Numerical Methods Final Exam Practice — Winter 2010

120 minutes ⋆ One **8.5×11 sheet** of notes is allowed. ⋆ Computers are allowed, but not the Internet.

(This practice test only has problems covering material since the last test, and most problems on the final exam will be on material since the last test. However, some problems will cover earlier material, so it is advisable to look over the earlier tests and practice tests.)

(1) Suppose that A is an $n \times n$ diagonalizable matrix. Show that $\det(A)$ is the product of the eigenvalues of A. (This is true for defective matrices as well, but is more subtle.)

Recall that if D is diagonal (or upper- or lower-triangular) then $\det(D)$ is the product of the entries on the diagonal, and that $\det(AB) = \det(A)\det(B)$ for square matrices A and B.

With A diagonalizable, we can write $A = V\Lambda V^{-1}$, where the diagonal of Λ is $\lambda_1, \ldots, \lambda_n$, giving us

$$\begin{aligned}\det(A) &= \det(V\Lambda V^{-1}) = \det(V)\det(\Lambda)\det(V^{-1})\\ &= \det(\Lambda) \quad \text{(since } \det(V)\det(V^{-1}) = \det(I) = 1\text{)}\\ &= \lambda_1 \cdots \lambda_n.\end{aligned}$$

That is a complete answer to the problem as stated, but for curiosity's sake, here is the more subtle argument that also works for defective matrices. In the characteristic polynomial $p(\lambda) = \det(A - \lambda I) = c_n\lambda^n + c_{n-1}\lambda^{n-1} + \cdots + c_0$, we must have $c_0 = p(0) = \det(A - 0I) = \det(A)$. We also have $p(\lambda) = (\lambda_1 - \lambda)(\lambda_2 - \lambda)\cdots(\lambda_n - \lambda)$, so $p(0) = \lambda_1 \cdots \lambda_n$. So $\det(A) = p(0) = \lambda_1 \cdots \lambda_n$. (The reason $p(\lambda)$ factors as it did above and not as, say, $p(\lambda) = -4(\lambda_1 - \lambda)\cdots(\lambda_n - \lambda)$, is that $c_n = (-1)^n$, since in $\det(A - \lambda I)$, the only way to get a λ^n term is by taking the product along the diagonal of $A - \lambda I$, so the λ^n term in the characteristic polynomial is always $(-\lambda)^n$.)

(2) (a) Show that the eigenvalues of an upper-triangular matrix are the entries on its diagonal. (Hint: What would the characteristic polynomial be?)

Let U be upper-triangular, with entries $d_1, \ldots, d_n$ on the diagonal. Then $U - \lambda I$ is also upper triangular, so the characteristic polynomial $\det(U - \lambda I)$ will be the product of the entries on the diagonal, which will be $(d_1 - \lambda)(d_2 - \lambda)\cdots(d_n - \lambda)$. The eigenvalues will be the roots of this polynomial, which are $d_1, \ldots, d_n$.

(b) Two $n \times n$ matrices A and B are *similar* if there is a matrix E such that $A = EBE^{-1}$. (Note that saying a matrix is diagonalizable is equivalent to saying it is similar to some diagonal matrix.) Show that similar diagonalizable matrices have the same eigenvalues.

Suppose $A = EBE^{-1}$. There are two direct approaches here.

The first is that if $B = V\Lambda V^{-1}$, then $A = EV\Lambda V^{-1}E^{-1} = (EV)\Lambda(EV)^{-1}$, so A and B both have the entries on the diagonal of Λ as their eigenvalues.

The second is to use the characteristic polynomial:

$$\begin{aligned}\det(A - \lambda I) &= \det(EBE^{-1} - \lambda I)\\ &= \det(EBE^{-1} - E\lambda IE^{-1}) \quad \text{(since } \lambda \text{ and } I \text{ commute with anything)}\\ &= \det(E(B - \lambda I)E^{-1})\\ &= \det(E)\det(B - \lambda I)\det(E^{-1})\\ &= \det(B - \lambda I) \quad \text{(since } \det(E)\det(E^{-1}) = \det(I) = 1\text{)}.\end{aligned}$$

So, since A and B have the same characteristic polynomial, they must have the same eigenvalues. This second approach has the advantage that it works even when A and B are not diagonalizable.

(c) The *Schur decomposition* of an $n \times n$ matrix A is $A = QUQ^*$ where Q is unitary and U is upper-triangular. The Schur decomposition always exists. Explain how you could use the Schur decomposition to compute the eigenvalues of A.

Given the Schur decomposition, $A = QUQ^*$, noting that $Q^* = Q^{-1}$ since Q is unitary, we have A similar to U, so A and U have the same eigenvalues. As seen above, the eigenvalues of U are the entries along its diagonal. In short, given $A = QUQ^*$, the eigenvalues of A lie on the diagonal of U.

(3) Find the unique cubic polynomial $p(x)$ that interpolates the points $(x_1, y_1) = (-2, 0)$, $(x_2, y_2) = (1, 0)$, $(x_3, y_3) = (2, 5)$, $(x_4, y_4) = (4, 0)$. (You do not need to simplify fully.)

We use the Lagrange formula. Fortunately, only y_3 is nonzero, leaving us with

$$\begin{aligned}p(x) &= y_3\frac{(x - x_1)(x - x_2)(x - x_4)}{(x_3 - x_1)(x_3 - x_2)(x_3 - x_4)}\\ &= 5\frac{(x + 2)(x - 1)(x - 4)}{(4)(1)(-2)}\end{aligned}$$

(4) What is the Vandermonde matrix of the vector $\mathbf{v} = \begin{bmatrix} 1 \\ i \\ -1 \\ -i \end{bmatrix}$?

$$V = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -i & -1 & i & 1 \\ -1 & 1 & -1 & 1 \\ i & -1 & -i & 1 \end{bmatrix}$$

(5) Suppose you have some discrete data that you need to interpolate. Describe a circumstance where you might prefer a natural cubic spline, and a circumstance where you might prefer PCHIP.

If we know that the underlying function is very smooth (relative to the spacing of our data points), we would prefer a natural cubic spline. An example of a situation like this might be interpolating the position of a projectile from measurements of its position.

If the data is not so smooth, if we want to avoid avoid overshooting the data, or if we want to preserve the monotonicity of the data, we would favor PCHIP. One possible situation where we might choose PCHIP: Suppose we are building a control system for a robotic arm, and the points we are interpolating specify where the arm should be at various times; the interpolation will be used to determine how the arm moves from one position to the next. If we overshoot the data, it could result in the arm moving beyond some range that it was expected to remain within. (Linear interpolation would also avoid overshooting the data, but could put a lot of stress on the robot because of the abrupt changes in velocity.)

(6) The following table contains barometric pressure from Albany International Airport on March 9, 2010. (The pressure units are inHg, inches of mercury.)

t	P
8:00pm	29.97
5:00pm	29.94
2:00pm	29.93

Find an approximation of how fast the pressure was changing at 8:00pm. What aspects of the data are restricting the accuracy of your approximation?

We can estimate the derivative by interpolating the 3 points with a quadratic function $q(t)$ and looking at $q'(8)$:

```
t = [8; 5; 2];
P = [29.97; 29.94; 29.93];
q = vander(t)\P; % The polynomial that interpolates the data.
% The interpolating polynomial is q(1)*t^2 + q(2)*t + q(3).
dqdt = 2*q(1)*8 + q(2)
```

This gives us $q'(8) = 0.013$. So we get the approximation $P'(8) \approx 0.013$ inHg/hour.

One aspect of the data undermining the accuracy is that, since the data is only given to two decimal places, the roundoff error in the data is quite large relative to the changes were are looking at. Another problem is that the data points are three hours apart, whereas accurate approximations of the derivative favor closely spaced data. (This would not be a problem if we were measuring a quantity whose change is fairly steady over such long time periods; however, barometric pressure is not that smooth.)

(7) Simpson's method works by interpolating a function with quadratic functions over many short intervals. Why not use, say, degree 50 polynomials?

As we have seen, high-degree polynomial interpolation is generally a bad idea (with the exception of Chebyshev polynomials): we run into bad numerical problems, and even if we could do perfect arithmetic, high-degree interpolating polynomials tend to have wild oscillations between data points.

That said, degree 50 Chebyshev polynomials would not be such a bad idea, and would generally yield a lot of accuracy. The drawback is that Chebyshev polynomials do not lend themselves well to adaptive algorithms.

(8) (a) Do one step of RK4 on the initial value problem

$$\begin{aligned}y' &= y^2\\ y(0) &= 1\end{aligned}$$

using $h = 1$.

```
f = @(t,y) y.^2;
h = 1;
t0 = 0;
y0 = 1;
s1 = f(t0, y0); % = 1
s2 = f(t0 + h/2, y0 + s1*h/2); % = 2.25
s3 = f(t0 + h/2, y0 + s2*h/2); % = 4.5156
s4 = f(t0 + h, y0 + s3*h); % = 30.422
y1 = y0 + h/6*(s1 + 2*s2 + 2*s3 + s4);
```

This yields $\boxed{y_1 = 8.4922}$, $t_1 = 1$.

(b) Do one step of Euler's method, also using $h = 1$, and use that to provide an error estimate for your answer to (a). (In practice, since RK4 is a 4th-order method, you would want to use a 3rd-order method to produce the error estimate.)

```
eulery1 = y0 + s1*h;
```

This yields $y_1 = 2$. So, our error estimate is $|8.4922 - 2| = \boxed{6.4922}$.

We will see in the next problem why $s_1, \ldots, s_4$ were so different from each other and we got such a large error estimate.

(9) (a) Verify that $y = 1/(1 - t)$ is an exact solution to the initial value problem from the previous problem.

$$\begin{aligned}y' &= -(1 - t)^{-2}(-1) = \frac{1}{(1 - t)^2} = y^2\\ y(0) &= 1/(1 - 0) = 1.\end{aligned}$$

(b) In this solution, $y \to \infty$ as $t \to 1^-$. What will happen if you try to solve the initial value problem over the interval $[0, 2]$ using an adaptive algorithm?

If we use an adaptive algorithm, once we get close to $t = 1$, any step that would reach $t = 1$ will have a large error estimate (as we had in the previous problem) and be rejected. The step size will get smaller and smaller, and we will get closer to $t = 1$, but we will never reach $t = 1$.