MATH7003-00: Assignment #3_rev.1

Sangman Jung

e-mail: sangmanjung@khu.ac.kr

Kyung Hee University —April 1, 2020

Problem. For the example on slide 20 (in week3-1), generate the same Table 6.8 using the Richardson error estimation. Discuss about the result. [1].

Example. Consider the problem

$$y' = -y^2 \qquad\qquad y(0) = 1$$

which has the solution $Y(x) = 1/(1+x)$. The results in Table 6.8 are for stepsizes $h = 0.25$ and $2h = 0.5$. The last column is the error estimate (6.5.27), and it is an accurate estimator of the true error $Y(x) - y_h(x)$.

Table 6.8 Trapezoidal method and Richardson error estimation

x	$y_{2h}(x)$	$Y(x) - y_{2h}(x)$	$y_h(x)$	$Y(x) - y_h(x)$	$\frac{1}{3}[y_h(x) - y_{2h}(x)]$
1.0	.483144	.016856	.496021	.003979	.004292
2.0	.323610	.009723	.330991	.002342	.002460
3.0	.243890	.006110	.248521	.001479	.001543
4.0	.194838	.004162	.198991	.001009	.001051
5.0	.163658	.003008	.165937	.000730	.000759

Solution. First, in order to estimate the numerical solution of the given differential equation, we have to find the numerical scheme of the equation. Using trapezoidal method, we obtain the numerical scheme of our equation as follows.

$$y_{n+1} = y_n - \frac{h}{2}\left(y_n^2 + y_{n+1}^2\right), \quad h > 0, \ n \geq 0.$$

We can see that the scheme is implicit method. There are many methods and formulas to compute the implicit scheme, we use the method as called **predictor-corrector method** in this problem[2][3]. According to [1], since the trapezoidal scheme is a nonlinear equation with root y_{n+1}, we adopt the root finding method of Chapter 2 in [1]. Hence, let $y_{n+1}^{(0)}$ be a good initial guess of the solution y_{n+1}, and define

$$y_{n+1}^{(j+1)} = y_n - \frac{h}{2}\left[y_n^2 + \left(y_{n+1}^{(j)}\right)^2\right] \ , \ j = 0,1,2,\cdots.$$

The initial guess is usually obtained using an explicit method such as Euler's method or midpoint method. For the successive iteration of this scheme, we have to investigate two conditions that

(1) choose the initial guess $y_{n+1}^{(0)} \simeq y_{n+1}$, **(2)** consider the iteration j such that $\left|y_{n+1} - y_{n+1}^{(j)}\right| = O(h^4)$.

In the textbook [1] of our lecture, the case of initial guess using the Euler's method has $O(h^2)$ and the case of midpoint method has $O(h^3)$ because of :

$$y_{n+1} - y_{n+1}^{(0)} = y_{n+1} - u_n(x_{n+1}) + u_n(x_{n+1}) - y_{n+1}^{(1)} = O(h^2) \text{ in Euler's method case,}$$
$$\because \; y_{n+1} - u_n(x_{n+1}) = O(h^3), \; u_n - y_{n+1}^{(1)} = O(h^2).$$

By the Lipschitz condition,

$$\left|y_{n+1} - y_{n+1}^{(1)}\right| \le \frac{hK}{2}\left|y_{n+1} - y_{n+1}^{(0)}\right| \le O(h^3)$$

in Euler's method case and furthermore, we can obtain

$$\left|y_{n+1} - y_{n+1}^{(2)}\right| \le \frac{hK}{2}\left|y_{n+1} - y_{n+1}^{(1)}\right| \le O(h^4).$$

Thus, this means that we just iterate a twice times for initial guess, using Euler's method. In the same way to Euler's case, we can catch the number of iteration by

$$\left|y_{n+1} - y_{n+1}^{(1)}\right| \le \frac{hK}{2}\left|y_{n+1} - y_{n+1}^{(0)}\right| = O(h^3) \le O(h^4) \text{ in midpoint method case.}$$

Now, more precisely, we let the initial guess $y_{n+1}^{(0)} = y_{pred}^{(0)} = y_{n-1} - 2hy_n^2$ using midpoint method. y_n is computed by using Euler's method as $y_n = y_{n-1} - hy_{n-1}^2$. Note that this initial guess is called **predictor**. From our trapezoidal scheme, now we can set the scheme as

$$y_{n+1}^{(j+1)} = y_n - \frac{h}{2}\left\{y_n^2 + \left[y_{n+1}^{(j)}\right]^2\right\} = y_n - \frac{h}{2}\left\{y_n^2 + \left[y_{pred}^{(j)}\right]^2\right\}, \; j = 0,1,2,\cdots, \; n \ge 0.$$

The **corrector** term is just the trapezoidal method of the scheme. Hence, finally we update $y_{n+1} = y_{correct}^{(j+1)}$ in the iteration, we can get the numerical solution of the equation as follows.

Table 6.8 Trapezoidal method and Richardson error estimation

x	y_{2h}(x)	Y(x) - y_{2h}(x)	y_{h}(x)	Y(x) - y_{h}(x)	[y_{h}(x) - y_{2h}(x)]/3
1.0	0.497132	0.002868	0.499407	0.000593	0.000759
2.0	0.332948	0.000386	0.333530	0.000197	0.000194
3.0	0.250216	0.000216	0.250254	0.000254	0.000013
4.0	0.200333	0.000333	0.200220	0.000220	0.000038
5.0	0.166995	0.000328	0.166847	0.000180	0.000049

fx >>

Figure 1. Numerical computation of Table 6.8 using MATLAB.

The error $\left[y_h(x) - y_{2h}(x)\right]/3$ is called **Richardson error** and it was used to predict the error and to obtatin a more rapidly convergent numerical integration method. This is a practical procedure for estimating the global error. More briefly, the Richardson error is the approximation of the true error if we don't know the exact solution of y'. Note that $Y(x_n) - y_h(x_n) \doteq [y_h(x) - y_{2h}(x)]/3$. In **Figure 1.**, we can see that the Richardson error is an accurate estimator of the true error $Y(x) - y_h(x)$.

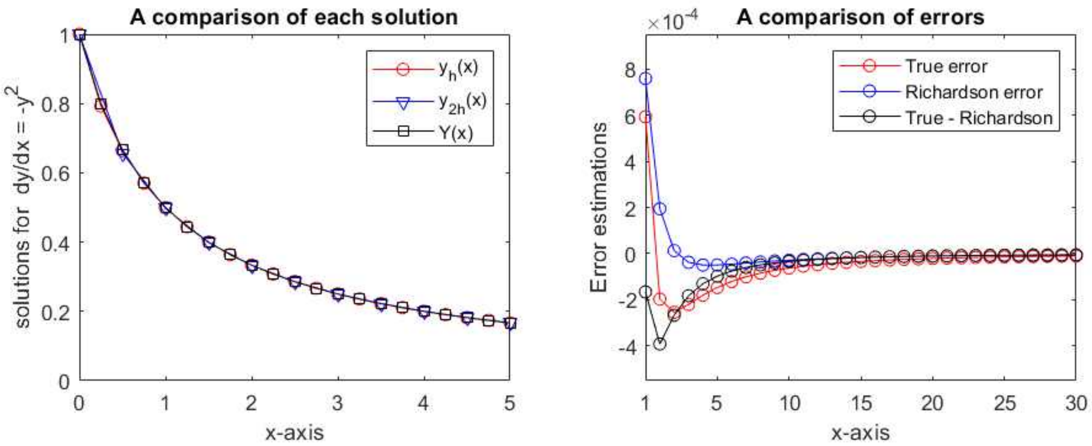

Figure 2. Two comparison graphs of the equation $y' = -y^2$. On the left side of this figure, we can see the comparison of three solutions : $y_h(x)$ (red color) and $y_{2h}(x)$ (blue color) for $h = 0.25$ and the exact solution, $Y(x)$ (black color), $x \in [0, 5]$. On the right side, we computed three error estimates : true error $e_T = Y(x) - y_h(x)$ at $h = 0.25$ (red color), Richardson error $e_{Richardson} = [y_h(x) - y_{2h}(x)]/3$ (blue color) and their difference $e_T - e_{Richardson}$ (black color), $x \in [0, 30]$.

In **Figure 2.**, the numerical scheme is nicely approximated to the exact solution of y'. It is sufficient to draw out the error comparison graph as shown above.

-(풀이 내용에 틀린 점이 있고, 구한 결과가 Table 6.8과 맞지 않아 아래에 다시 풀어보았습니다.)-

국문으로 간략히 풀이를 설명하면, initial guess $y_{n+1}^{(0)}$를 midpoint로 구한 후, (j)-iteration 동안 다음을 만족하는지 확인해야 합니다.

$$\left|y_{n+1} - y_{n+1}^{(1)}\right| \le \frac{hK}{2}\left|y_{n+1} - y_{n+1}^{(0)}\right| = O(h^3) \le O(h^4)$$

문제의 Table 6.8에서는, 0.000001 단위까지 보여줄 수 있어야 하므로, 10^{-7}보다 작아지면 (j)-iteration을 종료하도록 한 후, criterion에 맞추어 최적화된 $y_{n+1}^{(j)}$에 대하여 본 y_{n+1}을 업데이트하면 됩니다. 즉,

1. initial guess를 구하기 위해 Euler, midpoint 방법을 이용하여 (j)에 관한 초항을 구합니다.
2. 이 initial guess가 order of convergence를 만족하는지 확인합니다.
3. convergence criterion을 만족할 때까지 initial guess (predictor)를 가지고 trapezoidal corrector를 계산합니다.
4. 이 corrector 가 criterion에 최적화 되면, 그것을 구하고자 하는 numerical y_{n+1}로 업데이트합니다.
5. 이 과정을 반복하여 최종적으로 주어진 구간에서의 numerical y를 얻습니다.

다음 장에 새로 수정한 코드의 결과를 첨부합니다.

Table 6.8 Trapezoidal method and Richardson error estimation

x	y_{2h)(x)	Y(x) - y_{2h}(x)	y_{h}(x)	Y(x) - y_{h}(x)	[y_{h}(x) - y_{2h}(x)]/3
1.0	0.483145	0.016855	0.496021	0.003979	0.004292
2.0	0.323610	0.009723	0.330991	0.002342	0.002460
3.0	0.243890	0.006110	0.248521	0.001479	0.001543
4.0	0.195839	0.004161	0.198991	0.001009	0.001051
5.0	0.163658	0.003008	0.165937	0.000730	0.000759

fx >>

위 코드의 결과는 문제가 제시한 Table 6.8의 결과와 같으며, $O(h^4)$에 근접하게 줄어드는 것을 확인할 수 있습니다.

Trapezoidal_method_HW3-(1).m (except Figure 2. code) <-- 이전 코드

```
%% MATH7003-00: Assignment #3-(1), 2019310290 Sangman Jung
clear,clc

% Initial parameters, initial conditions
h = [0.25 0.5]; % step size
y(1) = 1; % initial value of y
Y = @(x) 1./(1+x)'; % exact solution of y'

% Trapezoidal method iteration to solve y'=-y^2
% Using predictor-corrector method, we can obtain the scheme as follows.
for h_iter = 1:2 % the loop of h and 2h
    x = 0:h(h_iter):5; % apply a different step size
    y = ones(size(x)); % reset the computed results of h=0.25 before
    y(2) = y(1)-h(h_iter)*y(1)^2; % define the term for midpoint, using Euler's method
    y_p(1)=y(1)-2*h(h_iter)*y(2)^2; % initial guess of y_p (midpoint method)
    for j =1:length(x)-1 % Trapezoidal method loop
        y_p(j+1)=y(j)-h(h_iter)*(y(j)^2+y_p(j)^2)/2; % predictor
        y(j+1)=y(j)-h(h_iter)*(y(j)^2+y_p(j+1)^2)/2; % corrector (In this code, numerical sol. of y')
    end
    fval(h_iter) = {y'}; % save the values of y have different size for y_h and y_2h
    xval(h_iter) = {x'}; % save the values of x have different size for y_h and y_2h
end

% Print out setting
x_h = cell2mat(xval(1)); % for the case of h = 0.25
x_2h = cell2mat(xval(2)); % for the case of h = 0.5
y_h = cell2mat(fval(1)); % numerical y for h = 0.25
y_2h = cell2mat(fval(2)); % numerical y for h = 0.5
y_ht = y_h(5:4:end);
y_2ht = y_2h(3:2:end);
t = 1:5;

% Print the Table 6.8
fprintf("Table 6.8  Trapezoidal method and Richardson error estimation\n");
fprintf("--------------------------------------------------------------------------------------------------------------------\n");
fprintf("|   x   |   y_{2h}(x)   |   Y(x) - y_{2h}(x)   |   y_{h}(x)   |   Y(x) - y_{h}(x)   |   [y_{h}(x) - y_{2h}(x)]/3   |\n");
fprintf("--------------------------------------------------------------------------------------------------------------------\n");
for i = 1:length(t)
    fprintf('  %1.1f     %1.6f        %1.6f           %1.6f         %1.6f                 %1.6f\n',...
        abs([t(i) y_2ht(i) Y(t(i))-y_2ht(i) y_ht(i) Y(t(i))-y_ht(i) (y_ht(i)-y_2ht(i))/3]));
end
fprintf("--------------------------------------------------------------------------------------------------------------------\n");
```

Trapezoidal_method_HW3-(1)_rev1.m <-- 수정된 코드

```
%% MATH7003-00: Assignment #3-(1), 2019310290 Sangman Jung _ revision 1.
clear,clc

% Initial parameters, initial conditions
h = [0.25 0.5]; % step size
Y = @(x) 1./(1+x)'; % exact solution of y'
iter = 100; % iteration number for the stop criterion

% Trapezoidal method iteration to solve y'=-y^2
% Using predictor-corrector method, we can obtain the scheme as follows.
for h_iter = 1:2 % the loop of h and 2h
    x = 0:h(h_iter):5; % apply a different step size
    y0 = 1; % initial condition
    y(1) = y0; % initialize
    for j = 1:length(x)-1 % Trapezoidal method loop
        y1 = y0-h(h_iter)*y0^2; % first term using Euler's method
        y2 = y0-2*h(h_iter)*y1^2; % second term using midpoint method
        y_p = y2; % the predictor
        for i = 1:iter % order of convergence criterion
            y_c = y0-h(h_iter)*(y0^2+y_p^2)/2; % the corrector (In case, y_c == numerical y)
            if abs(y_c-y_p) <= 10^-7 % I will present the results per 10^-6.
                break;
            end
            y_p = y_c; % update the predictor until satisfying the criterion
        end
        y0 = y_p; % set y^(j)_{n+1} term
        y(j+1) = y_c; % update our results
    end
    fval(h_iter) = {y'}; % save the values of y have different size for y_h and y_2h
    xval(h_iter) = {x'}; % save the values of x have different size for y_h and y_2h
end

% Print out setting
y_h = cell2mat(fval(1)); % numerical y for h = 0.25
y_2h = cell2mat(fval(2)); % numerical y for h = 0.5
y_h = y_h(5:4:end);
y_2h = y_2h(3:2:end);
t = 1:5;

% Print the Table 6.8
fprintf("Table 6.8  Trapezoidal method and Richardson error estimation\n");
fprintf("-----------------------------------------------------------------------------------------
-------------------\n");
fprintf("|   x    |    y_{2h)(x)    |    Y(x) - y_{2h}(x)    |    y_{h}(x)    |    Y(x) - y_{h}(x)    |    [y_{h}(x) -
y_{2h}(x)]/3    |\n");
fprintf("-----------------------------------------------------------------------------------------
-------------------\n");
for i = 1:length(t)
    fprintf('   %1.1f      %1.6f          %1.6f           %1.6f          %1.6f                  %1.6f\n',...
        abs([t(i) y_2h(i) Y(t(i))-y_2h(i) y_h(i) Y(t(i))-y_h(i) (y_h(i)-y_2h(i))/3]));
end
fprintf("-----------------------------------------------------------------------------------------
-------------------\n");
```

References.

[1] Atkinson, K. E. (2008). An introduction to numerical analysis. John wiley & sons.

[2] Atkinson, K., Han, W., & Stewart, D. E. (2011). Numerical solution of ordinary differential equations (Vol. 108). John Wiley & Sons.

[3] Atkinson, K. E., & Han, W. (1985). Elementary numerical analysis (p. 17). New York et al.: Wiley.