Numerical lienar algebra Homework 3
Model Order Reduction

Zirui YAN

Jan 3, 2024

Contents

Section 1 State Space Model

1.1 From ODEs to state space model

Ordinary Differential equation(ODE) is a differential equation dependent on only a single independent variable. **A system of ordinary differential equations (ODEs)** is a finite set of ordinary differential

equations. The **order** of an ODEs is the highest order of derivative of the unknown function that appears in the ordinary differential equations.

An m-th order ODEs can be written as

$$\mathbf{x}^{(m)} + A_1\mathbf{x}^{(m-1)} + \cdots + A_{m-1}\mathbf{x}^{(1)} + a_n\mathbf{x} = \mathbf{b}u(t) \tag{1}$$

There are many ways to make an ODEs into state space model. The realization is not unique. The most common realizations are Controllable Canonical Form, Observable Canonical Form and Jordan Canonical Form. We can use the way we want to change an n-th order ODEs into a first order ODEs (the number of equations will be n times.): (slightly abuse notation)

$$E\dot{\mathbf{x}} - A\mathbf{x} = \mathbf{b}u \tag{2}$$

Example 1.1. *Typical ODEs for mechanical systems take the form*

$$M\ddot{\mathbf{x}} + D\dot{\mathbf{x}} + K\mathbf{x} = \mathbf{b} \cdot u(t). \tag{3}$$

Here $\mathbf{x}(t) \in \mathbb{R}^n$ and $\mathbf{b} \cdot u(t) \in \mathbb{R}^n$ are vectors varying over time. Typically, $\mathbf{x}$ is a vector of displacements of degrees-of-freedom (dofs) arising from a Finite Element Method (FEM) discretization of the original continuous problem. The vector $\mathbf{b} \cdot u(t)$ then typically corresponds to a force exerted on the structure and is called the input. The matrices M, D and K are called the mass, damping and stiffness matrix respectively.

Let

$$\mathbf{q} := \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \end{bmatrix} := \begin{bmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{bmatrix} \tag{4}$$

then we have

$$\dot{\mathbf{q}} = \begin{bmatrix} \dot{\mathbf{q}}_1 \\ \dot{\mathbf{q}}_2 \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{x}} \\ \ddot{\mathbf{x}} \end{bmatrix} \tag{5}$$

Now equation (3) becomes

$$M\dot{\mathbf{q}}_2 = -\left(D\mathbf{q}_2 + K\mathbf{q}_1 - \mathbf{b}u(t)\right) \tag{6}$$

The mass matrix M is often invertible. With $\dot{q}_1 = \mathbf{q}_2$, we get

$$\dot{\mathbf{q}} = A\mathbf{q} + \mathbf{f} \cdot u(t) \tag{7}$$

where $A = \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{bmatrix}$ and $\mathbf{f} = \begin{bmatrix} \mathbf{0} \\ M^{-1}\mathbf{b} \end{bmatrix}$. Now the second order ODEs becomes the second order ODEs.

Example 1.2. *Consider*

$$M\ddot{\mathbf{x}} + D\dot{\mathbf{x}} + K\mathbf{x} = \mathbf{b} \cdot u(t). \tag{8}$$

again. Choose

$$\mathbf{q} := \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \end{bmatrix} := \begin{bmatrix} \mathbf{x} \\ M\dot{\mathbf{x}} \end{bmatrix} \tag{9}$$

Mimic what we have done in example (1.1), we get

$$E\dot{\mathbf{q}} = A\mathbf{q} + \mathbf{f} \cdot u(t) \tag{10}$$

where $E = \begin{bmatrix} M & 0 \\ D & I \end{bmatrix}$, $A = \begin{bmatrix} 0 & I \\ -K & 0 \end{bmatrix}$ and $\mathbf{f} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \end{bmatrix}$.

The biggest advantage of the state space model in example (1.2) is that it use the matrix(vector) M, D, K, $\mathbf{b}$ from original ODEs to construct matrix E, A,$\mathbf{f}$ instead of using $M^{-1}K$, $M^{-1}K$, $M^{-1}\mathbf{b}$. The two examples above show that an ODEs can be transformed into different state space models. **But all of them present the same system. This is a result of control theory. We will show that the above models is**

equivalent from equation (11) to (16). (Another way of proof is to multiply matrix E of example (1.2) on both side of equation (7) and notice $\mathbf{q}_2$ in two examples are different.)

Apply Laplace transform on both side of equation (2):

$$sE\mathbf{X} - E\mathbf{x}(0^-) - A\mathbf{X} = \mathbf{b}U \tag{11}$$

If $\mathbf{x}(0^-) = 0$, then

$$sE\mathbf{X} - A\mathbf{X} = \mathbf{b}U \tag{12}$$

Abuse notation in order to use the same notation as the assignment

$$sE\mathbf{x} - A\mathbf{x} = \mathbf{b}u \tag{13}$$

Then

$$\mathbf{x} = (sE - A)^{-1}\mathbf{b}u \tag{14}$$

For example (1.1),

$$\begin{aligned}\mathbf{q} = \begin{bmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{bmatrix} &= (sI - A)^{-1}\mathbf{f}u \\ &= (\begin{bmatrix} sI & -I \\ M^{-1}K & sI + M^{-1}D \end{bmatrix})^{-1} \begin{bmatrix} \mathbf{0} \\ M^{-1}\mathbf{b} \end{bmatrix} u \\ &= \begin{bmatrix} * & (s^2I + sM^{-1}D + M^{-1}K)^{-1} \\ * & * \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ M^{-1}\mathbf{b} \end{bmatrix} u \end{aligned} \tag{15}$$

So $\mathbf{x} = (s^2M + sD + K)^{-1}\mathbf{b}u$, which is the Laplace transform of equation (3).

For example (1.2), similarly

$$\begin{aligned}\mathbf{q} = \begin{bmatrix} \mathbf{x} \\ M\dot{\mathbf{x}} \end{bmatrix} &= (sE - A)^{-1}\mathbf{f}u \\ &= (\begin{bmatrix} sM & -I \\ sD + K & sI \end{bmatrix})^{-1} \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \end{bmatrix} u \\ &= \begin{bmatrix} * & (s^2M + sD + K)^{-1} \\ * & * \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \end{bmatrix} u \end{aligned} \tag{16}$$

So $\mathbf{x} = (s^2M + sD + K)^{-1}\mathbf{b}u$, which is the same as example (1.1).

We are not always interested in $\mathbf{x}$, but rather some functional applied to $\mathbf{x}$, which results in the state space model

$$sE\mathbf{x} - A\mathbf{x} = \mathbf{b}u \tag{17}$$

$$y = \mathbf{c}^T\mathbf{x}. \tag{18}$$

State space models of this kind will be referred to in this text as **basic state space models** (also Laplace domain state space model).

Apply Laplace transform on both side of equation (3), we get

$$\begin{aligned} s^2M\mathbf{x} + sD\mathbf{x} + K\mathbf{x} &= \mathbf{b}u \\ y &= \mathbf{c}^T\mathbf{x}. \end{aligned} \tag{19}$$

The model above is referred to as the **quadratic model of the mechanical system**. The quadratic model is also equivalent to the models given in examples 1.1 and 1.2. Equations (15) and (16) are enough to show this.

1.2 The residual form of transform functions

A **transfer function** of a system, sub-system, or component is a mathematical function that models the system's output for each possible input.

For continuous system, transfer function $H(s) := y(s)/u(s)$ where $y(s)$ is Laplace transform of the output $y(t)$ and $u(s)$ is Laplace transform of the output $u(t)$. If we know transfer function $H(s)$ of a system, we can get $y(t)$ with any input $u(t)$.

Let's start with basic state space models

$$sE\mathbf{x} - A\mathbf{x} = \mathbf{b}u \tag{20}$$

$$y = \mathbf{c}^T\mathbf{x}. \tag{21}$$

where $\mathbf{x}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^n$ and $E, A \in \mathbb{R}^{n\times n}$.

From equation (20), we have

$$\frac{\mathbf{x}}{u} = (sE - A)^{-1}\mathbf{b} \tag{22}$$

Therefore,

$$H(s) := y(s)/u(s) = \mathbf{c}^T(sE - A)^{-1}\mathbf{b} \tag{23}$$

Consider the generalized eigenvalue decomposition of matrix A with regard to matrix E

$$A = EP\Lambda Q^* \tag{24}$$

where $P = [\mathbf{p}_1 \cdots \mathbf{p}_n]$, $Q = [\mathbf{q}_1 \cdots \mathbf{q}_n]$ and $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. Assume $Q^* = P^{-1}$

Then,

$$\begin{aligned} sE - A &= sE - EP\Lambda Q^* \\ &= E(sI - P\Lambda Q^*) \\ &= E(sPQ^* - P\Lambda Q^*) \\ &= EP(sI - \Lambda)Q^* \\ &= E\sum_{i=1}^{n}(s - \lambda_i)\,\mathbf{p}_i\mathbf{q}_i^* \end{aligned} \tag{25}$$

Claim: Assume E is invertible, the inverse of $(sE - A)$ is $(\sum_{i=1}^{n}\frac{\mathbf{p}_i\mathbf{q}_i^*}{s-\lambda_i})E^{-1}$.

Proof.

$$\begin{aligned} (sE - A)(\sum_{i=1}^{n}\frac{\mathbf{p}_i\mathbf{q}_i^*}{s-\lambda_i})E^{-1} &= E(\sum_{i=1}^{n}(s-\lambda_i)\,\mathbf{p}_i\mathbf{q}_i^*)(\sum_{j=1}^{n}\frac{\mathbf{p}_j\mathbf{q}_j^*}{s-\lambda_j})E^{-1} \\ &= E(\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{p}_i\mathbf{q}_i^*\mathbf{p}_j\mathbf{q}_j^*)E^{-1} \end{aligned} \tag{26}$$

Notice $\mathbf{q}_i^*\mathbf{p}_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$, so

$$\begin{aligned} E(\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{p}_i\mathbf{q}_i^*\mathbf{p}_j\mathbf{q}_j^*)E^{-1} &= E(\sum_{i=1}^{n}\mathbf{p}_i\mathbf{q}_i^*)E^{-1} \\ &= EIE^{-1} \\ &= I \end{aligned} \tag{27}$$

□

Use the claim above, we get

$$H(s) = \sum_{i=1}^{n} \frac{\left(\mathbf{c}^T \mathbf{p}_i\right)\left(\mathbf{q}_i^* E^{-1}\mathbf{b}\right)}{s - \lambda_i} \tag{28}$$

This form is called **the residual form** of the transfer function. **The residual form can be achieved when matrix E is invertible and $Q^* = P^{-1}$.**

When $E = I$, the residual form of transfer function is

$$H(s) = \sum_{i=1}^{n} \frac{\left(\mathbf{c}^T \mathbf{p}_i\right)\left(\mathbf{q}_i^* \mathbf{b}\right)}{s - \lambda_i} \tag{29}$$

Example 1.3. *Consider the quadratic model of the mechanical system.*

$$\begin{aligned} s^2 M\mathbf{x} + sD\mathbf{x} + K\mathbf{x} &= \mathbf{b}u \\ y &= \mathbf{c}^T\mathbf{x}. \end{aligned} \tag{30}$$

Obviously, the transfer function

$$H(s) := \frac{y(s)}{u(s)} = \mathbf{c}^T(s^2M + sD + K)^{-1}\mathbf{b} \tag{31}$$

From examples 1.1 and 1.2, we know the quadratic model can be rewritten into the linear model. Then, use (generalized) eigenvalue decomposition, we can get the residual form of the quadratic model

$$H(s) = \sum_{i=1}^{2n} \frac{R_i}{s - \lambda_i} \tag{32}$$

But the cost of computing (generalized) eigenvalue decomposition is high ($\mathcal{O}\left(n^3\right)$) (It might have some easy way to compute since matrices M and K have special structures in practice.)

Notice $s = \lambda_i$ is a pole of the transfer function if and only if matrix $(s^2M + sD + K)$ is singular, which might also give a good algorithm to compute λ_i. Once we know λ_i, R_i can be computed by

$$R_i = \lim_{s \to \lambda_i} (s - \lambda_i) H(s) \tag{33}$$

The residual form is more useful compared to the original form of transform function.

In the original form (see equation 23), $(sE - A)^{-1}$ is a matrix of size $n \times n$ in variable s. And after left-multiplying $\mathbf{c}^T$ and right-multiplying $\mathbf{b}$, it becomes a rational fraction of the form $\frac{f(s)}{g(s)}$, where f and g are polynomials of very high order (when n is large). It is more convenient to use residual form since it can be computed more quickly when we know the poles and residues.

But for residual form, each term is a simple rational fraction of the form $\frac{R_i}{s-\lambda_i}$. It is much easier to compute the inverse Laplace transformation of $\frac{R_i}{s-\lambda_i}$ and then sum them together.

Section 2 Applications about State Space model

2.1 Spiral inductor

2.1.1 Introduction

Spiral inductors, which vary in structure, are among the most common types of on-chip inductors. Spiral inductors are usually characterized by the diameter, the line width, the number of turns, and the line-to-line space. In this case, the inductor has turns that are 40μm wide, 15μm thick, with a separation of 40μm. The spiral is suspended 55μm over a substrate by posts at the corners and centers of the turns in order to reduce the capacitance to the substrate. To make it also a proximity sensor, a 0.1μm plane of copper is added 45μm above the copper spiral. The overall extent of the suspended turns is 1.58 mm $\times$ 1.58 mm. The spiral inductor, including part of the overhanging copper plane, is shown in figure 1.

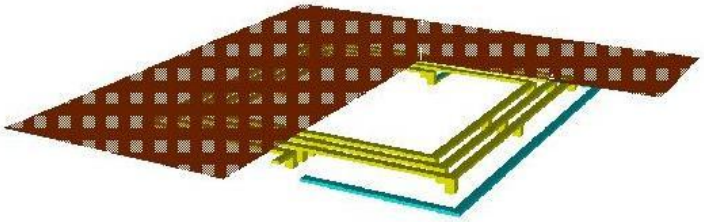

Figure 1: Spiral inductor

2.1.2 State space model of spiral inductor

The model is discretized using a FEM-like technique called partial element equivalent circuit method (PEEC), which results in mesh specific inductance and resistance matrices L and R respectively, corresponding to the following differential state-space form:

$$\begin{aligned} L\frac{di_m}{dt} &= Ri_m + Nv_p \\ i_p &= N^T i_m \end{aligned} \tag{34}$$

Here i_m is the mesh current and v_p and i_p are the voltage and current at nodes of interest, with N a 'natural' matrix mapping between these nodes and the mesh.

Apply Laplace transform on both side, we get (abusing notation)

$$\begin{aligned} sLi_m &= Ri_m + Nv_p \\ i_p &= N^T i_m \end{aligned} \tag{35}$$

The MATLAB code uses the same notation as [2]. And Altan's paper [3] explains all the details about practical **assive Reduced-order Interconnect Macromod- eling Algorithm(PRIMA)**. The implementation of PRIMA is in `prima.m` and `barnoldi.m`.

2.1.3 Skin effect and reduced model

`spiral_inductor.m` contains the code for this part.

In the multigigahertz frequency range, the so-called **'skin effect'** causes current to flow only at the surface of conductors (i.e. the copper coil and plane), leading to a decrease of wire inductance and an increase of resistance. Capturing the skin effect while also maintaining an accurate low frequency response is a challenge for many model reduction algorithms. For that reason the frequency range considered for this model is very wide: $\omega \in \left[1, 10^{10}\right]$.

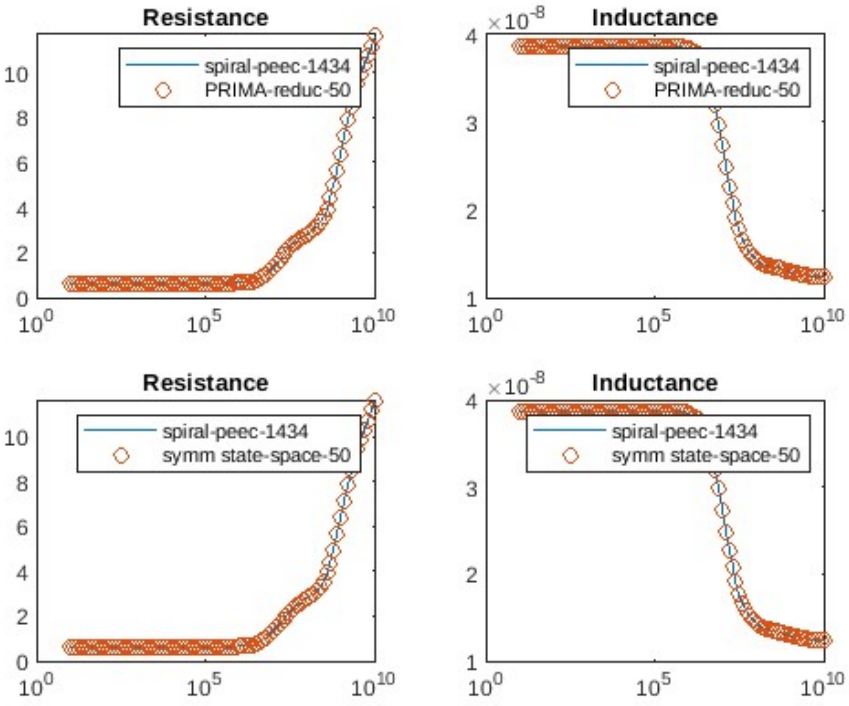

Figure 2: Reduced model with order 50

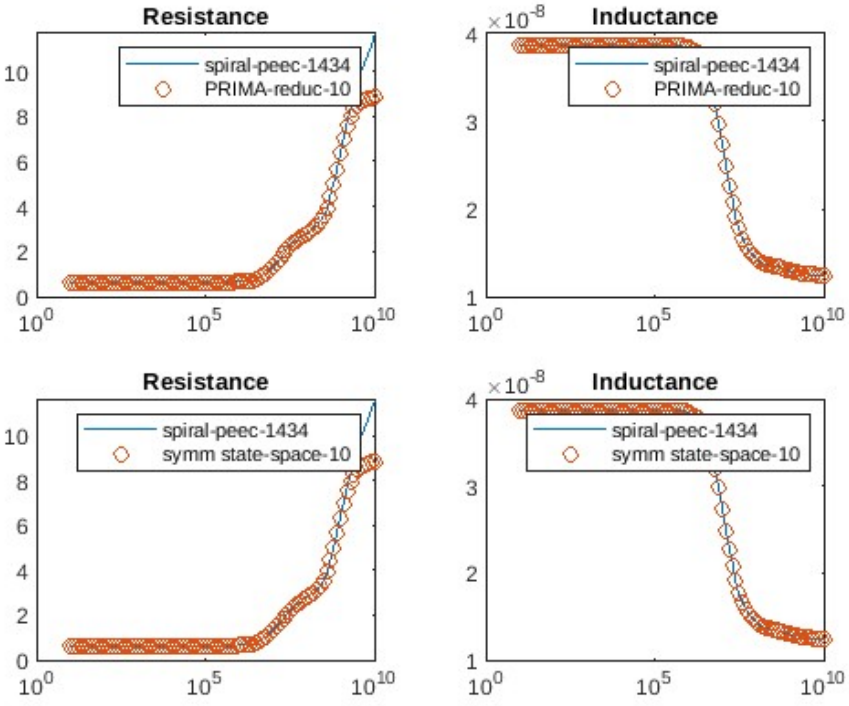

Figure 3: Reduced model with order 10

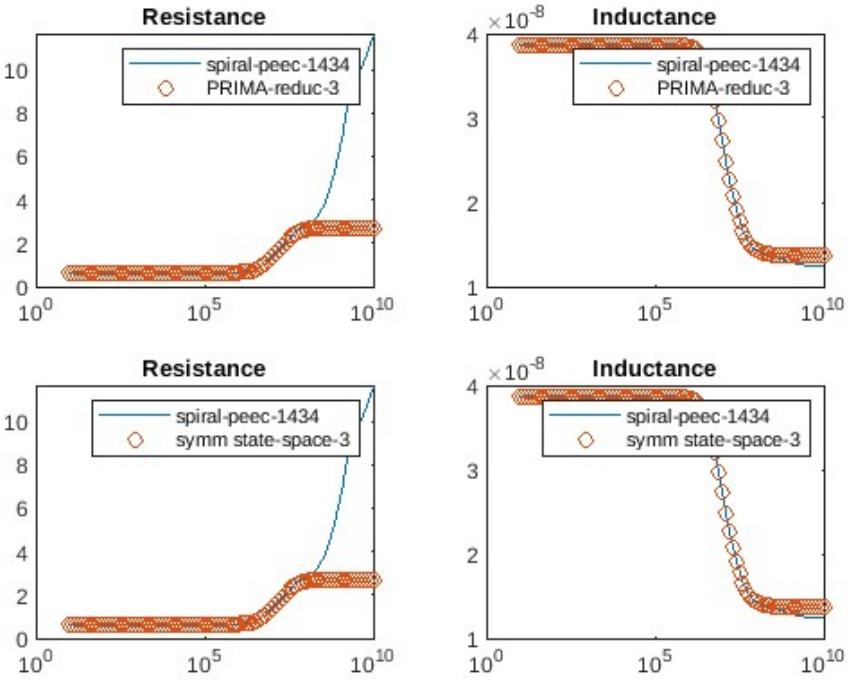

Figure 4: Reduced model with order 3

From Figure 2 to 4, we know reduced model of order 50 can capture 'skin effect' perfect. Some reduced models of lower order, e.g. order 10, can also perform very good but be unable to increase resistance as high as he original system. Models of even lower order, e.g., can almost give the same inductance but they totally lose 'skin effect'.

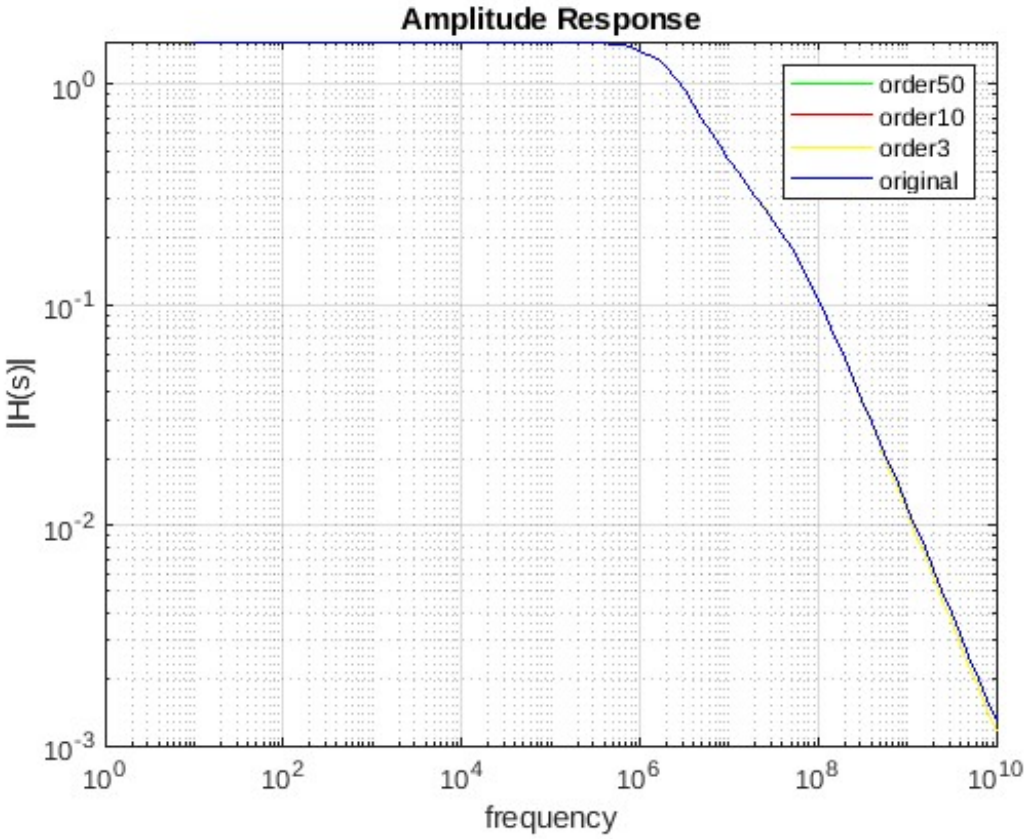

Figure 5: Amplitude response for order50, order10, order3 and original system

Figure 5 shows even reduced model of order 3 can give good ampliude response. But its phase response deviates from the response of original model when frequency is high. Reduced model of order 50 perform perfectly and totally overlap with the bode diagram of original model.

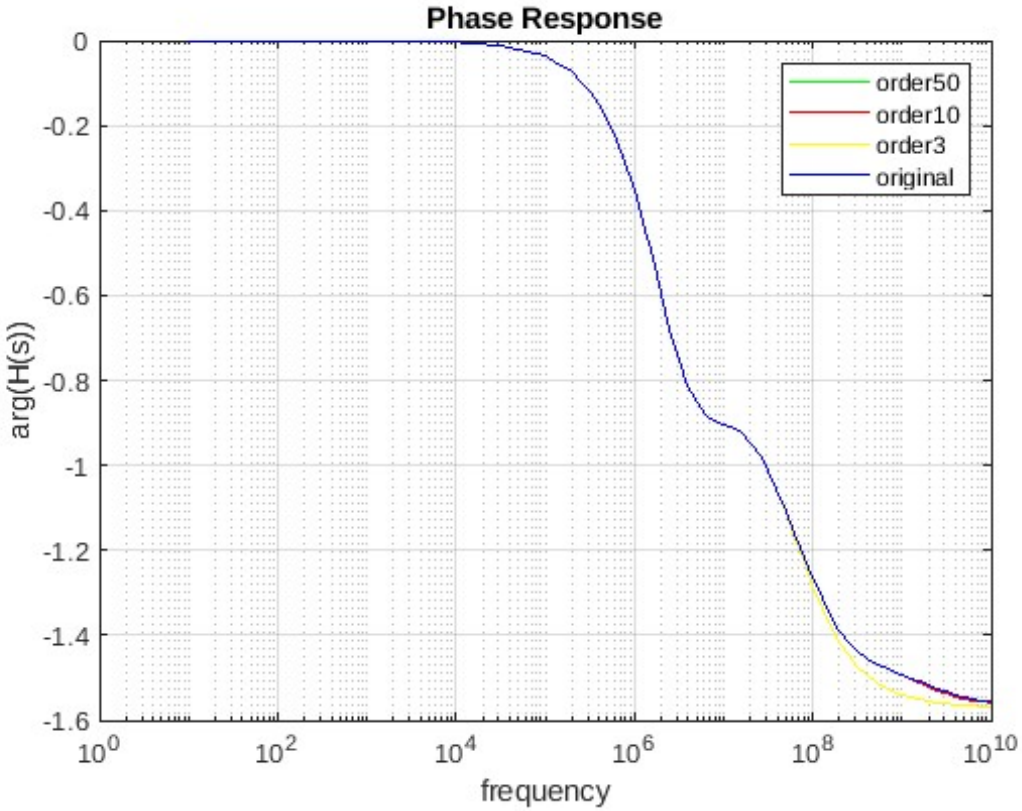

Figure 6: Phase response for order50, order10, order3 and original system

2.2 Butterfly Gyroscope

2.2.1 Introduction

In inertial navigation, rotation sensors, also called gyroscopes, are indispensible. One such sensor, at the high-end of the quality range is the Butterfly gyroscope. You can see a diagram of this gyro in Figure 7 By applying DC-biased AC-voltages, the Butterfly wings are forced to vibrate in anti-phase in the wafer plane. This vibrational mode is called the excitation mode. As the structure containing the gyro rotates about the axis of sensitivity (see Figure 7), each of the masses will be affected by a Coriolis acceleration. The Coriolis force induces an anti-phase motion of the wings out of the wafer plane, which can be measured via different electrodes. The external angular velocity can be related to the amplitude of this vibrational mode. Since this gyro must be fast and precise, it should be clear to you that a high-precision, but efficiently computable structural model of the gyro must be implemented.

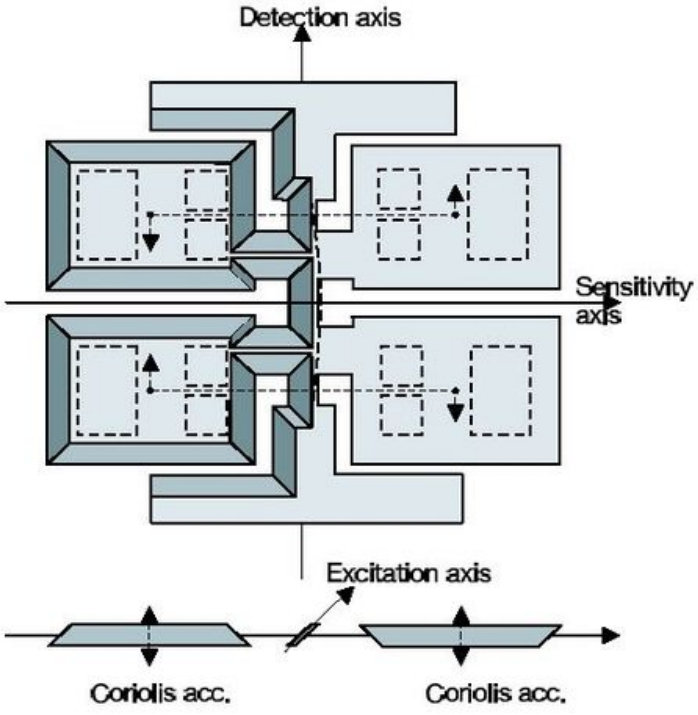

Figure 7: Butterfly gyroscope

2.2.2 State space model of butterfly gyroscope

The Gyro is discretized using FEM, resulting in a model of the form

$$\begin{aligned} M\ddot{\mathbf{x}} + \beta K\dot{\mathbf{x}} + K\mathbf{x} &= B \cdot u(t) \\ y(t) &= C\mathbf{x}(t) \end{aligned} \tag{36}$$

We can use MATLAB function `mechss` to get state space model.

2.2.3 Use PRIMA to reduce the model

(`butterflygyro.m` contains the code for this part.)

The original model is of very large order because FEM often needs large number of elements to work well. But this causes large cost for computation. We can use `barnoldi.m` and `prima.m` in the previous subsection on state space model (36) in order to get the reduced model. (In my implementation `butterflygyro.m`, I only deal with one output.)

The original model equals to

$$\begin{aligned} X'MXX'\ddot{\mathbf{x}} + \beta X'KXX'\dot{\mathbf{x}} + X'KXX'\mathbf{x} &= X'B \cdot u(t) \\ y(t) &= CXX'\mathbf{x}(t) \end{aligned} \tag{37}$$

where $X'MX$ is a Hessenberg matrix. Let $z = X'_m x$, then

$$\begin{aligned} M_m\ddot{\mathbf{z}} + \beta K_m\dot{\mathbf{z}} + K_m\mathbf{z} &= B_m \cdot u(t) \\ y(t) &= C_m\mathbf{z}(t) \end{aligned} \tag{38}$$

where $M_m = X'_m MX_m$, $K_m = X'_m KX_m$, $B_m = X'_m B$ and $C_m = CX_m$.

The given system has 1 input and 12 outputs. Here I choose the input1-output2 as an example. Figure 8 shows that the reduced model of order 4 performs almost the same as the origin system in the frequency range

$f \in [1, 10000]$Hz. Therefore, reduced model of order 4 is a good choice for the input1-output2 pair. Although we need to construct a reduced model for every input-output pair, this method allows us to significantly reduce the order of the model and highly reduce the cost of computing the response.

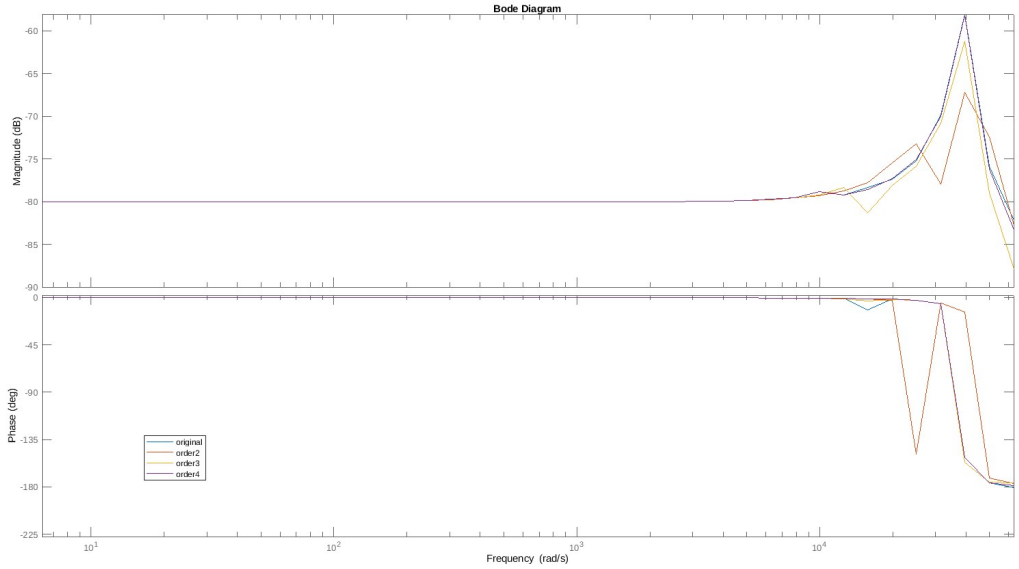

Figure 8: Bode diagram of reduced models and the original system for output2

2.3 International space station component

Control is another important source of model order reduction, since the complexity of a controller is dominated by the complexity of the corresponding system. In the International Space Station (ISS), assembly, testing and maneuvring is done in 'stages' i.e. parts of the ISS. One such stage, stage 12A, is depicted in figure 9. This stage leads to a large sparse system to be reduced in order.

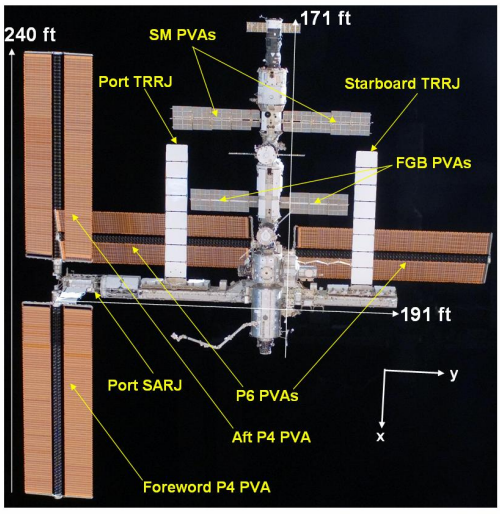

Figure 9: Stage 12A of the ISS. Source: First Ever Flight Demonstration of Zero Propellant Maneuver(TM) Attitude Control Concept, Bedrossian, N. & Bhatt, S., 2007

This system was determined using techniques from standard systems theory, and is of the form

$$s\mathbf{x} = A\mathbf{x} + B\mathbf{u}$$
$$\mathbf{y} = C\mathbf{x} \tag{39}$$

(There is a typo in homeworkMOR.pdf. It forgets $\mathbf{u}$. And here should be C instead of C^T)
(`iss.m` contains the code for this part.)

This state space model has 3 inputs and 3 outputs. This model is not too large, so we have 2 kinds of choices to reduce the model: 1. Divide this model into 9 sub-models with a single input and a single output, then do the model order reduction for every sub-model. 2. Do model order reduction on the original model.

Which one to use depends on our needs. If we are only interested in a single or few input-ouput pairs, we should apply model order reduction methods on sub-models we are interested in. If we are interested in all the pairs and the size of reduced model is not much bigger than a reduced sub-model, we can use the second kinds of choices.

Since $E = I$ in this state space model, we can use some functions given by MATLAB. For example, `balred`. `balred` returns the balanced truncated model. One of the advantages of balanced truncation is that it can preserve stability and passivity. From Figure 10 and 11, we know balanced truncated model of the original model needs much higher order to achieve the same performance of balanced truncated model of sub-model with a single input and a single output. This can be interpreted as each input-output is mainly decided by different part of the system. So we should use the first kind of method to do the model order reduction.

From Figure 12 and 13, we know that the amplitude response and phase response change very quickly and that reduced models with low order are hard to capture their quick changes. Balanced truncated model of input1-output1 sub-model with order 150 performs almost perfectly in the frequency range $\omega \in \left[10^{-2}, 10^{2}\right]$.

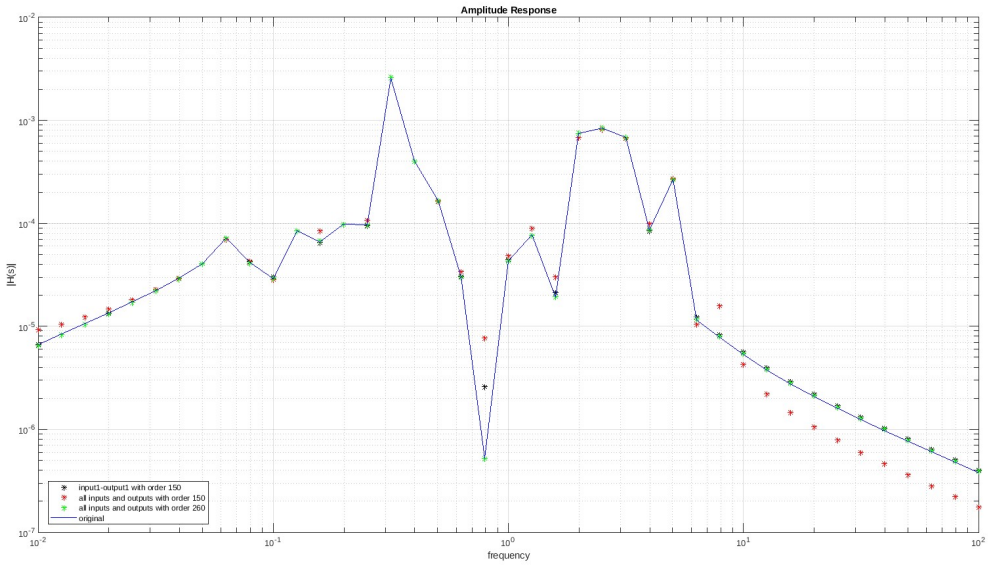

Figure 10: Amplitude response of reduced models and the original system. (Blue line) the original model; (black star) balanced truncated model of sub-model with order 150; (red star) balanced truncated model of the original model with order 150; (green star) balanced truncated model of the original model with order 260.

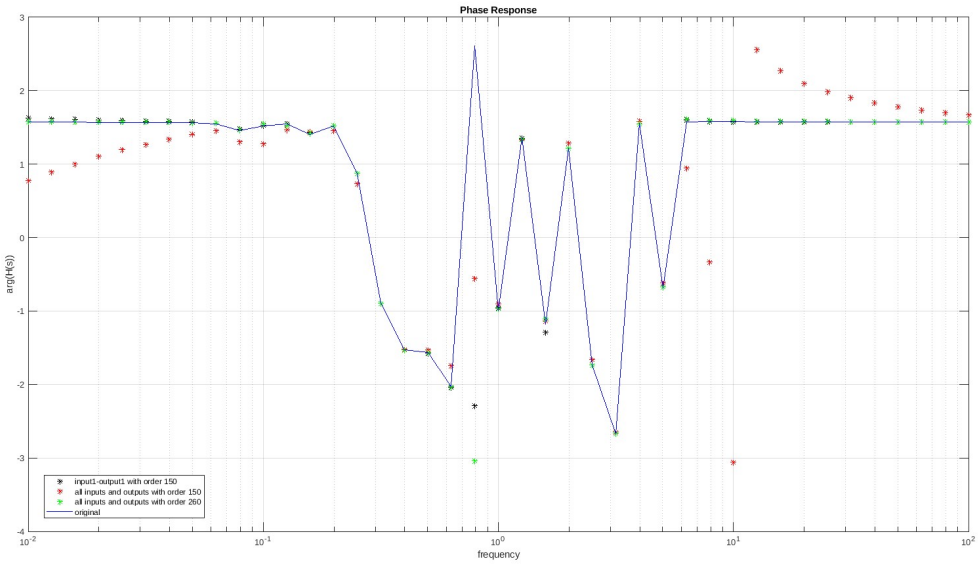

Figure 11: Phase response of reduced models and the original system. (Blue line) the original model; (black star) balanced truncated model of sub-model with order 150; (red star) balanced truncated model of the original model with order 150; (green star) balanced truncated model of the original model with order 260.

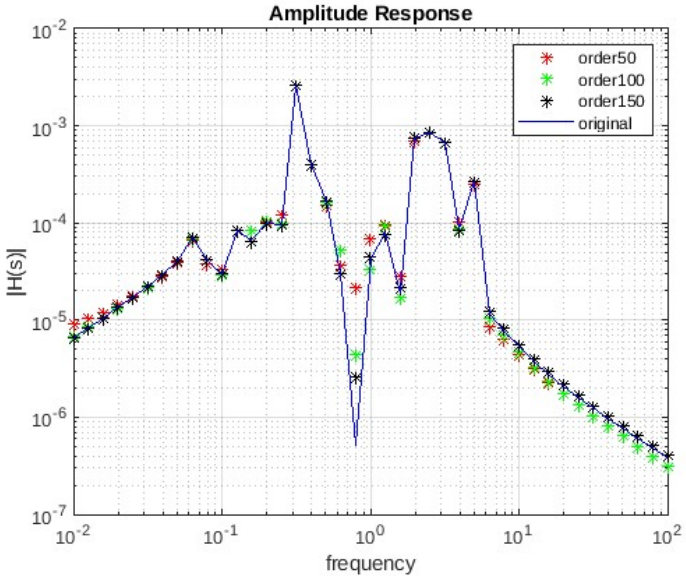

Figure 12: Amplitude response of reduced models and the original system. (Blue line) the original model; (black star) balanced truncated model of the sub-model with order 150; (green star) balanced truncated model of the sub-model with order 100; (green star) balanced truncated model of the sub-model with order 50.

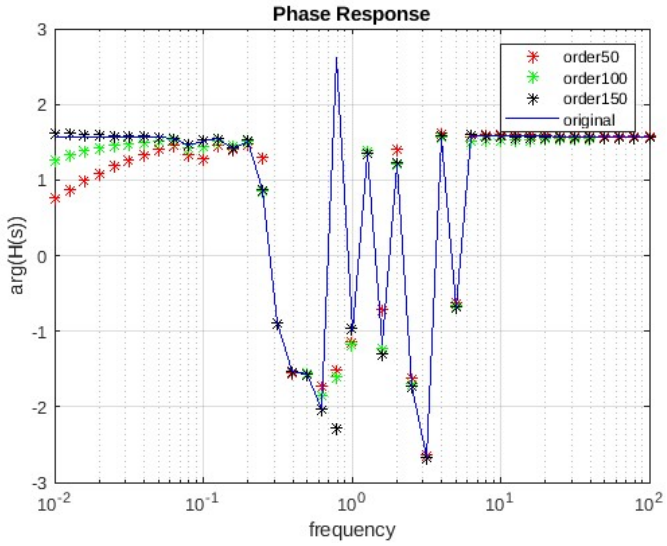

Figure 13: Phase response of reduced models and the original system. (Blue line) the original model; (black star) balanced truncated model of the sub-model with order 150; (green star) balanced truncated model of the sub-model with order 100; (green star) balanced truncated model of the sub-model with order 50.

Section 3 Dominant Pole Algorithm

The residual form of the transfer function

$$H(s) = \sum_{i=1}^{n} \frac{R_i}{s - \lambda_i} \tag{40}$$

Most of the time, we are most interested in the value of the transfer function when s is on the imaginary axis because it decides the peaks in the transfer function. Therefore, the term with larger $\frac{|R_i|}{\mathrm{Re}(\lambda_i)}$ is more important. $\frac{|R_i|}{\mathrm{Re}(\lambda_i)}$ is called **weighted residue**. The first k **dominant poles** are the poles $s = \lambda_i$ with the biggest k corresponding weighted residues.

We can approximate (40) by taking the first k dominant terms

$$H(s) \approx \sum_{i=1}^{k \ll n} \frac{R_i}{s - \lambda_i} \tag{41}$$

The **dominant pole algorithm(DPA)** is summarized in algorithm 1.

Algorithm 1: DPA

Input: System (E, A, b, c), initial pole estimate s_0, tolerance ϵ

Output: Approximate dominant pole $\hat{\lambda}$ and corresponding eigenpair (x, y)

init k:=0, $err = \infty$

while $err > tol$ **do**

  Solve $(s_k E - A)\,\mathbf{v}_k = \mathbf{b}$

  Solve $(s_k E - A)^*\,\mathbf{w}_k = \mathbf{c}$

  Compute the new pole estimate

  $s_{k+1} = s_k - \frac{\mathbf{c}^*\mathbf{v}_k}{\mathbf{w}_k^* E \mathbf{v}_k} = \frac{\mathbf{w}_k^* A \mathbf{v}_k}{\mathbf{w}_k^* E \mathbf{v}_k}$

  $\mathbf{x} := \mathbf{v}_k / \|\mathbf{v}_k\|$

  $\mathbf{y} := \mathbf{w}_k / \|\mathbf{w}_k\|$

  $err := \|A\mathbf{x} - s_{k+1} E \mathbf{x}\|_2$

  k:=k+1

This algorithm can be proven by Newton's method for finding a root of $Q(s) = H^{-1}(s)$. We have the iteration

$$\begin{aligned} s_{k+1} &= s_k - \frac{Q(s_k)}{Q'(s_k)} \\ &= s_k + \frac{H(s_k)}{H'(s_k)} \quad (Q' = -H'H^{-2}) \\ &= s_k - \frac{\mathbf{c}^*\mathbf{v}_k}{\mathbf{w}_k^* E\mathbf{v}_k} = s_k - \frac{\mathbf{w}_k^*\mathbf{b}}{\mathbf{w}_k^* E\mathbf{v}_k} \quad (H' = -\mathbf{c}^*(sE-A)^{-2}\mathbf{b}) \\ &= \frac{\mathbf{w}_k^* A\mathbf{v}_k}{\mathbf{w}_k^* E\mathbf{v}_k} \quad (s_k E\mathbf{v}_k - \mathbf{b} = A\mathbf{v}_k) \end{aligned} \tag{42}$$

where $\mathbf{v}_k = (s_k E - A)^{-1}\mathbf{b}$ and $\mathbf{w}_k = (s_k E - A)^{-*}\mathbf{c}$.

Simple DPA is not a very good or useful algorithm. Firstly, it only computes one dominant pole, which is not enough for approximating complex transfer functions; if we want to compute multiple dominant poles, we need to select many good initial pole estimate; and moreover, we can keep $\mathbf{v}_k$ and $\mathbf{w}_k$ after the iterations of one pole in order to get some information about the eigenvectors. Secondly, which dominant pole DPA converges to depends on the initial pole estimates s_0, but sometimes we do not know which initial pole estimates s_0 is able to return the pole we want. Thirdly, we require (A, E) to be a non-defective matrix pencil, otherwise $\mathbf{w}_k^* E\mathbf{v}_k$ might tend to 0 as $s_k \to \lambda_k$.

These problems motivate the **subspace accelerated DPA(SADPA)**. In SADPA, we keep $\mathbf{w}_k$ $\mathbf{v}_k$ in spaces W and V after we find a dominant pole, see [4].

Lemma 3.1. *(**Lemma 1** in homeworkMOR.pdf)*
Suppose $\left(s^2M + sD + K\right) := G(s)$ *is invertible. Then*

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}s}G^{-1}(s) &= -G^{-1}(s)G'(s)G^{-1}(s) \\ &= -\left(s^2M + sD + K\right)^{-1}(2sM + D)\left(s^2M + sD + K\right)^{-1} \end{aligned} \tag{43}$$

Proof. From $\left(s^2M + sD + K\right) := G(s)$, we know

$$G'(s) = \frac{\mathrm{d}}{\mathrm{d}s}\left(s^2M + sD + K\right) = 2sM + D. \tag{44}$$

Notice $\frac{\mathrm{d}}{\mathrm{d}s}(G(s)G^{-1}(s)) = \frac{\mathrm{d}}{\mathrm{d}s}I = 0$. So

$$\frac{\mathrm{d}}{\mathrm{d}s}(G(s)G^{-1}(s)) = \left(\frac{\mathrm{d}}{\mathrm{d}s}G(s)\right)G^{-1}(s) + G(s)\left(\frac{\mathrm{d}}{\mathrm{d}s}G^{-1}(s)\right) = 0. \tag{45}$$

Therefore,

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}s}G^{-1}(s) &= -G^{-1}(s)G'(s)G^{-1}(s) \\ &= -\left(s^2M + sD + K\right)^{-1}(2sM + D)\left(s^2M + sD + K\right)^{-1} \end{aligned} \tag{46}$$

□

Now we can construct a Newton method for a quadratic model like in the case of (regular) DPA. Take $Q(s) = H^{-1}(s)$ and $H(s) = \mathbf{c}^*(s^2M + sD + K)^{-1}\mathbf{b}$. Let's find the root of $Q(s)$ by Newton's iteration

$$\begin{aligned} s_{k+1} &= s_k - \frac{Q(s_k)}{Q'(s_k)} \\ &= s_k + \frac{H(s_k)}{H'(s_k)} \quad (Q' = -H'H^{-2}) \\ &= s_k - \frac{\mathbf{c}^*\mathbf{v}_k}{\mathbf{w}_k^*(2s_kM + D)\mathbf{v}_k} = s_k - \frac{\mathbf{w}_k^*\mathbf{b}}{\mathbf{w}_k^*(2s_kM + D)\mathbf{v}_k} \quad (H' = -\mathbf{c}^*\left(s^2M + sD + K\right)^{-1}(2sM + D)\left(s^2M + sD + K\right)^{-1}\mathbf{b}) \\ &= \frac{\mathbf{w}_k^*(s_k^2M - K)\mathbf{v}_k}{\mathbf{w}_k^*(2sM + D)\mathbf{v}_k} \quad (2s_k^2M\mathbf{v}_k + sD\mathbf{v}_k - \mathbf{b} = s_k^2M\mathbf{v}_k - K) \end{aligned} \tag{47}$$

where $\mathbf{v}_k = \left(s_k^2 M + s_k D + K\right)^{-1} \mathbf{b}$ and $\mathbf{w}_k = \left(s_k^2 M + s_k D + K\right)^{-*} \mathbf{c}$.

We call this algorithm QDPA. QDPA is summarized in algorithm 2.

Algorithm 2: QDPA

Input: Quadratic model of a mechanical system (M, D, K, b, c), initial pole estimate s_0, tolerance ϵ

Output: Approximate dominant pole $\hat{\lambda}$ and corresponding eigenpair (x, y)

init k:=0, $err = \infty$

while $err > tol$ **do**

- Solve $\left(s_k^2 M + s_k D + K\right) \mathbf{v}_k = \mathbf{b}$
- Solve $\left(s_k^2 M + s_k D + K\right)^* \mathbf{w}_k = \mathbf{c}$
- Compute the new pole estimate
- $s_{k+1} = s_k - \frac{\mathbf{c}^* \mathbf{v}_k}{\mathbf{w}_k^* (2sM+D)\mathbf{v}_k} = \frac{\mathbf{w}_k^* (s_k^2 M - K)\mathbf{v}_k}{\mathbf{w}_k^* (2sM+D)\mathbf{v}_k}$
- $\mathbf{x} := \mathbf{v}_k / \|\mathbf{v}_k\|$
- $\mathbf{y} := \mathbf{w}_k / \|\mathbf{w}_k\|$
- $err := \left\|(s_{k+1}^2 M + s_{k+1} D + K)\mathbf{x}\right\|_2 = \left\|(s_{k+1}^2 M - K)\mathbf{x} - s_{k+1}(2s_{k+1}M + D)\mathbf{x}\right\|_2$
- k:=k+1

QDPA has the same problems as DPA. We can specially tailor SADPA to quadratic model of a mechanical system, and we call this algorithm **SAQDPA**

3.1 Spiral inductor

(`application_dpa_peec.m` contains the code for this part.)

From the previous section, we know the reduced model of order 3 already has good performance except for the high-frequency part. By using MATLAB functions `dss` and `pole`, we get the poles of the reduced model of order 3 are $1e8 * [3.1469, -0.1403, -1.0908]$. I implement DPA in `dpa.m` and this function should return one of the poles since they are dominant.

I choose initial estimate s_0 with real part equal to 0. And the result of DPA is shown in Figure 14.

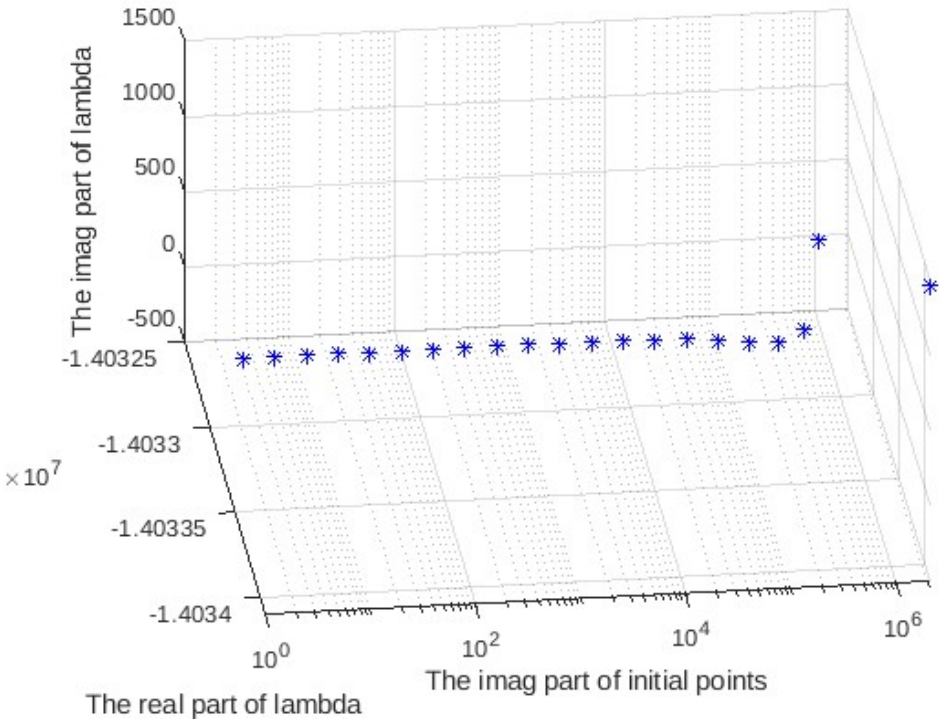

Figure 14: The results of DPA with different initial values.

We can see that λ converges to $-0.1403e8$ when the imaginary part of the initial pole estimate s_0 is smaller than $10^{4.4}$, which also implies the low frequency performance is mainly decided by $-0.1403e8$. If we want to find more poles, we need to try different initial pole estimate many times.

Figures 15 and 16 show the reduced model given by SADPA. Different from the reduced model of order 50 given by PRIMA, reduced model of order 50 given by SADPA still slightly deviates from the original model in the high-frequency region. But SADPA returns the dominant poles and corresponding residues, which enable us to compute $H(s)$ more efficiently.

In `application_dpa_peec.m`, I choose `options = struct("nwanted",nwanted,"tol",1e-5, displ",1, "strategy",'LR',"kmin",1,"kmax",15,"maxrestarts",100,"f_ax",'N',"f_ex",'N',"f_semax",'N', "f_semax_s",'N',"use_lu",0,"use_lu_w_amd",0,"dpa_bordered",0,"yEx_scaling",0, "rqitol",1e-4, "turbo_deflation",0)`. `nwanted` is a variable which I can easily change the value of it. Here I choose `nwanted=50` because I already know the reduced model of order 50 has good performance. Some other settings are chosen from default type. `strategy=LR` because we want to get the dominant poles. We can also change the `kmax` and `maxrestarters` in order to control when to restart and maximum number of restarts.

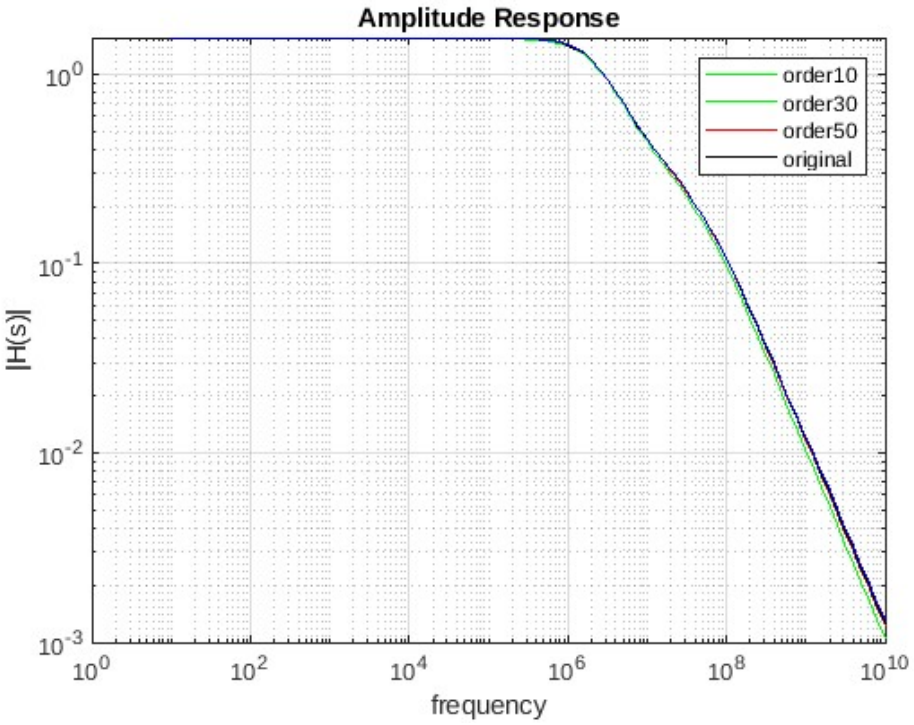

Figure 15: The amplitude response of the original model and reduced models given by SADPA.

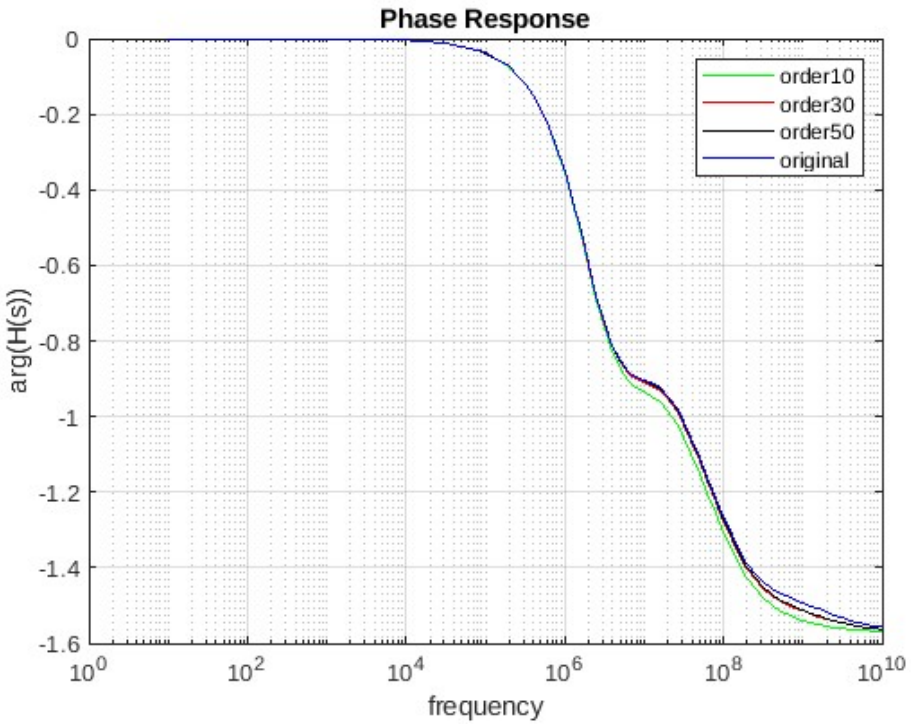

Figure 16: The phase response of the original model and reduced models given by SADPA.

We can also use $\frac{|H(s)-\hat{H(s)}|_\infty}{|H(s)|_\infty}$ to check the performance of the reduced model, see Figure 17. Using this Figure, we can easily choose the number of dominant poles we want.

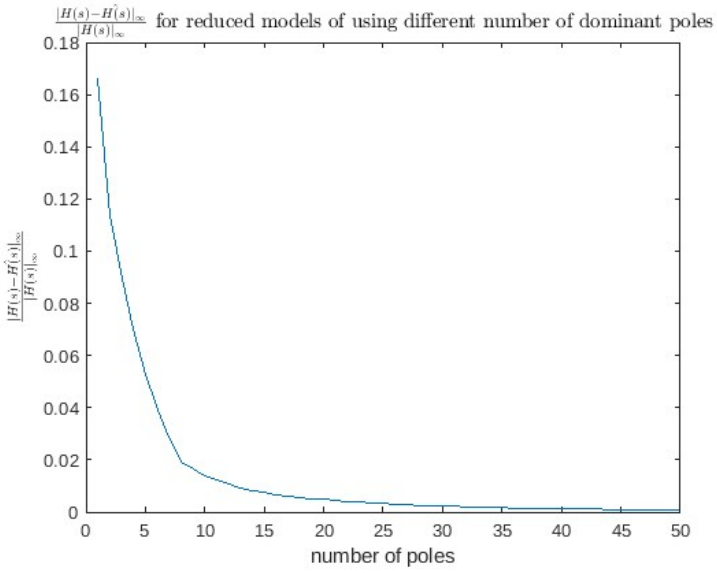

Figure 17: $\frac{|H(s)-\hat{H(s)}|_\infty}{|H(s)|_\infty}$ for reduced models using different number of dominant poles

3.2 Gyro

(`application_dpa_gyro.m` contains the code for this part.)

Although I've written down the function for QPDA, but it cannot converge for Gyro.

In subsection 2.2, we've already constructed reduced model(order 4 is enough for output2). Then I construct reduced models of order from 1 to 10. Then I know the order of each dominant pole. SAQPDA can return most

of dominant poles except for $-1298759.8945 + 954328.31114i$. When I set turbo_deflation == 0, it can find this pole, but also return some other poles we do not want. Figure 18 and 19 show the result of turbo_deflation == 1. We can see the two plot have some simlilarity in shape, but they are in different scales, which comes from that $-1298759.8945 + 954328.31114i$ did not be computed and there are some problems abut the result of residues.

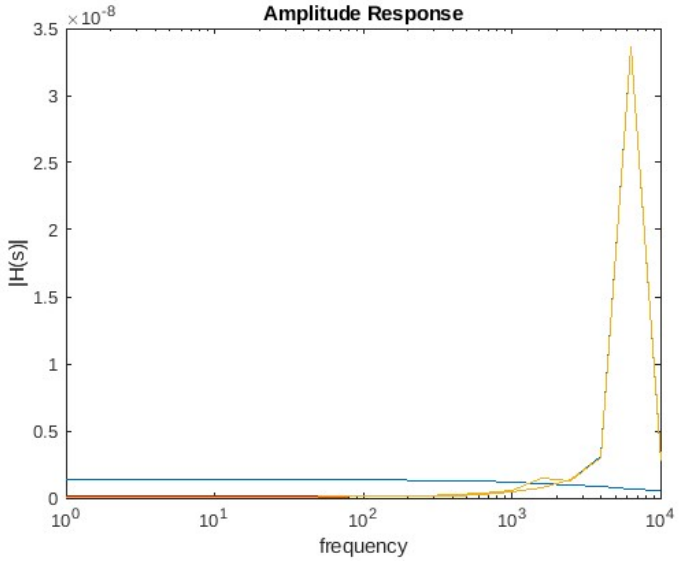

Figure 18: The amplitude response of reduced model of output2 of order 1 to 10 given by SAQPDA

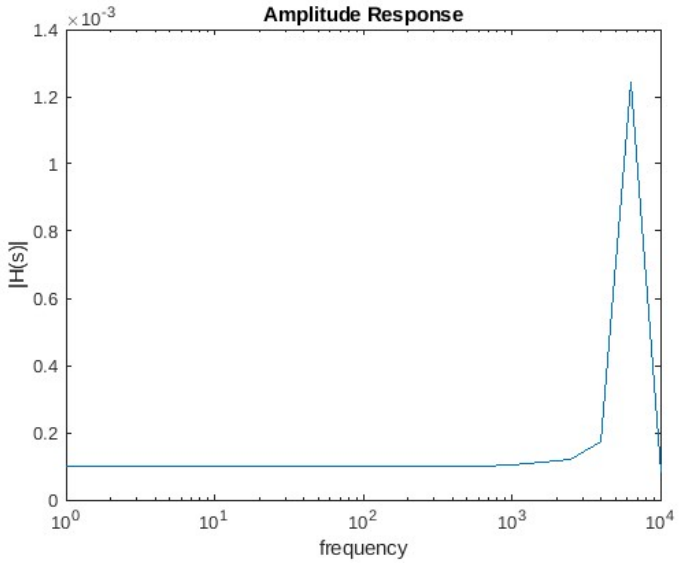

Figure 19: The amplitude response of the original model of output2.

3.3 ISS

(`application_dpa_iss.m` contains the code for this part.)

The transfer function of ISS has many oscillations. Compared to previous two applications, we need more poles to approximate it.

From Figure 20, we know that sometimes increasing the model order cannot get better performance and that sometimes it even makes the performance worse (order 3 to 16 in Figure 20). Reduced model of order 120 is good enough since higher order model cannot perform much better than it, see Figure 21 and 22.

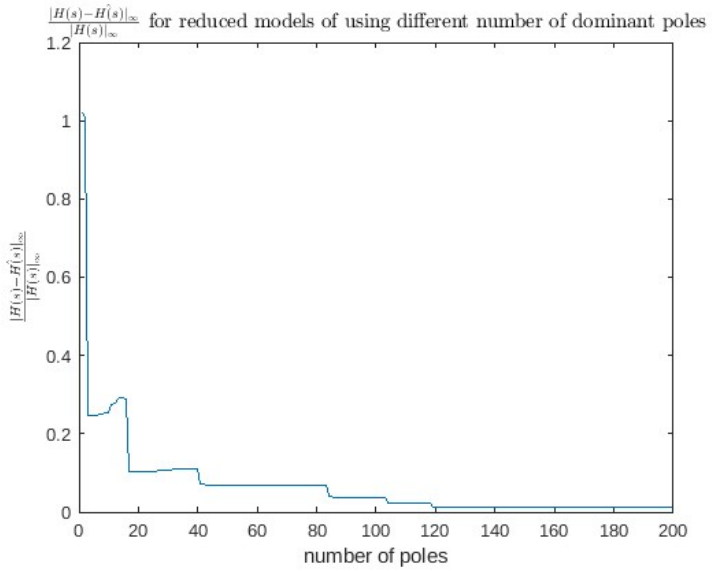

Figure 20: $\frac{|H(s)-\hat{H(s)}|_\infty}{|H(s)|_\infty}$ for reduced models using different number of dominant poles

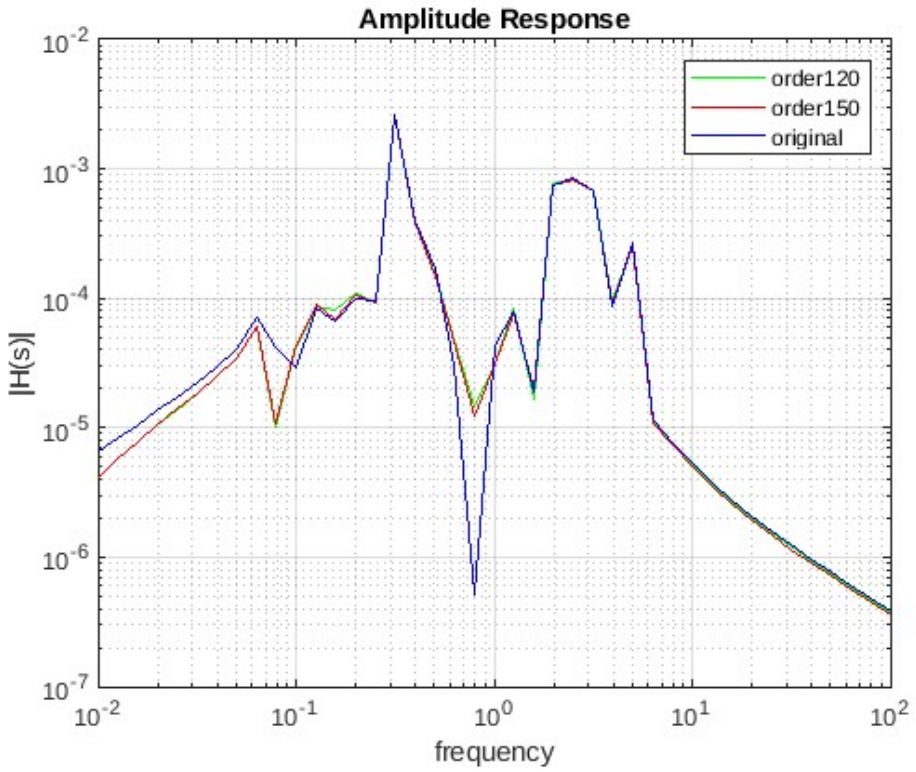

Figure 21: The amplitude response of the original model and reduced models given by SADPA.

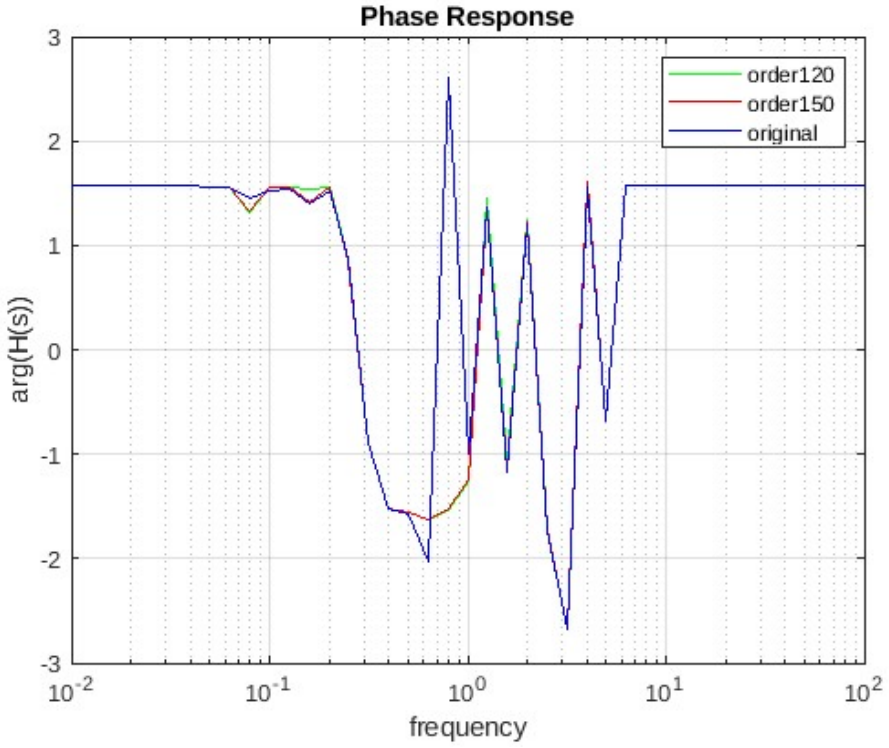

Figure 22: The phase response of the original model and reduced models given by SADPA.

Section 4 Iterative Rational Krylov Algorithm

4.1 Basics

Suppose we have a system of the form

$$\begin{aligned} s\mathbf{x} - A\mathbf{x} &= \mathbf{b}u \\ y &= \mathbf{c}^T\mathbf{x}. \end{aligned} \tag{48}$$

with associated transfer function

$$H(s) = \sum_{i=1}^{n} \frac{R_i}{s - \lambda_i}. \tag{49}$$

Theoretically, the Iterative Rational Krylov Algorithm (IRKA) aims at minimizing

$$\|H - \hat{H}\|_{\mathcal{H}_2} := \left(\int_{\infty}^{-\infty} |H(is) - \hat{H}(is)| ds \right)^{\frac{1}{2}} \tag{50}$$

over the set $\mathcal{V}_d$ of approximants

$$\hat{H}(s) = \sum_{i=1}^{d} \frac{\hat{R}_i}{s - \hat{\lambda}_i} \tag{51}$$

of fixed Macmillan degree d.

Lemma 4.1. *(**Lemma 2** in homeworkMOR.pdf)*
If

$$G(s) = \sum_{i=1}^{n} \frac{R_i}{s - \lambda_i} \tag{52}$$

then it holds that

$$\|G\|_{\mathcal{H}_2}^2 = \sum_{i=1}^{n} R_i G\left(-\lambda_i\right) \tag{53}$$

under mild technical assumptions that you can assume satisfied.

Now define the error function

$$\mathcal{E} := \|H - \hat{H}\|_{\mathcal{H}_2}^2. \tag{54}$$

Then,

$$\mathcal{E} = \left(\int_{\infty}^{-\infty} |H(is) - \sum_{i=1}^{d} \frac{\hat{R}_i}{is - \hat{\lambda}_i}| ds \right) \tag{55}$$

Observe that in the context of minimizing $\mathcal{E}$ over the set of approximate models $\hat{H}$ of fixed Macmillan degree d. Transfer function H has been given. s is the variable of integration. So Only $\hat{R}_i$ and $\hat{\lambda}_i$ decide the value of $\mathcal{E}$. **So $\mathcal{E}$ is a function of $2d$ variables $\hat{R}_i$ and $\hat{\lambda}_i$, $i = 1, \ldots, d$.**

Let

$$G(s) := H(s) - \hat{H}(s) = \sum_{i=1}^{n} \frac{R_i}{s - \lambda_i} - \sum_{j=1}^{d} \frac{\hat{R}_j}{s - \hat{\lambda}_j}. \tag{56}$$

So $G(s)$ is of residual form. By Lemma 4.1, we have

$$\begin{aligned} \mathcal{E} &= \|G\|_{\mathcal{H}_2}^2 \\ &= \sum_{k_1=1}^{n} R_{k_1} \left(\sum_{i_1=1}^{n} \frac{R_{i_1}}{-\lambda_{k_1} - \lambda_{i_1}} - \sum_{j_1=1}^{d} \frac{\hat{R}_{j_1}}{-\lambda_{k_1} - \hat{\lambda}_{j_1}} \right) + \sum_{k_2=1}^{d} \hat{R}_{k_2} \left(\sum_{i_2=1}^{n} \frac{R_{i_2}}{-\hat{\lambda_{k_2}} - \lambda_{i_2}} - \sum_{j_2=1}^{d} \frac{\hat{R}_{j_2}}{-\hat{\lambda_{k_2}} - \hat{\lambda}_{j_2}} \right) \end{aligned} \tag{57}$$

If $\hat{R}_i$ and $\hat{\lambda}_i$, $i = 1, \ldots, d$ can minimize $\mathcal{E}$, then (by some simple calculus)

$$\frac{\partial \mathcal{E}}{\partial \hat{R}_i} = 2\left(\hat{H}\left(-\hat{\lambda}_i\right) - H\left(-\hat{\lambda}_i\right) \right) = 0 \tag{58}$$

and

$$\frac{\partial \mathcal{E}}{\partial \hat{\lambda}_i} = 2\hat{R}_i \left(\hat{H}'\left(-\hat{\lambda}_i\right) - H'\left(-\hat{\lambda}_i\right) \right) = 0 \tag{59}$$

Thus,

$$H\left(-\hat{\lambda}_i\right) = \hat{H}\left(-\hat{\lambda}_i\right) \tag{60}$$

$$H'\left(-\hat{\lambda}_i\right) = \hat{H}'\left(-\hat{\lambda}_i\right) \tag{61}$$

is necessary conditions for ⌉ to be minimized. These two conditions are called **Meier–Luenberger conditions**.

Meier–Luenberger conditions also imply that we are actually doing the first-order moment matching and interpolation for fixed points $-\hat{\lambda}_i$,$i = 1, \ldots, d$.

But we are not able to solve the system of equations 59 and 61 directly because they are too complex. And we do not know anything about the reduced model. **So it is impossible to directly make Meier–Luenberger conditions be satisfied.** Therefore, we must use an iterative algorithm to solve this problem. Based on these, **iterative rational Krylov algorithm(IRKA)** is developed.

Algorithm 3: IRKA

Input: System (E, A, b, c), selection of k distinct interpolation points $\{\sigma_i\}_{i=1}^{k}$, tolerance ϵ

Output: Approximate model $(\hat{E}, \hat{A}, \hat{b}, \hat{c})$

while norm of σ-update $>$ tol **do**
- $V = \text{orth}\{\mathbf{x}(\sigma_1), \ldots, \mathbf{x}(\sigma_k)\}$
- $W = \text{orth}\{\mathbf{z}(\sigma_1), \ldots, \mathbf{z}(\sigma_k)\}$
- $\hat{E} = W^T E V, \hat{A} = W^T A V, \hat{\mathbf{b}} = W^T \mathbf{b}, \hat{\mathbf{c}} = V^T \mathbf{c}$
- calculate generalized eigenvalues $\left\{\hat{\lambda}_1, \ldots, \hat{\lambda}_k\right\}$ of $(\hat{A}, \hat{E})$
- set $\sigma_i := -\hat{\lambda}_i$ for all $i = 1, \ldots, k$

$V = \text{orth}\{\mathbf{x}(\sigma_1), \ldots, \mathbf{x}(\sigma_k)\}$

$W = \text{orth}\{\mathbf{z}(\sigma_1), \ldots, \mathbf{z}(\sigma_k)\}$

$\hat{E} = W^T E V, \hat{A} = W^T A V, \hat{\mathbf{b}} = W^T \mathbf{b}, \hat{\mathbf{c}} = V^T \mathbf{c}$

Consider

$$\begin{aligned} sE\mathbf{x} - A\mathbf{x} &= \mathbf{b} \\ H &= \mathbf{c}^T \mathbf{x}. \end{aligned} \tag{62}$$

We have $\mathbf{x} = (sE - A)^{-1}\mathbf{b}$ For each $s = \sigma_i$, we can get $x(\sigma_i)$. Build space V such that Range(V) = span$\{\mathbf{x}(\sigma_1), \ldots, \mathbf{x}(\sigma_k)\}$.

Also for ajoint problem

$$\begin{aligned} sE^T\mathbf{z} - A^T\mathbf{z} &= \mathbf{c} \\ H &= \mathbf{b}^T \mathbf{z}. \end{aligned} \tag{63}$$

We have $\mathbf{z} = (sE - A)^{-T}\mathbf{c}$ For each $s = \sigma_i$, we can get $z(\sigma_i)$. Build space V such that Range(W) = span$\{\mathbf{z}(\sigma_1), \ldots, \mathbf{z}(\sigma_k)\}$.

Next, we compute $\hat{E} = W^T E V, \hat{A} = W^T A V, \hat{\mathbf{b}} = W^T \mathbf{b}, \hat{\mathbf{c}} = V^T \mathbf{c}$.

Then, we get a reduced model

$$\begin{aligned} s\hat{E}\hat{\mathbf{x}} - \hat{A}\hat{\mathbf{x}} &= \hat{\mathbf{b}} \\ \hat{H} &= \hat{\mathbf{c}}^T \hat{\mathbf{x}}. \end{aligned} \tag{64}$$

Theorem 4.2. *If σ_i is not an eigenvalue of A and $\hat{A}$, then*

$$H(\sigma_i) = \hat{H}(\sigma_i) \tag{65}$$

$$H'(\sigma_i) = \hat{H}'(\sigma_i) \tag{66}$$

The proof of this theorem can be found on page 25 of the slides. So in every iteration, $\sigma_1, \ldots, \sigma_k$ can satisfy Meier–Luenberger conditions. In the next iteration, we use $\sigma_i = -\hat{\lambda}_i$ where $\left\{\hat{\lambda}_1, \ldots, \hat{\lambda}_k\right\}$ are generalized eigenvalues of $(\hat{A}, \hat{E})$.

The convergence analysis can be found in [1].

4.2 Tailor IRKA to quadratic models

Consider

$$\begin{aligned} s^2 M\mathbf{x} + sD\mathbf{x} + K\mathbf{x} &= \mathbf{b} \\ H &= \mathbf{c}^T \mathbf{x}. \end{aligned} \tag{67}$$

We have $\mathbf{x} = (s^2 M + sD + K)^{-1}\mathbf{b}$ For each $s = \sigma_i$, we can get $\mathbf{x}(\sigma_i)$. Build space V such that Range(V) = span$\{\mathbf{x}(\sigma_1), \ldots, \mathbf{x}(\sigma_k)\}$.

Also for ajoint problem

$$\begin{aligned} s^2 M^T\mathbf{z} + sD^T\mathbf{z} + K^T\mathbf{z} &= \mathbf{c} \\ H &= \mathbf{b}^T \mathbf{z}. \end{aligned} \tag{68}$$

We have $\mathbf{z} = (s^2M + sD + K)^{-T}\mathbf{c}$ For each $s = \sigma_i$, we can get $\mathbf{z}(\sigma_i)$. Build space V such that Range(W) = span$\{\mathbf{z}(\sigma_1), \ldots, \mathbf{z}(\sigma_k)\}$.

Next, we compute $\hat{M} = W^TMV, \hat{D} = W^TDV, \hat{K} = W^TKV, \hat{\mathbf{b}} = W^T\mathbf{b}, \hat{\mathbf{c}} = V^T\mathbf{c}$.

Then, we get a reduced model

$$\begin{aligned} s^2\hat{M}\hat{\mathbf{x}} + s\hat{D}\hat{\mathbf{x}} + \hat{K}\hat{\mathbf{x}} &= \hat{\mathbf{b}} \\ \hat{H} &= \hat{\mathbf{c}}^T\hat{\mathbf{x}}. \end{aligned} \tag{69}$$

Using Example 1.2, we can calculate generalized eigenvalues $\left\{\hat{\lambda}_1, \ldots, \hat{\lambda}_k\right\}$ of $(\hat{A}, \hat{E})$ where $\hat{E} = \begin{bmatrix} \hat{M} & 0 \\ \hat{D} & I \end{bmatrix}$, $A = \begin{bmatrix} 0 & I \\ -\hat{K} & 0 \end{bmatrix}$. Set $\sigma_i = \hat{\lambda}_i$. And do all these things iteratively. This is summarized in Algorithm 4.

Algorithm 4: IRKA for quadratic model

Input: Quadratic model of a mechanical system (M, D, K, b, c), selection of k distinct interpolation points $\{\sigma_i\}_{i=1}^k$, tolerance ϵ

Output: Approximate model $(\hat{M}, \hat{D}, \hat{K}, \hat{b}, \hat{c})$

while norm of σ-update $>$ tol **do**
- $V = \text{orth}\{\mathbf{x}(\sigma_1), \ldots, \mathbf{x}(\sigma_k)\}$
- $W = \text{orth}\{\mathbf{z}(\sigma_1), \ldots, \mathbf{z}(\sigma_k)\}$
- $\hat{M} = W^TMV, \hat{D} = W^TDV, \hat{K} = W^TKV, \hat{\mathbf{b}} = W^T\mathbf{b}, \hat{\mathbf{c}} = V^T\mathbf{c}$
- $\hat{E} = \begin{bmatrix} \hat{M} & 0 \\ \hat{D} & I \end{bmatrix}$, $A = \begin{bmatrix} 0 & I \\ -\hat{K} & 0 \end{bmatrix}$
- calculate generalized eigenvalues $\left\{\hat{\lambda}_1, \ldots, \hat{\lambda}_k\right\}$ of $(\hat{A}, \hat{E})$
- set $\sigma_i := -\hat{\sigma}_i'$ for all $i = 1, \ldots, k$

$V = \text{orth}\{\mathbf{x}(\sigma_1), \ldots, \mathbf{x}(\sigma_k)\}$

$W = \text{orth}\{\mathbf{z}(\sigma_1), \ldots, \mathbf{z}(\sigma_k)\}$

$\hat{M} = W^TMV, \hat{D} = W^TDV, \hat{K} = W^TKV, \hat{\mathbf{b}} = W^T\mathbf{b}, \hat{\mathbf{c}} = V^T\mathbf{c}$

The implementation of Algorithm 4 is in `qirka.m`.

4.3 Spiral inductor

(`application_irka_peec.m` contains the code for this part.)

From Figure 15 to 17, we know that the reduced model of order 30 given by SADPA is good but does not perform very well in the high-frequency domain. But IRKA can give reduced model which has 'skin-effect' but with much lower order.

From Figure 23 and 24, we know that the reduced model of order 4 given by IRKA can already give extremely good performance.

$\frac{|H(s)-\hat{H(s)}|_\infty}{|H(s)|_\infty}$:

- Reduced model of order 3: 0.0012
- Reduced model of order 4: 2.1267e-04
- Reduced model of order 5: 1.4714e-05

The results given by IRKA are much better than the results given by SADPA. IRKA try to minimize the $\mathcal{E}$. So $\hat{H(s)}$ is very close to $H(s)$.

Figure 25 shows the resistance and inductance of these three reduced model. We can see that although reduced model of order 4 performs very well, it is still not totally able to capture the 'skin effect'.

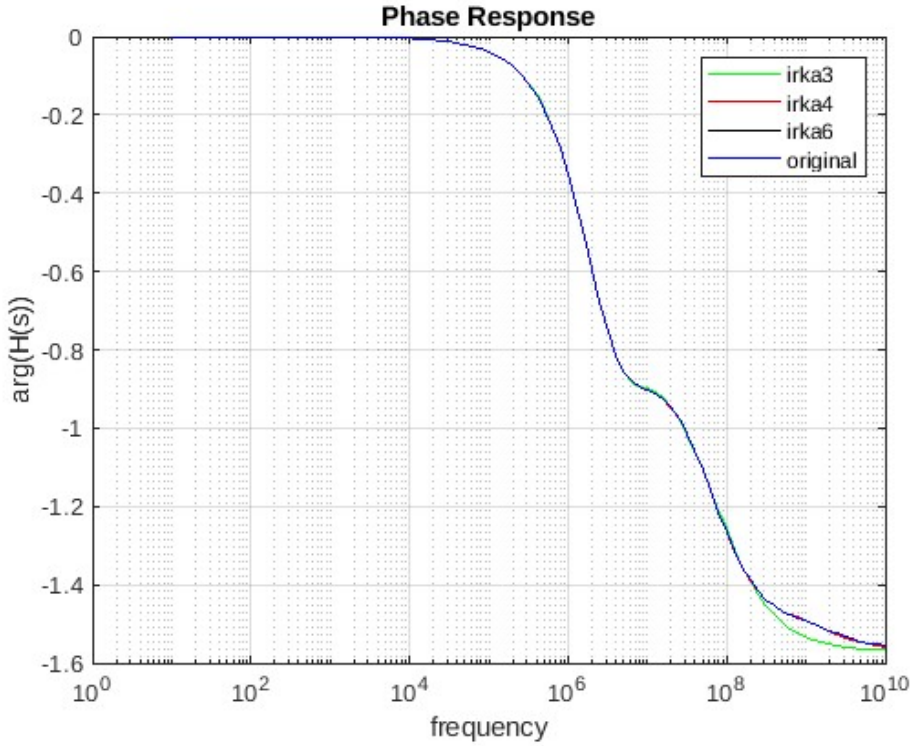

Figure 23: The amplitude response of the original model and reduced models given by IRKA.

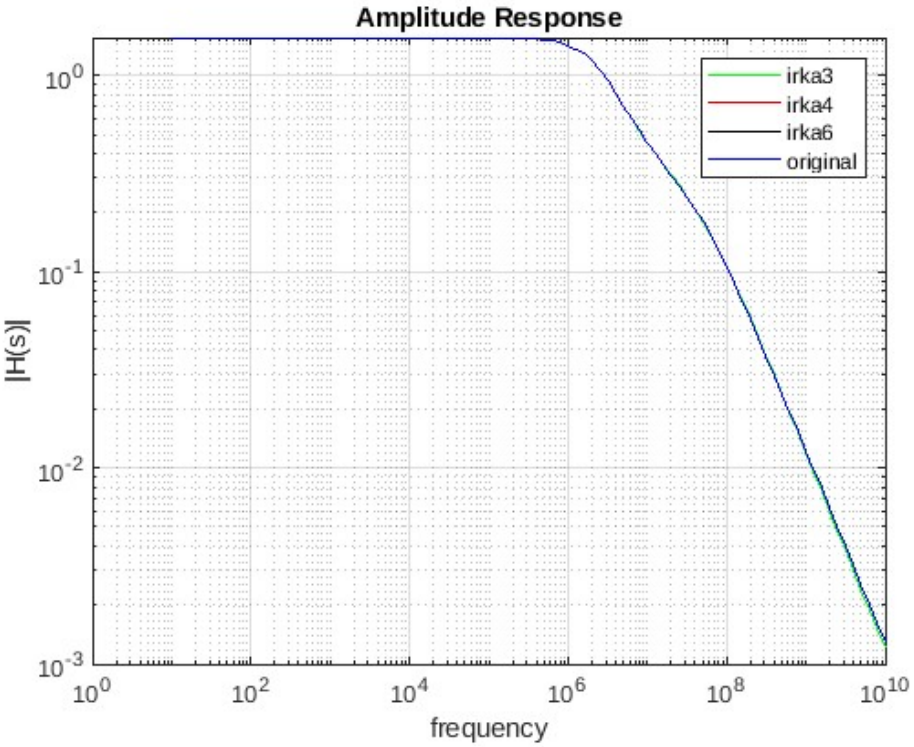

Figure 24: The phase response of the original model and reduced models given by IRKA.

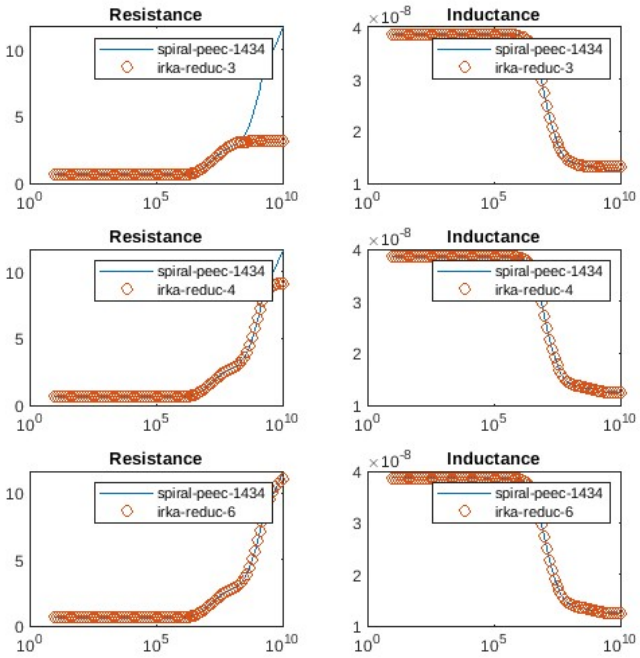

Figure 25: Resistance and inductance given by IRKA.

4.4 Gyro

(`application_irka_gyro.m` contains the code for this part.)

Since this is a quadratic model, I choose even number of initial points to do Algorithm 4.

From Figure 26, we know that 4 initial points is enough for IRKA to capture the second-order behaviour of this system. The only unsatisfactory part of reduced model of order 4 is that its phase response does not have a small decrease when some $f \in [10^4, 10^{4.1}]$ like the original system.

The results given by IRKA is much better than DPA. QDPA even cannot converge for Gyro model.

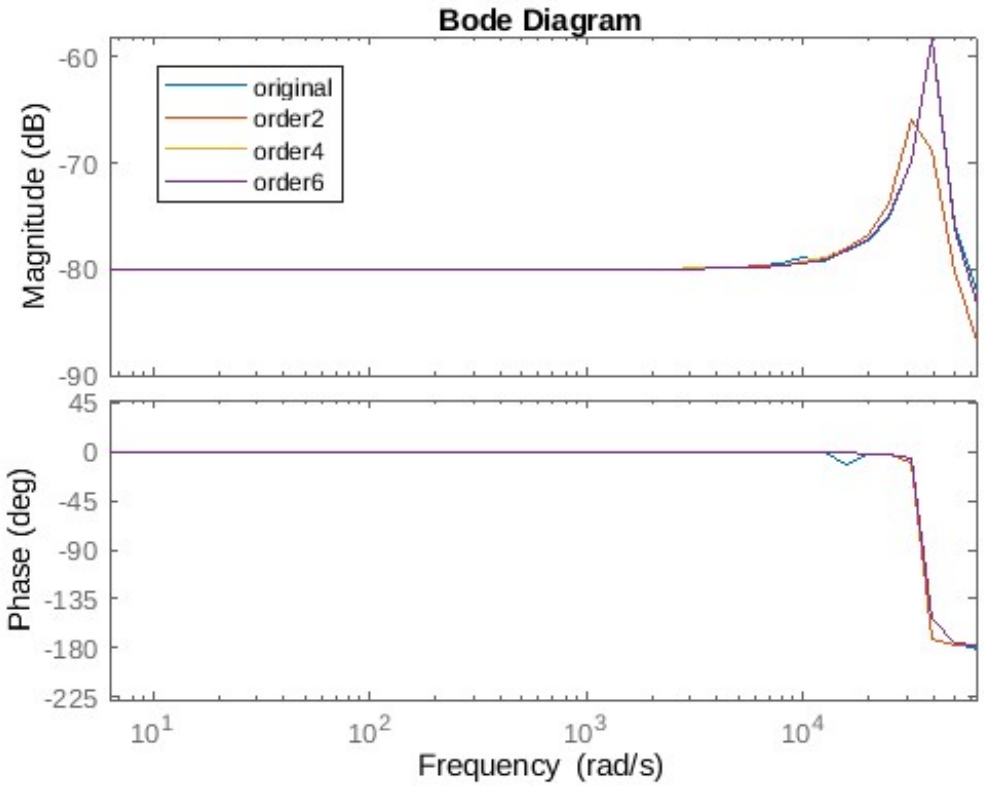

Figure 26: Bode diagram of the original model and the reduced model given by IRKA.

4.5 ISS

(`application_irka_iss.m` contains the code for this part.)

Although IRKA has very good performance in the previous two cases, its performance for ISS model is not good enough, see Figure 27. The previous two applications do not have high oscillation in Bode diagram, so they do not need as many initial points as ISS. IRKA requires the initial points are not close to eigenvalues of A and $\hat{A}$, otherwise the matrices will be close to singular or badly scaled and results are inaccurate. However, in this case, ISS model has too many poles, which make selection of initial points to be a little bit difficult when the number of the initial points is large. And using SADPA is more convenient and accurate in this case.

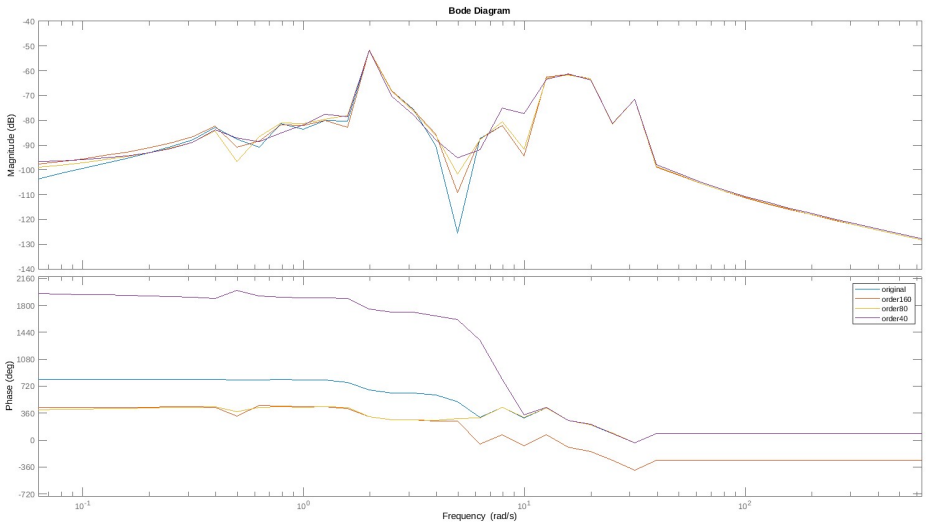

Figure 27: Bode diagram of the original model and the reduced model given by IRKA.

Section 5 Greedy Rational Krylov Methods

The disadvantage of IRKA:

- IRKA needs many iterations to converge and many LU factorizations
- We have to decide the number of the initial points (model order) before the iterations.

This motivates **Greedy Rational Krylov algorithm (GRKA)**.
The Greedy Rational Krylov method finds $\sigma_1, \ldots, \sigma_k$ such that the residual norm

$$\|(sE - A)V\hat{\mathbf{x}} - \mathbf{b}\|_2 \tag{70}$$

is minimized with $V = \{\mathbf{x}(\sigma_1), \ldots, \mathbf{x}(\sigma_k)\}$.

Algorithm 5: Greedy Rational Krylov Algorithm

Input: System (E, A, b, c), one initial point σ_1, tolerance ϵ
Output: Approximate model $(\hat{E}, \hat{A}, \hat{b}, \hat{c})$

for $k = 1, 2, \ldots$ **do**
  Compute $\mathbf{x}(\sigma_k)$ and $\mathbf{z}(\sigma_k)$
  $V = \text{orth}\{\mathbf{x}(\sigma_1), \ldots, \mathbf{x}(\sigma_k)\}$
  $W = \text{orth}\{\mathbf{z}(\sigma_1), \ldots, \mathbf{z}(\sigma_k)\}$
  $\hat{E} = W^T EV, \hat{A} = W^T AV, \hat{\mathbf{b}} = W^T\mathbf{b}, \hat{\mathbf{c}} = V^T\mathbf{c}$ (only need to compute the new row and the new column.)
  $\sigma_{k+1} = \text{argmax}_s \|(s\hat{E} - \hat{A})\hat{\mathbf{x}} - \hat{\mathbf{b}}\|_2$.
  if $\|(\sigma_{k+1}\hat{E} - \hat{A})\hat{\mathbf{x}} - \hat{\mathbf{b}}\|_2 / \|\hat{b}\| < \epsilon$ **then**
    break;
$V = \text{orth}\{\mathbf{x}(\sigma_1), \ldots, \mathbf{x}(\sigma_k)\}$
$W = \text{orth}\{\mathbf{z}(\sigma_1), \ldots, \mathbf{z}(\sigma_k)\}$
$\hat{E} = W^T EV, \hat{A} = W^T AV, \hat{\mathbf{b}} = W^T\mathbf{b}, \hat{\mathbf{c}} = V^T\mathbf{c}$

5.1 Tailor GRKA to quadratic models

Now our residual norm is

$$\|(s^2M + sD + K)V\hat{\mathbf{x}} - \mathbf{b}\|_2 \tag{71}$$

The other part of the algorithm can use the same trick as the one in the previous section.

Algorithm 6: GRKA for quadratic models

Input: Quadratic model of a mechanical system (M, D, K, b, c), one initial point σ_1, tolerance ϵ

Output: Approximate model $(\hat{M}, \hat{D}, \hat{K}, \hat{b}, \hat{c})$

for $k = 1, 2, \ldots$ **do**
- Compute $\mathbf{x}(\sigma_k)$ and $\mathbf{z}(\sigma_k)$
- $V = \text{orth}\{\mathbf{x}(\sigma_1), \ldots, \mathbf{x}(\sigma_k)\}$
- $W = \text{orth}\{\mathbf{z}(\sigma_1), \ldots, \mathbf{z}(\sigma_k)\}$
- $\hat{M} = W^T M V, \hat{D} = W^T D V, \hat{K} = W^T K V, \hat{\mathbf{b}} = W^T \mathbf{b}, \hat{\mathbf{c}} = V^T \mathbf{c}$ (only need to compute the new row and the new column.)
- $\sigma_{k+1} = \text{argmax}_s \|(s^2 M + sD + K) V \hat{\mathbf{x}} - \mathbf{b}\|_2$.
- **if** $\|(\sigma_{k+1}^2 M + \sigma_{k+1} D + K) V \hat{\mathbf{x}} - \mathbf{b}\|_2 / \|\hat{b}\| < \epsilon$ **then**
  - break;

$V = \text{orth}\{\mathbf{x}(\sigma_1), \ldots, \mathbf{x}(\sigma_k)\}$

$W = \text{orth}\{\mathbf{z}(\sigma_1), \ldots, \mathbf{z}(\sigma_k)\}$

$\hat{M} = W^T M V, \hat{D} = W^T D V, \hat{K} = W^T K V, \hat{\mathbf{b}} = W^T \mathbf{b}, \hat{\mathbf{c}} = V^T \mathbf{c}$

We can use $\mathcal{H}_2$-norm or $\mathcal{H}_\infty$-norm to measure the performance of transfer functions in later discussions, but it is just the same as the one in the previous section.

5.2 Spiral inductor

(`application_grka_peec.m` contains the code for this part.)

We need to choose sigma1, smin, smax, scount, tol for function `grka`. I used the following combinations of parameters: (scount=200, tol=1e-2)

- sigma1 = -1e7; smin=-1e10; smax=-1e5;
- sigma1 = -1e-6; smin=-1e10; smax=0;
- sigma1 = -1; smin=-1e5; smax=0;

The first two combinations return a model of order higher than 200. Although the performance of this two models is good, the order is too high and we prefer lower order model. The last combination only needs 3 iterations, but it does not capture the 'skin effect'.

The result of GRKA depends on the initial choices, and it is less likely to give the best reduced model.

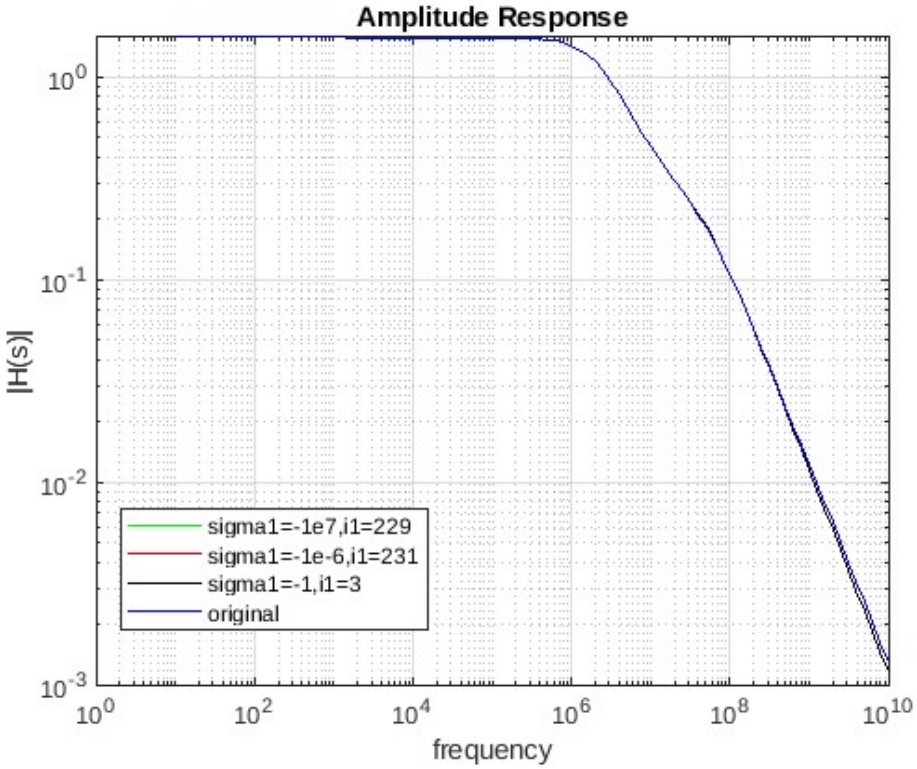

Figure 28: The amplitude response of the original model and reduced models given by GRKA.

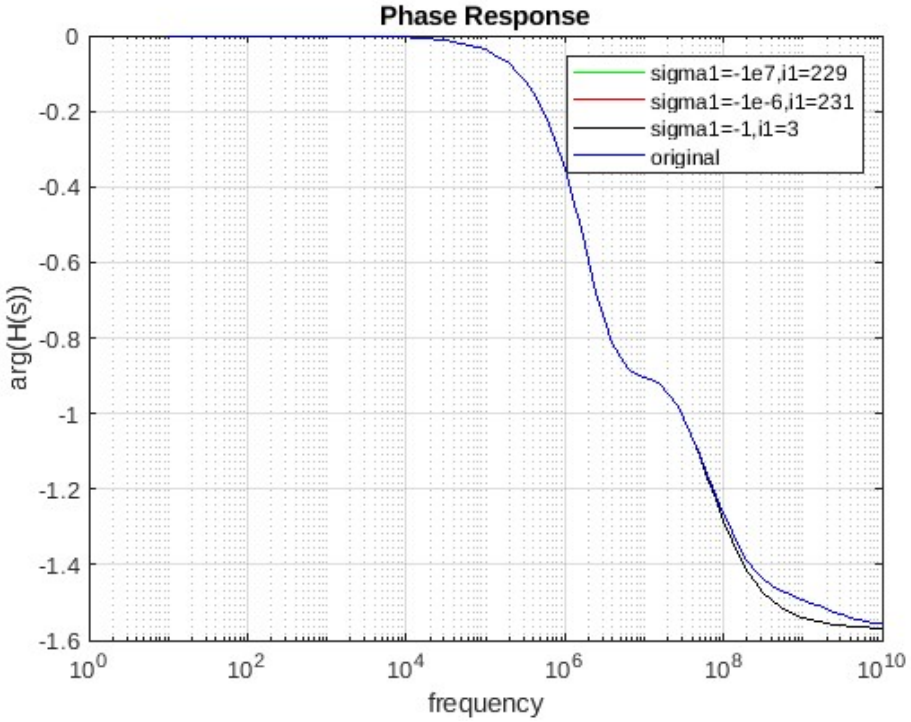

Figure 29: The phase response of the original model and reduced models given by GRKA.

5.3 Gyro

(`application_grka_gyro.m` contains the code for this part.)

I used the following combinations of parameters: (scount=200, tol=1e-2)

- sigma1 = -1; smin=-1e4; smax=0;
- sigma1 = -1e4; smin=-1e6; smax=0;
- sigma1 = -1e7; smin=-1e10; smax=0;

The first combination needs 3 iterations, the second needs 23 and the third cannot converge and show some warning: "Matrix is close to singular or badly scaled. Results may be inaccurate."

Although the reduced model of order 3 is slight different from the original model in bode diagram, it is enough to use most of the time. GRKA allows us quickly get the reduced models for each input-output pair.

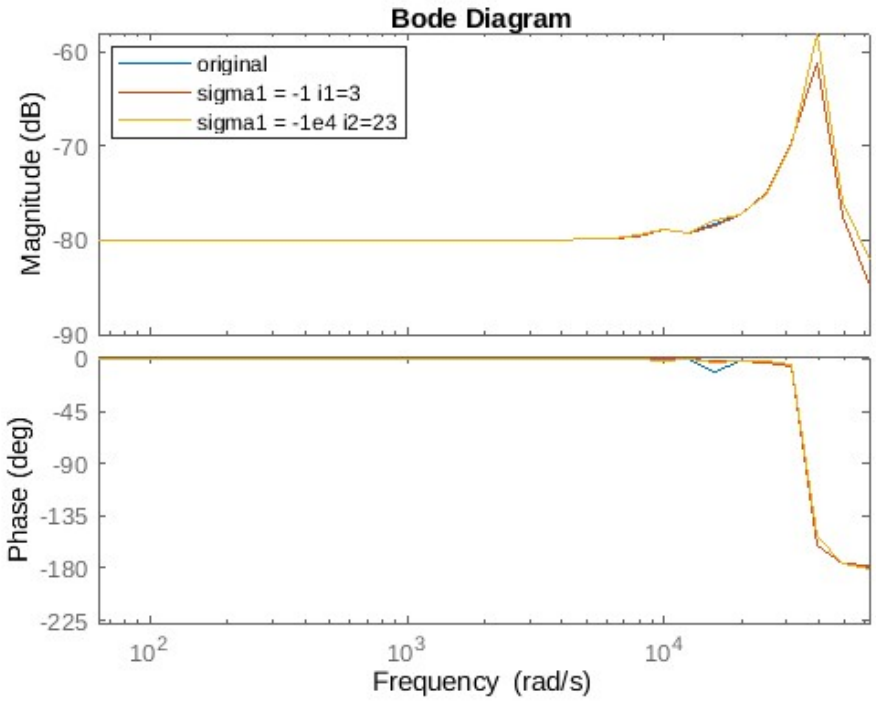

Figure 30: Bode diagram of the original model and the reduced model given by GRKA.

5.4 ISS

(`application_grka_iss.m` contains the code for this part.)

I tried many combinations of parameters, but none of they can return satisfying result (The order of the result model is too low). Decreasing the tol can slightly alleviate this problem, but the algorithm cannot converge when tol is too low because some matrices become singular.

The results of GRKA are not useful.

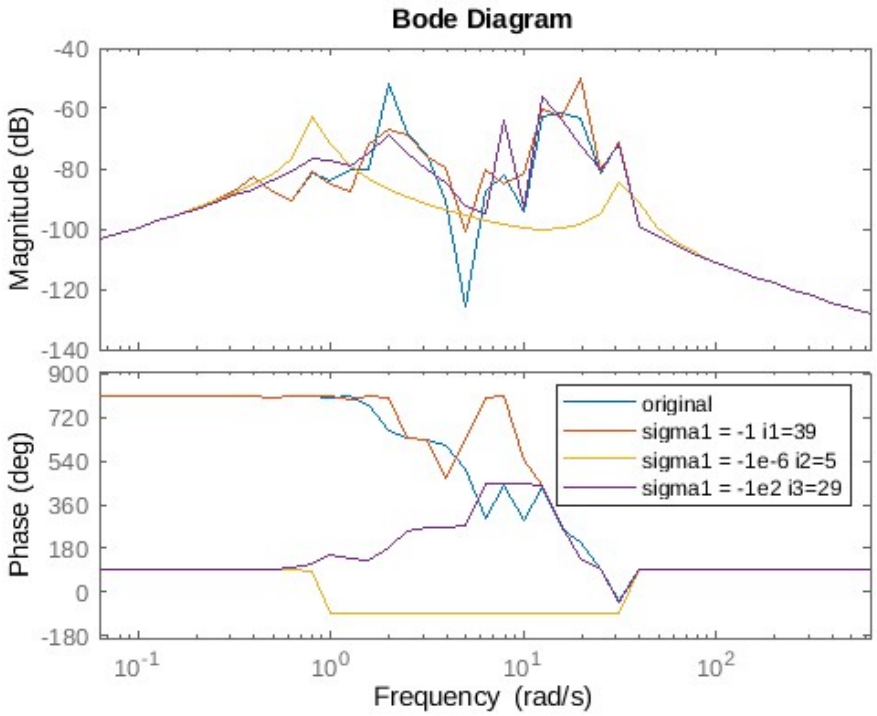

Figure 31: Bode diagram of the original model and the reduced model given by GRKA when tol=1e-5.

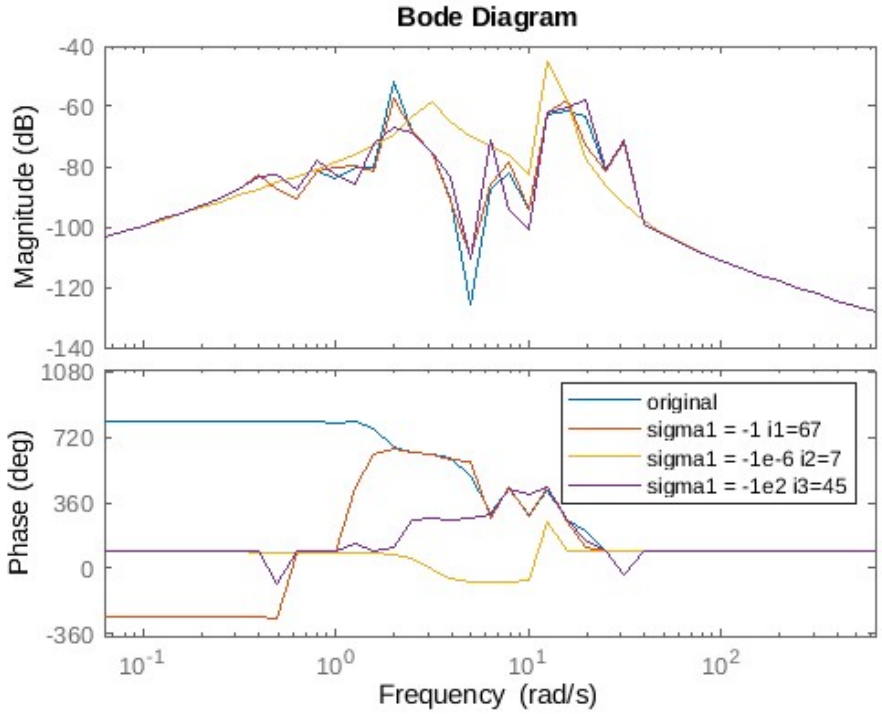

Figure 32: Bode diagram of the original model and the reduced model given by GRKA when tol=1e-11.

References

[1] Flagg, G., Beattie, C., and Gugercin, S. Convergence of the iterative rational krylov algorithm. Systems & Control Letters 61, 6 (2012), 688–691.

[2] Li, J.-R., and Kamon, M. Peec model of a spiral inductor generated by fasthenry. In Dimension Reduction of Large-Scale Systems: Proceedings of a Workshop held in Oberwolfach, Germany, October 19–25, 2003 (2005), Springer, pp. 373–377.

[3] Odabasioglu, A., Celik, M., and Pileggi, L. Practical considerations for passive reduction of rlc circuits. In 1999 IEEE/ACM International Conference on Computer-Aided Design. Digest of Technical Papers (Cat. No.99CH37051) (1999), pp. 214–219.

[4] Rommes, J., and Martins, N. Efficient computation of transfer function dominant poles using subspace acceleration. IEEE Transactions on Power Systems 21, 3 (2006), 1218–1226.