1. (BFB 5.6.13) The initial-value problem
\[ y' = e^y, \quad 0 \leq t \leq 0.20, \quad y(0) = 1, \]
has solution
\[ y(t) = 1 - \ln(1 - et) \]
Applying the three-step Adams-Moulton method to this problem is equivalent to finding the fixed point \( w_{i+1} \) of
\[ g(w) = w_i + \frac{h}{24} (9e^w + 19e^{w_i} - 5e^{w_i-1} + e^{w_i-2}) . \]
(a) With \( h = 0.01 \), obtain \( w_{i+1} \) by functional iteration for \( i = 2, \ldots, 19 \), using exact starting values \( w_0, w_1 \) and \( w_2 \). At each step use \( w_i \) initially to approximate \( w_{i+1} \).
(b) Will Newton’s method speed the convergence over functional iteration?

2. (BFB 5.6.16) (a) Derive two-step Adams-Bashforth by using the Lagrange form of the interpolating polynomial.
(b) Derive four-step Adams-Bashforth by using Newton’s backward-difference form of the interpolating polynomial.

3. Suppose \( y(t) \) is the exact solution of the initial value problem
\[ y'(t) = f(t, y(t)), \]
\[ y(0) = y_0, \]
and \( u(t) \) is any approximation to \( y(t) \) with \( u(0) = y(0) \). Define the error \( e(t) = y(t) - u(t) \).
(a) Show that \( e(t) \) satisfies the initial value problem
\[ e'(t) = f(t, u(t) + e(t)) - u'(t) \]
\[ e(0) = 0 \]
(b) Suppose \( f(t, y) = \lambda y \) for some constant \( \lambda \). Solve the initial value problem from (a) exactly to show that \( u(t) + e(t) = y(t) \).

4. Define a family of explicit Runge-Kutta methods parametrized by order \( p \), by applying \( p - 1 \) passes of deferred correction to \( p \) steps of Euler’s method. I.e. starting from \( u_n \) define the uncorrected solution by
\[ u^1_{n+j+1} = u^1_{n+j} + hf(t_{n+j}, u^1_{n+j}) \]
for \( 0 \leq j \leq p - 1 \). Let \( u(t) = U_1(t) \) be the degree-\( p \) polynomial that interpolates the \( p + 1 \) values \( u^1_{n+j} \) at the \( p + 1 \) points \( t = t_{n+j} \) for \( 0 \leq j \leq p \). Solve the error equation from question 3 by Euler’s method, yielding approximate errors \( e^1_{n+1}, e^1_{n+2}, \ldots, e^1_{n+p} \). Produce a second-order accurate corrected solution
\[ u^2_{n+j} = u^1_{n+j} + e^1_{n+j} \]
for \(1 \leq j \leq p\). Repeat the procedure to produce \(u_{n+1}^2, \ldots, u_{n+p}^p\).

(a) Verify that \(p = 1\) gives Euler’s method.

(b) For \(p = 2\) express your method as a Runge-Kutta method in the form

\[
\begin{align*}
  k_1 &= f(t_n, u_n) \\
  k_2 &= f(t_n + c_2 h, u_n + 2ha_21k_1) \\
  k_3 &= f(t_n + c_3 h, u_n + 2h(a_{31}k_1 + a_{32}k_2)) \\
  u_{n+2} &= u_n + 2h(b_1k_1 + b_2k_2 + b_3k_3).
\end{align*}
\]

Find all the constants \(c_i, a_{ij}\) and \(b_j\) and arrange them in a Butcher array.

(c) For \(p = 2\), ignore the \(t\) argument of \(f(t, u)\) and Taylor expand \(k_2(h)\) and \(k_3(h)\) to \(O(h^2)\). Show that your method has local truncation error \(\tau = O(h^2)\) and find the coefficient of the \(O(h^2)\) term.

(d) For arbitrary \(p\), verify that your method is equivalent to using fixed point iteration to solve an implicit Runge-Kutta method.

5 Write, test and debug a matlab function

```matlab
function yb = idec(a, b, ya, f, r, p, n)
% a,b: interval endpoints with a < b
% ya: vector y(a) of initial conditions
% f: function handle f(t, y) to integrate (y is a vector)
% r: parameters to f
% p: number of euler substeps / correction passes
% n: number of time steps
% yb: output approximation to the final solution vector y(b)

which approximates the final solution vector \(y(b)\) of the vector initial value problem

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

by the method you derived in problem 4, with \(u_0 = y_a\).

(a) Use idec.m with orders \(p = 1\) through 7 and \(N = 10000, 20000, 40000\) and 80000 steps to approximate the final solution vector \(u(T)\) of the initial value problem derived in problem 6 of problem set 8. Tabulate the errors

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

Estimate the constant \(C_p\) such that the error behaves like \(C_p h^p\).

(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 the order \(p\) which give three, six or twelve digits of accuracy for minimal CPU time.