Numerical Analysis (CS 450)

Homework Set 5

Due: April 18, 2014 (Friday, because of Exam 2) · Out: April 6, 2014

Problem 1: Interpolation, Newton and Cubic Spline (20 points)

(a) (9 points) Prove that the formula using divided differences:

$$f[t_1, t_2, \ldots, t_k] := \frac{f[t_2, t_3, \ldots, t_k] - f[t_1, t_2, \ldots, t_{k-1}]}{t_k - t_1}$$
$$f[t_j] := f(t_j)$$

indeed gives the coefficient of the jth basis function in the Newton interpolation polynomial.

Hint: Use induction.

(b) ($4 \times 1.5 = 6$ points) Given the three data points $(-1, 1)$, $(0, 0)$, $(1, 1)$, determine the interpolating polynomial of degree two:

(i) Using the monomial basis

(ii) Using the Lagrange basis

(iii) Using the Newton basis

(iv) Show that the three representations give the same polynomial.

(c) (5 points) Consider interpolating a given set of data points (x_i, y_i), $i = 1, \ldots, n$ using natural cubic splines. Write a code to set up and solve the linear system that performs this interpolation. Plot the resulting cubic spline along with the data. For the data, pick $n = 6$ random points $(x_i)_{i=1}^n$ on $[0, 1)$ with values $(y_i)_{i=1}^n$ in $[0, 1)$.

Hint: Make sure you sort the x_i's after you draw the random numbers and before you start constructing the spline, to avoid confusing your spline construction code.

Problem 2: Numerical Quadrature (25 points)

The goal of this problem is to compute π using numerical integration using the following equation:

$$\int_0^1 \frac{4}{1+x^2}\, dx = \pi$$

and several different quadrature schemes. Your task is to write a function that accomplishes this using each of the following different schemes.

(a) Composite midpoint rule.

Hint: Perform your own tests by comparing using
`scipy.integrate.fixed_quad(f, a, b, n=1)`.
No need to report these.

(continued on next page. . .)

(b) Composite trapezoid rule. (Compare with `scipy.integrate.trapz(f(X), X)` and don't report the test results.)

(c) Composite Simpson rule. (Compare with `scipy.integrate.simps(f(X), X)`.)

(d) Monte Carlo method.

The Monte Carlo methods work by drawing n uniform samples $(x_i)_{i=1}^{n}$ using a uniform distribution on the integration domain ($[0,1]$ in this case) and using the approximation

$$\int_0^1 f(x)\,dx \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i).$$

For the purpose of comparing with the other methods, use $h = 1/n$ as a rough analog of the 'mesh spacing' h.

For parts (a) through (d), answer the following prompts for each method:

(i) Compute the approximate value for π using the method with various values of h.

(ii) Characterize the error for each method as a function of h.

Use two different approaches:

- Create a clearly labeled log-log plot. Use the same plot (with a legend). Use the same plot for all four parts.
- Compute the *empirical order of convergence*. Use this procedure to estimate this quantity:

  We assume that the error depends on the mesh spacings h as $E(h) \approx Ch^p$ for some unknown power p. Taking the log of this approximate equality reveals a linear function in p:

  $$E(h) \approx Ch^p \quad \Longleftrightarrow \quad \log E(h) \approx \log(C) + p\log(h).$$

  You can now either do a least-squares fit for $\log C$ and p from a few data points $(h, E(h))$ (more accurate, more robust), or you can use just two grid sizes h_1 and h_2, and estimate the slope: (less accurate, less robust)

  $$p \approx \frac{\log(E(h_2)/E(h_1))}{\log(h_2/h_1)}.$$

  This is called the *empirical order of convergence* or *EOC*.

(iii) Is there a point beyond which decreasing h yields no further improvement?

 (continued on next page...)

Problem 3: Gaussian Quadrature (10 + 5 + 5 = 20 points)

(a) Let p be a real polynomial of degree n such that:

$$\int_a^b p(x)\, x^k\, dx = 0, \quad k = 0, \ldots, n-1.$$

(i) Show that the n zeros of p are real, simple and lie in the open interval (a, b).
Hint: Consider the polynomial $q_k(x) = (x - x_1)(x - x_2) \cdots (x - x_k)$, where x_i, $i = 1, \ldots, k$ are the roots of p in (a, b). Where does $p(x)q_k(x)$ change signs?

(ii) Show that the n-point interpolatory quadrature on $[a, b]$ whose nodes are the zeros of p has degree $2n - 1$.
Hint: Consider the quotient and remainder polynomials when a given polynomial is divided by p.

(b) Write a function that computes a given integral using Gaussian quadrature. Your Gaussian quadrature function should nominally have the following signature:

```
def gauss_quad(f, n, ...):
    # add code here
```

You can obtain the nodes using the following snippet:

```
nodes = scipy.scpecial.legendre(n).weights[:, 0]
```

(c) Use `gauss_quad` to integrate the following two functions for various quadrature orders $n = 1, 2, \ldots, 100$. For the integrals of both these functions, create a log-plot of error in the Gaussian quadrature versus the order. Does the error obey $E(h) \approx Ch^p$ for some p, in each of the cases?

(i) $f(x) = \sin 2\pi x$ on $[-1, 1]$

(ii) $g(x) = |x|$ on $[-1, 1]$

(continued on next page...)

Problem 4: Numerical Differentiation (15 points)

(a) (5 points) Given a sufficiently smooth function $f : \mathbb{R} \to \mathbb{R}$, use Taylor series to derive a second order accurate, one-sided difference approximation to $f'(x)$ in terms of the values of $f(x)$, $f(x+h)$ and $f(x+2h)$.

(b) (5 points) Write a Python function to implement this difference scheme for a function f defined on $[-1, 1]$.

(c) (5 points) Discretize $(-1, 1)$ using a uniformly spaced mesh with spacing $h = 2^{-k}, k = 3, \dots, 20$ and obtain the derivative at the sampled points using the above function for $f(x) = \sin x$. Obtain the error in the derivative using the max norm (i.e. error is measured to be the maximum of the absolute differences) and make a plot of error versus h. What is the expected order of convergence? What is the convergence rate that you obtain? Is the error monotonically decreasing? Explain your observations.

Problem 5: Initial Value Problems (20 points)

Consider the initial value problem

$$y' = -200\, t\, y^2, \quad \text{with} \quad y(0) = 1.$$

This IVP has the analytical solution

$$y(t) = \frac{1}{1 + 100\, t^2}.$$

Implement the following methods to numerically integrate the ODE from $t = 0$ to $t = 1$:

(a) Forward Euler method,

(b) Backward Euler method,

(c) Fourth-order Runge-Kutta method.

For each method do the following:

- Use step sizes of $h = 0.125, 0.25, 0.5, 1$.
- Plot the numerical solution y_h versus t for these various values of h. You may plot y on same plot for comparison.
- Compute the error at $t = 1$ between y_h and y for each h and plot versus h on a loglog plot.
- Based on the error data, what is the order of accuracy for each method?
- Explain whether each method is stable or not.

Your writeup should consist of plots of y_h against t (one plot for each method and each of the values of h), and three error plots–one per method. You should also provide comments on stability and accuracy of each method.