AMATH 352 Coding Final

Tom Trogdon

Due Friday, August 17

Rules:

- Absolutely no late submissions will be accepted. This part of the exam is due Friday, August 17 by 10:50am.
- NO COLLABORATION. This exam should be your own work. You may not discuss the problems or results with anyone other than our TA or myself. Violating this has serious consequences, see `http://www.washington.edu/uaa/advising/help/academichonesty.php`.
- This exam is worth 20% of your grade (2/3 of the whole final).
- Choose **THREE** of the four problems below. If you do all four I will just grade the first three I see.
- All code written should be uploaded on the Moodle page. Filenames MUST be of the form `LastName_ExerciseNumber.m`. Points will be deducted for not following this convention.

Exercise 1

In this question we consider applying Gram-Schmidt to the inverse of the Hilbert matrix. In MATLAB this matrix is constructed by calling `invhilb(n)` for an integer n. For any given n, write code that will orthonormalize the columns of `invhilb(n)` using the following algorithms:

1. Using formula (5.19) in the text and normalizing all the columns **after** making them orthogonal (Classical GS).
2. Using the more stable version (page 234, see also homework 6) of Gram-Schmidt (Modified GS).
3. Applying Classical GS twice to the same matrix (Two Step GS).
4. Using MATLAB 's built-in QR algorithm `[Q,R] = qr(A)` (Built-in GS).

Now, apply each of these algorithms to to `invhilb(n)` for n in `N = 2:20`. If `Q` is the othogonalized matrix then measure the error by `norm(Q'*Q - eye(n))`. For each algorithm and each n you will have an associated error. Plot these errors on a semilog plot versus n using

```
clf
figure(1)
hold on
semilogy(N,ModifiedError,'.g')
semilogy(N,ClassicalError,'.r')
semilogy(N,TwoStepError,'.b')
semilogy(N,BuiltinError,'.k')
```

Also include the plot of the condition number `cond(invhilb(n))` of the inverse Hilbert matrix on the same figure. Comment on what you see and rank the algorithms in terms of effectiveness/efficiency.

Exercise 2

Code the full $PA = LU$ algorithm (pseudo code given in the text). Apply your algorithm to the matrices

```
%(a)
a = [0,1,1,0,10;
    0,1,2,5,4;
    1,-1,0,0,11;
    3,1,1,-15,1;
    1,-1,-1,1,-1];
%(b)
b = [0,1,1,0,10;
    0,1,2,5,4;
    1,-1,0,0,11;
    3,1,1,-15,1;
    1,3,4,5,45];
%(c)
c = [1,1,1,0,10;
    0,1,2,5,4;
    1,-1,0,0,11;
    3,1,1,-15,1;
    1,2,4,5,0];
```

Comment on what you see for each matrix. You can find these matrices in an m-file on the Moodle page.

Exercise 3

Code the following algorithm for computing the largest (in modulus) eigenvalue of a matrix A.

Choose a vector v randomly
set `ratio` to be zero
for $i = 1$ to `max`
    set v_{new} to be Av

set `rationew` to be $\langle v_{new}, v\rangle / \langle v, v\rangle$
if `rationew` differs from `ratio` by less than 10^{-15} then break out of loop
set `ratio` to be `rationew`
set v to be $v_{new}/\|v_{new}\|$
end
print out `ratio`

Note: This can also be done with a while loop.

The variable `ratio` will be an approximation of the largest eigenvalue and v will be an approximate eigenvector. Additionally, when you break out of the loop i will give you the number of iterations you needed. Apply this algorithm to `A = rand(10)` many times and comment on what you see in terms of the eigenvalue found and i. Running the full algorithm many times and then averaging the output may help.

Exercise 4

In this exercise we will numerically solve the differential equation

$$\Psi''(x) - \sin^2(x)\Psi(x) = \cos(x), \quad x \in [0, 2\pi],$$

for a periodic solution $\Psi(x)$. You do not need to know how to solve differential equations to do this problem!

- MATLAB has an implementation of the Fast Fourier Transform (FFT), called by `fft(v)` for a vector $v \in \mathbb{C}^n$. The FFT is a linear function. Write code to find its matrix representation for any given n by applying the FFT to each basis vector. Use the variable `FFTMAT` to denote this matrix.
- The inverse FFT (iFFT) is called by `ifft(v)` so that `v - ifft(fft(v)) = 0`. Write code to find the matrix representation of the iFFT. Call this matrix `iFFTMAT`.
- Check that `iFFTMAT*FFTMAT` produces the identity matrix.
- Use the code

  ```
  X = linspace(0,2*pi,n+1);
  x = X(1:n);
  d = diag([0 1:floor(n/2) -fliplr(1:floor((n-1)/2))]);
  a = -diag(sin(x).^2);
  diffop = -iFFTMAT*d*d*FFTMAT+a;
  rhs = cos(x)';
  sol = diffop\rhs;
  plot(x,sol)
  ```

  to obtain and plot an approximate solution `sol`. Plot this solution for $n = 5, 10, 20, 40$ on this same figure.