

**Problem [1]. Approximation of the image: saturn.png.**

1. The singular values on a logarithmic scale are:

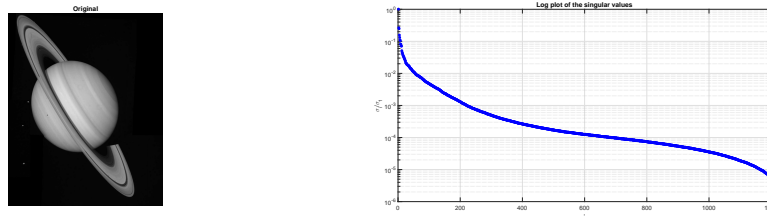


Figure 1: Original picture and the Singular values decay.

2. In order to compute the optimal rank  $r$  approximation we need to compute the Singular Value Decomposition of the image matrix. the approximation errors were computed using the matlab comand `norm(Z-Zr,2)`. The maximum singular value is  $\sigma_1 = 369.0955$ . The ranks of the approximants, the approximation errors (2-norm) and the compression ratios for each desired relative error are provided in the next table below:

Relative % error	Approximation rank	Approximation error	R
10	7	30.17	0.0105
5	12	16.79	0.0180
2	27	7.310	0.0405

**Problem [2]. Ellipse fitting using LS and TLS.**

- (a) Least square (LS) and total least square (TLS) solution:

```
% (LS) Least square approach A1*w=b1      w=Api*b;
A=[xn.^2.' yn.^2.'];                      % (TLS) Total Least Square A2*q=0
b=ones(N,1);                               At=[xn.^2.' yn.^2.' ones(N,1)];
[U,S,V]=svd(A,0);                          [Ut,St,Vt]=svd(At,0);
Api=V*S^(-1)*U';                           wt=Vt(:,3)/Vt(3,3);
```

$N = 63$ points	Error	Coefs	Std of Noise
LS	0.9867	$(\alpha, \beta) = (0.9480, 0.2470)$	0.08
TLS	0.7647	$(\bar{\alpha}, \bar{\beta}, \bar{\gamma}) = (0.9656, 0.2458, -1)$	0.08

- (b) Next is the plot with the two approximants.

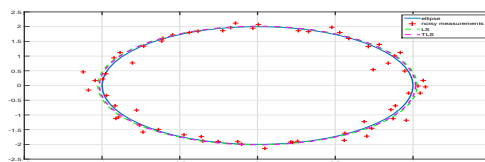


Figure 2:

### Problem [3]. Power Method for computing the PageRank.

In this problem, we will use the Power Method for ranking MPI-related webpages.

(a) Here, a database of webpages starting from an initial URL has to be constructed. For this task, we will use the m-file `surfer.m` (which is attached to the collection of m files provided in this course). Simply type the following command to create database:

```
[U,G] = surfer('http://www.mpi-magdeburg.mpg.edu',\,1000);
```

The code starts at the specified URL (`http://www.mpi-magdeburg.mpg.edu` in this case), and surfs the Web until it has visited  $n = 1000$  pages.

The `surfer.m` function will return an  $n \times 1$  cell array `U` of URLs which stores the visited webpages. The  $n \times n$  sparse connectivity matrix `G` shows how the webpages are linked to each other. Below, we can visualize the structure of this connectivity matrix (using the command `spy(G)`).

For  $n = 1000$ , the processing time for constructing the data base was in our case around 2 hours. So it is advisable to start with values of  $n$  around 20 to 40 first. Alternatively, one can directly load the `G` and `U` matrices (corresponding to the  $n = 1000$  case) by using the command

```
load('dataMPIsurfer1000.mat');
```

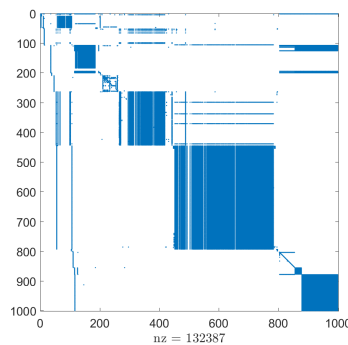


Figure 3: `spy(G)`

(b) Next, we construct the transition matrix  $A$  by using the following code sequence

```
n = size(G,2);
p = 0.85; % The probability we used in the class
delta = (1-p)/n;
c = sum(G,1);
k = find(c~=0);
D = sparse(k,k,1./c(k),n,n);
e = ones(n,1);
z = ((1-p)*(c~=0) + (c==0))/n;
A = p*G*D + e*z; % This corresponds to p*M + (1-p)/n*e*e' discussed in class
```

(c) The PageRank corresponds to entries of the left eigenvector  $v$  corresponding to the dominant eigenvalue 1 of  $A$ . The implementation of the Power Method can be found in the m file `powermat.iter.mat`. We use an initial vector  $v^{(0)} = \text{ones}(n,1)/n$ , corresponding to initially equal probabilities and run the iteration with tolerance value  $10^{-4}$ . The iteration converged after about 20 steps.

```

x0 = ones(n,1)/n; tol = 10^(-4);

% x = right eigenvector corresponding to lambda
[x,stp,lambda]=powermat_iter(A,x0,tol);

% the dominant eigenvalue of A equal to 1
display('The estimated largest eigenvalue is:')
lambda

% check to see if this (lambda,x) is an eig/eigv right pair
norm(A*x-lambda*x)

% check to see if x is indeed a probability vector
sum(x)

```

The estimated value for  $\lambda$  was, in exact arithmetic, equal to 0.999955099977393. Then, we can compute  $\|Ax - \lambda\| = 2.11 \cdot 10^{-5}$ . Note that the  $x$  vector is a probability vector, i.e. the sum of its entries is 1. The top five webpages in this PageRanking problem and the probabilities of being visited are listed below:

The first website:	The probability to be visited:
ans =	ans =
<a href="http://alpha.mixi.co.jp">http://alpha.mixi.co.jp</a>	0.0119
The second website:	The probability to be visited:
ans =	ans =
<a href="http://pr.mixi.co.jp">http://pr.mixi.co.jp</a>	0.0119
The third website:	The probability to be visited:
ans =	ans =
<a href="http://ogp.me/ns/fb#">http://ogp.me/ns/fb#</a>	0.0075
The fourth website:	The probability to be visited:
ans =	ans =
<a href="http://ogp.me/ns#">http://ogp.me/ns#</a>	0.0065
The fifth website:	The probability to be visited:
ans =	ans =
<a href="http://gmpg.org/xfn/11">http://gmpg.org/xfn/11</a>	0.0060

**Problem [4]. Power Method for computing the PageRank (Problem 4).**

(a) We are given a matrix  $\mathbf{A} \in \mathbb{R}^{n \times n}$  with eigenvalues  $\lambda_1, \dots, \lambda_n$ . Amongst these, choose the first  $k$  ones:  $\lambda_1, \dots, \lambda_k$  (and call them the *preferred* eigenvalues). Partition the eigenvector matrix as follows:  $\mathbf{V} = [\mathbf{V}_1 \ \mathbf{V}_2]$  where  $\mathbf{V}_1^{(j)} \in \mathbb{C}^n$  corresponds to  $\lambda_j$ , i.e.  $\mathbf{A}\mathbf{V}_1^{(j)} = \lambda_j \mathbf{V}_1^{(j)}$  for  $j \in \{1, 2, \dots, k\}$ . Additionally,  $\mathbf{V}_2^{(\ell)} \in \mathbb{C}^n$  corresponds to  $\lambda_{k+\ell}$ , i.e.  $\mathbf{A}\mathbf{V}_2^{(\ell)} = \lambda_{k+\ell} \mathbf{V}_2^{(\ell)}$ , for  $\ell \in \{1, 2, \dots, n-k\}$ .

Let  $\mathbf{W}_1 \in \mathbb{C}^{n \times (n-k)}$  so that its columns represent an orthonormal basis for the orthogonal complement of matrix  $\mathbf{V}_1$ , i.e.  $\mathbf{W}_1^* \mathbf{V}_1 = \mathbf{0}$  and  $\mathbf{W}_1^* \mathbf{W}_1 = \mathbf{I}_{n-k}$ .

We will show that the compressed matrix  $\mathbf{M} = \mathbf{W}_1^* \mathbf{A} \mathbf{W}_1 \in \mathbb{C}^{(n-k) \times (n-k)}$  has eigenvalues  $\lambda_{k+1}, \dots, \lambda_n$  which correspond to eigenvectors  $\mathbf{W}_1^* \mathbf{V}_2^{(1)}, \dots, \mathbf{W}_1^* \mathbf{V}_2^{(n-k)}$  respectively.

In order to do that, we need to show that for all  $\ell \in \{1, 2, \dots, n-k\}$ , the following holds

$$\mathbf{M} \mathbf{W}_1^* \mathbf{V}_2^{(\ell)} = \lambda_{k+\ell} \mathbf{W}_1^* \mathbf{V}_2^{(\ell)} \Leftrightarrow \mathbf{W}_1^* \mathbf{A} \mathbf{W}_1 \mathbf{W}_1^* \mathbf{V}_2^{(\ell)} = \lambda_{k+\ell} \mathbf{W}_1^* \mathbf{V}_2^{(\ell)} \quad (1)$$

Let  $\mathbf{P} = \mathbf{I}_n - \mathbf{W}_1 \mathbf{W}_1^* \in \mathbb{C}^{n \times n}$ . Note that  $\mathbf{P}$  is a *projection*, i.e.  $\mathbf{P}^2 = \mathbf{P}$ . By multiplying  $\mathbf{P} = \mathbf{I}_n - \mathbf{W}_1 \mathbf{W}_1^*$  to the left with  $\mathbf{W}_1^*$  and using that  $\mathbf{W}_1^* \mathbf{W}_1 = \mathbf{I}_{n-k}$ , it hence follows that  $\mathbf{W}_1^* \mathbf{P} = \mathbf{0}$ . Furthermore, by multiplying the same relation to the right with  $\mathbf{V}_1$  and using that  $\mathbf{W}_1^* \mathbf{V}_1 = \mathbf{0}$ , it hence follows that  $\mathbf{P} \mathbf{V}_1 = \mathbf{V}_1$ .

Rewriting (1) by substituting  $\mathbf{W}_1 \mathbf{W}_1^*$  in the left term with  $\mathbf{I}_n - \mathbf{P}$ , we need to show that

$$\mathbf{W}_1^* \mathbf{A} (\mathbf{I}_n - \mathbf{P}) \mathbf{V}_2^{(\ell)} = \lambda_{k+\ell} \mathbf{W}_1^* \mathbf{V}_2^{(\ell)} \Leftrightarrow \mathbf{W}_1^* \underbrace{(\mathbf{A} \mathbf{V}_2^{(\ell)} - \lambda_{k+\ell} \mathbf{V}_2^{(\ell)})}_{\mathbf{0}_n} = \mathbf{W}_1^* \mathbf{A} \mathbf{P} \mathbf{V}_2^{(\ell)} \quad (2)$$

Since  $\mathbf{V}_1$  is a block eigenvector matrix of  $\mathbf{A}$ , we can write  $\mathbf{A} \mathbf{V}_1 = \mathbf{V}_1 \mathbf{\Lambda}_1$ , with  $\mathbf{\Lambda}_1 = \text{diag}(\lambda_1, \dots, \lambda_k)$ . By multiplying this equality with  $\mathbf{W}_1^*$  to the left, and using that  $\mathbf{W}_1^* \mathbf{V}_1 = \mathbf{0}$  and that  $\mathbf{P} \mathbf{V}_1 = \mathbf{V}_1$ , we have that

$$\mathbf{A} \mathbf{V}_1 = \mathbf{V}_1 \mathbf{\Lambda}_1 \Rightarrow \mathbf{W}_1^* \mathbf{A} \mathbf{V}_1 = \mathbf{W}_1^* \mathbf{V}_1 \mathbf{\Lambda}_1 = \mathbf{0} \Rightarrow \mathbf{W}_1^* \mathbf{A} \mathbf{P} \mathbf{V}_1 = \mathbf{0} \quad (3)$$

Similarly, one can show that  $\mathbf{W}_1^* \mathbf{A} \mathbf{P} \mathbf{V}_2 = \mathbf{0}$  and hence  $\mathbf{W}_1^* \mathbf{A} \mathbf{P} \mathbf{V}_2^{(\ell)} = \mathbf{0}$ ,  $\forall 1 \leq \ell \leq n-k$ . Finally, from (2) it follows that the statement is proven (the matrix  $\mathbf{M}$  has eigenvalues  $\lambda_{k+1}, \dots, \lambda_n$  which correspond to the eigenvectors  $\mathbf{W}_1^* \mathbf{V}_2^{(1)}, \dots, \mathbf{W}_1^* \mathbf{V}_2^{(n-k)}$  respectively). ■

(b) We will use the result in part (a) to compute the second largest eigenvalue of the Google matrix (constructed in Problem 3, part (b)) by means of the power iteration method.

```
% V is the eigenvector matrix and Lam is the eigenvalue matrix
[V,Lam] = eig(A);
```

```
% the 1 eigenvalue corresponds to the first column of V
V1 = V(:,1); V2 = V(:,2:n);
```

```
% the preferred eigenvalues (and the one which will be removed)
display('The largest eigenvalue is:');
Lam(1,1)
```

```
[W1f,S1,Z1] = svd(V1);
```

```
% select the last n-k columns of the nxn left singular vector matrix W1f
W1 = W1f(:,(k+1):n);
```

```
% check to see if these values are indeed 0
norm(W1'*V1);
norm(W1'*W1-eye(n-1));
```

```

% eliminate the largest eigenvalue of matrix A (the matrix M is (n-1)x(n-1))
M = W1'*A*W1;

x0 = ones(n-1,1)/n; tol = 10^(-4);
[x,stp,lambda2]=powermat_iter(M,x0,tol);

% display the second largest eigenvalue
display('The estimated second largest eigenvalue is:')
lambda2

% check if the pair (lambda2,x) is indeed an eigenvalue/eigenvector pair
norm(M*x-lambda2*x)

```

The second largest eigenvalue computed by means of the above sequence of code is, in exact arithmetic, equal to  $\lambda_2 = 0.848708826007528$ . If we decrease the tolerance value from  $10^{-4}$  (the one that determines the number of steps performed in the power iteration function), then the value of  $\lambda_2$  will approach  $p = 0.85$ .

### Problem [5]. Clamped Beam.

Download the file beam.m which contains the system matrices. This clamped beam problem is a Linear Time Invariant (LTI) dynamical system with dimension 348. The purpose of this exercise is for someone to get familiar with such a linear model by computing some fundamental quantities. Lets start by using the command `load('beam.mat')`.

(a) In order to compute the system poles we need to solve the following eigenvalue problem.

```
poles=eig(A);
```

Next is the plot with the poles in the complex plane. As long as we have a real model,

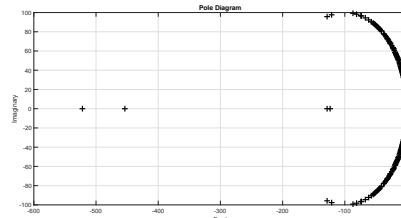


Figure 4: Poles for the beam model. Poles appeared in the left half plane, so, this indicates the stability.

the underneath transfer function it is real. So, the poles will be the roots of a polynomial with real coefficients. That proves the conjugates pairs.

(b) Next is the transfer function.

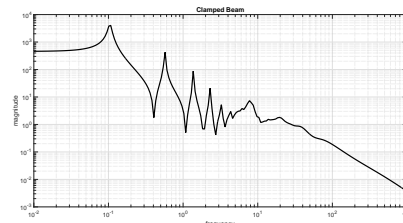


Figure 5: Transfer function for the beam model. 200 frequency values between  $[10^{-2}, 10^3]$ .

(c) Next is the impulse response evaluated in 500 points inside  $[0, 500]$ .

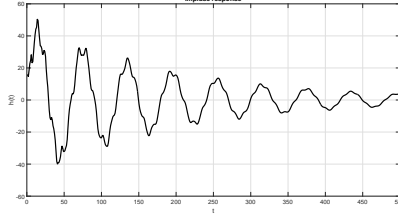


Figure 6: Impulse response for the beam model inside  $[0, 500]$ .

- (d) Next is the output after we approximate  $\dot{x}(t)$  with the Backward Euler scheme and use as an input  $u = \text{ones}(1, 2000)$  with  $u(1:1000) = -1$ .

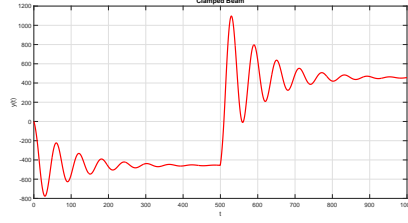


Figure 7: Output  $y(t)$  assuming  $\mathbf{x}(0) = 0$ .

#### Problem [6]. (Optimal $\mathcal{H}_2$ model reduction for Clamped Beam)

- (a) Next is the magnitude of the transfer function evaluated in  $[10^{-2}, 10^2]$  for the beam superimposed with the IRKA approximant.

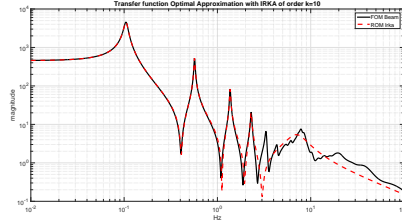


Figure 8: IRKA approximant with order  $k=10$ .

- (b) Impulse responses for the reduced model and the FOM.

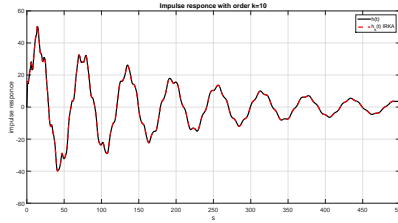


Figure 9: IRKA Impulse response with order  $k=10$ .

- (c) Unfortunately, IRKA does not always converge. If you reach the MAXITER which is 100 it does not mean that the method has converged.

**Problem [7]. (Reduced models from measurements (Loewner method))**

(a) One good way to partition the data is the following:

```
mu=s(1:2:end);v=H(1:2:end);
la=s(2:2:end);w=H(2:2:end);
```

To ensure that our reduce models are real we need to include the complex conjugate values of the measurements. (for real symmetry we assume  $\bar{H}(s) = H(\bar{s})$ ).

```
% get a real model (assumed real symmetry)
mu=s(1:2:end);muc=conj(mu);v=H(1:2:end).';vc=conj(v);
la=s(2:2:end);lac=conj(la);w=H(2:2:end);wc=conj(w);
MU=zeros(1,N);V=zeros(1,N);
MU(1:2:end)=mu;V(1:2:end)=v;
MU(2:2:end)=muc;V(2:2:end)=vc;
V=V.';
LA=zeros(1,N);W=zeros(1,N);
LA(1:2:end)=la;W(1:2:end)=w;
LA(2:2:end)=lac;W(2:2:end)=wc;
```

Next, form the loewner matrices. Notice that the Loewner matrices still have complex entries. In order to obtain matrices with real entries apply the following transformation

```
L=ones(N,1);
R=L.';
J=(1/sqrt(2))*[1 -1i;1 1i];
Jb=blkdiag(kron(eye(floor(N/2)),J));
LL=Jb'*LL*Jb;
LLs=Jb'*LLs*Jb;
V = Jb'*V;
W = W*Jb;
R = R*Jb; % tangential direction
L = Jb'*L;% tangential direction
```

After all the above, you should have only real quantities (please check!).

(b) Applying SVD of the loewner matrix  $\mathbb{L}$  or for the pairs  $[\mathbb{L} \ \mathbb{L}_s]$  and  $[\mathbb{L}; \mathbb{L}_s]$ . Then, from the singular value decay you could decide the truncation order ( $r$ ) and also you should get the left and right projectors with the first  $r$  columns.

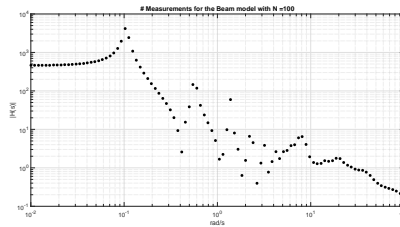


Figure 10: Measurements from the clamped beam.

(c) Providing only data we succeed to identify a Linear System and we were able to reduced it to a much smaller one with dimension only 10. The method we used is the Loewner Framework which is a data-driven method.

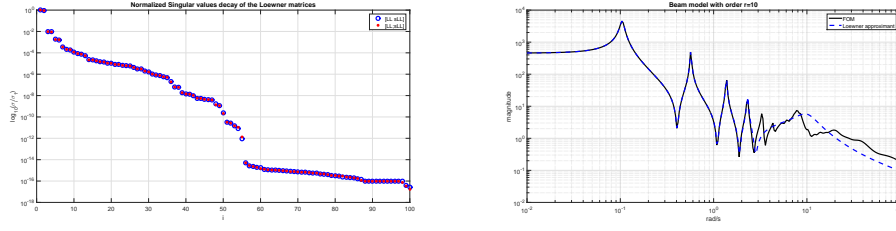


Figure 11: The Loewner approximant with the FOM.

(d) Next is the pole/zero diagram for the FOM and ROM.

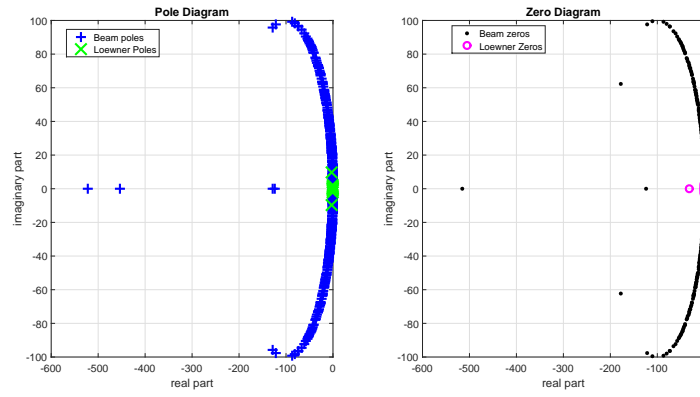


Figure 12: Poles and Zeros for the reduced in comparison with the FOM.

(f) Next are computed the impulse responses.

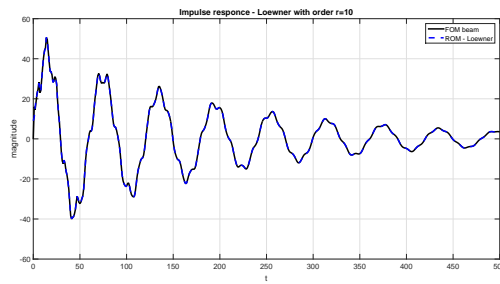


Figure 13: Impulse responses superimposed.



### Problem [8]. (Balanced truncation)

Here, we compute reduced order models using *balanced truncation*.

- (a) First, we load the system matrices A,B,C stored in beam.mat and then use the lyapchol command to compute the Cholesky factors for the controllability gramian P and observability gramian Q.

```
% load the beam linearized model of size 348
load beam

n = length(A);

%% Part a: Gramian computation
% here you can use lyapchol() or lyap() to compute the gramians

% the controllability gramian P with Cholesky factor U:  $P = U*U'$ ;
Ur=lyapchol(A,B);
U=Ur';
P = U*U';

% the observability gramian Q with Cholesky factor L:  $Q = L*L'$ ;
Lr=lyapchol(A',C');
L=Lr';
Q = L*L';
```

- (b) Here, we display the eigenvalues of the symmetric positive definite gramian matrices P and Q, as well as the Hankel singular values of the system.

```
[Y,S,X]=svd(U'*L);
Sigma = diag(S);

figure(1);
semilogy(sort(abs(eig(P)),'descend'),'-r. ');hold on
semilogy(sort(abs(eig(Q)),'descend'),'-b. ');
semilogy(sort(abs(Sigma),'descend'),'-g. ');
legend('eig(P)','eig(Q)','Hankel sv');
```

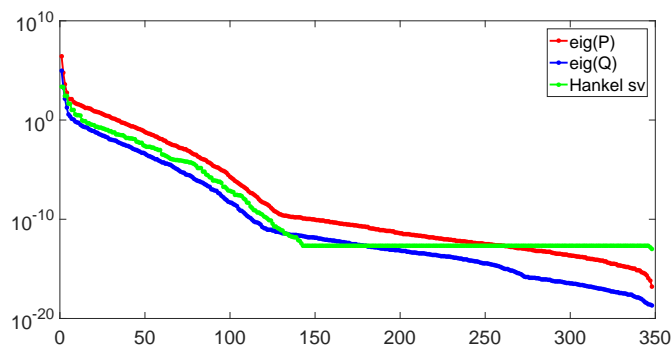


Figure 14: Eigenvalues of P,Q and Hankel singular values

- (c) For several values of  $k$ , i.e. 5,10,15,20,25, we produce frequency response plots comparing the original system with the reduced order models.

```
% the range of the desired order of the reduced model
rangek = [5 10 15 20 25];
```

```

% the range of the frequency samples used to compute the frequency response
freq = logspace(-2,2,200);

% vector of color entries
col = ['r','g','b','m','c'];

for ii = 1:length(freq)
H(ii) = C*(1i*freq(ii)*speye(n)-A)^(-1)*B;
end

figure(2);
loglog(freq,abs(H),'k');hold on;

for jj = 1:length(rangek)

k = rangek(jj);

%left and right truncated singular vector matrices
Yk = Y(:,1:k);
Xk = X(:,1:k);

% truncated diagonal matrices containing only the first k dominant Hankel svcs
Sk = S(1:k,1:k);

% compute the projector matrices

Wk = L*Xk*Sk^(-1/2);
Vk = U*Yk*Sk^(-1/2);

% the system matrices of the balanced truncation ROM
Ak = Wk'*A*Vk;
Bk = Wk'*B;
Ck = C*Vk;

for ii = 1:length(freq)
Hk(ii) = Ck*(1i*freq(ii)*speye(k)-Ak)^(-1)*Bk;
end

figure(2);
loglog(freq,abs(Hk),'r--','color',col(jj)); hold on;

% compute the spectral abscissa of Ak
SpecAbs(jj) = max(real(eig(Ak)));

end

Below, a plot showing the spectral abscissa of  $A_k$  as a function of  $k$ .

figure(3)
plot(rangek,SpecAbs,'-ro','markerfacecolor','k')
title('Spectral abscissa of Ak as a function of k');
xlabel('k');

```

- (d) For  $k = 20$ , we produce a plot of the error between the original system and the balanced truncation reduced model, i.e. plot  $|\mathbf{H}(j\omega) - \mathbf{H}_k(j\omega)|$ . On the same plot, the balanced

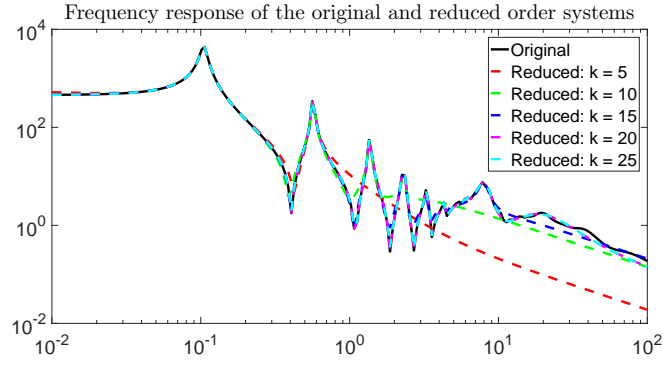


Figure 15: Frequency responses of the original and reduced order systems

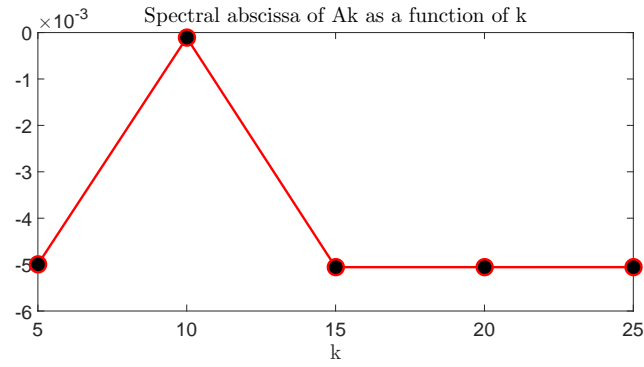


Figure 16: Spectral abscissa of the reduced order A matrix

```

truncation upper error bound, i.e.  $2(\sigma_{k+1} + \dots + \sigma_n)$ , is superimposed.

% take a fixed particular value for the truncation order
k = 20;

%left and right truncated singular vector matrices
Yk = Y(:,1:k); Xk = X(:,1:k);

% truncated diagonal matrices containing only the first k dominant Hankel svcs
Sk = S(1:k,1:k);

% compute the projector matrices

Wk = L*Xk*Sk^(-1/2); Vk = U*Yk*Sk^(-1/2);

% the system matrices of the balanced truncation ROM
Ak = Wk'*A*Vk; Bk = Wk'*B; Ck = C*Vk;

upperB = 2*sum(Sigma(2:n));

% compute frequency response of the error system
for ii = 1:length(freq)
Hk(ii) = Ck*(1i*freq(ii)*speye(k)-Ak)^(-1)*Bk;
Hdif(ii) = H(ii) - Hk(ii);
end

```

```

figure(4);
loglog(freq,abs(Hdif),'m'); hold on;
loglog(freq,ones(1,length(freq))*upperB,'k');
title('Frequency response of the error system and upper bound');
legend('Reduced: k = 20','Upper bound for BT');

```

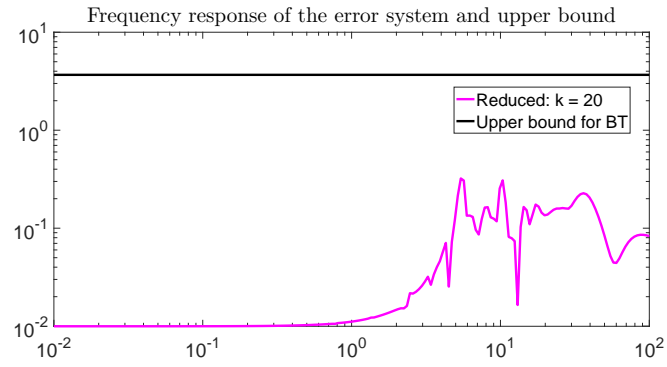


Figure 17: The error for  $k = 20$  and the balanced truncation upper bound

**Problem [9]. Data-Driven (Optimal) Model Reduction** In this problem is given a PDE which describe the Heat diffusion through a perfectly insulated, heat-conducting rod. The goal is to bypass the discretization (FEM,FDM, etc) by applying a Laplace tranformation and get the approximate solution by interpolating the transfer function.

(a) Applying Laplace transformation...

$$\begin{aligned}\frac{\partial T}{\partial t}(x, t) &= \frac{\partial^2 T}{\partial x^2}(x, t) \Rightarrow \\ \mathcal{L}\left\{\frac{\partial T}{\partial t}(x, t)\right\} &= \mathcal{L}\left\{\frac{\partial^2 T}{\partial x^2}(x, t)\right\} \Rightarrow \\ sT(x, s) - T(x, 0) &= \frac{\partial^2 T}{\partial x^2}(x, s).\end{aligned}$$

with  $T(x, 0) = 0$  we obtain  $sT(x, s) = \frac{\partial^2 T}{\partial x^2}(x, s)$  which is:

$$T(x, s) = C_1 e^{\sqrt{s}x} + C_2 e^{-\sqrt{s}x}. \quad (4)$$

From the 1st boundary condition  $\frac{\partial T}{\partial t}(0, t) = 0 \Rightarrow T(0, s) = 0$  with Eq.[1] we compute  $C_1 = C_2 = C$ . So Eq.[1] becomes

$$T(x, s) = C(e^{\sqrt{s}x} + e^{-\sqrt{s}x}). \quad (5)$$

From the 2nd boundary condition  $\frac{\partial T}{\partial x}(1, t) = u(t) \Rightarrow \frac{\partial T}{\partial x}(1, s) = U(s)$ . We compute the partial derivative in respect to  $s$  from Eq.[2].

$$\frac{\partial T}{\partial x}(x, s) = C(\sqrt{s}e^{\sqrt{s}x} - \sqrt{s}e^{-\sqrt{s}x}) \quad (6)$$

For  $x = 1$  we obtain:

$$\frac{\partial T}{\partial x}(1, s) = C(\sqrt{s}e^{\sqrt{s}} - \sqrt{s}e^{-\sqrt{s}}) = U(s). \quad (7)$$

Last part is  $y(t) = T(0, t) \Rightarrow Y(s) = T(0, s)$  which from Eq.[2] we get  $Y(s) = 2C$ . If we form now  $H(s) = \frac{Y(s)}{U(s)} = \frac{1}{\sqrt{s}(e^{\sqrt{s}} - e^{-\sqrt{s}})} = \frac{1}{\sqrt{s} \sinh \sqrt{s}}$ .

- (b) Solve the denominator equal to zero you get the poles of the system. Writing  $\sqrt{\cdot}$  and  $\exp\{\cdot\}$  as an infinite series you can conclude to the pole residue form.
- (c) If we take the first 200 terms ( $k = 200$ ) without the  $\frac{1}{s}$  and we apply TF-IRKA we get an optimal  $\mathcal{H}_2$  model with order 2. Then, if we add back the pole at zero we get the following results:

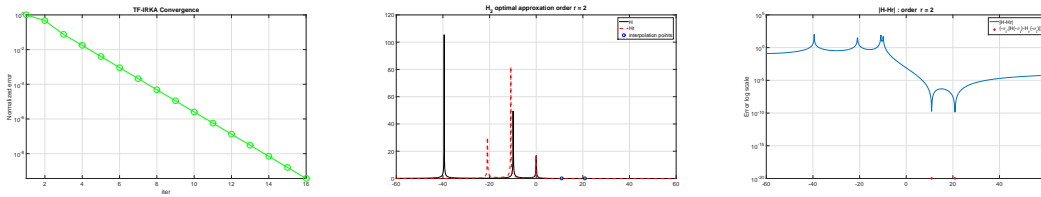


Figure 18: Left fig. is TF-IRKA convergence with  $tol = 1e - 8$ , middle fig. is the transfer function approximant with the irrational superimposed and right figure is the evaluation of the approximation on the real axis.

### Problem [10]. RLC circuit - MIMO - Band-stop filter

(a) Results with  $D = 0$ .

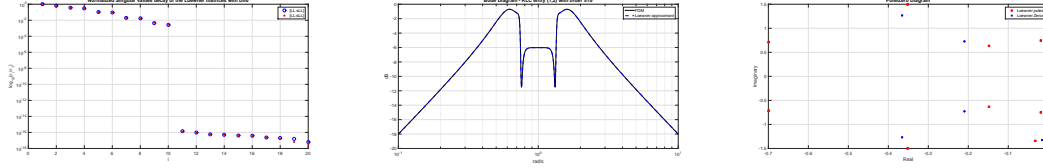


Figure 19: With  $D = 0$  we are able to recover the system with order 10.

(b) Results with  $D = -\frac{1}{2}$  (BandStop).

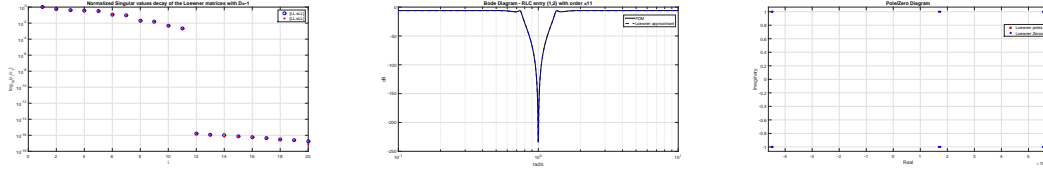


Figure 20: With  $D \neq 0$  we are able to recover the system with order 11. (dim+1)

(c) For the full order model MIMO with have:

$$P = Q = \begin{pmatrix} \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \end{pmatrix}$$

And the Hankel Singular Values are:

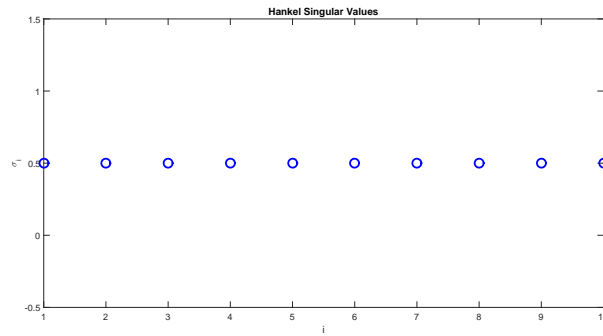


Figure 21: Hankel singular values.

As long as, there is no decay this system cannot be approximated using balanced truncation.