Problem 1 Write, test and debug a matlab function

function [p, q] = pcoeff( t, n, k )
% t: solution times t(1) < t(2) < ... < t(n) < t( n+1 )
% n+1: new time step
% k: number of previous steps t(n-k+1)..t(n)

which computes coefficients p and q for the k-step predictor-corrector method

\[ v_{n+1} = u_n + \int_{t_n}^{t_{n+1}} p(t)dt = u_n + p_1f_n + p_2f_{n-1} + \cdots + p_kf_{n-k+1}, \]

\[ u_{n+1} = u_n + \int_{t_n}^{t_{n+1}} q(t)dt = u_n + q_1f(t_{n+1}, v_{n+1}) + q_2f_n + \cdots + q_kf_{n-k+2}. \]

Here \( p(t) \) is the degree \( k - 1 \) polynomial which interpolates the values \( f_j = f(t_j, u_j) \) for \( n - k + 1 \leq j \leq n \) and \( q(t) \) is the degree \( k - 1 \) polynomial which interpolates the values \( f_j \) for \( n - k + 2 \leq j \leq n \) and also the predicted slope \( f(t_{n+1}, v_{n+1}) \) at \( t_{n+1} \). Thus

\[ p_j = \int_{t_n}^{t_{n+1}} \prod_{i \neq j} \frac{t - t_{n-i+1}}{t_{n-j+1} - t_{n-i+1}} \]

and

\[ q_j = \int_{t_n}^{t_{n+1}} \prod_{i \neq j} \frac{t - t_{n-i+2}}{t_{n-j+2} - t_{n-i+2}} \]

for \( 1 \leq j \leq k \). Tabulate the coefficients \( p \) and \( q \) with constant step size \( h = 1 \) and \( k \leq 5 \) and verify against Adams-Bashforth and Adams-Moulton methods.

(Hint: the integrands are polynomials of degree \( k - 1 \) for which \( \lceil k/2 \rceil \) Gaussian integration points and weights will give an exact result.)
Problem 2 Write, test and debug a matlab function

function [ t, u ] = pcode(a, b, ua, f, r, k, N)
% a,b: interval endpoints with a < b
% ua: vector u_1 = y(a) of initial conditions
% f: function handle f(t, u, r) to integrate
% r: parameters to f
% k: number of previous steps to use at each regular time step
% N: total number of time steps,
% t: output times for numerical solution u_n ~ y(t_n), t(1) = a, t(N) = b
% u: numerical solution at times t
which uses pcoeff to approximate the solution vector y(t) of the vector initial value problem

\[ y' = f(t, y, r) \]
\[ y(a) = y_a \]

by the family of methods you derived in problem 1, with \( u_1 = y_a \). Start with \( k_1 = 1 \) and a tiny step size

\[ h_1 = (b - a) \left( \frac{h}{b-a} \right)^{k/2} \]

which brings the one-step error in line with the \( O(h^k) \) error. Increase the step size smoothly (e.g. by \( h_1 \leftarrow (1 + 1/k)h_1 \)) and increase \( k_1 \) (e.g. by steps of 1 up to \( k \)) until \( h_1 \geq h = (b-a)/(N-1) \) and then continue with uniform step sizes. (To save CPU time, (a) when the most recent \( k \) step sizes are uniform, the predictor-corrector coefficients \( p \) and \( q \) can be frozen and (b) many values of \( f \) can be saved rather than re-evaluated.)

(a) Use pcode.m with odd \( k = 1 \) through 11 and \( N = 10000, 20000, 40000, 80000 \) and 160000 to approximate the final solution vector \( u(T) \) of the initial value problem derived in problem 4 of problem set 8. Tabulate the errors

\[ E_{kN} = \max_{1\leq j\leq 4} |u_j(T) - u_j(0)|. \]

Estimate the constant \( C_k \) such that the error behaves like \( C_k h^k \).

(b) Measure the CPU time for each run and estimate the total CPU time necessary to obtain an orbit which is periodic to three-digit, six-digit and twelve-digit accuracy.

(c) Plot some inaccurate solutions and some accurate solutions and draw conclusions about values of \( k \) which give three, six or twelve digits of accuracy for minimal CPU time.

(d) Compare to the results of euler.m and idec.m.
**Problem 3** Consider a differential equation
\[ y'(t) = f(t, y(t)), \]
where \( f \) satisfies the condition
\[ (u - v)(f(t, u) - f(t, v)) \leq 0 \]
for all \( u \) and \( v \).

(a) Suppose \( U(t) \) and \( V(t) \) are exact solutions. Show that
\[ |U(t) - V(t)| \leq |U(0) - V(0)| \]
for all \( t \geq 0 \).

(b) Suppose \( W \) satisfies a perturbed differential equation
\[ W'(t) = f(t, W(t)) + r(t) \]
for \( t \geq 0 \). Show that
\[ |U(t) - W(t)| \leq |U(0) - W(0)| + \int_0^t |r(s)|ds \]
for \( t \geq 0 \).

(c) Show that two numerical solutions \( u_n \) and \( v_n \) generated by implicit Euler (e.g. with different initial values) satisfy
\[ |u_n - v_n| \leq |u_0 - v_0| \]
for all \( n \geq 0 \).

(d) Show that the local truncation error \( \tau_{n+1} \) of the implicit Euler method
\[ u_{n+1} = u_n + hf(t_{n+1}, u_{n+1}) \]
is given by
\[ \tau_{n+1} = \frac{y_{n+1} - y_n}{h} - f(t_{n+1}, y_{n+1}) = -\frac{h}{2}y''(\zeta) \]
where \( y_n = y(t_n) \) is the exact solution and \( \zeta \) is an unknown point.

(e) Show that the numerical solution \( u_n \) generated by implicit Euler with \( u_0 = y_0 \) satisfies
\[ |u_n - y_n| \leq nh\tau \]
for \( 0 \leq nh < \infty \), where \( \tau = Mh/2 \) and \( |y''| \leq M \).
Problem 4 Consider the linear initial value problem

\[ y' = -L(y(t) - \varphi(t)) + \varphi'(t) \]

\[ y(0) = y_0 \]

where \( \varphi(t) = \cos(30t) \).

(a) Solve the initial value problem exactly.

(b) Use euler.m to solve the initial value problem with \( y(0) = 2 \) for \( 0 \leq t \leq 1 \) with \( L = 10^k \) for \( k = 1 \) to 5. For each \( L \) use \( h = 10^{-j} \) with \( j = 1 \) to 6. Tabulate the errors.

(c) Write a matlab script ieuler.m which uses the implicit Euler method to solve the initial value problem with \( y(0) = 2 \) for \( 0 \leq t \leq 1 \) with \( L = 10^k \) for \( k = 1 \) to 5. For each \( L \) use \( h = 10^{-j} \) with \( j = 1 \) to 6. Tabulate the errors. Plot an accurate solution for each \( L \).
Problem 5 (cf. BFB 6.1.12) Write, test and debug a matlab code

function [t, u] = solveinteq( a, b, kernel, rhs, p, n )
% a, b: endpoints of interval
% kernel: function handle for kernel K = kernel( t, s ) of integral equation
% rhs: function handle for right-hand side f = rhs( t, p ) of integral equation
% p: parameters for rhs
% n: number of quadrature points and weights
% t: evaluation points in [a,b]
% u: solution values at evaluation points

which uses \( n \)-point Gaussian quadrature points \( t_i \) and weights \( w_i \) on \([a,b]\) (generated by gaussint.m) to approximate the solution \( y(t) \) of the integral equation

\[
y(t) + \int_a^b K(t, s)y(s) \, ds = f(t, p)
\]

on the interval \( a \leq t \leq b \). Your code should set up the \( n \times n \) linear system

\[
u_i + \sum_{j=1}^{n} K(t_i, t_j)w_ju_j = f(t_i, p)
\]

for approximate values \( u_i \approx y(t_i) \) and solve it by Gaussian elimination with partial pivoting.

(a) Suppose \([a, b] = [0, 1] \) and the kernel \( K \) is given by \( K(t, s) = \cos(t) \sin(s) \). For any positive real number \( m \), find a right-hand side \( f(t, m) \) such that the exact solution \( y(t) \) of the integral equation (1) is given by \( y_m(t) = \cos(mt) \).

(b) Solve the problem in (a) numerically by solveinteq, using even \( n = 2 \) through 16 and odd integers \( m = 1 \) through 9. Tabulate the errors at integration points

\[
E_n = \max_{1 \leq i \leq n} |u_i - y_m(t_i)|
\]

vs. \( m \) and \( n \).

(c) For an arbitrary right-hand side \( f \) and the specific kernel \( K(t, s) = \cos(t) \sin(s) \) in (a), find a formula for the exact solution \( u \) of the linear system of equations (2).

(d) Use the error formula for Hermite interpolation to show that the local truncation error in (a)

\[
\tau_i = y(t_i) + \sum_{j=1}^{n} w_jK(t_i, t_j)y(t_j) - f(t_i)
\]

is bounded by

\[
|\cos(t_i)| \left( \int_0^1 \prod_{i=1}^{n} (t - t_i)^2 \, dt \right) \frac{|\phi^{(2n)}(\xi)|}{(2n)!}
\]
as $n \to \infty$, where $v(s) = \sin(s)y(s)$.

(e) Assume that all the derivatives of the exact solution $y$ in (a) are bounded by 

$$|y^{(n)}(t)| \leq m^n$$

for some fixed $m > 0$. Use (d) and (c) to prove that 

$$E_n \leq 2 \max_i |\tau_i| \to 0$$

as $n \to \infty$. 