ME 11Lab: MPX Compilation Problems

Prepared by: Asst.Prof. Ryan M. Cabrera
Email: `rmcabrera.updme@gmail.com`

Instructions:

- Read each problem carefully.
- You are strictly prohibited from copying the work of students in other sections of the same subject.
- HOWEVER, you may collaborate with your partner ONLY.
- Save your work (as M-files) and name it accordingly, for instance: `MPX0_A_Prob1.m`. Compile each MPX set and send to me through my email. For instance, one compilation for all MPX0-A problems; another compilation for MPX0-B problems, and so on.

MPX0-A Problems:

1. Create a script file that will require the user to input the following functions, and their common interval $[a, b]$. The output must be a combined plot of these functions over the interval $[a, b]$, for real-values only. Hint: Determine the most appropriate domain such that no function will get values over $[a, b]$ which are either undefined or imaginary values.

   (a) $f(x) = x^3 - 4x + 5x - 1$
   (b) $g(x) = \frac{x^2+1}{e^x-1}$
   (c) $h(x) = \frac{x^2-1}{\sqrt{2x-1}}$

2. Create a script file that will require that will require the user to input the function, say $f(x)$, its interval $[a, b]$, and intermediate value K, such that the following is satisfied:

$$f(a) < K < f(b)$$

   where $K = f(c)$, and $c \in (a, b)$. The output will be a statement whether the function satisfies the *intermediate value theorem* (IVT) or not. Use the following test functions and their corresponding given intervals:

   (a) $f(x) = x \cos x - 2x^2 + 3x - 1$, $[0.2, 0.4]$
   (b) $f(x) = (x - 2)^2 - \ln x$, $[e, e + 1]$

(c) $f(x) = 2x\cos 2x - (x-2)^2$, $[3, 4]$

(d) $f(x) = x - (\ln x)^2$, $[4, 5]$

3. Create a script file that will require the user to input the function, $f(x)$ and its interval $[a, b]$. The output will be its plot and the values of c_1 and c_2, such that

$$f(c_1) \leq f(x) \leq f(c_2)$$

where $f(c_1)$ and $f(c_2)$ are the minimum and maximum values of $f(x)$ over the interval $[a, b]$.

4. The kinematic equations of a projectile is given by:

$$x(t) = x_o + v_o \cos\theta_o t$$
$$y(t) = y_o + v_o \sin\theta_o t - \frac{1}{2}gt^2$$

(a) Create a function file that will require the following input variables:

$$(x_o, y_o, v_o, \theta_o, t)$$

The output will be the kinematic coordinates $[x, y]$ of the projectile. Note the following nomenclature: (x_o, y_o) are the coordinates of the projectile's initial position, in [m], v_o is the initial velocity, in [m/s], θ_o is the angle of release, in [°] and, t is the time of flight, in [s]. Name this function file `ProjectileMotion`.

(b) Create a script file that will make use of the function `ProjectileMotion`; the input should be (x_o, y_o), v_o, for various angles of release, $0 \leq \theta_o \leq 90$, over the time interval $0 \leq t \leq t_\text{f}$, where t_f total time of flight.

MPX0-B Problems:

1. Create a script file that will require the user to input the function $f(x)$ and its interval $[a, b]$. Also, the tolerance ε must be an input. Using the *bisection method*, obtain the value of $c \in (a, b)$ such that

$$f(x)|_{x=c} = 0$$

Note: To ensure that the solution c exists in (a, b), incorporate the IVT checker developed in Problem 2 of MPX0-A. Also, plot the function $f(x)$ over the interval $[a, b]$.

2. Redo the previous problem (Prob. 1 of MPX0-B) using the *secant method*.

MPX1-A Problems:

1. Create a script file that will require the user to input a function $f(x)$, the *pivot* point x_o, and the order of Taylor polynomial n; also, the interval $[a, b]$ must also be an input. The output will be the Taylor approximating polynomial using the following:

(a) $P_{\mathrm{M}}(x) \to$ the Taylor polynomial using the MATLAB® built-in function `taylor`.

(b) $P_u(x) \to$ the Taylor polynomial defined by the following equation:

$$P_u(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_o)}{k!}(x - x_o)^k$$

(c) Plot of $f(x)$, $P_{\mathrm{M}}(x)$, and $P_u(x)$ versus $x \in [a, b]$. Place x- and y- axes labels as well as plot legend and title.

2. Solve the following ODEs using MATLAB® built-in function called `dsolve`:

(a) $y' - ty = 0$

(b) $y' - ty = 0$; $y(0) = 5$

(c) $\frac{\mathrm{d}^2 y}{\mathrm{d}x^2} - \cos 2x + y = 0$, $y(0) = 1$, $y'(0) = 0$.

(d) $\left(\frac{\mathrm{d}x}{\mathrm{d}t} + x\right)^2 - 1 = 0$, $x(0) = 0$.

MPX2-A Problems:

1. Create a script file that will require the user to input a function $f(x)$ and the vector $\{\mathbf{X}\}$ containing nodes X_j. These nodes serve as the interpolating points for the *Vandermonde* polynomial $\mathcal{V}_n(x)$ of order n, where $(n+1)$ is the total number of nodes. The output should be the following:

   (a) The Vandermonde polynomial $\mathcal{V}_n(x)$.

   (b) The plot of $f(x)$ and $\mathcal{V}_n(x)(x)$ versus $[a, b]$, where $a = X_1$ and $b = X_{n+1}$.

2. Create a script file that will require the user to input a function $f(x)$ and the vector $\{\mathbf{X}\}$ containing nodes X_j. These nodes serve as the interpolating points for the *Lagrange* polynomial $\mathcal{L}_n(x)$ of order n, where $(n+1)$ is the total number of nodes. The output should be the following:

   (a) The Lagrange polynomial $\mathcal{L}_n(x)$.

   (b) The plot of $f(x)$ and $\mathcal{L}_n(x)(x)$ versus $[a, b]$, where $a = X_1$ and $b = X_{n+1}$.

MPX2-B Problems:

1. Create a script file that will require the user to input the x_j and $f(x_j)$ data. The output will be to compute the approximation of a first-order derivative of $f(x)$ at each node x_j, *i.e.*, to compute for $f'(x_j)$ using the following:

   (a) three-point endpoint (forward) at x_1.

   (b) three-point midpoint at x_j, for $j = 2, 3, ..., N$

   (c) three-point endpoint (backward) at x_{N+1}

   Note: N is the number of subdivisions over the interval $[x_1, x_{N+1}]$; which means there are $N+1$ points in total. **Suggestion:** Create function files for each of the approximation schemes.

MPX3 Problems:

1. One problem on numerical solutions of an ODE problem of the *first* order using Taylor and Runge-Kutta methods (to follow).....

MPX4 Problems:

1. One problem on numerical solutions of an ODE problem of the *second* order using Taylor and Runge-Kutta methods (to follow).....