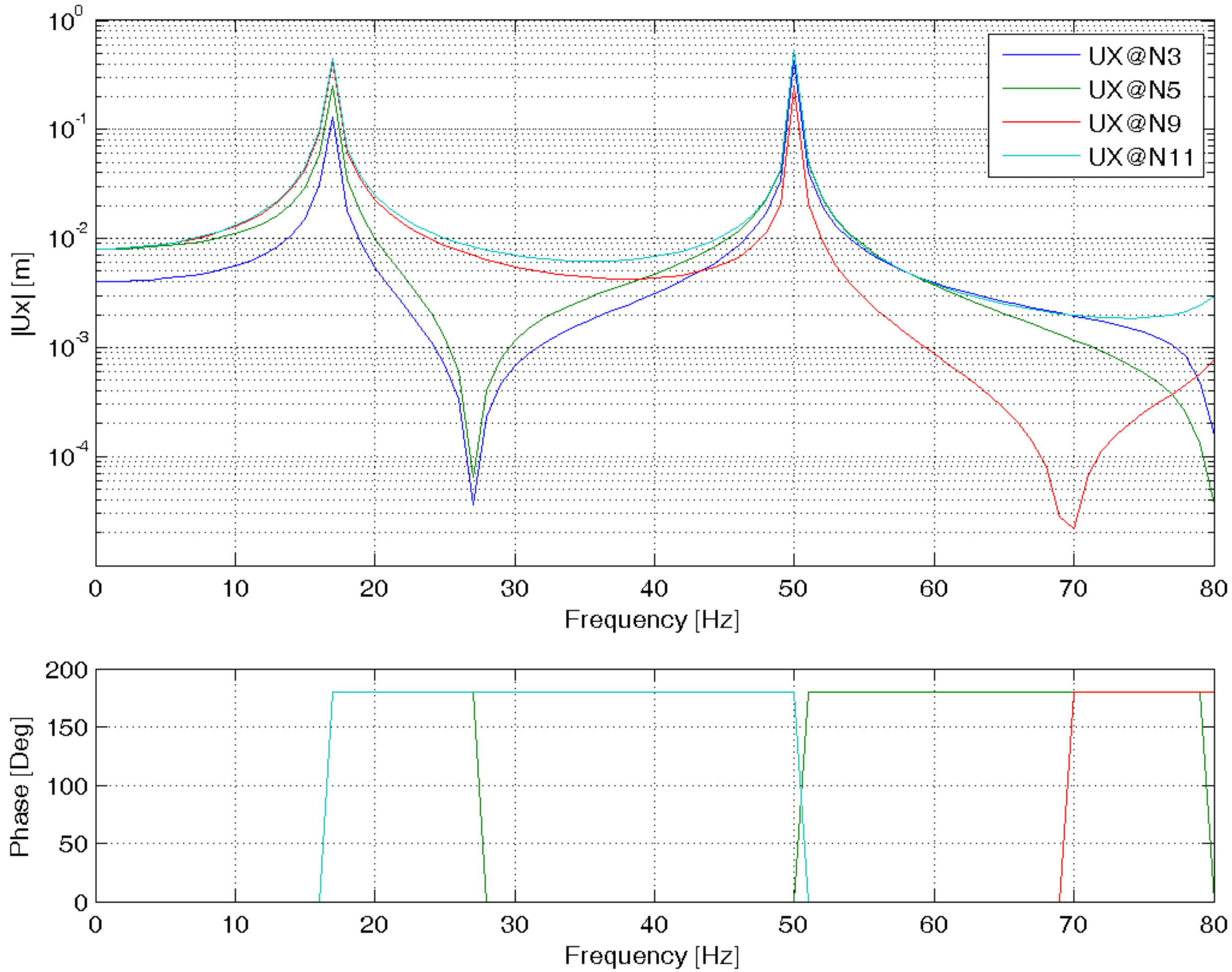
$$\mathbf{y}(t)=\begin{Bmatrix} x_{m_2}(t) \\ x_{m_4}(t) \\ x_{m_8}(t) \\ x_{m_{10}}(t) \end{Bmatrix}$$

@n9

(4×1)

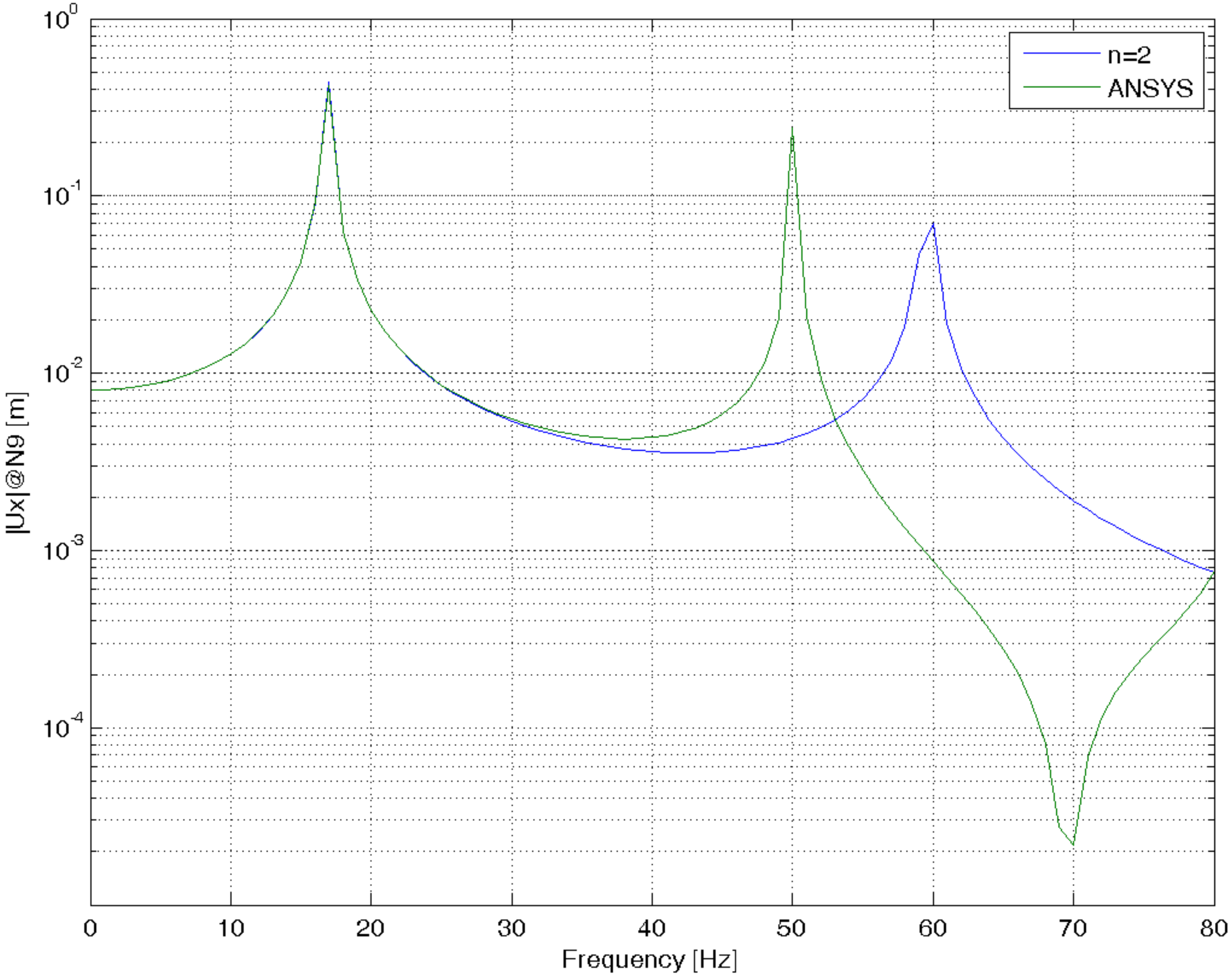
Tutorial (1)

- Frequency response function (ANSYS, N=10)
  - Range : 0~80 Hz (@81 frequencies)



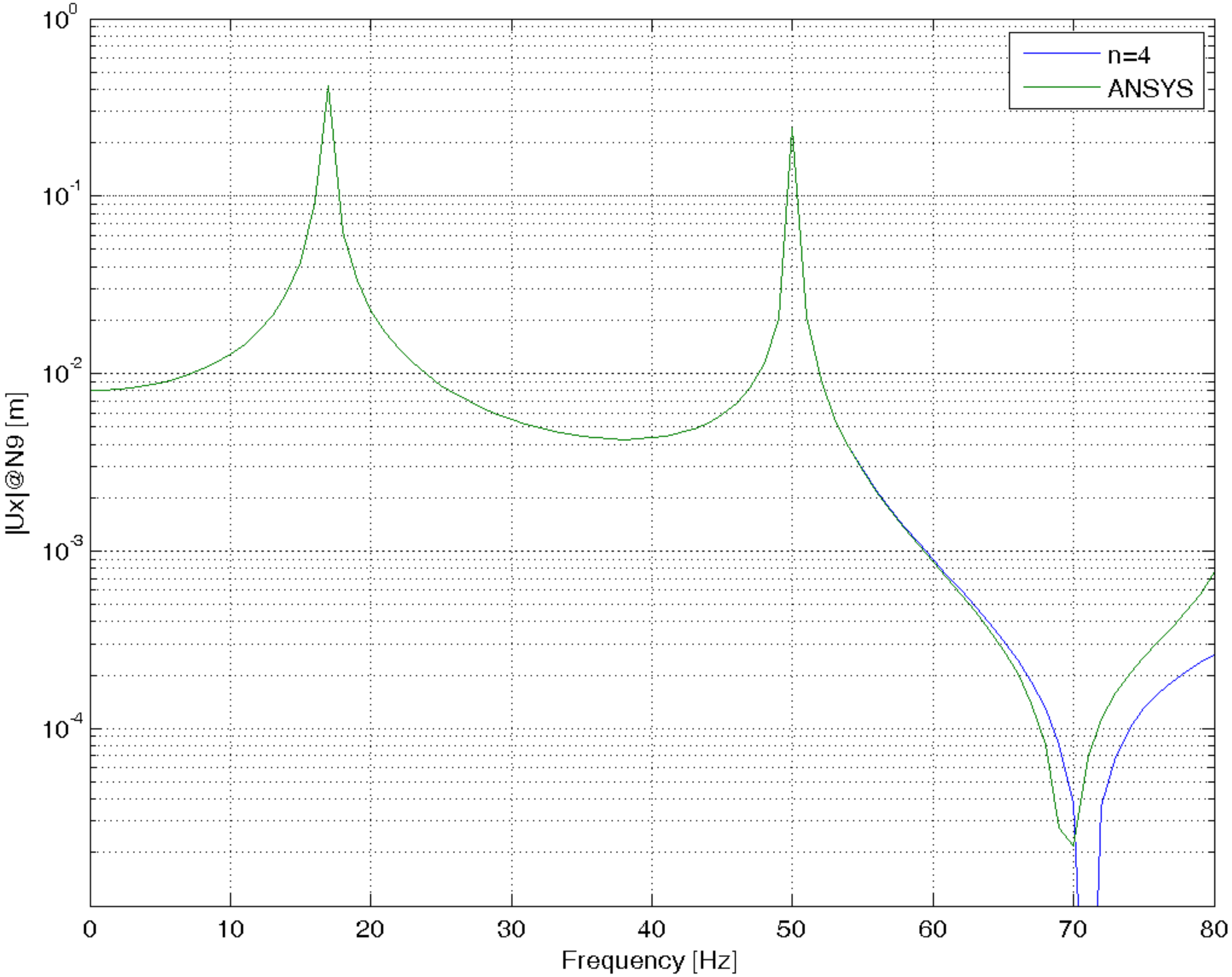
Tutorial (1)

- Frequency response @n9 (m_8) (n=2)



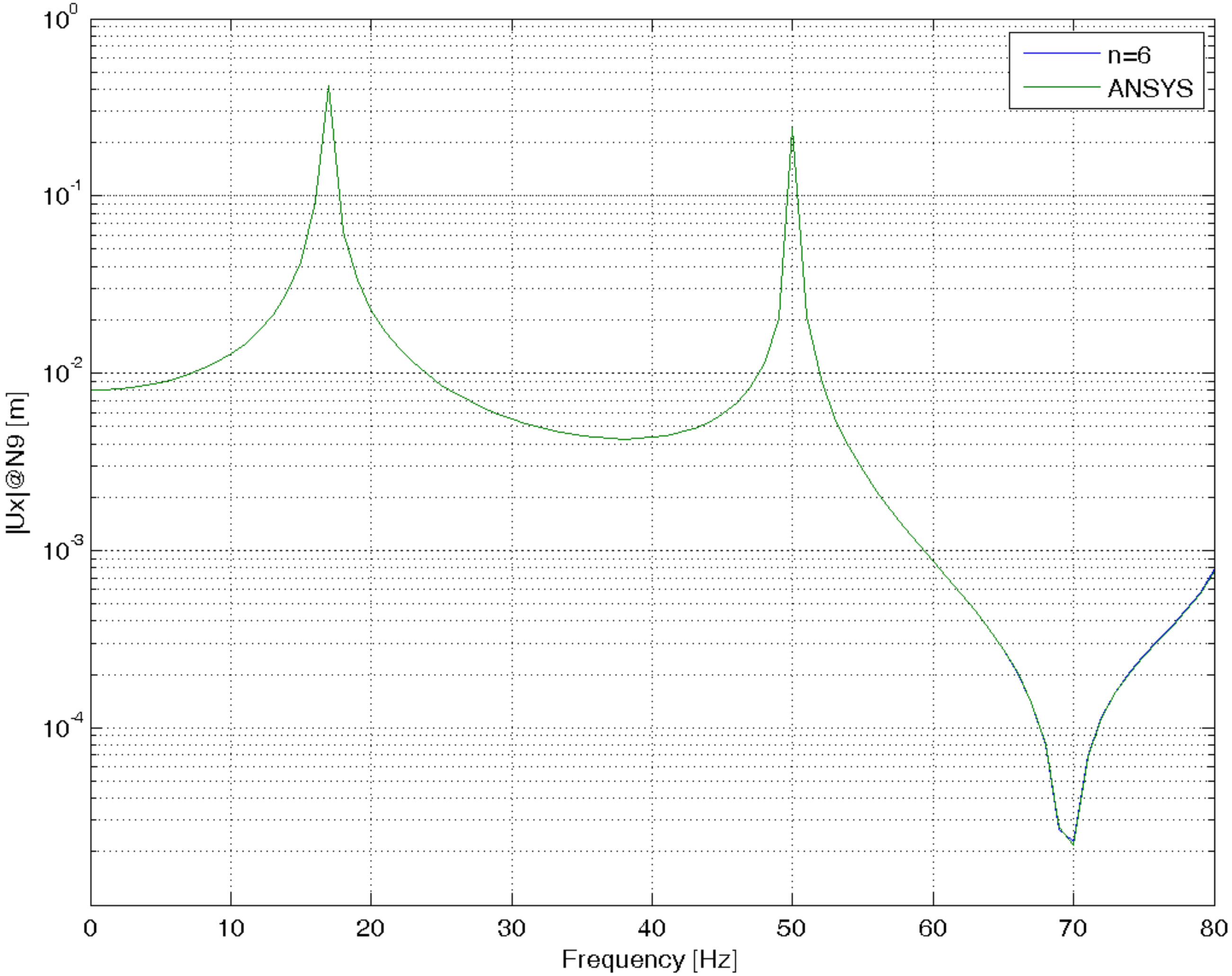
Tutorial (1)

- Frequency response @n9 (m_8) (n=4)



Tutorial (1)

- Frequency response @n9 (m_8) (n=6)



Tutorial (1) : MATLAB Code

```
%%% set a working directory --> change it according to your case
clear all; close all;
cd d:\tutorial_1

%%% read original system matrices from binary MAT-file
whos -file mk10.mat;
clear all;
load mk10.mat;

%%% load results by the full-size ansys model
mag_ansys = load('mag_ansys.txt');
phs_ansys = load('phs_ansys.txt');

figure(1);
subplot(3,1,1:2); semilogy(mag_ansys(:,1),mag_ansys(:,2:end));
axis([0 80 1E-5 1E0]); grid on;
legend(NAMES);
xlabel('Frequency [Hz]'); ylabel('|Ux|  [m]');
subplot(3,1,3); plot(phs_ansys(:,1),phs_ansys(:,2:end));
axis([0 80 0 200]); grid on;
xlabel('Frequency [Hz]'); ylabel('Phase [Deg]');
saveas(gcf,'frf_ansys.png','png');
```

Read the system matrices!

Read the FRF results by ANSYS

Plot the FRF by the original ANSYS model

Tutorial (1) : MATLAB Code

```
%%% perform model order reduction by arnoldi algorithm
n = 6;          % change the order of reduced model (n<N)

s0 = -(2*pi*0.)^2; % f=0 hz
KK = K + s0*M;

[L, U] = lu(KK);   % LU matrix factorization (KK = L*U)

v = U\(L\B);       % the starting vector by left division
v = (1/norm(full(v)))*v;     % normalizing the starting vector

% generate krylov vectors up to n
for j = 2:n
    v(:,j) = U\(L\(M*v(:,j-1)));
    for k = 1:j-1
         hv = v(:,k)'*v(:,j);
         v(:,j) = v(:,j) - hv*v(:,k);
    end
    v(:,j) = v(:,j)/norm(v(:,j));
end

diff_v = norm(v'*v - eye(n));     % check orthonormality
```

Arnoldi process through the modified Gram-Schmidt algorithm

Tutorial (1) : MATLAB Code

```
%%% generate reduced system matrices by projection
Mr = full(v'*M*v);
Kr = full(v'*K*v);
Br = full(v'*B);
Cr = full(C*v);
NAMESr = NAMES;
% damping with a proportional damping
alpha = 0.; beta = 0.;  % no damping in tutorial_1
Er = alpha*Mr + beta*Kr;
```

Construct a reduced system using the generated orthonomal matrix V through projection

```
%%% perform frequency responses with the reduced system
nstep = (80+1); fstart = 0.; fend = 80.;
fdel = (fend - fstart)/(nstep - 1);

for k = 1:nstep
    kfrq = fstart + (k-1)*fdel;
    komg = 2*pi*kfrq;

    Kc = Kr - (komg^2)*Mr + i*komg*Er;

    kXc = Kc\Br;
    kYc = Cr*kXc;

    Yc(k,:) = [kfrq, kYc'];
end
```

Perform frequency response analyses at each freqeuency 'kfrq' using the reduced system of order 'n'; 0~80 hz

Tutorial (1) : MATLAB Code

```
% plot the harmonic responses from the reduced system
FRQ = Yc(:,1);
MAG = abs(Yc(:,2:end));
PHS = (180./pi)*angle(Yc(:,2:end));
```

Save the results in a matrix-format

```
% UXs
figure(2);
semilogy(FRQ, MAG); grid on;
legend(NAMESr);
xlabel('Frequency [Hz]'); ylabel('|Ux| [m]');
saveas(gcf,'mag_n.png','png');
```

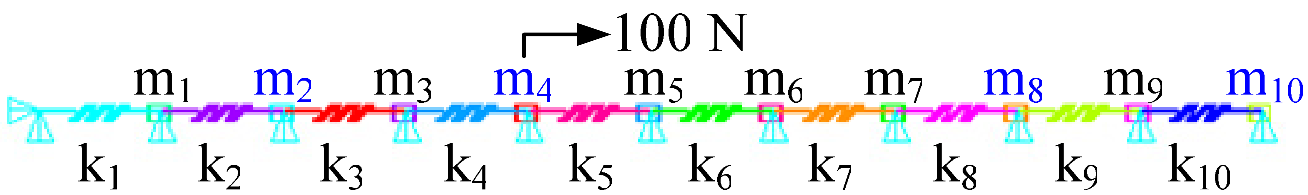
Plot the results from the reduced models

```
% UXs@N9(m8)
ux = [MAG(:,end-1), mag_ansys(:,end-1)];
figure(3);
semilogy(FRQ, ux); grid on;
axis([0 80 1E-5 1E0]);
legend(['n=',num2str(n)], 'ANSYS');
xlabel('Frequency [Hz]'); ylabel('|Ux|@N9 [m]');
saveas(gcf,'ux@n9_n.png','png');
```

Compare 'ux@n9' between the reduced and ANSYS results and plot them

Tutorial (2)

- 10 DoF mass-spring system (proportional damping)



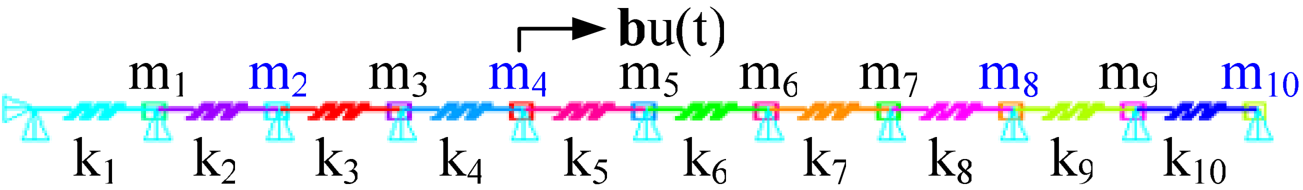
  - m_i = 0.1 kg (mass21 elements in ANSYS)
  - k_i = 50 kN/m (combin14 elements in ANSYS)
  - $E = \alpha M + \beta K$ (α=2.5 ms^{-1}, β=0.25 ms)

- Calculate frequency response functions using Krylov-based model order reduction
- Compare the results with those by an ANSYS original finite element model

Tutorial (2)

- **System matrices (from ANSYS)**

$$\mathbf{M}\ddot{\mathbf{x}}(t)+\mathbf{E}\dot{\mathbf{x}}(t)+\mathbf{K}\mathbf{x}(t)=\mathbf{b}u(t)$$
$$\mathbf{y}(t)=\mathbf{C}\mathbf{x}(t)$$

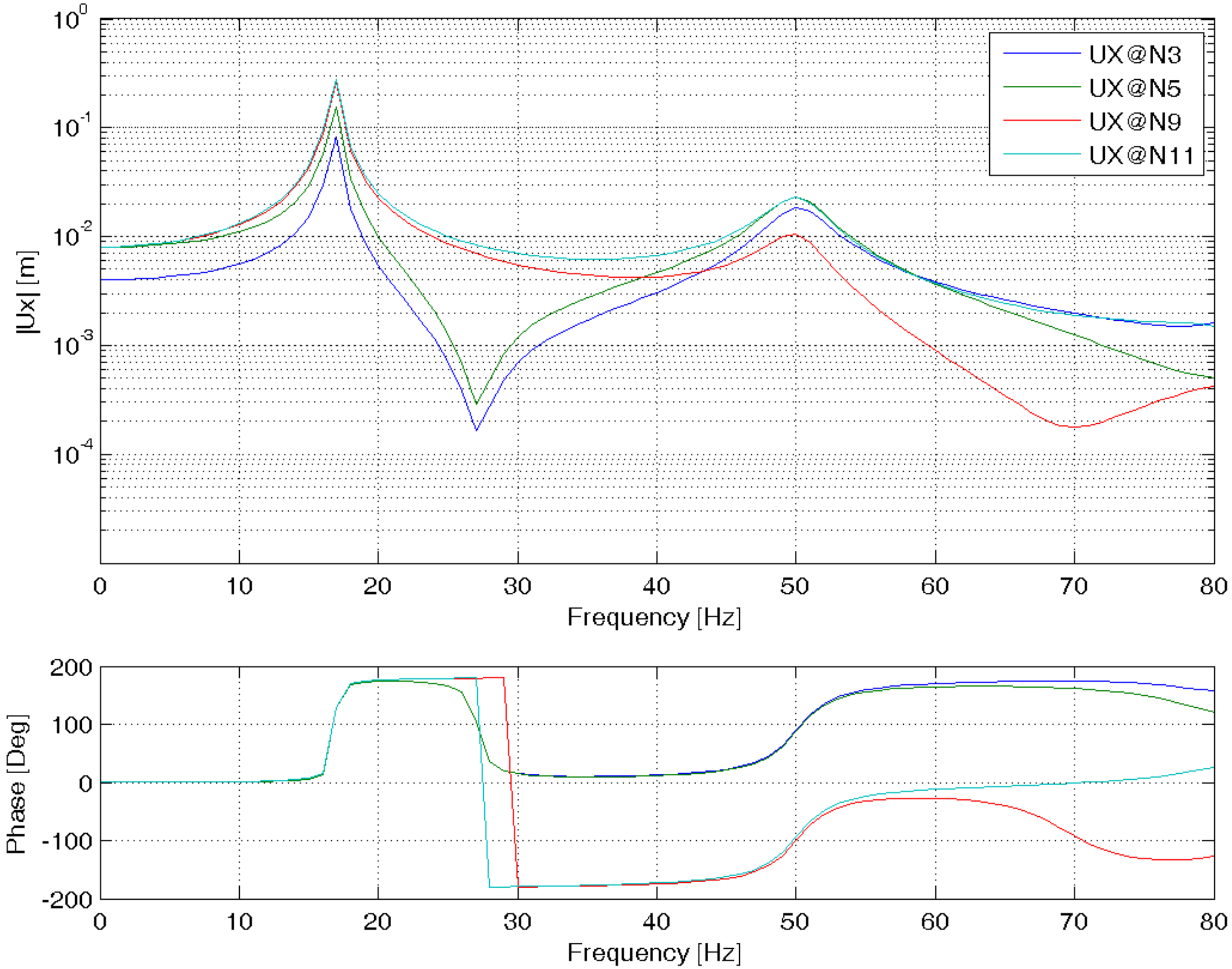


M, K, b, C, E=αM+βK

$$\mathbf{E}=\begin{pmatrix}
25.0003 & -12.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
-12.5 & 25.0003 & -12.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & -12.5 & 25.0003 & -12.5 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & -12.5 & 25.0003 & -12.5 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & -12.5 & 25.0003 & -12.5 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & -12.5 & 25.0003 & -12.5 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & -12.5 & 25.0003 & -12.5 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & -12.5 & 25.0003 & -12.5 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -12.5 & 25.0003 & -12.5\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -12.5 & 12.5003
\end{pmatrix}$$

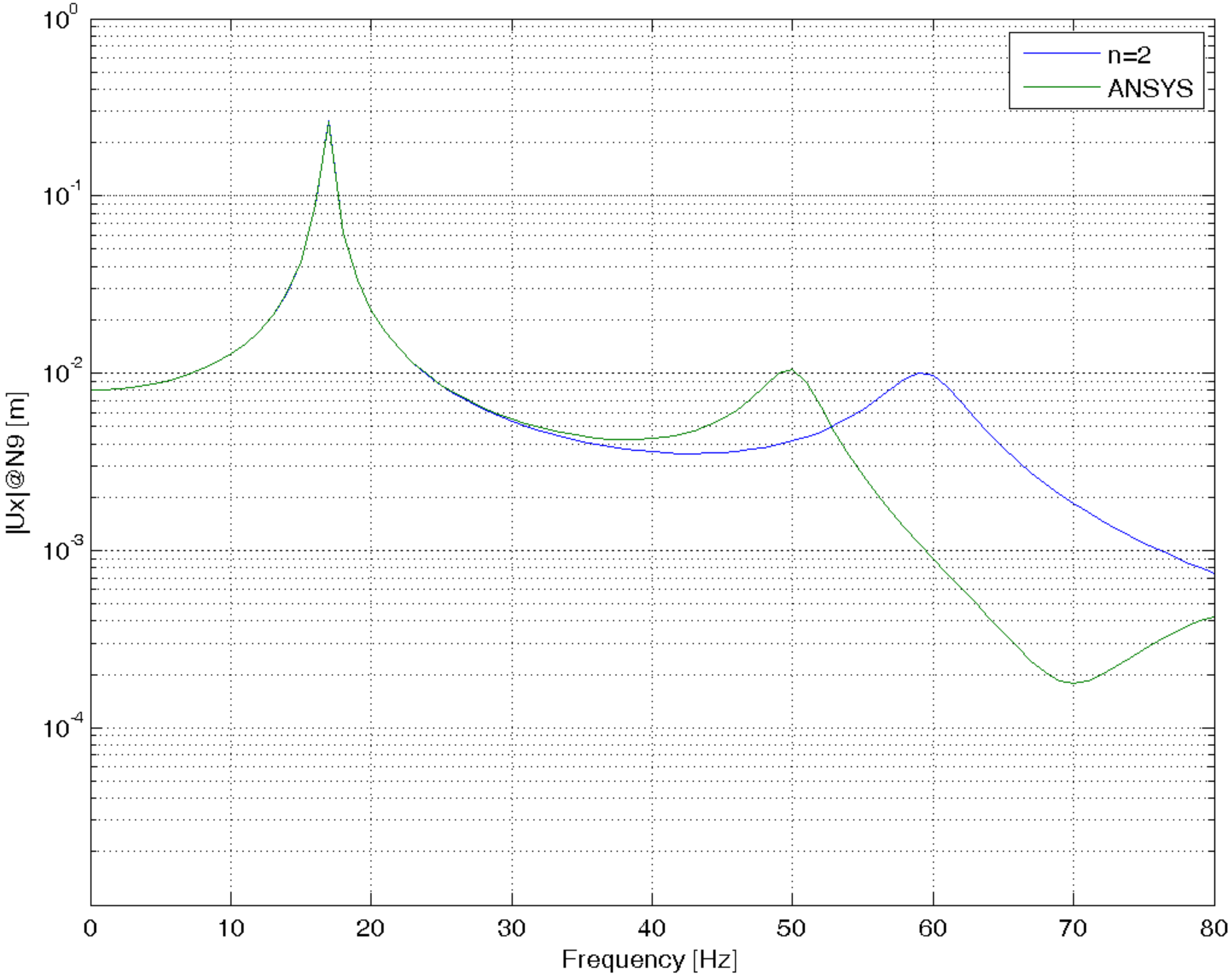
Tutorial (2)

- **Frequency response function (ANSYS, N=10)**
  - **Range : 0~80 Hz (@81 frequencies)**



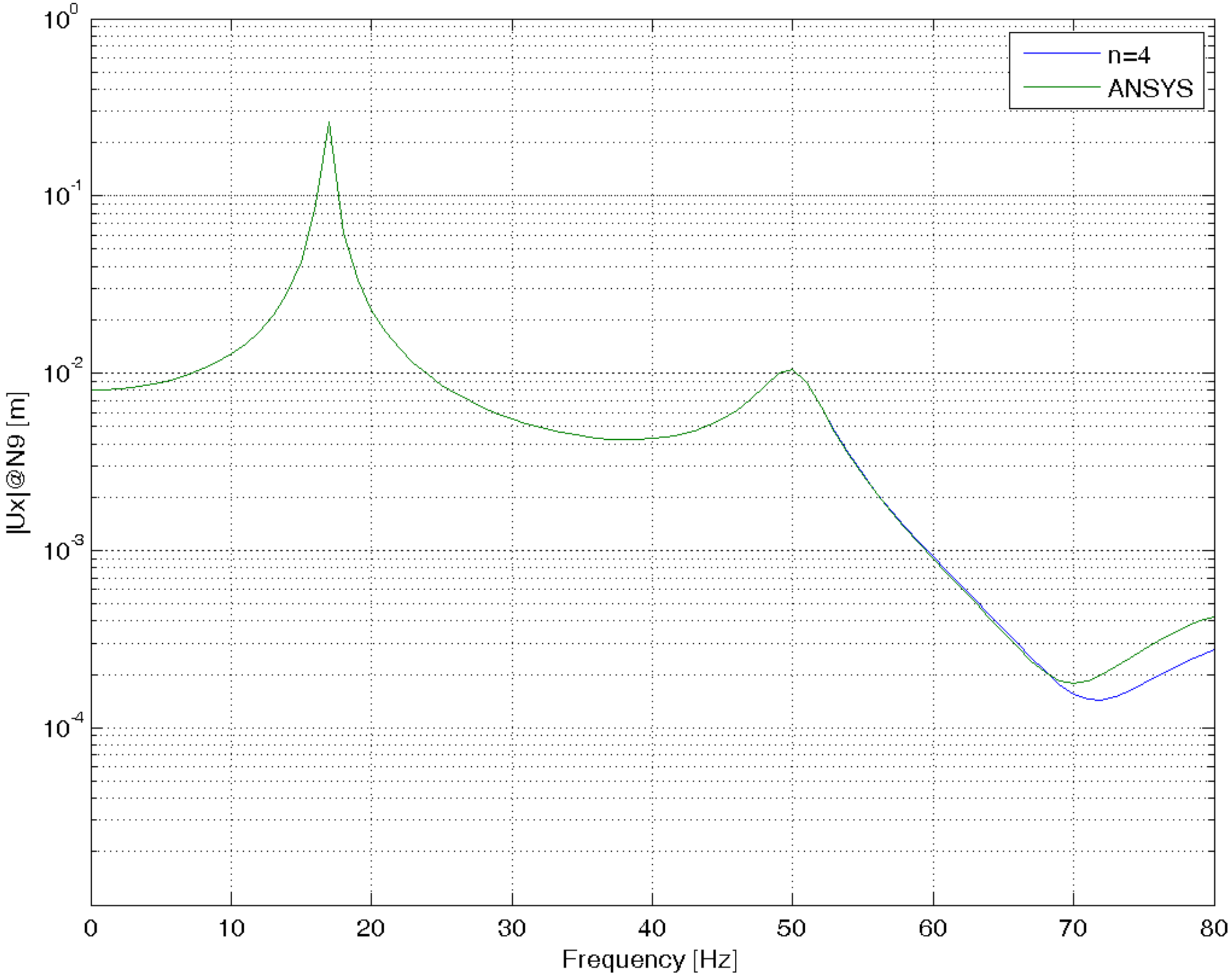
Tutorial (2)

- **Frequency response @n9 (m_8) (n=2)**



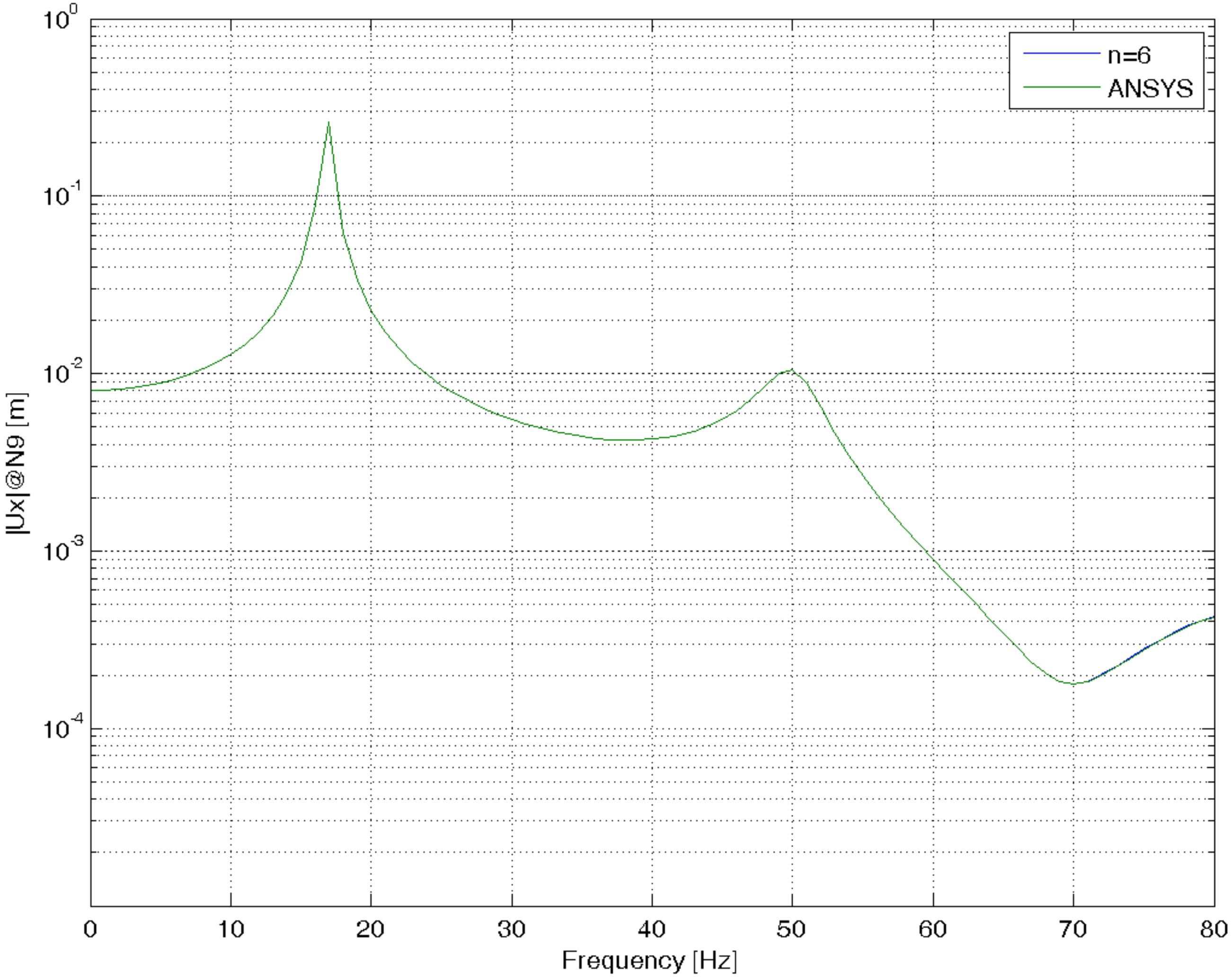
Tutorial (2)

- Frequency response @n9 (m_8) (n=4)



Tutorial (2)

- **Frequency response @n9 (m_8) (n=6)**



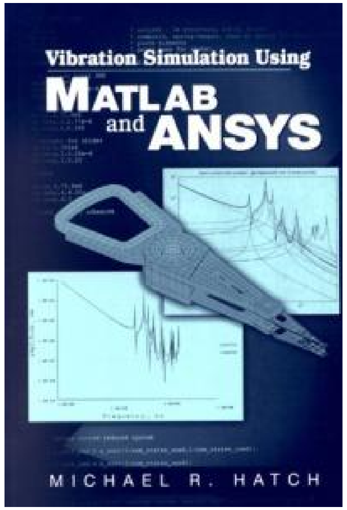
Tutorial (2) : MATLAB Code

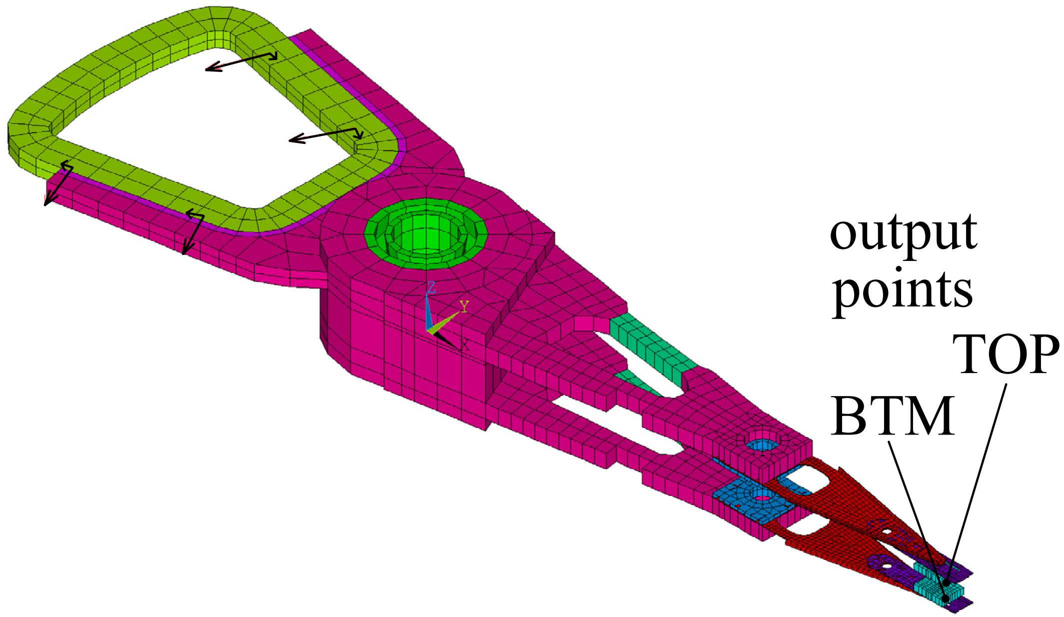
- Use the MATLAB code for tutorial 1
- But in order to include a damping effect, change the values α and β for the proportional damping coefficients in the code as follows;

```
% damping with a proportional damping
alpha = 2.5E-3; beta = 2.5E-4;
Er = alpha*Mr + beta*Kr;
```

Tutorial (3)

- HDD Actuator/Suspension [Hatch, 2001]



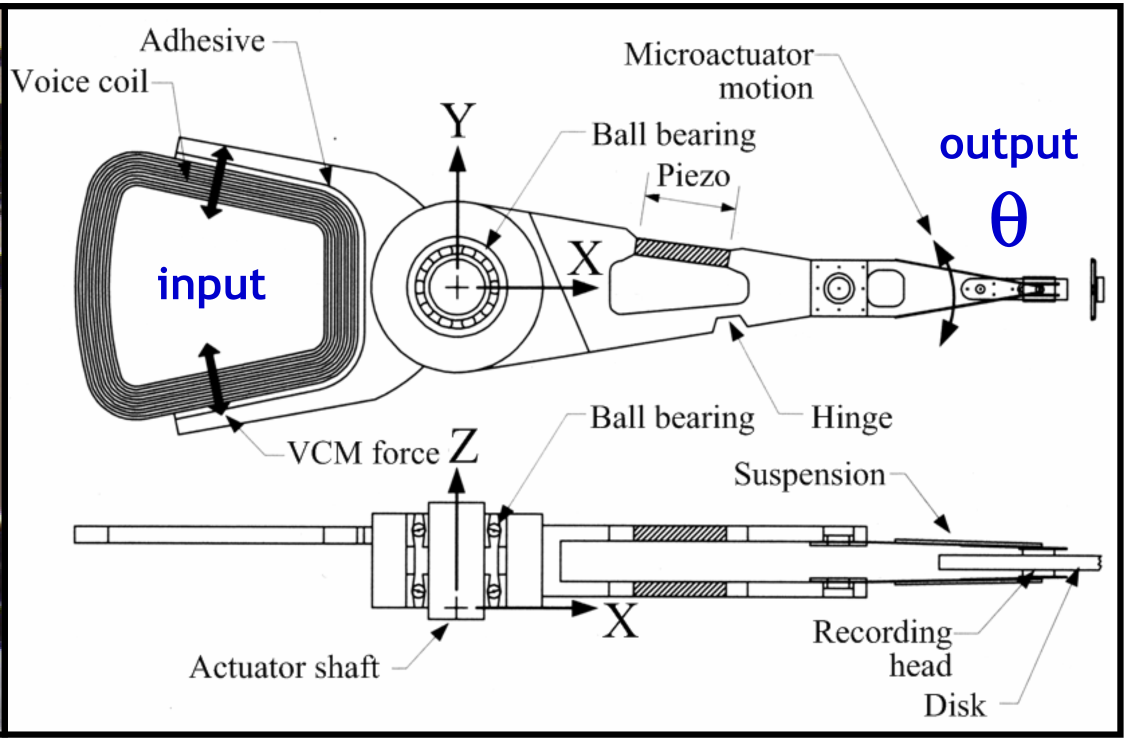


No. of nodes : 7,336
No. of DoFs : 21,203
No. of elements : 3,338 (SOLID45) + 8 (COMBIN14)

- Calculate frequency response functions using Krylov-based model order reduction
- Compare the results with those by the ANSYS full-size finite element model

HDD Actuator/Suspension System

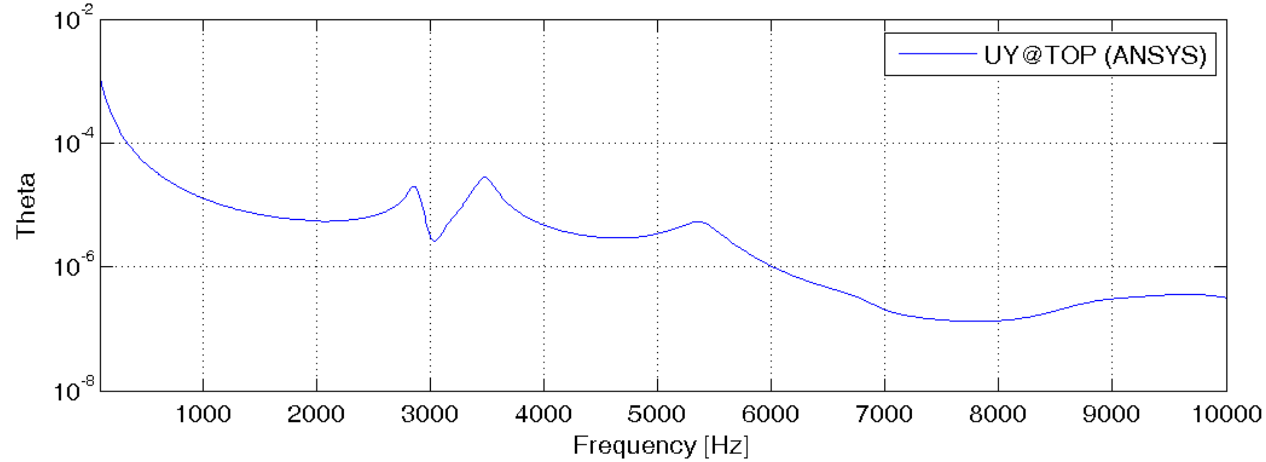
- Overview

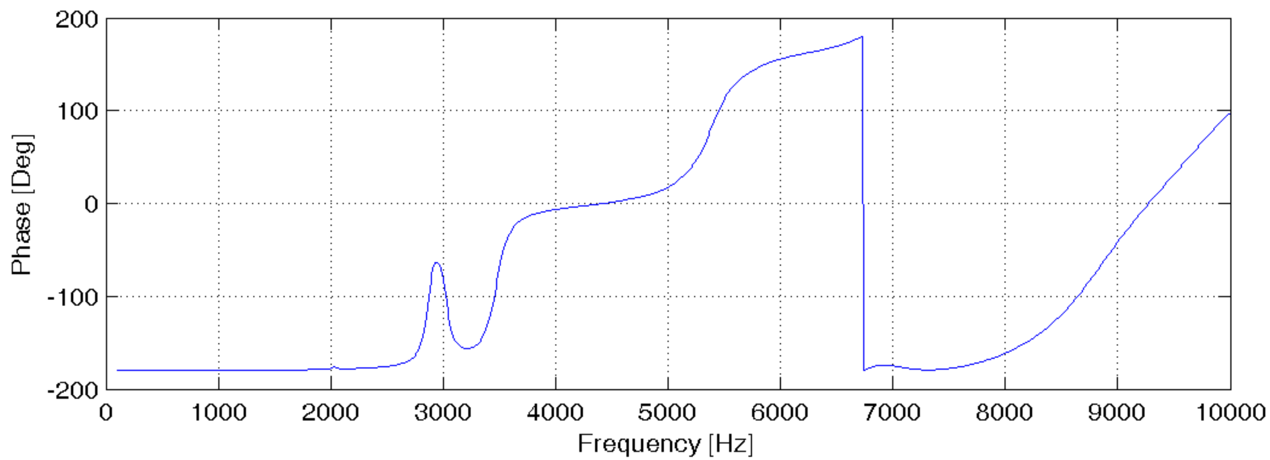


Hatch (2001)

Tutorial (3)

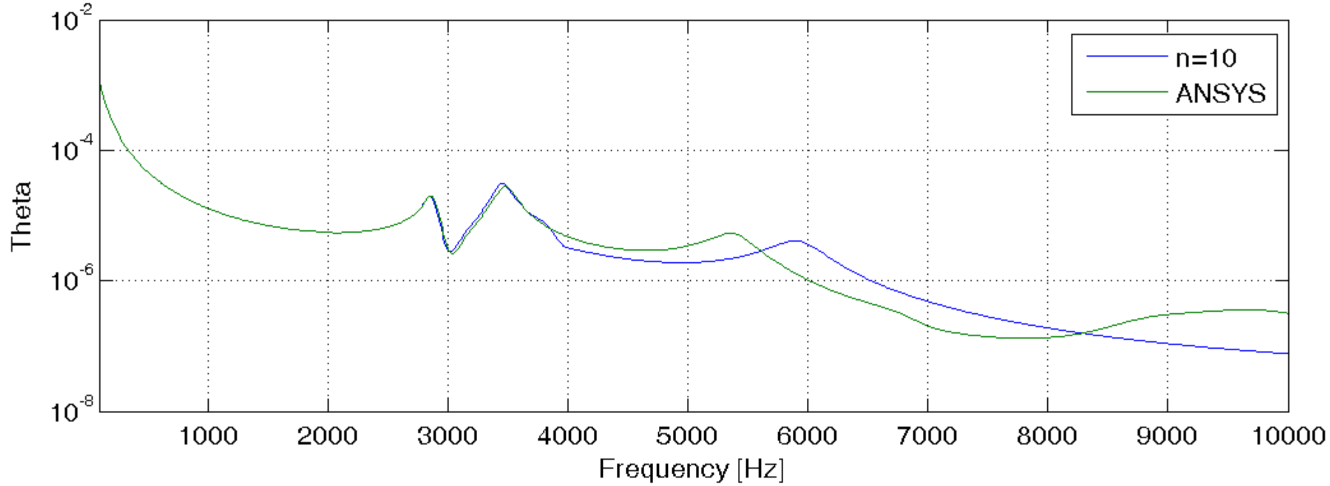
- **Frequency response function (ANSYS, N=21,203)**
  - **Range : 100~10,000 Hz (@991 frequencies)**
  - **Proportional damping : α=2 μs^{-1}, β=2 μs**

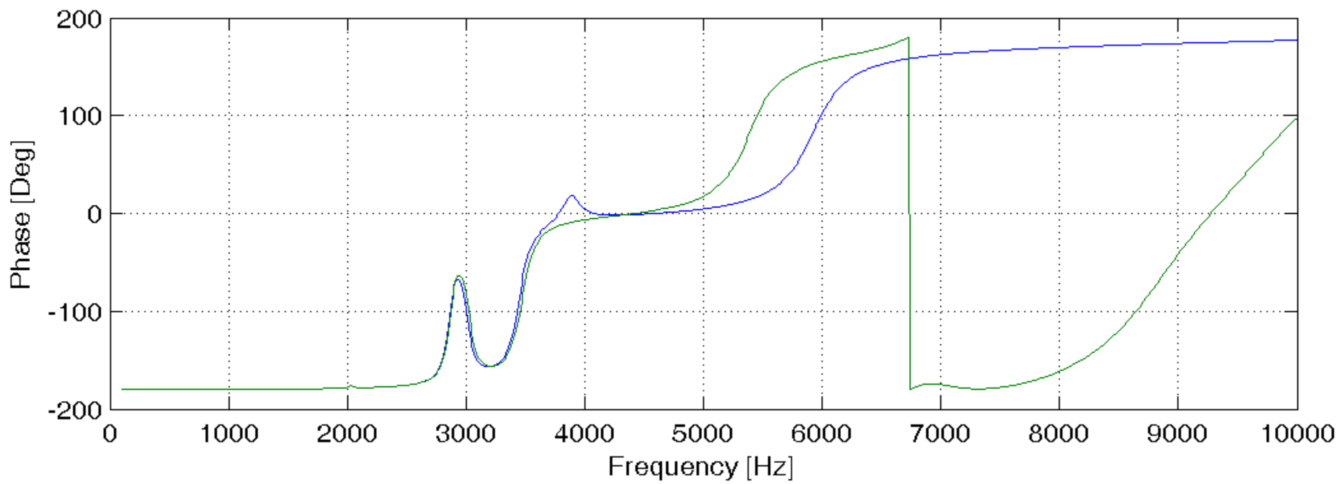




Tutorial (3)

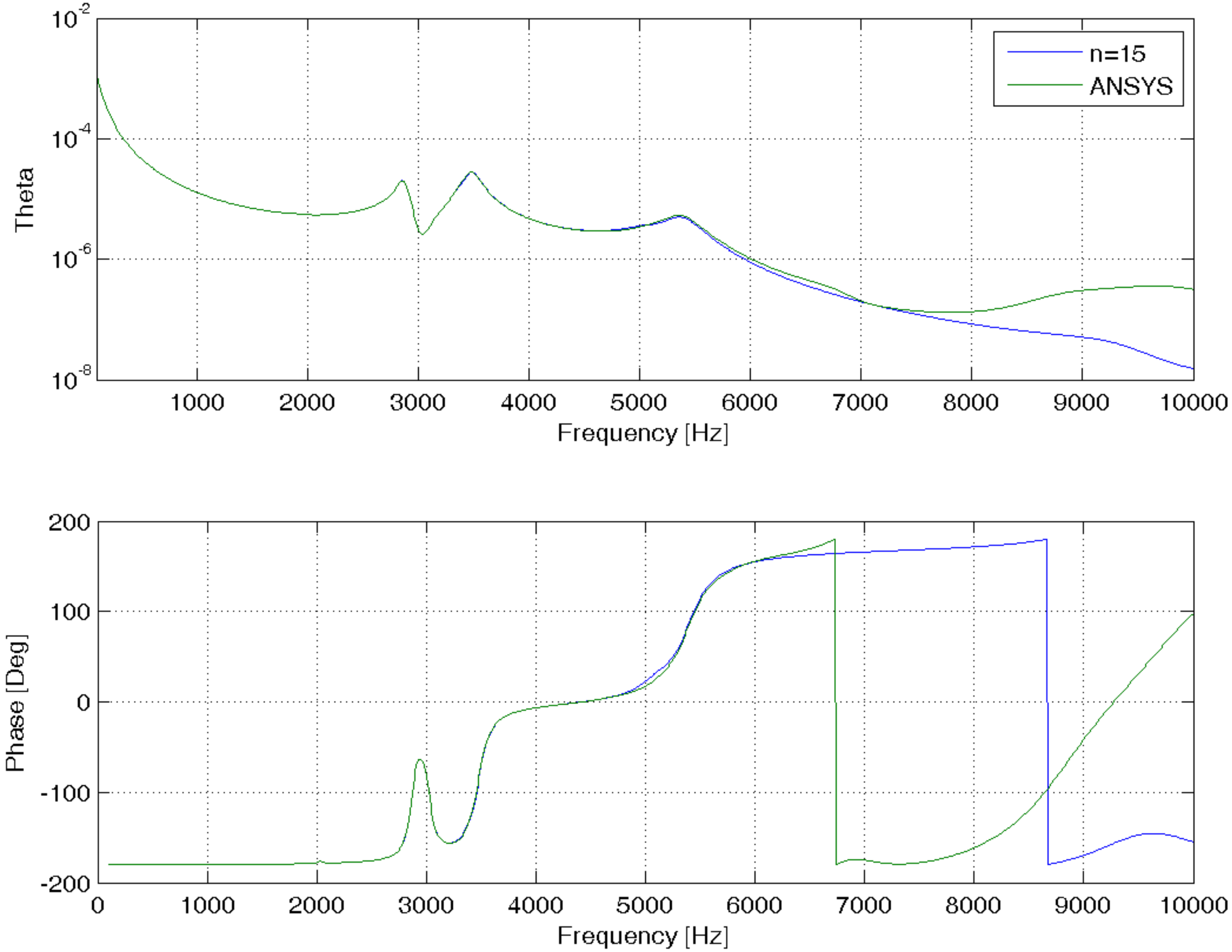
- **Frequency response θ@TOP (n=10)**





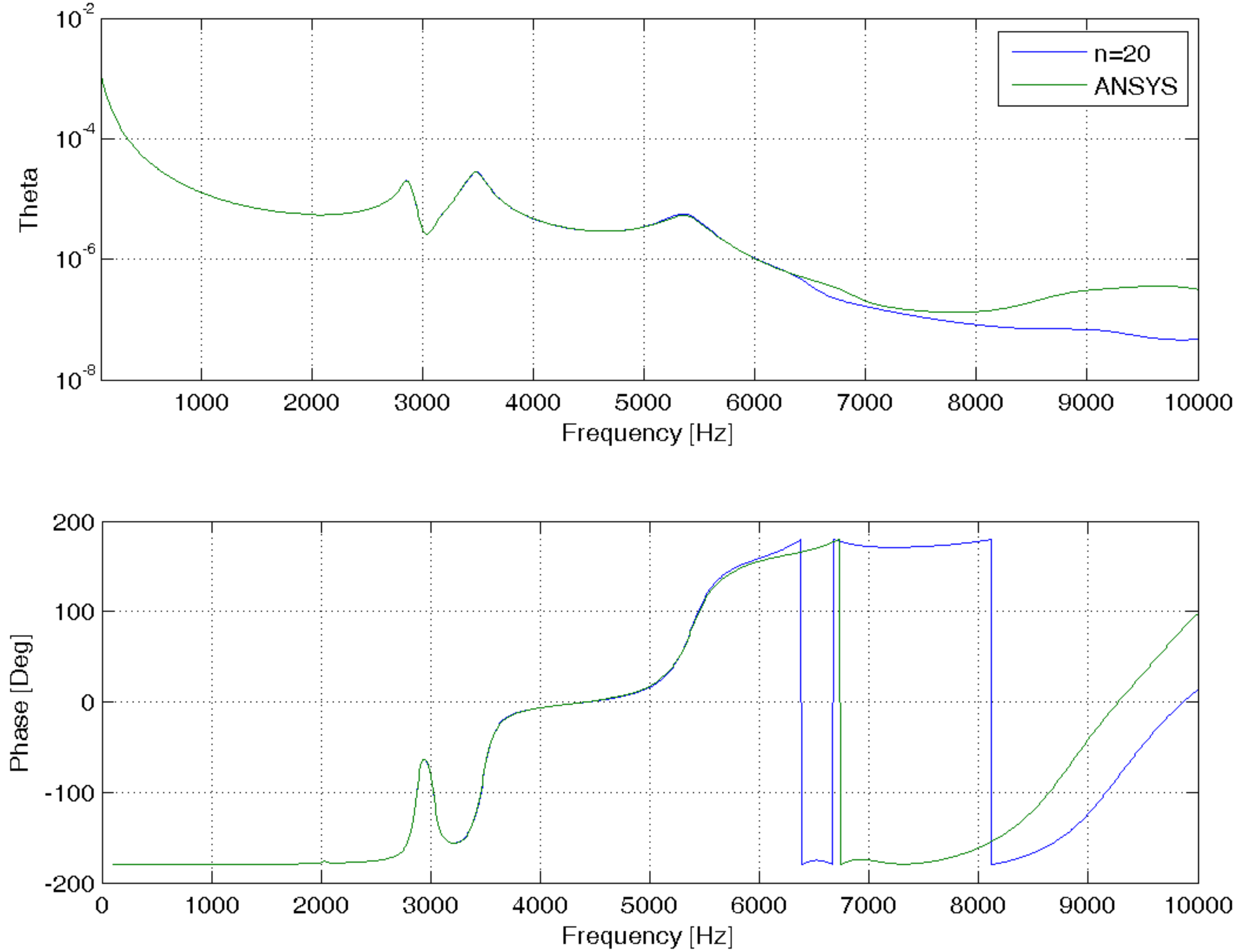
Tutorial (3)

- **Frequency response θ@TOP (n=15)**



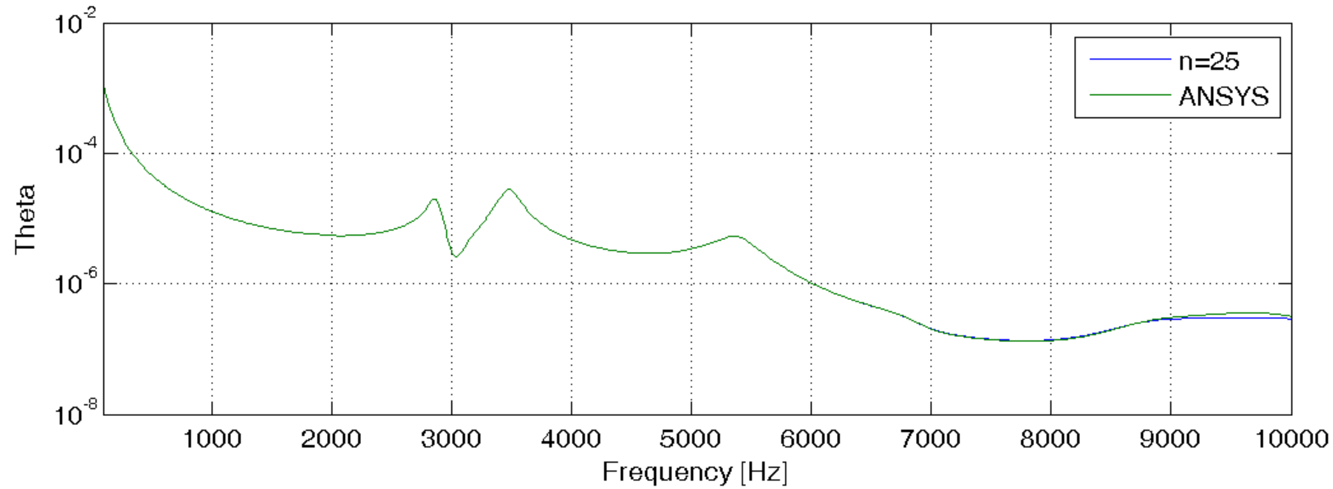
Tutorial (3)

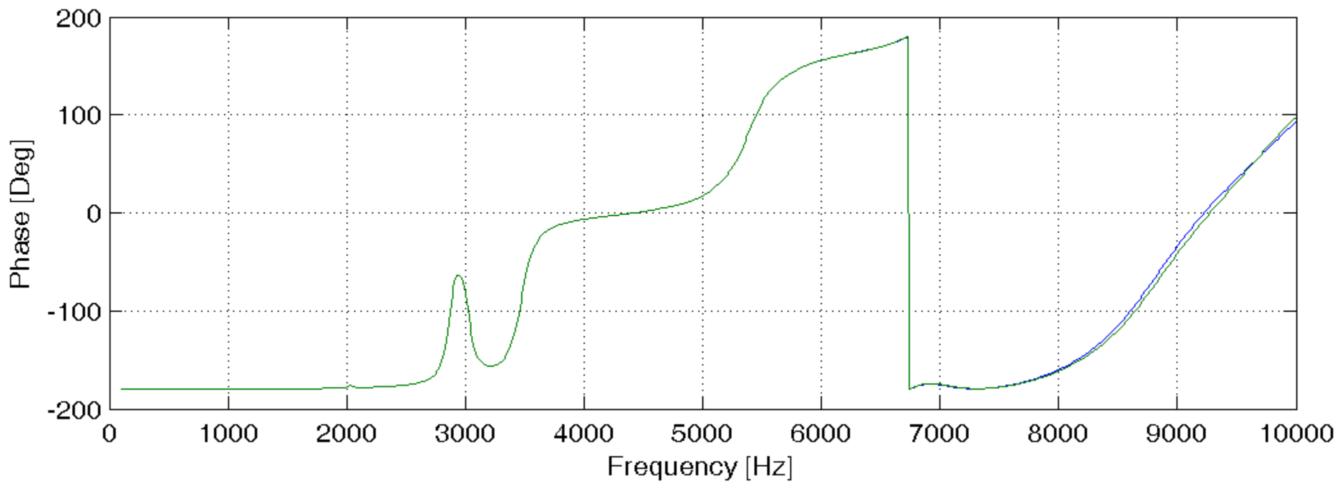
- **Frequency response θ@TOP (n=20)**



Tutorial (3)

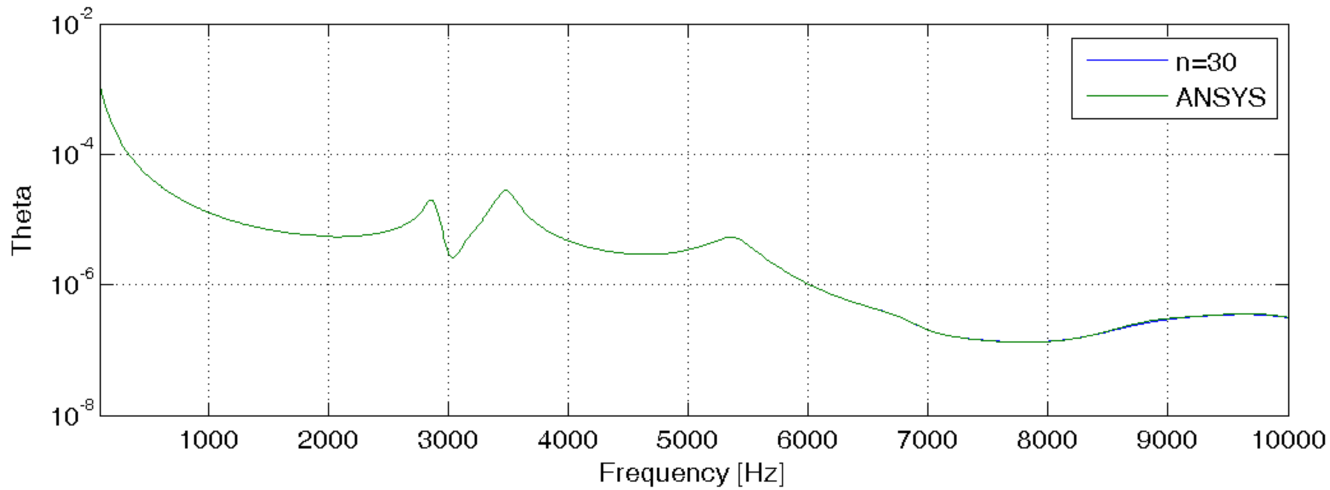
- **Frequency response θ@TOP (n=25)**

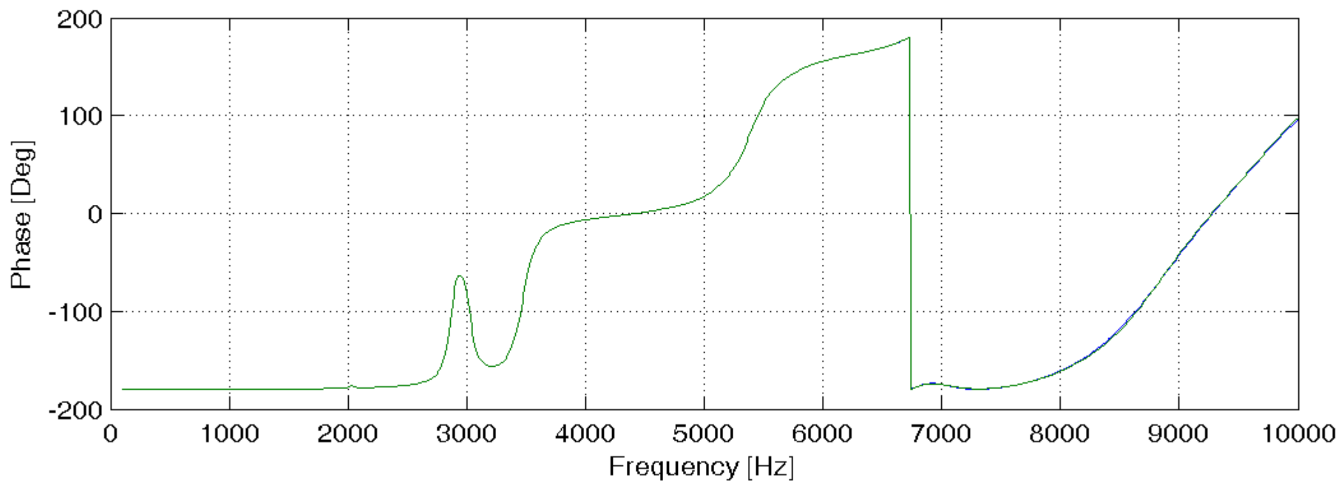




Tutorial (3)

- **Frequency response θ@TOP (n=30)**





Tutorial (3) : MATLAB Code

```
%%% set a working directory --> change it according to your case
clear all; close all;
cd d:\tutorial_3

%%% read original system matrices from binary MAT-file
whos -file simo.mat;
clear all;
load simo.mat;

%%% load results by the full-size ansys model
uy_top_ansys = load('uy_top_ansys.txt');

figure(1);
subplot(2,1,1);
semilogy(uy_top_ansys(:,1), uy_top_ansys(:,2)); grid on;
legend('UY@TOP (ANSYS)');
xlabel('Frequency [Hz]'); ylabel('Theta');
axis([100 10000 1E-8 1E-2]);
subplot(2,1,2);
plot(uy_top_ansys(:,1), uy_top_ansys(:,3)); grid on;
xlabel('Frequency [Hz]'); ylabel('Phase [Deg]');
saveas(gcf,'uy@top_ansys.png','png');
```

Read the system matrices!

Read the FRF results by ANSYS

Plot the FRF by the original ANSYS model

Tutorial (3) : MATLAB Code

```
%%% perform model order reduction by arnoldi algorithm
n = 10;        % change the order of reduced model (n<N)

s0 = -(2*pi*100)^2;      % f=100 hz
KK = K + s0*M;

[L, U] = lu(KK);   % LU matrix factorization (KK = L*U)

v = U\(L\B);         % the starting vector by left division
v = (1/norm(full(v)))*v;      % normalizing the starting vector

% generate krylov vectors up to n
for j = 2:n
    v(:,j) = U\(L\(M*v(:,j-1)));
    for k = 1:j-1
         hv = v(:,k)'*v(:,j);
         v(:,j) = v(:,j) - hv*v(:,k);
    end
    v(:,j) = v(:,j)/norm(v(:,j));
end

diff_v = norm(v'*v - eye(n));     % check orthonormality
```

Use a non-zero expansion point (f=100 hz) because of a rigid-body motion

Arnoldi process through the modified Gram-Schmidt algorithm

Tutorial (3) : MATLAB Code

```
%%% generate reduced system matrices by projection
Mr = full(v'*M*v);
Kr = full(v'*K*v);
Br = full(v'*B);
Cr = full(C*v);
NAMESr = NAMES;
% damping with a proportional damping
alpha = 2E-6; beta = 2E-6;
Er = alpha*Mr + beta*Kr;
```

Construct a reduced system using the generated orthonomal matrix V through projection.
Damping matrix E_r is also constructed!

```
%%% perform frequency responses with the reduced system
nstep = (990+1); fstart = 100; fend = 10000;
fdel = (fend - fstart)/(nstep - 1);

for k = 1:nstep
    kfrq = fstart + (k-1)*fdel;
    komg = 2*pi*kfrq;

    Kc = Kr - (komg^2)*Mr + i*komg*Er;

    kXc = Kc\Br;
    kYc = Cr*kXc;

    Yc(k,:) = [kfrq, kYc'];
end
```

Perform frequency response analyses at each freqeuency 'kfrq' using the reduced system of order 'n'; 100~10,000 hz

Tutorial (3) : MATLAB Code

```
% plot the harmonic responses from the reduced system
FRQ = Yc(:,1);
MAG = abs(Yc(:,2:end));
PHS = (180./pi)*angle(Yc(:,2:end));
```

Save the results in a matrix-format

```
% UYs
figure(2);
semilogy(FRQ, MAG); grid on;
legend(NAMESr);
xlabel('Frequency [Hz]'); ylabel('Theta');
axis([100 10000 1E-8 1E-3]);
saveas(gcf,'uy_n.png','png');
```

Plot the results from the reduced models

```
% UYs@TOP
uy_top = [MAG(:,1), uy_top_ansys(:,2)];
ph_top = [PHS(:,1), uy_top_ansys(:,3)];
figure(3);
subplot(2,1,1); semilogy(FRQ, uy_top); grid on;
legend(['n=',num2str(n)], 'ANSYS');
xlabel('Frequency [Hz]'); ylabel('Theta');
axis([100 10000 1E-8 1E-2]);
subplot(2,1,2); plot(FRQ, ph_top); grid on;
xlabel('Frequency [Hz]'); ylabel('Phase [Deg]');
saveas(gcf,'uy@top_n.png','png');
```

Compare 'θ@TOP' between the reduced and ANSYS results and plot them

Final Remarks

- Model Order Reduction with MATLAB is
  - Very accurate!
  - Highly efficient!
  - Useful to mechanical + control systems
  - Applicable to many engineering areas

- Please contact me at jshan@andong.ac.kr
  if you want any questions and collaboration!