Due Thursday, March 3rd at 11:59pm

Problem 1

Solve the following initial value problem:

$$\dot{x}(t) = \cos(t) - x \quad \text{with} \quad x(0) = 1. \tag{1}$$

It is easy to check that the true solution to this equation is

$$x_{\text{true}}(t) = \frac{1}{2}\left[\cos(t) + \sin(t) + e^{-t}\right]. \tag{2}$$

We will use this true solution to calculate the error of our approximations.

(1) Solve equation (1) from $t = 0$ to $t = 10$ with a time step of $\Delta t = 0.1$ using the forward Euler method. Make a 1×101 row vector containing all of your approximations of x and save it in a variable named `A1`. (Don't forget to use reshape in python.)

For each approximation x_n that you found with the forward Euler method, calculate the error $|x_n - x_{\text{true}}(t_n)|$ between your approximation and the true solution at the corresponding time and create a 1×101 row vector of these errors. Save this row vector in a variable named `A2`. (Don't forget to use reshape in python.)

(2) Solve equation (1) from $t = 0$ to $t = 10$ with a time step of $\Delta t = 0.1$ using the backward Euler method. At each step, you will have to solve an implicit equation for x_{n+1}. This equation should be easy to solve by hand. Make a 1×101 row vector containing all of your approximations of x and save it in a variable named `A3`. (Don't forget to use reshape in python.)

For each approximation x_n that you found with the backward Euler method, calculate the error $|x_n - x_{\text{true}}(t_n)|$ (notice the absolute value) between your approximation and the true solution at the corresponding time and create a 1×101 row vector of these errors. Save this row vector in a variable named `A4`. (Don't forget to use reshape in python.)

(3) Solve equation (1) from $t = 0$ to $t = 10$ using `ode45` (in MATLAB) or `solve_ivp` (in Python). Specify that the solver should produce approximations for the points `tspan = [0:0.1:10]` (in MATLAB) or using the `t_eval = np.arange(0, 10 + 0.1, 0.1)` option (in python). Make a 1×101 row vector containing all of your approximations of x and save it in a variable named `A5`. (Don't forget to use reshape in python.)

For each approximation x_n that you found with `ode45` or `solve_ivp`, calculate the error $|x_n - x_{\text{true}}(t_n)|$ between your approximation and the true solution at the corresponding time and create a 1×101 row vector of these errors. Save this row vector in a variable named `A6`. (Don't forget to use reshape in python.)

Problem 2

Consider the initial value problem

$$\dot{x}(t) = a\sin(x) \quad \text{with} \quad x(0) = \pi/4, \tag{3}$$

where a is a constant. You can check that the solution to this differential equation is

$$x_{\text{true}}(t) = 2\arctan\left(\frac{e^{at}}{1+\sqrt{2}}\right). \tag{4}$$

We will use this problem to explore the accuracy of the forward and backward Euler methods.

Throughout this problem, use $a = 8$.

(1) Solve equation (3) from $t = 0$ to $t = 2$ with a time step of $\Delta t = 0.01$ using the forward Euler method. Make a 1×201 row vector containing all of your approximations of x and save it in a variable named `A7`. (Don't forget to use reshape in python.)

Find the maximum error between your approximations and the true solution. That is, find $\max(|x_n - x_{\text{true}}(t_n)|)$. Save this maximum error in a variable named `A8`. Repeat for $\Delta t = 0.001$, and save the ratio of the old maximum error to the new maximum error as `A9`.

(2) Solve equation (3) from $t = 0$ to $t = 2$ with a time step of $\Delta t = 0.01$ using the backward Euler method. At each step of the backward Euler method, you will need to solve an equation of the form $x_{n+1} = x_n + a\Delta t\sin(x_{n+1})$. This equation cannot be solved by hand, but we can use forward Euler to calculate (or more accurately "predict") x_{n+1}. This is called the predictor-corrector method. In one iteration, first approximate x_{n+1} using forward Euler, then use it to approximate $\sin(x_{n+1})$, finally, use $x_{n+1} = x_n + a\Delta t\sin(x_{n+1})$ to approximate the corrected x_{n+1}. Make a 1×201 row vector containing all of your approximations of x and save it in a variable named `A10`. (Don't forget to use reshape in python.)

Find the maximum error between your approximations and the true solution. That is, find $\max(|x_n - x_{\text{true}}(t_n)|)$. Save this maximum error in a variable named `A11`. Repeat for $\Delta t = 0.001$, and save the ratio of the old maximum error to the new maximum error as `A12`.

(3) Solve equation (3) from $t = 0$ to $t = 2$ using `ode45` or `solve_ivp`. Specify that the solver should produce approximations for the points `tspan = [0:0.01:2]` (in MATLAB) or using the `t_eval = np.arange(0, 2 + 0.01, 0.01)` option (in python). Make a 1×201 row vector containing all of your approximations of x and save it in a variable named `A13`. (Don't forget to use reshape in python.)

Find the maximum error between your approximations and the true solution. That is, find $\max(|x_n - x_{\text{true}}(t_n)|)$. Save this maximum error in a variable named `A14`. Repeat for $\Delta t = 0.001$, and save the ratio of the old maximum error to the new maximum error as `A15`.