AMATH 586 SPRING 2023
HOMEWORK 2 — DUE APRIL 21 ON GRADESCOPE BY 11PM

Be sure to do a `git pull` to update your local version of the `amath-586-2023` repository.
Homeworks must be typeset and uploaded to `Gradescope` for submission.
The submitted homework must include plots and descriptions of your code.
Code should be uploaded to `GitHub`.
You must include your name and `GitHub` username on your assignment.

Problem 1: Write a second-order accurate method to solve the following boundary-value problem

$$\begin{cases} (\kappa(x)u'(x))' = g(x), \\ u(0) = 1, \\ u(1) = 1, \end{cases}$$

where $\kappa(x) = \exp(-x/\epsilon) + \exp((x-1)/\epsilon)$, $g(x) = x(1-x)$. Numerically verify your method is second-order accurate. How will you do this if you do not know the solution explicitly? Plot solutions for increasingly small values of ϵ.

Problem 2: Nonlinear pendulum.

(a) Write a program to solve the boundary value problem for the nonlinear pendulum as discussed in LeVeque, Section 2.16. See if you can find yet another solution for the boundary conditions illustrated in Figures 2.4 and 2.5.

(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".

Problem 3: (a) Consider the function

$$u(x,y) = \cos(k\pi x)\sin(k\pi y)\exp(-(x-y)^2), \quad k \in \mathbb{N}. \tag{1}$$

Determine h_0, h_1, g_0, g_1 and f such that

$$\begin{cases} u_{xx}(x,y) + u_{yy}(x,y) = f(x,y), \\ u(x,0) = g_0(x), \\ u(x,1) = g_1(x), \\ u(0,y) = h_0(y), \\ u(1,y) = h_1(y). \end{cases} \tag{2}$$

(b) Construct a second-order accurate method for this problem and verify the order of accuracy numerically in both the ∞-norm and the grid 2-norm. How does the convergence depend on k?

Problem 4: Consider the iterative solution the linear system that results from (2) (when (1) is the solution).

(a) Implement the Jacobi iteration to solve the linear system.

(b) For a varying number of grid points, compare the convergence rate of the Jacobi iteration to the conjugate gradient algorithm.

(c) For a varying number of grid points, compare the convergence rate of the Jacobi iteration to the diagonally-preconditioned conjugate gradient algorithm.

Problem 5: Construct a fourth-order accurate method to solve (2) (when (1) is the solution) and verify the order of accuracy in the ∞-norm and the grid 2-norm. Is the conjugate gradient algorithm good for this problem? How about the Jacobi iteration?

```
## Julia
function prand(m)
  p = x -> -(2.0/3)*x.+4.0/3 .+ .5sin.(2*pi*x)
  B = 1.7
  out = fill(0.,m)
  for j = 1:m
    u = 10.
    y = 0.
    while u >= p(y)/B
      y = rand()
      u = rand()
    end
    out[j] = y
  end
  out
end
```

```
%% Matlab
function out = prand(m)
   p = @(x) -(2/3)*x + 4/3 + .5*sin(2*pi*x);
   B = 1.7;
   out = zeros(m,1);
   for j = 1:m
       u = 10.;
       y = 0.;
       while u >= p(y)/B
```

```
            y = rand();
            u = rand();
        end
        out(j) = y;
    end
end
```

```
## Python
import numpy as np

def psamp(m):
  p = lambda x: -(2.0/3)*x +4.0/3 + 0.5*np.sin(2*np.pi*x)
  B = 1.7
  out = np.zeros(m)
  for j in np.arange(m):
    u = 10.
    y = 0.
    while u >= p(y)/B:
      y = np.random.rand()
      u = np.random.rand()
    out[j] = y
  return out
```