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 \(t(n-k+1)\ldots t(n)\)

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

\[v_{n+1} = u_n + \int_{t_n}^{t_{n+1}} p(t) \, dt = u_n + p_1 f_n + p_2 f_{n-1} + \cdots + p_k f_{n-k+1},\]
\[u_{n+1} = u_n + \int_{t_n}^{t_{n+1}} q(t) \, dt = u_n + q_1 f(t_{n+1}, v_{n+1}) + q_2 f_n + \cdots + q_k f_{n-k+2} .\]

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 \([k/2]\) Gaussian integration points and weights will give an exact result.)

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 \(u_n \sim y(t_n)\), \(t(1) = a, t(N) = b\)
% u: numerical solution at times \(t\)

which uses \texttt{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 \texttt{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 6 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 \texttt{rich.m} and \texttt{idec.m}.

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 \( |y''| \leq M \).

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 \( L \).