Prof. Mark D Shattuck
Physics 39907 Computational Physics
November 7, 2024

Problem Set 6

Question 1. *Eigen-decomposition:* Find the matrix A with eigenvalues $\lambda_1 = 5$, $\lambda_2 = 3$ and eigenvectors $y_1 = (1, 0)$, $y_2 = (1, 1)$. Use MATLAB to find `[S e]=eig(A)` to show your answer is correct.

Question 2. *Markov Matrices:* A Markov matrix is a special matrix where each column sums to 1. This is a 2×2 example:

$$A = \begin{bmatrix} \frac{8}{10} & \frac{3}{10} \\ \frac{2}{10} & \frac{7}{10} \end{bmatrix}.$$

A Markov chain uses a Markov matrix to evolve a state at time n, u_n, to a state at $n+1$ according to this rule:

$$u_{n+1} = Au_n.$$

For example, $u = [N, S]^T$ might represent the number of people N who live in the north and S the number in the south. During 1 year 8/10 of the people living in the north, stay in the north, and 2/10 move to the south. 7/10 of those living in the south stay in the south, and 3/10 move to the north.

(1) What does $u_0 = \begin{bmatrix} 10 \\ 0 \end{bmatrix}$ represent?
(2) Show that the Markov chain rule is consistent with the moving habits described above, by finding u_1 for $u_0 = \begin{bmatrix} 10 \\ 0 \end{bmatrix}$ and $u_0 = \begin{bmatrix} 0 \\ 10 \end{bmatrix}$.
(3) Show that $u_n = A^n u_0$.
(4) Find the eigenvalues and eigenvectors of A.
(5) Express the equation for u_n in terms of the eigen-decomposition $A = S\Lambda S^{-1}$.
(6) Find u_1, u_2, and u_3 given that $N = 1,000,000$ people live in the North at $n = 0$ and zero people live in the south $S = 0$.
(7) Use the eigen-decomposition of A to find A^{100} and u_{100} how does it compare to the infinite time steady-state (fixed-point) A^∞ and u_∞.
(8) For n large how does the state u_n depend on the initial state u_0?

Question 3. *Eigen-system of K:* The k-th eigenvector y_k of K_N is:

$$y_k = (\sin(k\pi h), \sin(2k\pi h), ..., \sin(Nk\pi h)),$$

where $h = 1/(N + 1)$.

(1) Find the first eigenvalue of K_N by direct multiplication of the first row of K_N by y_1. (Useful Identity: $\sin 2x = 2 \sin x \cos x$).
(2) Use MATLAB to find `eig(K5)`, where `K5`$= K_5$. Show that it matches the general equation for the eigenvalues of K_N:

$$\lambda_k = 2(1 - \cos k\pi h).$$

`e=eig(K)` returns a column vector. It is useful to express $\lambda = (\lambda_1, ..., \lambda_N)$ as column vector `lam` in MATLAB as well. Then `e-lam` should be a column vector of zeros (possibly with round-off errors of order `eps`).

Question 4. *Linear-Constant-Coefficient-Finite-Difference-Ordinary-Differential-Equation-Solver (lccfdodes):* We discussed a number of integration schemes to solve the ordinary differential equation:

$$\dot{u} \equiv \frac{du}{dt} = Au,$$

where u is an $M \times 1$ vector and A is a an $M \times M$ constant matrix. Follow the steps below to write a MATLAB function that solves $\dot{u} = Au$ for initial condition u_0 with time-step dt and N steps.

(1) Here is a start:

```
function u=lccfdodes(A,u0,dt,N)
% lccfdodes <Linear-Constant-Coefficient-Finite-Difference-
%  Ordinary-Differential-Equation-Solver (lccfdodes)>
% Usage:: u=lccfdodes(A,u0,dt,N)
%
% Solves du/dt=Au with u(0)=u0 t=(0:N-1)*dt; u(:,n) and u0 are column vectors

% revision history:
% 11/01/2023 Mark D. Shattuck <mds> lccfdodes.m

%% Main

M=???;       % number of equations
u=???;       % initialize u(t) to zeros, one Mx1 vector for each of N times
u(:,1)=???; % set initial condition

G=???; % define growth factor G

% loop over times 1 through N-1
for n=1:N-1
  u(:,n+1)=???;  % update u_n+1 using G and u_n
end
```

(2) For the growth Factor, discretize the the time derivative to first order:

$$\frac{du}{dt} \simeq \frac{u(t+\Delta) - u(t)}{\Delta} + \mathcal{O}(\Delta)$$
$$= \frac{u_{n+1} - u_n}{\Delta},$$

where $u_n = u(n\Delta)$, and $t = n\Delta$. For the right-hand side start with the Forward Euler (`FE`) approximation:

$$\frac{u_{n+1} - u_n}{\Delta} = Au_n.$$

For `G` in the code solve this equation for u_{n+1} and find G such that: $u_{n+1} = Gu_n$. Fill in `G=???;` with the G you found, using `dt` for the scalar Δ. Note: For a matrix B and vector v, $(I+B)v = v+Bv$.

(3) Test your code on the equations for a simple harmonic oscillator:

$$\dot{x} = v$$
$$\dot{v} = -x$$

with initial condition $x = 1$ and $v = 0$. The following commands (script) should produce a x-v phase space plot like the one in figure 1, when you fill in the correct values for `A` and `u0`:

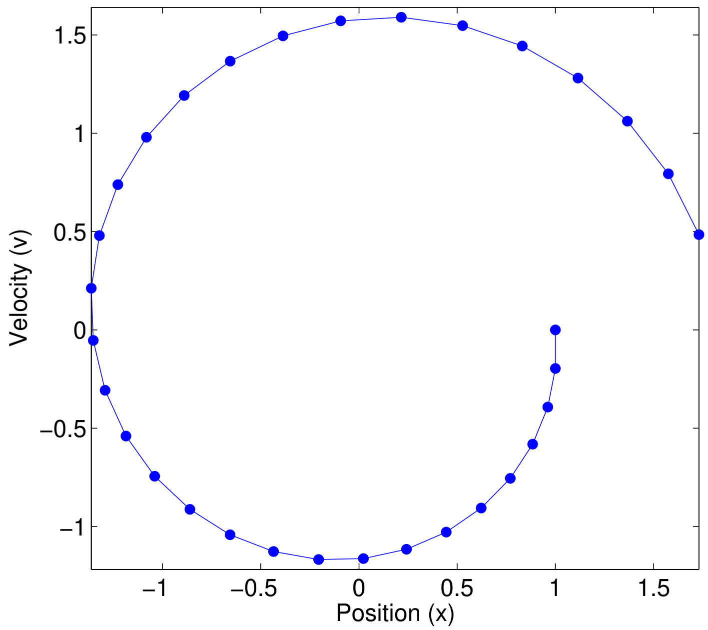

FIGURE 1. Phase-space trajectory for simple harmonic oscillator using forward Euler.

```
A=???;  % fill in SHM matrix A from du/dt=Au;
u0=???; % fill in initial conditions

N=32;            % Number of time points
dt=2*pi/(N-1);   % Time step

u=lccfdodes(A,u0,dt,N);   % solve the equation

%% Make a phase space x-v plot
h=plot(u(1,:),u(2,:),'.-');
set(h,'markersize',20);   % increase marker size
axis('equal')
set(gca,'fontsize',15);   % make font larger
xlabel('Position (x)');
ylabel('Velocity (v)');
```

(4) Add the exact solution to the plot.
(5) Find the `G` for Backward Euler (`BE`) using the approximation:

$$\frac{u_{n+1} - u_n}{\Delta} = Au_{n+1},$$

and solving for u_{n+1} such that: $u_{n+1} = Gu_n$. Note that u_{n+1} is on the right-hand-side this time. You may need to use inverses. However, in MATLAB use \ instead of `inv`. Add the `BE` solution to the plot.
(6) (*Optional:*) It might be useful to add a new input to your `lccfdodes` code to allow you to change the integrator from `FE` to `BE` and others (see below). One easy way is to use the `switch-case` statement. Here is an example. Add a new input `itype` to the function:

```
function u=lccfdodes(A,u0,dt,N,itype)
% lccfdodes <Linear-Constant-Coefficient-Finite-Difference-
%  Ordinary-Differential-Equation-Solver (lccfdodes)>
% Usage:: u=lccfdodes(A,u0,dt,N,itype{['FE'],'BE','TP','LF'})
```

Then in place of `G=???;` add the following:

```
% define growth factor G
switch itype
  case 'FE'
    G=???;  % forward Euler
  case 'BE'
    G=???;  % backward Euler
  case 'TP'
    G=???;  % trapazoid method 2nd-order
  case 'LF'
    G=???;  % leapfrog
end
```

The `switch`-`case` statement is a shorthand for cascading `if..then..else..end` statements. It executes only the code under the `case` if `case` `cond==itype`. You can read more in the documentation for `switch`. It is often useful to have a default choice for `itype`. To implement that add:

```
%% Parse Input
if(~exist('itype','var') || isempty(itype))
    itype='FE';
end
```

before you use `itype`. Then the function call `lccfdodes(A,u0,dt,N)` is the same as the function call `lccfdodes(A,u0,dt,N,'FE')`. Note the way this is set up the case of `itype` matters. So `'FE'~='fE'`. You could use the command `upper` to modify this behavior.

(7) Find the `G` for the trapezoid method (`TP`) using the approximation:

$$\frac{u_{n+1}-u_n}{\Delta} = A\frac{u_n+u_{n+1}}{2},$$

and solving for u_{n+1} such that: $u_{n+1} = Gu_n$. Add this solution to the plot.

(8) Find the `G` for the explicit modified Euler method (`ME`). To see the pattern start with `FE` for SHM:

$$\begin{aligned}
\frac{x_{n+1}-x_n}{\Delta} &= v_n, \\
\frac{v_{n+1}-v_n}{\Delta} &= -x_n. \\
\frac{1}{\Delta}\left(\begin{bmatrix} x \\ v \end{bmatrix}_{n+1} - \begin{bmatrix} x \\ v \end{bmatrix}_{n}\right) &= \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}\begin{bmatrix} x \\ v \end{bmatrix}_{n}. \\
\frac{u_{n+1}-u_n}{\Delta} &= Au_n. \\
u_{n+1}-u_n &= A\Delta u_n. \\
u_{n+1} = u_n + A\Delta u_n &= (I + A\Delta)u_n = G_{FE}u_n.
\end{aligned}$$

To make the modification replace $-x_n$ on the *rhs* of the second equation with $-x_{n+1}$. This is still explicit since x_{n+1} can be calculated from the first equation.

$$\begin{aligned}
\frac{x_{n+1}-x_n}{\Delta} &= v_n, \\
\frac{v_{n+1}-v_n}{\Delta} &= -x_{n+1};. \\
x_{n+1}-x_n &= v_n\Delta, \\
v_{n+1}-v_n &= -x_{n+1}\Delta. \\
x_{n+1} &= x_n + v_n\Delta, \\
v_{n+1} &= v_n - x_{n+1}\Delta.
\end{aligned}$$

Collecting $n+1$ terms on the left:

$$\begin{aligned}
x_{n+1} &= x_n + v_n\Delta \\
v_{n+1} + x_{n+1}\Delta &= v_n.
\end{aligned}$$

Converting to matrix form and solving:

$$\begin{aligned}
\begin{bmatrix} 1 & 0 \\ \Delta & 1 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix}_{n+1} &= \begin{bmatrix} 1 & \Delta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix}_n . \\
\begin{bmatrix} 1 & 0 \\ \Delta & 1 \end{bmatrix} u_{n+1} &= \begin{bmatrix} 1 & \Delta \\ 0 & 1 \end{bmatrix} u_n. \\
u_{n+1} &= \begin{bmatrix} 1 & 0 \\ \Delta & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & \Delta \\ 0 & 1 \end{bmatrix} u_n. \\
u_{n+1} &= \begin{bmatrix} 1 & 0 \\ -\Delta & 1 \end{bmatrix} \begin{bmatrix} 1 & \Delta \\ 0 & 1 \end{bmatrix} u_n. \\
u_{n+1} &= \begin{bmatrix} 1 & \Delta \\ -\Delta & 1-\Delta^2 \end{bmatrix} u_n. \\
u_{n+1} &= G u_n, \\
G &= \begin{bmatrix} 1 & \Delta \\ -\Delta & 1-\Delta^2 \end{bmatrix}.
\end{aligned}$$

To see the general pattern notice that A can be broken up into a strictly lower triangular part L and an upper triangular part $U = A - L$ such that $A = L + U = L + A - L = A$. To find L in MATLAB use the function `tril(A,-1)`. `tril(A,k)` returns a lower-triangular matrix from A starting at the k-th diagonal. $k = 0$ is the main diagonal, $k > 0$ is above the diagonal, and $k < 0$ is below the main diagonal. Using this decomposition and returning to the generic forward Euler and replacing A:

$$\begin{aligned}
u_{n+1} &= (I + A\Delta)u_n = (I + (L+U)\Delta)u_n \\
&= L\Delta u_n + (I + U\Delta)u_n.
\end{aligned}$$

Now all of the terms $L\Delta u_n$ can be replaced by previously calculated terms $L\Delta u_{n+1}$ since L has only non-zero terms below the main diagonal:

$$\begin{aligned} u_{n+1} &= L\Delta u_{n+1} + (I + U\Delta)u_n. \\ u_{n+1} - L\Delta u_{n+1} &= (I + U\Delta)u_n. \\ (I - L\Delta)u_{n+1} &= (I + U\Delta)u_n. \\ u_{n+1} &= (I - L\Delta)^{-1}(I + U\Delta)u_n. \end{aligned}$$

Implement this formula and add to your plot. You should see an ellipse instead of a circle. Notice that we could have defined `L=tril(L,0)` and then $U = A - L$ would be strictly upper-triangular. Then we could replace Uu_n with Uu_{n+1}. In fact there are many ways to chose the order of evaluation since any permutation of A does not change the equations. So there are $N!$ ways, where N is the rank of A. For the 2×2 we have been using there are 2 ways. One gives an ellipse tipping left and the other to the right.

(9) (*Optional:*) Here is the last scheme that we discussed `LF`:

Leapfrog:

$$\frac{u_{n+1} - u_{n-1}}{2\Delta} = Au_n.$$

(10) Include all of your code and a single plot of the exact solution with the all 4 schemes `FE`, `BE`, `TP`, and `ME` (and `LF` if you did it) on one plot.

Question 5. *Magnetic Dipole in a Magnetic Field:* The equations for a magnetic moment vector $m = (m_x, m_y, m_z)$ in a magnetic field $B = (0, 0, 1)$ is a good test problem for the code `lccfdodes` from the previous problem. The moment experiences a torque in the magnetic field and evolves according to the Bloch equations:

$$\begin{aligned} \frac{dm}{dt} &= m \times B - Rm + M_0, \\ R &= \begin{bmatrix} \frac{1}{T_2} & 0 & 0 \\ 0 & \frac{1}{T_2} & 0 \\ 0 & 0 & \frac{1}{T_1} \end{bmatrix}, \\ M_0 &= \left(0, 0, \frac{1}{T_1}\right). \end{aligned}$$

R is a relaxation matrix of positive relaxations times T_1 and T_2 with $T_1 \geq T_2$. $m \times B$ is the cross product.

(1) Rewrite the equation for m in this matrix form:

$$\frac{dm}{dt} = Am + b,$$

and find A and b in terms of T_1 and T_2.

(2) The current function `lccfdodes(A,u0,dt,N)` does not allow for the constant term b. To handle this case define u such that $m = u - A^{-1}b$, and show that:

$$\frac{du}{dt} = Au.$$

(3) If the initial condition $m(0) = m_0$, what is the initial condition for u?

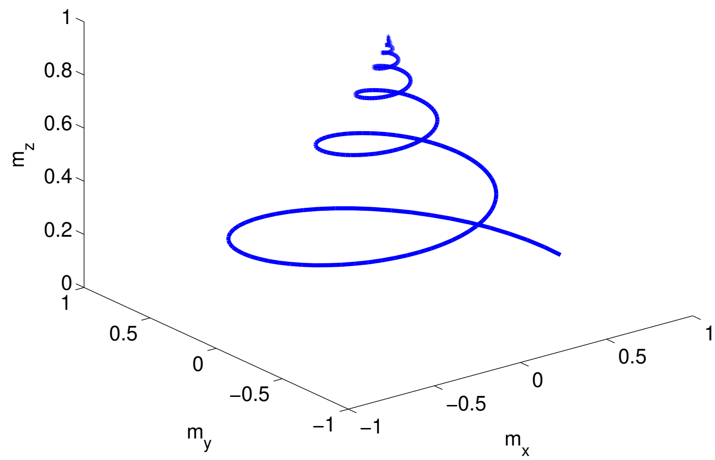

FIGURE 2. Solution to the Bloch equations.

(4) How can you recover the real solution m from the solution u that comes from: `u=lccfdodes(A,u0,dt,N);`? What MATLAB command will you use to account for the fact that `u` is a list of vectors at each of `N` time points?
(5) Solve this system with initial conditions `m0=[1;0;0]`, `T1=10`, `T1=8`, for a total time of `T=100`, with `dt=.1`, to produce plot like figure 2, using `plot3(m(1,:),m(2,:),m(3,:))`.
(6) Comment on the effect of changing T_1 and T_2.
(7) Comment on the effect of changing integration schemes? Chose one to make a plot to turn in.

Question 6. *Linear Predator-Prey Model:* The population of rabbits r grows at a rate of $6r$ from births, but decreases at a rate of $-2f$ due to predation from the population of foxes f. The fox population grows at a rate $2r + f$ due to increase of food and birth. This leads to the following equations:

$$\frac{dr}{dt} \equiv \dot{r} = 6r - 2f$$
$$\dot{f} = 2r + f.$$

(1) Define $u = (r, f)$ and convert these equations to matrix form $\dot{u} = Au$.
(2) What is A?
(3) What are the eigen-values Λ and eigen-vectors S of A?
(4) Check your answer using MATLAB: `[S,e]=eig(sym(A)).` `e`$\equiv \Lambda$.
(5) Rewrite the equation using the eigen-decomposition of A.
(6) Substitute $y = S^{-1}u$ into the equation, and show it reduces to $\dot{y} = \Lambda y$, using the fact that differentiation and matrix multiplication are linear so that $B\dot{u} = \dot{(Bu)}$, for any constant matrix B.
(7) Using $y = (y_1, y_2)$, rewrite $\dot{y} = \Lambda y$ as two equations and solve for y_1 and y_2 with initial conditions y_1^0 and y_2^0. The first equations should be $\dot{y_1} = \lambda_1 y_1$.
(8) Solve this model for analytically u given $u_0 = u(0)$.
(9) Show the solution is equivalent to $u = Se^{\Lambda t}S^{-1}u_0$. Note: this e is Euler's constant not the eigenvalue matrix.

(10) In MATLAB there are 2 different functions to find the exponential of matrix. `exp(e)` is the element-wise exponentiation, where each element of the matrix is exponentiated. `expm` is the matrix exponentiation. It uses the Taylor expansion:

$$\texttt{expm(A)} \equiv \exp A = I + A + \frac{1}{2}A^2 + ... + \frac{1}{N!}A^N.$$

For the solutions to differential equation we need the matrix version. If A is diagonal then the Taylor expansion is simplified since `diag(v)^k` equals `diag(v^k)`. Look at `exp(e)` and `expm(e)` in MATLAB and explain the difference. Here `e` is the eigenvalue matrix Λ.

(11) This model predicts that rabbits and foxes will grow without bound, which is only a good model for early times when rabbit food is plentiful. However it does predict the ratio of rabbits to foxes. Plot the solution $r(t)/f(t)$ for the initial condition of $r = 10$ and $f = 10$ using `dt=1/20` for the interval [0 T], where `T=5`. To evaluate a matrix exponential at many times you will need a loop. For example to find $q(t) = e^{At}$ for `t=0:dt:T` use:

```
t=0:dt:T;
q=zeros(1,N);
for n=1:N
    q(n)=expm(A*t(n));
end
```

(12) Compare the exact solution to `lccfdodes`. Rate each of the 4 integrators that we discussed.

(13) Compare to MATLAB's integrator `ode45`. The code to get the solution is:

```
t=0:dt:T; % list of times to find the solution
sol=deval(ode45(@(t,u) A*u,[0 T],u0),t);
```

Where `A` is the same matrix, and `u0` is the initial conditions.