553.481/681 Numerical Analysis

Homework 6 Solutions
**Problem 1.** This problem concerns the general 2nd-order Runge-Kutta method

\[
\mathbf{F}(t, \mathbf{y}; h, \mathbf{f}) = \gamma_1 \mathbf{f}(t, \mathbf{y}) + \gamma_2 \mathbf{f}(t + \alpha h, \mathbf{y} + \beta h \mathbf{f}(t, \mathbf{y}))
\]

with

\[
\gamma_1 + \gamma_2 = 1, \quad \gamma_2 \alpha = \gamma_2 \beta = \frac{1}{2}
\]

(a) Show that the truncation error is minimized for the choice \( \gamma_2 = 3/4 \). We call this the “optimal 2nd-order Runge-Kutta method”.

(b) Modify the MATLAB script heun.m to implement instead the optimal 2nd-order method in part (a). Apply your code to the following test problem:

\[
\dot{y} = y - e^{-t} y^2, \quad y(1) = e
\]

with exact solution \( Y(t) = e^t / t \) on the time-interval \([1, 3]\), using a number of steps \( N = 4, 8, 16, \ldots, 2048\). By considering the ratio of errors in the final value \( y(3) \) verify that this method is indeed asymptotically 2nd-order.

(c) For the same test problem as in part (b), plot the error over the time interval \([1, 3]\) for the optimal 2nd-order Runge-Kutta method. Also plot the error for Heun’s method applied to the same problem. Are the errors for the “optimal” method always smaller than those for Heun?

(a) The 2nd-order Runge-Kutta iterate for the ODE \( \mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t)) \) is \( y_{n+1} = y_n + h \mathbf{F}(t_n, y_n; h, \mathbf{f}) \).

The truncation error is, with \( y_n = \mathbf{y}(t_n) \),

\[
\mathbf{T}_n = \mathbf{y}(t_{n+1}) - y_{n+1} = \mathbf{y}(t_{n+1}) - \mathbf{y}(t_n) - h \mathbf{F}(t_n, \mathbf{y}; h, \mathbf{f}).
\]

By Taylor’s expansion,

\[
y(t_{n+1}) = y(t_n) + h y'(t_n) + \frac{h^2}{2} y''(t_n) + \frac{h^3}{6} y'''(t_n) + o(h^3)
\]

\[
\begin{align*}
= & y(t_n) + h \mathbf{f}(t_n, y_n) + \frac{h^2}{2} (\mathbf{f}(t_n, y_n) + \gamma_1 \mathbf{f}(t_n, y_n) + \gamma_2 \mathbf{f}(t_n, y_n)) + \frac{h^3}{6} \left[ \gamma_1 \mathbf{f}(t_n, y_n) + \gamma_2 \mathbf{f}(t_n, y_n) + \gamma_1 \mathbf{f}(t_n, y_n) + \gamma_2 \mathbf{f}(t_n, y_n) \right] + o(h^3).
\end{align*}
\]

So with \( \gamma_1 + \gamma_2 = 1, \gamma_2 \alpha = \gamma_2 \beta = \frac{1}{2} \),

\[
\mathbf{T}_n = (1 - \gamma_1 - \gamma_2) \mathbf{f}(t_n, y_n) + \left( \frac{1}{2} - \gamma_2 \alpha \right) \mathbf{f}_x(t_n, y_n) + \left( \frac{1}{2} - \gamma_2 \beta \right) \mathbf{f}_y(t_n, y_n) \mathbf{f}(t_n, y_n) h^2
\]

\[
+ \left( \frac{1}{3} - \gamma_2 \alpha \beta \right) \mathbf{f}_{xy}(t_n, y_n) \mathbf{f}(t_n, y_n) + \left( \frac{1}{6} - \gamma_2 \beta^2 \right) \mathbf{f}_{y}(t_n, y_n) \mathbf{f}(t_n, y_n) h^2
\]

\[
+ \left( \frac{1}{6} - \frac{1}{2} \gamma_2 \alpha^2 \right) \mathbf{f}_{tx}(t_n, y_n) + \left( \frac{1}{3} \right) \mathbf{f}_{yy}(t_n, y_n) \mathbf{f}(t_n, y_n) \mathbf{f}(t_n, y_n) + \left( \frac{1}{6} \right) \mathbf{f}_{tt}(t_n, y_n)
\]

\[
+ \left( \frac{1}{6} \right) \mathbf{f}_{x}(t_n, y_n) \mathbf{f}_y(t_n, y_n) + \left( \frac{1}{6} \right) \mathbf{f}_y(t_n, y_n) \mathbf{f}(t_n, y_n) h^3 + o(h^3).
\]

Denote \( \mathbf{c}(\gamma_2) = \begin{pmatrix} 1/2 - \gamma_2 \alpha/2 - \gamma_2 \beta/2 \\ \frac{1}{6} - \frac{1}{4} \gamma_2 \end{pmatrix} \), \( \mathbf{g}(t_n, y_n; \mathbf{f}) = \begin{pmatrix} \mathbf{f}_{tx}(t_n, y_n) \\ \mathbf{f}_{xy}(t_n, y_n) \mathbf{f}(t_n, y_n) \\ \mathbf{f}_{t}(t_n, y_n) \mathbf{f}_y(t_n, y_n) \\ \mathbf{f}_{y}(t_n, y_n) \mathbf{f}(t_n, y_n) \\ \mathbf{f}_{yy}(t_n, y_n) \mathbf{f}(t_n, y_n) \end{pmatrix} \), then

\[
|\mathbf{T}_n| = |\mathbf{c}^T \mathbf{g}(t_n, y_n; \mathbf{f})| h^3 + o(h^3) \leq \|c(\gamma_2)\| \cdot \|\mathbf{g}(t_n, y_n; \mathbf{f})\| h^3 + o(h^3) \quad \text{(by the Cauchy-Schwartz inequality)}.
\]

We want a minimal upper bound for arbitrary \( \mathbf{f} \), which is done by minimizing \( \|\mathbf{c}(\gamma_2)\| \), or, equivalently, minimizing \( \|\mathbf{c}(\gamma_2)\|^2 \):

\[
\|\mathbf{c}(\gamma_2)\|^2 = \left( \frac{1}{6} - \frac{1}{4\gamma_2} \right)^2 + \left( \frac{1}{3} - \frac{1}{4\gamma_2} \right)^2 + \left( \frac{1}{6} - \frac{1}{8\gamma_2} \right)^2 + \left( \frac{1}{6} \right)^2 + \left( \frac{1}{6} \right)^2
\]
\[
\frac{3}{2} \left( \frac{1}{3} - \frac{1}{4\gamma_2} \right)^2 + \frac{1}{18},
\]

The minimum is attained for \( \gamma_2 = \frac{3}{4} \). Alternatively, this choice sets to zero the largest number of terms proportional to \( h^3 \) in the truncation error.

(b) Modified `heun.m`:

```matlab
function [ t,y ] = heun(f,tspan,y_0,N_s,yes)
% solve the ODE dy/dt = f(t,y) by Heun’s method with N_s steps

% set up initial values

% define the time steps

% define the number of steps

% start the loop

% if yes == 1

% define the order

% plot the results

% return

% define the function

% define the time span

% define the initial condition

% define the order

% perform the calculation

% calculate the error

Using the following code one can see the order is 2.
```
Heun method
\[ N_s = 4, [y_1(1,:), Y_1(1,:), E_1(1), NaN] \]
for \( i = 2:10 \)
\[ N_s = 2^{(i+1)}, [y_1(i,:), Y_1(i,:), E_1(i), \log_2(E_1(i-1)/E_1(i))] \]
end

Heun approximation, exact solution, error magnitude, base-2 log of error ratio

(c) We find for errors vs step size:

\[ \begin{align*}
&\text{for } i = 1:10, \\
&\quad N_s = 2^{(i+1)}; \\
&\quad [t, y] = \text{mrk2}(f, tspan, y_0, N_s, 0); \\
&\quad s = y(1+N_s,:); \\
&\quad y_1(i,:) = s; \\
&\quad S = \text{feval}(Y, t(1+N_s)); \\
&\quad Y_1(i,:) = S; \\
&\quad E_1(i) = \text{norm}(s - S); \\
&\end{align*} \]
end

err2 = y - Y(t);

optimal 2nd-order Runge-Kutta
\[ \begin{align*}
&\text{for } i = 2:10 \\
&\quad N_s = 2^{(i+1)}, [y_1(i,:), Y_1(i,:), E_1(i), \log_2(E_1(i-1)/E_1(i))] \\
&\end{align*} \]
end

MRK2 approximation, exact solution, error magnitude, base-2 log of error ratio

pause

figure
plot(t, err1, '-b', t, err2, '-r')
legend('Heun method', 'optimal RK2', 'Location', 'NorthWest')
**Problem 2.** (a) Consider the three-stage Runge-Kutta formula

\[ F(t, y, h, f) = \gamma_1 k_1 + \gamma_2 k_2 + \gamma_3 k_3 \]

\[ k_1 = f(t, y), \quad k_2 = f(t + \alpha_2 h, y + h \beta_21 k_1), \quad k_3 = f(t + \alpha_3 h, y + h (\beta_31 k_1 + \beta_32 k_2)) \]

Determine the set of equations that the coefficients \( \{\gamma_j, \alpha_j, \beta_{ji}\} \) must satisfy if the formula is to be of order 3.

(b) The Runge-Kutta integration scheme with coefficients

\[ \alpha_2 = 1/2, \quad \beta_{21} = 1/2 \]
\[ \alpha_3 = 3/4, \quad \beta_{31} = 0, \quad \beta_{32} = 3/4, \]
\[ \gamma_1 = 2/9, \quad \gamma_2 = 1/3, \quad \gamma_3 = 4/9 \]

is employed as part of the Bogacki-Shampine algorithm implemented in MATLAB’s integrator ode23. Use the result of part (a) to check whether this scheme satisfies the conditions to be third-order.

(c) Write a MATLAB script `rk3.m` to implement the scheme in part (b). Apply your code to solve the initial-value problem

\[ \dot{y}_1 = -y_2, \quad \dot{y}_2 = y_1, \quad y_1(0) = 1, \quad y_2(0) = 0 \]

with solution \( Y_1(t) = \cos(t), \quad Y_2(t) = \sin(t) \) on the time-interval \([0, 2\pi]\), using a number of steps \( N = 4, 8, 16, ..., 2048\). Use the results to estimate the order of convergence. Does your numerical finding agree with your theoretical conclusion in part (b)?

(a) The 3rd-order Runge-Kutta iterate for the ODE \( y'(t) = f(t, y(t)) \) is \( y_{n+1} = y_n + h F(t_n, y_n; h, f) \) with \( F(t_n, y_n; h, f) = \gamma_1 k_1 + \gamma_2 k_2 + \gamma_3 k_3 \).

The truncation error is \( T_n = y(t_{n+1}) - y_{n+1} = y(t_{n+1}) - y(t_n) - h (\gamma_1 k_1 + \gamma_2 k_2 + \gamma_3 k_3) \).

By Taylor’s expansion,

\[
y(t_{n+1}) = y(t_n) + h y'(t_n) + \frac{h^2}{2} y''(t_n) + \frac{h^3}{6} y'''(t_n) + o(h^3)
\]

\[
= y(t_n) + h f(t_n, y_n) + \frac{h^2}{2} (f(t_n, y_n) + f_y(t_n, y_n) f(t_n, y_n)) + \frac{h^3}{6} (f_{tt}(t_n, y_n) + 2 f_{ty}(t_n, y_n) f_y(t_n, y_n) + f_{yy}(t_n, y_n) f_{ty}(t_n, y_n)) + O(h^4)
\]

\[
k_1 = f(t_n, y_n),
\]
\[
k_2 = \gamma_2 f(t_n + \alpha_2 h, y_n + \beta_{21} k_1),
\]
\[
= f(t_n, y_n) + h (\gamma_2 f(t_n, y_n) + \beta_{21} k_1 f_y(t_n, y_n)) + \frac{h^2}{2} \left( \gamma_2^2 k_1^2 f_{yy}(t_n, y_n) \right) + O(h^3)
\]
\[
k_3 = f(t_n + \alpha_3 h, y_n + h (\beta_{31} k_1 + \beta_{32} k_2))
\]
\[
= f(t_n, y_n) + h (\gamma_3 f(t_n, y_n) + f_y(t_n, y_n) \beta_{31} k_1 + \beta_{32} k_2)) + \frac{h^2}{2} \left( \gamma_3^2 f^2(t_n, y_n) \beta_{31}^2 k_1^2 + \beta_{32}^2 k_2^2 \right) + O(h^3)
\]
\[ + (\beta_{31} + \beta_{32})^2 \mathbf{f}^2(t_n, y_n) \mathbf{f}_{yy}(t_n, y_n) + O(h^3). \]

As a result,
\[
T_n = [(1 - \gamma_1 - \gamma_2 - \gamma_3) \mathbf{f}(t_n, y_n)] h
+ \left[ \left( \frac{1}{2} - \gamma_2 \alpha_2 - \gamma_3 \alpha_3 \right) \mathbf{f}(t_n, y_n) + \left( \frac{1}{2} - \gamma_3 (\beta_{31} + \beta_{32}) \right) \mathbf{f}(t_n, y_n) \mathbf{f}_y(t_n, y_n) \right] h^2
+ \left[ \left( \frac{1}{6} - \frac{1}{2} (\gamma_2 \alpha_2^2 + \gamma_3 \alpha_3^2) \right) \mathbf{f}_tt(t_n, y_n) + \left( \frac{1}{3} - \gamma_2 \alpha_2 \beta_{21} - \gamma_3 \alpha_3 (\beta_{31} + \beta_{32}) \right) \mathbf{f}(t_n, y_n) \mathbf{f}_y(t_n, y_n) \right. \\
+ \left( \frac{1}{6} - \frac{1}{2} (\gamma_2 \beta_{21}^2 + \gamma_3 (\beta_{31} + \beta_{32})^2) \right) \mathbf{f}^2(t_n, y_n) \mathbf{f}_{yy}(t_n, y_n) + \left( \frac{1}{6} - \gamma_3 \alpha_2 \beta_{32} \right) \mathbf{f}_t(t_n, y_n) \mathbf{f}_y(t_n, y_n)
+ \left( \frac{1}{6} - \gamma_3 \beta_{21} \beta_{32} \right) \mathbf{f}(t_n, y_n) \mathbf{f}_{yy}(t_n, y_n) \right] h^3 + O(h^4).
\]

To guarantee convergence of the 3rd order, we want all the coefficients to be 0 of the terms of lower order for arbitrary \( \mathbf{f} \), which leads to:
\[
\begin{align*}
\gamma_1 + \gamma_2 + \gamma_3 &= 1, \\
\gamma_2 \alpha_2 + \gamma_3 \alpha_3 &= \frac{1}{2}, \\
\gamma_2 \beta_{21} + \gamma_3 (\beta_{31} + \beta_{32}) &= \frac{1}{2}, \\
\gamma_2 \alpha_2^2 + \gamma_3 \alpha_3^2 &= \frac{1}{3}, \\
\gamma_2 \alpha_2 \beta_{21} + \gamma_3 \alpha_3 (\beta_{31} + \beta_{32}) &= \frac{1}{3}, \\
\gamma_2 \beta_{21}^2 + \gamma_3 (\beta_{31} + \beta_{32})^2 &= \frac{1}{3}, \\
\gamma_3 \alpha_2 \beta_{32} &= \gamma_3 \beta_{21} \beta_{32} = \frac{1}{6}.
\end{align*}
\]

(b) By plugging into the conditions in (a) we find it’s consistent.

(c) The 3-stage Runge–Kutta can be seen to be asymptotically order 3.

```matlab
f=inline ('[-y(2); y(1)] ', 't','y ') tsnap=[0 2*pi] y_0=[1; 0] Y=inline ('[cos(t), sin(t)] ','t ') pause
for i=1:10 ,
    N_s = 2^(i+1);
    [ t, y ] = rk3(f,tsnap,y_0,N_s,0);
    s=y(1+N_s,:);
    y1(i,:) = s. ';
    S=feval(Y,t(1+N_s));
    Y1(i,:) = S;
    E1(i)=norm(s-S);
end
err1=y-Y(t);

'RK3 method'
N_s=4, [y1(1,:) Y1(1,:) E1(1) NaN]
for i=2:10
    N_s = 2^(i+1), [y1(i,:) Y1(i,:) E1(i) log2(E1(i-1)/E1(i))]
end

'RK3 approximation, exact solution, error magnitude, base-2 log of error ratio'
```
Problem 3. (a) Use the Heun method to solve the initial-value problem
\[ y' = y + y^{1/3}, \quad y(10^{-16}) = Y(10^{-16}) \]
whose exact solution is \( Y(t) = e^t(1 - e^{-2t/3})^{3/2} \). Solve the equation on \( [10^{-16}, 1] \), using a number of steps \( N = 50, 100, 200, \ldots, 6400 \) increasing by factors of 2. Plot in one figure the base-10 logarithms of the errors for all of the step sizes. Then calculate the base-2 logarithm of the ratios by which errors in \( y(1) \) decrease for successive approximations. How does this compare with the usual quadratic rate of convergence proved for the method? Explain your results.

(b) Repeat part (a) using the classical 4th-order Runge-Kutta method.

(a) We see that the order is below quadratic. This is due to \( f(t, y) := y + y^{1/3} \) not being twice continuously differentiable, which was an assumption for our convergence proof.
(b) We again see lower-than-expected order for the same reason as above.

```matlab
f=inline('y+y.^(1/3)', 't', 'y');
Y=inline('exp(t).*(1-exp(-2*t/3)).^(3/2)', 't');
y_0=Y(1e-16);
tspan=[1e-16 1];
figure
hold on
for i =1:7,
    N_s=50*2^i;
    [t,y]=rk4(@(t,y) f(t,y), tspan, y_0, N_s, 0);
    y1(i,:)=y(1+N_s,:);
    s=y(1+N_s,:);
    S=feval(@(t) Y(t), t(1+N_s));
    y1(i,:)=S;
    E1(i)=norm(s-S);
    ee=y-Y(t);
    plot(t, log10(abs(ee)),'-')
end
xlabel('time t')
ylabel('log base-10 of the error')
title('errors for the RK4 method with decreasing steps')
N_s=100, [y1(1,:) Y1(1,:) E1(1) NaN]
for i=2:7
    N_s=50*2^i, [y1(i,:) Y1(i,:) E1(i) log2(E1(i-1)/E1(i))]
end
```
Problem 4*. In this problem you will establish an asymptotic formula for the error in the Euler approximation \( y_n = y(t_n) \) to the solution of the initial-value problem

\[
\dot{y} = f(t, y), \quad y(0) = y_0.
\]

whenever \( f(t, y) \) is \( C^2 \) in the variables \( t \) and \( y \).

(a) If \( e_n = y(t_n) - y_n \) is the error in the Euler approximation, then prove that it satisfies the iteration equation

\[
e_{n+1} = \left[ 1 + h \frac{\partial f}{\partial y}(t_n, y_n) \right] e_n + \frac{h^2}{2} \ddot{y}(t_n) + b_n
\]

where \( b_n = O(h^3) \). To prove this result, you will need to expand \( y(t_{n+1}) = y(t_n + h) \) in \( h \) up to second-order terms with a remainder \( O(h^3) \) and expand \( f(t_n, y(t_n)) = f(t_n, y_n + e_n) \) to first order in \( e_n \) with a remainder \( O(|e_n|^2) \).

(b) Let \( \delta_n = \delta(t_n) \) be the Euler approximation to the solution of the initial-value problem for the linearized equation

\[
\dot{\delta} = \frac{\partial f}{\partial y}(t, y(t)) \delta + \frac{1}{2} h^2 \ddot{y}(t), \quad \delta(t_0) = \delta_0.
\]

Explain why \( h[\delta(t_n) - \delta_n] = O(h^2) \) so that

\[
e_n - \delta(t_n)h = e_n - \delta_n h + O(h^2)
\]

(c) Based on the results of part (a), show that the quantity \( k_n = e_n - \delta_n h \) satisfies the recurrence relation

\[
k_{n+1} = k_n + h \frac{\partial f}{\partial y}(t_n, y_n)k_n + b_n.
\]

and use this result to show that, if the initial error in Euler’s method satisfies \( e_0 = \delta_0 h + O(h^2) \), then also for all \( n > 0 \) the asymptotic error formula holds

\[
e_n = \delta(t_n)h + O(h^2),
\]

where \( \delta(t) \) is the solution of (*) in part (b).

(d) As an application of the formula in (c), consider the initial-value problem

\[
\dot{y} = -y, \quad y(0) = 1
\]

and argue that the Euler error \( e_n \) is decreasing in \( t_n \) while the relative error \( e_n/y(t_n) \) is growing slowly \( \propto t_n \).

(a) From Taylor’s theorem

\[
y(t_{n+1}) = y(t_n) + hy'(t_n) + \frac{h^2}{2} y''(t_n) + \frac{h^3}{6} y'''(t_n^*), \quad \text{for} \quad t_n \leq t^*_n \leq t_{n+1}.
\]

So

\[
e_{n+1} = e_n + h [f(t_n, y(t_n)) - f(t_n, y_n)] + \frac{h^2}{2} y''(t_n) + \frac{h^3}{6} y'''(t_n).
\]

From Taylor’s theorem for multivariate functions:

\[
f(t_n, y(t_n)) - f(t_n, y_n) = \frac{\partial}{\partial y} f(t_n, y_n) e_n + O(h^2),
\]

thus

\[
e_{n+1} = \left[ 1 + h \frac{\partial f}{\partial y}(t_n, y_n) \right] e_n + \frac{h^2}{2} y''(t_n) + O(h^3).
\]

(b) Let \( \delta_n \) satisfy

\[
\delta_{n+1} = \delta_n + h \left[ \frac{\partial}{\partial y} f(t_n, y_n) \delta_n + \frac{1}{2} y''(t_n) \right], \quad y(t_0) - y_0 = \delta_0 h + O(h^2).
\]

Then from the convergence of Euler’s method,

\[
\delta(t_n) - \delta_n = O(h), \quad h[\delta(t_n) - \delta_n] = O(h^2), \quad e_n - \delta(t_n)h = k_n + O(h^2),
\]

the last part following since \( h\delta_n = h \delta(t_n) + O(h^2) \).
(c) By subtracting $h$ times (2) from (1):

$$k_{n+1} = k_n + h \frac{\partial}{\partial y} f(t_n, y_n) k_n + O(h^2), \quad k_0 = O(h^2) \implies k_n = O(h^2).$$

Combine with the last result in (b) to conclude.

(d) Note $y(t) = \exp(-t)$ and $y''(t) = y(t)$. We have

$$y_{n+1} = y_n - h y_n = (1 - h) y_n = \cdots = (1 - h)^{n+1} y_0,$$

and

$$e_n = y_n - y(t_n) = (1 - h)^n - \exp(-hn) \to 0.$$

However

$$\frac{e_n}{y(t_n)} = \exp(hn)(1 - h)^n - 1 = (1 + hn)(1 - h)^n + O(h^2) = 2hn + O(h^2) \propto t_n,$$

where we expand $\exp(hn)$ as a Taylor series then $(1 - h)^n$ as a binomial sum.