MATH 151B Homework 3

Zheng Wang (404855295)

May 22, 2019

Question 1

(a) The Euler's Method gives

$$\begin{cases} w_{i+1} = w_i + h \cdot f(x_i, w_i) \\ w_0 = 0 \end{cases}$$

Since $x_i = ih$, we then break down the original formula to

$$\begin{cases} w_{i+1} = w_i + h(ih - (ih)^2) \\ w_0 = 0 \end{cases}$$

Therefore, we have

$$\begin{aligned} w_{i+1} &= w_i + h \cdot (ih - (ih)^2) \\ &= w_{i-1} + h \cdot ((i-1)h - (i-1)^2 h^2) + h \cdot (ih - (ih)^2) \\ &= w_{i-1} + (i-1) \cdot h^2 - (i-1)^2 \cdot h^3 + i \cdot h^2 - i^2 \cdot h^3 \\ &= w_{i-1} + h^2 \cdot [i + (i-1)] - h^3 \cdot [i^2 + (i-1)^2] \\ &= w_0 + h^2 \cdot \left(\sum_{j=0}^{i} j \right) - h^3 \cdot \left(\sum_{j=0}^{i} j^2 \right) && \text{(By telescoping)} \\ &= h^2 \cdot \frac{i \cdot (i+1)}{2} - h^3 \cdot \frac{i(i+1)(2i+1)}{6} \end{aligned}$$

Therefore, $\boxed{w_i = h^2 \cdot \dfrac{i(i-1)}{2} - h^3 \cdot \dfrac{i(i-1)(2i-1)}{6}}$ ■

(b) We take $x = ih$, where x is an arbitary fixed point

Then we have the following:

$$\begin{aligned}
\lim_{h\to 0} |w_i - y(x)| &= \lim_{h\to 0} \left| \frac{i(i-1)}{2} \cdot h^2 - \frac{i(i-1)(2i-1)}{6} \cdot h^3 - \frac{i^2}{2} \cdot h^2 + \frac{i^3}{3} \cdot h^3 \right| \\
&= \lim_{h\to 0} \left| \frac{\cancel{i^2} - i - \cancel{i^2}}{2} \cdot h^2 + \frac{-\cancel{2i^3} + 3i^2 - i + \cancel{2i^3}}{6} \cdot h^3 \right| \\
&= \lim_{h\to 0} \left| \frac{-ih}{2} \cdot h + \frac{(ih)^2}{2} \cdot h - \frac{ih}{6} \cdot h^2 \right| \\
&= \lim_{h\to 0} \left| -\frac{x}{2} \cdot h + \frac{x^2}{2} \cdot h - \frac{x}{6} \cdot h^2 \right| \\
&= \left| \lim_{h\to 0} \left(-\frac{x}{2} \cdot h + \frac{x^2}{2} \cdot h - \frac{x}{6} \cdot h^2 \right) \right| && \text{(By continuity)} \\
&= 0 && \text{(Since } x \text{ is a constant)}
\end{aligned}$$

Thus, Euler's method is convergent. ■

Question 2

The RKF12 method is implemented with the following code.

```
% Math 151b
% Homework 3, Question 2

% run the program with TOL = 10^-4
disp(run_rkf12(0, 1, 1, 10^-4, 0.5, 10^-7, @f))

% a demo function, a = 0, b = 1, alpha  = 1
function dydt = f(t,y)
    dydt = y^2*exp(-t);
end

% Implement RKF12 method with Euler's method and Modified Euler Method
% INPUTS:
% a,b - endpoints; alpha - initial condition; TOL - tolerance
% hmax - maximum step size; hmin - min step size
% func - function to be solved
function y = run_rkf12(a, b, alpha, TOL, hmax, hmin, func)
    t = a;
    w = alpha;
    h = hmax;
    FLAG = 1;
    disp([t, w])
    while FLAG == 1
        K1 = h * func(t,w);
        K2 = func(t + h, w + K1);
        K3 = h/2 * ( func(t,w) + K2 );
        % Euler: w_i+1 = w_i + K1
```

```
        % M_Euler: w_i+1 = w_i + K3
        R = 1/h * abs(K3 - K1);
        if R <= TOL
            t = t + h;
            w = w + K1;
            disp([t,w,h]);
        end
        q = (1/2) * (TOL/R);
        % Adjust the step size
        if q <= 0.1
            h = 0.1 * h; % prevent delta to be too small and h goes to 0
        elseif q >= 4
            h = 4 * h;  % prevent delta become to large and h increase to fast
        else
            h = q * h; % normal case, just set h = qh
        end
        % bound h by hmax
        if h > hmax
            h = hmax;
        end
        % terminating conditions
        if t >= b
            FLAG = 0; % reach the end point
        elseif t + h > b
            h = b - t; % adjust the final step
        elseif h < hmin
            FLAG = 0;
            fprintf("Minimun h exceeded, Procedure completed unsuccessfully.")
        end
    end
    y = w;
end
```

Explanation:

From formula of Modified Euler Method and Euler Method, we have the following:

$$
\begin{aligned}
\tilde{w}_{i+1} &= w_i + \frac{h}{2}\left[f(t_i, w_i) + f(t_i + h, w_i + hf(t_i, w_i))\right] \\
w_{i+1} &= w_i + h \cdot f(t_i, w_i)
\end{aligned}
$$

Then, we compute τ_{i+1} and $\tilde{\tau}_{i+1}$:

$$
\begin{aligned}
\tau_{i+1} &= \frac{y(t_{i+1}) - w_i}{h} - f(t_i, w_i) \\
&= \frac{y(t_{i+1}) - (w_i + hf(t_i, w_i))}{h} = \frac{1}{h}\left(y(t_{i+1}) - w_{i+1}\right) \\
\tilde{\tau}_{i+1} &= \frac{y(t_{i+1}) - (w_i + \frac{h}{2}\left[f(t_i, w_i) + f(t_i + h, w_i + hf(t_i, w_i))\right])}{h} \\
&= \frac{1}{h}\left(y(t_{i+1}) - \tilde{w}_{i+1}\right)
\end{aligned}
$$

Next, we arrive at the following:

$$\begin{aligned}\tau_{i+1} &= \frac{1}{h}\left(y(t_{i+1}) - w_{i+1}\right) = \frac{1}{h}\left[(y(t_{i+1}) - \tilde{w}_{i+1}) + (\tilde{w}_{i+1} - w_{i+1})\right] \\ &= \tilde{\tau}_{i+1} + \frac{1}{h}(\tilde{w}_{i+1} - w_{i+1})\end{aligned}$$

As $\tilde{\tau}_{i+1}$ is $\mathcal{O}(h^2)$, but τ_{i+1} is $\mathcal{O}(h)$. Thus, $\frac{1}{h}(\tilde{w}_{i+1} - w_{i+1})$ is $\mathcal{O}(h)$. Therefore, $\tau_{i+1} \approx \frac{1}{h}(\tilde{w}_{i+1} - w_{i+1})$. With this equation, we can estimate the local truncation error as:

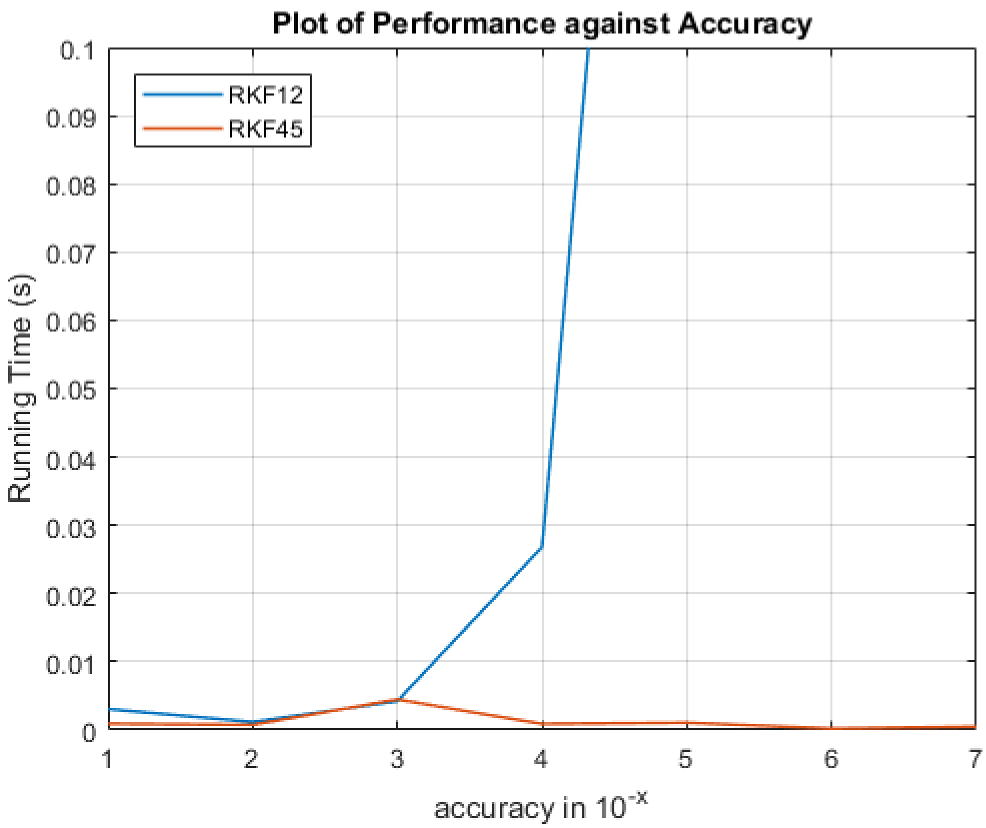
$$R = \frac{1}{h}\left|\tilde{w}_{i+1} - w_{i+1}\right| \approx \tau_{i+1}$$

In our program, since we have $\tilde{w}_{i+1} = w_i + K3$ and $w_{i+1} = w_i + K1$, Therefore, we have $R = \frac{1}{h}\left|K3 - K1\right|$. Finally, by taking $\tau_{i+1}(h) \approx Kh$, by solving $\tau_{i+1}(qh) \approx Kqh \approx q\tau_{i+1}(h) \approx qR \leq \epsilon$, we get $q \leq \frac{\epsilon}{R}$. To be more conservative, we make q even smaller by let $q = \frac{\epsilon}{2R}$. Since in the program TOL represent ϵ, we use $q = TOL/2R$ in the program.

After these, to make sure the q is not too small nor too large , we also bound q with 0.1 and 4. To make sure that h is reasonable, we also fix hmin $\leq h \leq$ hmax in the program. From the demo I did in the program, the final estimation is 2.7181, which is close to the real solution 2.7183 to degree 10^{-4}.

In terms of the run-time of the RKF12 and RKF45 method, when the accuracy is low (like 10^{-2}), the two method will have similar performance, sometimes RKF12 can run faster than RKF45. This is because we have fixed the maximum step size in the program, and even though RKF45 can achieve the desired accuracy level with large step size, the program with still have to run the a minimum number of steps. Moreover, since RKF45 must take more calculations per iteration than RKF12, it might have very similar or even lower performance than RKF12.

But when accuracy is fixed high (like 10^{-7}), the RKF45 can run much faster than RKF12. This is becuase to achieve the desired accuracy level, RKF12 will need to take much smaller step size than RKF45. Even though RKF45 needs more evaluation per iteration, there are much less iterations to run. Therefore, it is much faster than RKF12. The general trend in the performance is given by the following plot:



The following code is used to do the test:

```
% Math 151b
% Homework 3, Question 2
time_rkf12 = zeros(1,7);
time_rkf45 = zeros(1,7);

for i=1:7
    tic
    disp(run_rkf12(0, 1, 1, 10^-i, 0.5, 10^-10, @f))
    time_rkf12(i) = toc;

    tic
    disp(run_rkf45(0, 1, 1, 10^-i, 0.5, 10^-10, @f))
    time_rkf45(i) = toc;
end
% make the plot
figure;
plot((1:7),time_rkf12,'Linewidth', 1.1);
hold on;
plot((1:7),time_rkf45,'Linewidth', 1.1);
ylim([0,0.1]);
xlabel('accuracy in 10^{-x}');
ylabel('Running Time (s)');
legend({'RKF12','RKF45'},'Location','northwest')
```

```
title('Plot of Performance against Accuracy');
grid on;
hold off;

% a demo function, a = 0, b = 1, alpha  = 1
function dydt = f(t,y)
    dydt = y^2*exp(-t);
end

% Implement RKF12 method with Euler's method and Modified Euler Method
% INPUTS:
% a,b - endpoints; alpha - initial condition; TOL - tolerance
% hmax - maximum step size; hmin - min step size
% func - function to be solved
function y = run_rkf12(a, b, alpha, TOL, hmax, hmin, func)
    t = a;
    w = alpha;
    h = hmax;
    FLAG = 1;
    %disp([t, w])
    while FLAG == 1
        K1 = h * func(t,w);
        K2 = func(t + h, w + K1);
        K3 = h/2 * ( func(t,w) + K2 );
        % Euler: w_i+1 = w_i + K1
        % M_Euler: w_i+1 = w_i + K3
        R = 1/h * abs(K3 - K1);
        if R <= TOL
            t = t + h;
            w = w + K1;
            %disp([t,w,h]);
        end
        q = (1/2) * (TOL/R);
        % Adjust the step size
        if q <= 0.1
            h = 0.1 * h; % prevent delta to be too small and h goes to 0
        elseif q >= 4
            h = 4 * h;  % prevent delta become to large and h increase to fast
        else
            h = q * h; % normal case, just set h = qh
        end
        % bound h by hmax
        if h > hmax
            h = hmax;
        end
        % terminating conditions
        if t >= b
            FLAG = 0; % reach the end point
        elseif t + h > b
            h = b - t; % adjust the final step
        elseif h < hmin
            FLAG = 0;
            fprintf("Minimun h exceeded, Procedure completed unsuccessfully.")
        end
    end
    y = w;
```

```
end


% Implement RKF45 method
% INPUTS:
% a,b - endpoints; alpha - initial condition; TOL - tolerance
% hmax - maximum step size; hmin - min step size
% func - function to be solved
function y = run_rkf45(a, b, alpha, TOL, hmax, hmin, func)
    t = a;
    w = alpha;
    h = hmax;
    FLAG = 1;
    %disp([t, w])
    while FLAG == 1
        K1 = h * func(t,w);
        K2 = h * func(t + h/4, w + K1/4);
        K3 = h * func(t + 3/8*h, w + 3/32*K1 + 9/32*K2 );
        K4 = h * func(t + 12/13*h, w + 1932/2197*K1 - 7200/2197*K2 + 7296/2197*K3);
        K5 = h * func(t + h, w + 439/216*K1 -8*K2 + 3680/513*K3 - 845/4104*K4);
        K6 = h * func(t + h/2, w - 8/27*K1 + 2*K2 - 3544/2565*K3 + 1859/4104*K4 -
            11/40*K5);
        R = 1/h * abs(1/360*K1 - 128/4275*K3 - 2197/75240*K4 + 1/50*K5 + 2/55*K6);
        if R <= TOL
            t = t + h;
            w = w + 25/216*K1 + 1408/2565*K3 + 2197/4104*K4 - 1/5*K5;
            %disp([t,w,h]);
        end
        q = 0.84 * (TOL/R)^(1/4);
        % Adjust the step size
        if q <= 0.1
            h = 0.1 * h; % prevent delta to be too small and h goes to 0
        elseif q >= 4
            h = 4 * h;  % prevent delta become to large and h increase to fast
        else
            h = q * h; % normal case, just set h = qh
        end
        % bound h by hmax
        if h > hmax
            h = hmax;
        end
        % terminating conditions
        if t >= b
            FLAG = 0; % reach the end point
        elseif t + h > b
            h = b - t; % adjust the final step
        elseif h < hmin
            FLAG = 0;
            fprintf("Minimun h exceeded, Procedure completed unsuccessfully.")
        end
    end
    y = w;
end
```