MAT 226B, Winter 2020

Final Project

(due by Sunday, March 22, 11:59 pm)

General Instructions

- You are required to submit your final project via file upload to Canvas: a single pdf file of your final project and a single zip file that contains all your Matlab files. To prepare the zip file, create a directory that contains all your Matlab files and then zip that directory. For each problem that requires Matlab computations include a single driver file so that I can run and check your program.
- If at all possible, use a text processing tool (such as LaTeX) for the preparation of your final project.
- You are required to solve the problems on the final project yourself! If there are students with solutions or program codes that were obviously copied or are trivial modifications of each other, then each involved student (regardless of who copied from whom) will only get the fraction of the points corresponding to the number of involved students.

Consider linear time-invariant descriptor systems

$$\begin{aligned} E\,\frac{d}{dt}x &= A\,x + b\,u(t), \quad t \geq 0, \\ y(t) &= c^T x(t), \end{aligned} \tag{1}$$

where $u : [\,0,\infty\,) \mapsto \mathbb{R}$ is a given input, $y : [\,0,\infty\,) \mapsto \mathbb{R}$ is the unknown output, $A, E \in \mathbb{R}^{n\times n}$ are given matrices, and $b, c \in \mathbb{R}^n$, $b, c \neq 0$, are given vectors. We assume that the matrix pair (E, A) is regular. This assumption guarantees that the matrix $sE - A$ is singular for at most n values of $s \in \mathbb{C}$.

In frequency domain, the input-output behavior of (1) is described by the transfer function

$$Z(s) = c^T\bigl(sE - A\bigr)^{-1}b, \quad s \in \mathbb{C}. \tag{2}$$

Note that $Z(s)$ is a rational function of type $(n-1, n)$. Let $s_0 \in \mathbb{C}$ be any fixed number such that the matrix $s_0E - A$ is nonsingular. We can then rewrite (2) as follows:

$$Z(s) = c^T\bigl(I - (s - s_0)M\bigr)^{-1}r, \quad s \in \mathbb{C},$$

where I denotes the $n \times n$ identity matrix,

$$M := \bigl(A - s_0 E\bigr)^{-1} E, \quad \text{and} \quad r := -\bigl(A - s_0 E\bigr)^{-1} b. \tag{3}$$

Let $k \leq n$, and let Z_k be a rational function of type $(k-1, k)$. The function Z_k is said to be a k-th Padé approximant of Z at the expansion point s_0 if

$$Z_k(s) = Z(s) + \mathcal{O}\bigl((s - s_0)^{2k}\bigr).$$

In class, we discussed two approaches for computing a k-th Padé approximant Z_k of Z:

- In the 'textbook' algorithm, one first computes the leading $2k$ moments

  $$\mu_j = c^T M^j r, \quad j = 0, 1, \ldots, 2k-1, \tag{4}$$

  of Z and then obtains the coefficients of the numerator and denominator polynomials of Z_k by solving a system of $2k$ linear equations for the $2k$ unknown coefficients.

- In the Lanczos approach, one runs the nonsymmetric Lanczos process (applied to the matrix M and the vectors r and c) for k steps to generate the $k \times k$ tridiagonal matrix T_k and then obtains Z_k via the formula

  $$Z_k(s) = \bigl(c^T r\bigr)\, e_1^T \bigl(I_k - (s - s_0) T_k\bigr)^{-1} e_1, \quad s \in \mathbb{C}, \tag{5}$$

  where e_1 is the first unit vector of length k and I_k denotes the $k \times k$ identity matrix.

The main goal of the final project is to implement these two approaches for computing Z_k and to run some numerical experiments using these two examples:

- Example 1 is a small descriptor system (1) with $n = 308$ and is provided in the Matlab file "`FP_Ex1.mat`";

- Example 2 is a moderately large descriptor system (1) with $n = 110460$ and is provided in the Matlab file "`FP_Ex2.mat`.

Problems

1. Let T_k denote the $k \times k$ tridiagonal matrix, V_k the $n \times k$ matrix of right Lanczos vectors, and W_k the $n \times k$ matrix of left Lanczos vectors generated by k steps of the nonsymmetric Lanczos process applied to M, r, c. Recall that in exact arithmetic, the matrix $D_k := W_k^T V_k$ is diagonal. We assume that no breakdowns occur in the algorithm, and so D_k is nonsingular.

   (a) Prove that

   $$M^j r = \beta_1 V_k T_k^j e_1 \quad \text{for all} \quad j = 0, 1, \ldots, k-1,$$

and

$$M^k r = \beta_1 V_k T_k^k e_1 + \beta_1 \beta_{k+1} \big(e_k^T T_k^{k-1} e_1 \big) v_{k+1}.$$

Here,

$$e_k := \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \in \mathbb{R}^k \quad \text{and} \quad \beta_1 := \|r\|_2.$$

(b) Prove that

$$c^T M^j = \gamma_1 \delta_1 e_1^T T_k^j D_k^{-1} W_k^T \quad \text{for all} \quad j = 0, 1, \dots, k-1.$$

Here, $\gamma_1 := \|c\|_2$ and $\delta_1 := w_1^T v_1$.

(c) Use parts (a) and (b) to prove that Z_k is a k-th Padé approximant of Z at the expansion point s_0

2. Write two Matlab functions for the efficient computation of matrix-vector products $q = Mv$ and $q = M^T v$ for $v \in \mathbb{C}^n$, respectively, where M is the matrix defined in (3). Note that $s_0 \in \mathbb{C}$ is fixed. Exploit this fact by pre-computing a single LU factorization of the sparse matrix $S := A - s_0 E$. To this end, use the Matlab command

```
[L,U,p,q,D] = lu(S,'vector').
```

Here D is a diagonal scaling matrix that Matlab constructs in order to enhance the stability and sparsity of the LU factorization. This means that the triangular matrices L and U and the permutations 'vectors' p and q correspond to the sparse LU factorization of the scaled matrix $D^{-1}S$.

3. Write a Matlab function that for given k computes the $2k$ moments (4). This function should use your function from Problem 2 for the efficient computation of r and for the matrix-vector products with M. Write a second Matlab function with $s \in \mathbb{C}$ and your computed values for the $2k$ moments (4) as inputs that employs the 'textbook' algorithm to compute $Z_k(s)$.

4. Write a Matlab function with $s \in \mathbb{C}$ amd T_k as inputs that employs the Lanczos approach to compute $Z_k(s)$. This function should use your functions from Problem 2 for the efficient computation of r and for the matrix-vector products with M and M^T.

5. Use your Matlab functions to compute approximations $Z_k(s)$ of the transfer function $Z(s)$ of Example 1. Note that this example is small enough so that you can easily generate the values $Z(s)$ of the exact transfer function.

Set up your Matlab driver program so that you can generate plots of

$$\log_{10} \big| Z_k(s) \big| \quad \text{and} \quad \log_{10} \big| Z(s) \big| \tag{6}$$

for all values of s of the form

$$s = 2\pi\,\mathtt{i}\,f, \quad f_{\min} = 0 \le f \le 8 \times 10^9 = f_{\max},$$

where $\mathtt{i} := \sqrt{-1}$. Your plots should show (6) vs. f (using the Matlab command "`semilogy`"). In order to resolve all the features of the transfer function, compute the values (6) for 1001 evenly-spaced values of f in the frequency range $f_{\min} \le f \le f_{\max}$.

First compare the 'textbook' algorithm and the Lanczos approach. Choose an expansion points s_0 such that the Lanczos approach converges reasonably fast. For this s_0, then run both the 'textbook' algorithm and the Lanczos approach for $k = 1, 2, \ldots$ until you find a value of k for which the two approaches give different numerical results. Report the value of that k. Can you explain why the numerical results produced by the two approaches are different?

Now use only the Lanczos approach. Experiment with different expansion points s_0 and try to find values for which you get fast convergence, *i.e.*, good approximations for relative small k. Also, compare the effects of real vs. complex values s_0 on computing times and speed of convergence. Employ the Matlab command "`cputime`" to obtain computing times for each of your runs.

6. Use your Matlab functions for the Lanczos approach to compute approximations $Z_k(s)$ of the transfer function $Z(s)$ of Example 2. Note that this example is large enough so that you cannot easily compute the values $Z(s)$ of the exact transfer function for a large number of values of s. In order to decide if $Z_k(s)$ has 'converged' to $Z(s)$, compare the values of $Z_k(s)$ and $Z_{k-1}(s)$ and declare 'convergence' if these values are 'close' enough.

   Set up your Matlab driver program so that you can generate plots of

$$\log_{10}\bigl|Z_k(s)\bigr| \tag{7}$$

   for all values of s of the form

$$s = 2\pi\,\mathtt{i}\,f, \quad f_{\min} = 10^9 \le f \le 4 \times 10^{10} = f_{\max},$$

   where $\mathtt{i} := \sqrt{-1}$. Your plots should show (7) vs. $\log_{10} f$ (using the Matlab command "`loglog`"). In order to resolve all the features of the transfer functions, compute the values (7) for 1001 values of f in the frequency range $f_{\min} \le f \le f_{\max}$. Choose these 1001 values such that $\log_{10} f$ is evenly-spaced.

   Experiment with different expansion points s_0 and try to find values for which you get fast convergence, *i.e.*, good approximations for relative small k. Also, compare the effects of real vs. complex values s_0 on computing times and speed of convergence. Employ the Matlab command "`cputime`" to obtain computing times for each of your runs.