Problem 1. (20) Let

\[ A = \frac{1}{\eta^2} \text{tridiag}(1, -2, 1) \]

be a tridiagonal matrix (having the value \(-2\) in all diagonal entries, \(-1\) in all sub-diagonal and super-diagonal entries, and 0 elsewhere) of order \(n\), where \(\eta = \frac{1}{n+1}\). This problem concerns the initial value problem \(y' = Ay, \ y(0) = y_0\). We will consider the case \(n = 63\) here and \(y_0\) given by

\[ [y_0]_j = x_j(1 - x_j), \quad \text{where} \ x_j = j/(n + 1), \ j = 1, \ldots, n. \]

To do this problem, you will need to read MATLAB’s documentation for its ODE solvers.

a. Show that \(Av^{(k)} = \lambda_k v^{(k)}\) where for \(k = 1, \ldots, n\), the eigenvectors and eigenvalues of \(A\) are

\[ v^{(k)}_j = \sin k\pi j\eta, \ j = 1, \ldots, n, \quad \lambda_k = \frac{4}{\eta^2} \sin^2 \frac{k\pi\eta}{2}. \]

b. Use the result of part (a) to present a closed-form solution to the initial value problem. Note that this solution is a vector-valued function of \(t\) of length \(n\).

c. Use the four MATLAB routines \texttt{ode45}, \texttt{ode23s}, \texttt{ode15s} and \texttt{ode4} to solve the problem on the interval \([0, 2]\). The first three of these are generally robust methods of various types that use adaptive stepsize control; the last is a fourth-order Runge-Kutta method that does not do any error estimation, instead using a fixed user-specified set of time steps. This routine can be obtained from the web site

\[ \text{http://www.mathworks.com/support/tech-notes/1500/1510.html#fixed} \]

You should examine several aspects of these routines:

(i) for the first three, identify the number of time steps used to solve the problem, and compare the value of the computed solution \(y_N(2)\) (the discrete solution at the final time step \(N\), where \(N\) will vary with the solution algorithm) with the exact solution obtained from part (b). For this, you can evaluate the accuracy by plotting the error vector

\[ (y_N(2) - y(2)) / \|y(2)\|; \]

(ii) for \texttt{ode15s}, allow the solver to choose the order of the method, or, alternatively, restrict its order to be at most two;

(iii) for \texttt{ode4}, choose any step size that you want and describe in words what happens.

For parts (i) and (ii), there are a number of results to plot, and it may be possible to plot some of them on a single graph. You may also find it instructive to look at the absolute error \(y_N(2) - y(2)\) rather than the relative error. (Why?)

d. Give an explanation for the different outcomes you see for the solvers.

This problem actually corresponds to the numerical solution to the heat equation \(u_t = u_{xx}\) on the interval \(x \in [0, 1]\), where the spatial derivative is replaced by a discrete approximation on a spatial grid of width \(\eta\).
Problem 2. (10) This problem concerns the Monte-Carlo quadrature rule

\[ I(f) \equiv \int_0^1 f(x)dx \approx I_N^{(MC)}(f) \equiv \frac{1}{N} \sum_{i=1}^{N} f(x_i), \tag{1} \]

where \( \{x_i\} \) are uniformly distributed points in \([0, 1]\). We will use \( f(x) = \sin(\pi x) \).

a. In this setting, if \( X \) is a uniformly distributed random variable on the sample space \( \Omega = [0, 1] \), then \( \sin \pi X \) is also a random variable on \( \Omega \). What is the standard deviation of \( \sin \pi X \)?

b. Let \( E_N(f) = I(f) - I_N^{(MC)}(f) \). We know that this error behaves in a probabilistic sense like \( \sigma/\sqrt{N} \) as \( N \to \infty \), where \( \sigma \) is the standard deviation from part (a). Write a program that explores this issue for the sine function. That is, using logarithmic data in some way, produce a graphical image comparing the error \( E_N \) with \( \sigma/\sqrt{N} \) to see if the expected trend is visible. You may need to use very large \( N \) to discern the patterns. How do the trends shown compare to what one sees in conventional error bounds in scientific computation?

Problem 3. (10) Consider the function

\[ \rho(x) \equiv \begin{cases} 
4x & 0 \leq x < 1/2, \\
4 - 4x & 1/2 \leq x < 1.
\end{cases} \]

It is easily checked that this is a density function on \([0, 1]\). The accept/reject sampling method can be used to sample a random variable having the density function \( \rho \). Write a program that implements this method for \( \rho \). Generate a histogram of accepted samples to demonstrate that your implementation produces the correct density.

Problem 4. (15) We can approximate \( I(f) \) in (1) by sampling from a distribution with density function \( \rho \) from Problem 3. Under appropriate circumstances, this method, importance sampling, improves the performance of Monte-Carlo quadrature.

a. In particular, we want

\[ \hat{\sigma} = \left( \int_0^1 \frac{[f(x)/\rho(x) - I(f)]^2 \rho(x)dx}{\sigma} \right)^{1/2} \]

to be smaller than \( \sigma \). Show that this is true. You can do this analytically or computationally using MATLAB’s quadrature methods.

b. Using the sampling routine from Problem 3, repeat the experiment from Problem 2b to show that importance sampling improves performance.