1. (nonlinear pendulum)

(a) Write a program to solve the boundary value problem for the nonlinear pendulum as discussed in the text. See if you can find yet another solution for the boundary conditions illustrated in Figures 2.4 and 2.5.

See Matlab code nonlinear_pendulum.m. To see another solution, enter 3 when it asks which initial guess. (Initial guesses 1 and 2 lead to the results in Figures 2.4 and 2.5.) On the next page is a plot of the third solution that I found, along with the solution in part (b).

(b) Find a numerical solution to this BVP with the same general behavior as seen in Figure 2.5 for the case of a longer time interval, say, \( T = 20 \), again with \( \alpha = \beta = 0.7 \). Try larger values of \( T \). What does \( \max_i \theta_i \) approach as \( T \) is increased? Note that for large \( T \) this solution exhibits “boundary layers”.

I was able to get Newton’s method to converge with \( T = 20 \) by first solving the problem with \( T = 2\pi \) and then using that as the initial guess for the \( T = 20 \) problem. (In the code nonlinear_pendulum.m, you can first enter 2 when it asks which initial guess. When the code has run, type \texttt{thetasave=theta;}. Then run the code again and enter 0 when it asks which initial guess. Then type \texttt{thetasave} when it asks for theta.) I then used the solution from the \( T = 20 \) problem as the initial guess for problems with larger \( T \). For each of these problems I used 100 subintervals. The maximum value of \( \theta \) approached \( \pi \) as \( T \) was increased. For \( T = 50 \), the maximum value of \( \theta \) was surprisingly close to \( \pi \): 3.14159265349159.
2. (Richardson extrapolation) Use your code from problem 6 in assignment 1, or download the code from the course web page to do the following exercise. Run the code with \( h = .1 \) (10 subintervals) and with \( h = .05 \) (20 subintervals) and apply Richardson extrapolation to obtain more accurate solution values on the coarser grid. Record the \( L_2 \)-norm or the \( \infty \)-norm of the error in the approximation obtained with each \( h \) value and in that obtained with extrapolation.

Using \( h = .1 \), I got 0.0018 for the \( L_2 \)-norm of the error. Using \( h = .05 \), I got \( 4.2635e - 4 \) for the \( L_2 \)-norm of the error – a reduction of about a factor of 4, indicating second order accuracy. Therefore to do the extrapolation, I took \( (4/3) \) times the fine grid solution (at the even nodes) minus \( (1/3) \) times the course grid solution. I obtained an approximate solution on the course grid for which the \( L_2 \)-norm of the error was \( 5.4607e - 6 \).

Suppose you assume that the coarse grid approximation is piecewise linear, so that the approximation at the midpoint of each subinterval is the average of the values at the two endpoints. Can one use Richardson extrapolation with the fine grid approximation and these interpolated values on the coarse grid to obtain a more accurate approximation at these points? Explain why or why not?

Although the error in the approximation at the midpoint of each subinterval is \( O(h^2) \), the constant \( C_f \) multiplying this error is different from the constant \( C \) multiplying the \((h/2)^2\) error term in the fine grid approximation. Hence if one tries to take \( 4/3 \) times the fine grid approximation minus \( 1/3 \) times
the interpolated coarse grid approximation, the $O(h^2)$ error terms will not cancel. If one used a higher order interpolation method, such as quadratic interpolation, where the interpolation error is $O(h^3)$, then one should be able to use Richardson extrapolation to eliminate the $O(h^2)$ error terms. Alternatively, one could try to estimate the constant $C_I$ and relate it to $C$. Then one could eliminate the $O(h^2)$ error by taking $(4C_I)/(4C_I - C)$ times the fine grid approximation minus $C/(4C_I - C)$ times the interpolated value from the coarse grid.