APM120, Homework #3

Applied Linear Algebra and Big Data

Last updated: Thursday 9th February, 2023, 09:36

Assigned Feb 14, due Feb 21, 1:00 pm, via gradescope, **in pdf $\leq$ 20Mb, $\leq$ 30 pages.**

Eigenvalues, eigenvectors II: ODEs, transient growth, Jordan form

Show all steps in all calculations explicitly. Attach code used, well documented, and relevant plots and Matlab/python output, attaching code and figures *immediately following* the relevant question solution. A code printout is not a substitute for complete solutions, your solution should stand alone without the Matlab/python code or output. See needed python preliminaries at end of this HW[1]. It is fine to use Matlab/python unless a hand calculation (using only a simple calculator) is required *in orange.* For all questions, make sure you show all steps as if you are doing the problem by hand, and do not use library functions unless explicitly allowed in the question. Make sure you can do all calculations using no more than a hand-held calculator. See the end-note for guidelines and examples of hand-calculations.[2]

1. **Linear ordinary differential equations and matrix exponentiation:**

   (A) Show, in general, that the following two forms solve $d\mathbf{x}/dt = \mathsf{A}\mathbf{x}$,

   $$\mathbf{x}(t) = e^{\mathsf{A}t}\mathbf{x}_0; \quad \mathbf{x}(t) = \sum_i a_i \mathbf{e}_i e^{\lambda_i t}.$$

   Consider, `A=[-0.7108,-2.995;1.336,-0.7892];` $d\mathbf{x}/dt = \mathsf{A}\mathbf{x}$; $\mathbf{x}(0) = \mathbf{x}_0 = [-5; 2]$.

   (B) Calculate λ_i, $\mathbf{e}_i$ and then a_i for the above A.

   (C) Calculate $e^{\mathsf{A}t}$ for $t = 2.0$ using the eigenvalues and eigenvectors of A, compare to the result using `expm`/`scipy.linalg.expm` in Matlab/python. Then calculate the solution at the above t.

   (D) Draw *schematically* the phase space arrows and the phase space trajectory.

   (E) Derive the second-order ODE that is equivalent to this set of two first-order ODEs.

   (F) Learn what to avoid: **(i)** Consider the above as a Romeo & Juliet scenario à la Strogatz. Given that at the 2nd anniversary, $\mathbf{x}(t = 2) = [-2; -2]$, find the initial conditions at $t = 0$ leading to this frustrating celebration. **(ii)** Did they have a chance to begin with...? That is, would different initial mutual feelings lead to a more favorable state on that particular anniversary? Does the future look any brighter?

   (G) ***Optional extra credit challenge Let the above solution to (C) be denoted $\mathbf{x}(t) = (x_1(t), x_2(t))$. Use Matlab/python to: **(i)** Plot the three curves of $x_1(t)$, $x_2(t)$ and $|\mathbf{x}(t)|$ as function of t, for $t = (0, \ldots, 15)$, all on the same set of axes. **(ii)** On a second set of axes, plot $x_2(t)$ as function of $x_1(t)$ (the "phase space" trajectory). **(iii)** Plot the phase space arrows (quiver) for the above problem.

2. **Linear ordinary differential equations and transient growth:** Consider $d\mathbf{x}/dt = \mathsf{A}\mathbf{x}$ with,

```
A=[ -64.0859,  -22.6535;
    177.346,  62.6859];
```

(A) Calculate the eigenvalues and eigenvectors of A. What do you expect the solution $\mathbf{x}(t) = (x_1(t), x_2(t))$ to look like as $t \to \infty$ and why? Discuss the eigenvectors.

(B) Find the optimal initial conditions that lead to the maximum amplitude at $t = 1.5$.

(C) Write the explicit solution in the form $\sum_i a_i \mathbf{e}_i e^{\lambda_i t}$ using the initial conditions you calculated in the previous item to calculate the coefficients a_i.

(D) Plot time series of the solution norm $|\mathbf{x}(t)|^2 = x_1^2 + x_2^2$, and of the two coefficients of the two eigenvectors, $(a_1 e^{\lambda_1 t}, a_2 e^{\lambda_2 t})$. Explain the behavior of the solution starting from the initial conditions in terms of the eigenvectors and eigenvalues.

3. **Jordan form:** let,

```
A=[5,  -1,  -1; -10,  5,  3; 16,  -2,  -1];
```

(A) Calculate its eigenvalues and eigenvectors. Let V be the matrix whose columns are the eigenvectors, does V^{-1} exist so that it can be used to diagonalize A (check the determinant of V)? What is a Jordan form and what is the substitute for the standard diagonalization procedure? Calculate the Jordan form using Matlab: `[M,J]=jordan(A)` or python: `from sympy import Matrix; Amatrix=Matrix(A); M,J=Amatrix.jordan_form(); M=np.array(M).astype(float); J=np.array(J).astype(float);`.

(B) Consider `A1=A+[0 0 0.0000001; 0 0 0; 0 0 0];` Calculate its eigenvalues and eigenvectors. Discuss the sensitivity of the Jordan form to noise.

The next items are ***** optional extra credit:** A self-study task: read the appropriate section in the notes and consider the following questions.

(C) Calculate the generalized eigenvectors of A. *Hint:* if $(\mathsf{A} - \lambda \mathsf{I})^3$ returns an empty matrix in Matlab/python, you are seeing the effect of round-off errors on the null command. In that case, calculate $\mathsf{B}_3 = (\mathsf{A} - \lambda \mathsf{I})^3$, and set any entry that is smaller in absolute value than 10^{-13} to zero using Matlab: `B3(abs(B3)<1.e-13)=0` or python: `B3[np.abs(B3)<1.e-13]=0` and then calculate the null space using `B3null=null(B3)`, or, in python: get `null.py` from the course Sources, then: `from null import null; B3null=null(B3)`.

(D) Bring A to a Jordan form J using a similarity transformation $\mathsf{J} = \mathsf{M}^{-1}\mathsf{A}\mathsf{M}$, where M is composed of the generalized eigenvectors you calculated above. Compare your resulting Jordan form to Matlab/python.

* [What's the point of *****optional extra credit** challenge problems: apart from the fun of doing them, they may bring the total score of this HW assignment up to 110%, making up for problems you may have missed in this or other HW assignments…]

Python preliminaries & notes

1 python commands within the HW assume you have first used the followings: `import numpy as np; from numpy import linalg; import scipy as scipy; from scipy import linalg; import matplotlib.pyplot as plt; import matplotlib;`
Input a matrix A, column vector **b** and row vector **c** into python in the form
`A=np.array([[a11,a12,a13],[a21,a22,a23],[a31,a32,a33]]; b=np.array([[b1],[b2],[b3]]); c=np.array([[c1,c2,c3]]);` or convert Matlab arrays given in HW directly to python arrays using, e.g., `A=np.array(np.matrixlib.defmatrix.matrix('[1 2 3; 4 5 6]'));`

2 **Hand calculations**, in which you are asked to use only a simple hand-held calculator, are required only when we want to make sure you understand exactly what each step of an algorithm is doing. These also prepare you for the quizzes that involve similar hand calculations. **How much work to show?** Just don't use scratch paper, show all steps that you actually use for the hand calculations, but no more, carrying out calculations to **three significant digits**. Trust the graders to be reasonable, they were students in the course last year. **Examples: (i)** If you are asked, *not* in orange, to calculate the LU decomposition of a matrix, you may use Matlab/python as in `A(2,:)=A(2,:)-A(1,:)*(A(2,1)/A(1,1))` etc, but you may not use a library routine as in `[L,U,P]=lu(A)` except for checking your results. **(ii)** If required to calculate by hand the element $C_{2,3}$ of a matrix product $\mathsf{C} = \mathsf{AB}$, you need to explicitly multiply using a hand calculator the second row of A with the third column of B. You may not use Matlab/python to calculate `C=A*B` and then take the needed element from that product, nor to calculate `C23=A(2,:)*B(:,3)`.