MATH7003-00: Assignment #4

Sangman Jung

e-mail: sangmanjung@khu.ac.kr

Kyung Hee University —April 14, 2020

Problem. For the example on slide 17 (in week3-2), generate the same Table 6.9 using the algorithm Detrap. Discuss about the result. [1].

Example. Consider the problem

$$y' = \frac{1}{1+x^2} - 2y^2 \qquad\qquad y(0) = 0$$

$$Y(x) = \frac{x}{1+x^2}$$

This is an interesting problem for testing ***Detrap***, and it performs quite well. The equation was solved on $[0,10]$ with $h_{\min} = 0.001$, $h_{\max} = 1.0$, $h = 0.1$, and $\epsilon = 0.0005$. Table 6.9 contains some of the results, including the true global error and the true local error labeled True le, The latter was obtained by using another more accurate numerical method. Only selected sections of output are shown because of space.

Table 6.9 Example of algorithm *Detrap*

x_n	h	y_n	$Y(x_n) - y_n$	trunc	True le
.0227	.0227	.022689	5.84E − 6	5.84E − 6	5.84E − 6
.0454	.0227	.045308	1.17E − 5	5.83E − 6	5.84E − 6
.0681	.0227	.067787	1.74E − 5	5.76E − 6	5.75E − 6
.0908	.0227	.090060	2.28E − 5	5.62E − 6	5.61E − 6
.2725	.0227	.253594	5.16E − 5	2.96E − 6	2.85E − 6
.3065	.0340	.280125	5.66E − 5	6.74E − 6	6.79E − 6
.3405	.0340	.305084	6.01E − 5	6.21E − 6	5.73E − 6
.3746	.0340	.328411	6.11E − 5	4.28E − 6	3.54E − 6
.4408	.0662	.369019	5.05E − 5	−6.56E − 6	−5.20E − 6
.5070	.0662	.403297	2.44E − 5	−1.04E − 5	−2.12E − 5
.5732	.0662	.431469	−2.03E − 5	−2.92E − 5	−4.21E − 5
.6138	.0406	.445879	−2.99E − 5	−1.12E − 5	−1.10E − 5
1.9595	.135	.404982	−1.02E − 4	−1.64E − 5	−1.67E − 5
2.0942	.135	.388944	−1.03E − 4	−1.79E − 5	−2.11E − 5
2.3172	.223	.363864	−6.57E − 5	1.27E − 5	8.15E − 6
2.7632	.446	.319649	3.44E − 4	4.41E − 4	3.78E − 4
3.0664	.303	.294447	3.21E − 4	9.39E − 5	8.41E − 5
7.6959	.672	.127396	3.87E − 4	8.77E − 5	1.12E − 4
8.6959	1.000	.113100	3.96E − 4	1.73E − 4	1.57E − 4
9.6959	1.000	.101625	4.27E − 4	1.18E − 4	1.68E − 4
10.6959	1.000	.092273	4.11E − 4	9.45E − 5	1.21E − 4

Discussion. **Detrap** : a kind of low-order predictor-corrector algorithm, is the techniques involved in constructing a variable-stepsize predictor-corrector algorithm. It is also simpler to understand than algorithms based on higher order methods. It uses the trapezoidal method, and it controls the size of the local error by varying the stepsize h [1].

This algorithm has three steps : (1) Choose an initial stepsize $\Rightarrow$ (2) The regular predictor-corrector step $\Rightarrow$ (3) Changing the stepsize. The detail of this algorithm is given in [1].

In **Figure 1**, for $x \in [0,10]$ such that $D_3 y \doteq Y^{(3)}(x) = 0$, we can see that h is increasing with the errors. In detail, $Y^{(3)}(x)$ is calculated from $Y(x)$ as

$$Y^{(3)}(x) = -\frac{6(x^4 - 6x^2 + 1)}{(1+x^2)^4}.$$

Note that $h = \sqrt{6\epsilon / |D_3 y|}$ when the new stepsize h is chosen. Such points satisfying $Y^{(3)}(x) = 0$ are $x = 0.414$ and $x = 2.414$. In Table 6.9, the near points $x_n = 0.4408$ and $x_n = 2.3172$ of $x = 0.414$ and $x = 2.414$, respectively, the stepsize h is increasing with the errors after next step. Thus, at $x_n = 2.7632$ in the table need to avoid a misleadingly large value of h. As can be observed, the local error at $x_n = 2.7632$ increases greatly, due to the larger value of h. At the next step, h is decreased to reduce the size of the local error.

The algorithm can be made more sophisticated in order to detect the problems of too large an h, but setting a reasonably sized $h_{\max}$ will also help [1].

Table 6.9 Example of algorithm Detrap

x_{n}	h	y_{n}	Y(x_{n})-y_{n}	trunc
0.0227	0.0227	0.022689	5.84e-06	5.84e-06
0.0454	0.0227	0.045308	5.84e-06	5.83e-06
0.0681	0.0227	0.067787	1.17e-05	5.76e-06
0.0908	0.0227	0.090060	1.74e-05	5.62e-06
0.1135	0.0227	0.112059	2.28e-05	5.43e-06
0.1362	0.0227	0.133723	2.80e-05	5.19e-06
0.1589	0.0227	0.154991	3.29e-05	4.90e-06
0.1817	0.0227	0.175808	3.73e-05	4.57e-06
0.2044	0.0227	0.196121	4.12e-05	4.20e-06
0.2271	0.0227	0.215884	4.47e-05	3.81e-06
0.2498	0.0227	0.235054	4.75e-05	3.39e-06
0.2725	0.0227	0.253594	4.99e-05	2.96e-06
0.3065	0.0340	0.280125	5.66e-05	2.53e-06
0.3405	0.0340	0.305084	5.66e-05	6.21e-06
0.3746	0.0340	0.328411	6.01e-05	4.28e-06
0.4408	0.0662	0.369019	5.05e-05	2.25e-06
0.5070	0.0662	0.403297	5.05e-05	-1.04e-05
0.5732	0.0662	0.431469	2.44e-05	-2.92e-05
0.6139	0.0406	0.445879	-2.99e-05	-4.39e-05
0.6545	0.0406	0.458253	-2.99e-05	-1.11e-05
0.6951	0.0406	0.468720	-4.06e-05	-1.20e-05
0.7358	0.0406	0.477415	-5.13e-05	-1.25e-05
0.7764	0.0406	0.484477	-6.15e-05	-1.26e-05
0.8171	0.0406	0.490045	-7.08e-05	-1.24e-05
0.8577	0.0406	0.494253	-7.92e-05	-1.21e-05
0.8983	0.0406	0.497233	-8.63e-05	-1.16e-05
0.9390	0.0406	0.499108	-9.23e-05	-1.10e-05
0.9796	0.0406	0.499995	-9.70e-05	-1.03e-05
1.0203	0.0406	0.500002	-1.01e-04	-9.55e-06
1.0609	0.0406	0.499232	-1.03e-04	-8.81e-06
1.1016	0.0406	0.497775	-1.04e-04	-8.07e-06
1.1422	0.0406	0.495718	-1.05e-04	-7.36e-06
1.1828	0.0406	0.493137	-1.05e-04	-6.67e-06
1.2235	0.0406	0.490102	-1.04e-04	-6.02e-06
1.2641	0.0406	0.486675	-1.02e-04	-5.42e-06
1.3229	0.0588	0.481146	-1.02e-04	-4.86e-06
1.3817	0.0588	0.475066	-1.02e-04	-1.23e-05
1.4405	0.0588	0.468560	-1.05e-04	-1.10e-05
1.4992	0.0588	0.461733	-1.06e-04	-9.26e-06
1.5580	0.0588	0.454677	-1.05e-04	-7.73e-06
1.6470	0.0889	0.443732	-1.04e-04	-6.42e-06
1.7359	0.0889	0.432639	-1.04e-04	-1.61e-05
1.8248	0.0889	0.421544	-1.07e-04	-1.32e-05
1.9595	0.1347	0.404982	-1.02e-04	-9.69e-06
2.0943	0.1347	0.388944	-1.02e-04	-1.79e-05
2.3172	0.2230	0.363864	-6.57e-05	-1.23e-05
2.7632	0.4459	0.319649	3.44e-04	-4.20e-06
3.0664	0.3032	0.294447	3.21e-04	2.41e-04
3.3696	0.3032	0.272441	3.21e-04	5.84e-05
3.6728	0.3032	0.253184	3.09e-04	5.47e-05
3.9760	0.3032	0.236264	2.98e-04	4.74e-05
4.2792	0.3032	0.221324	2.82e-04	3.93e-05
4.7458	0.4666	0.201458	2.97e-04	3.20e-05
5.2124	0.4666	0.184709	2.97e-04	9.01e-05
5.6790	0.4666	0.170459	3.31e-04	7.52e-05
6.3513	0.6723	0.153281	3.59e-04	5.62e-05
7.0236	0.6723	0.139156	3.59e-04	1.08e-04
7.6959	0.6723	0.127396	3.93e-04	8.77e-05
8.6959	1.0000	0.113100	3.96e-04	6.22e-05
9.6959	1.0000	0.101696	3.55e-04	1.18e-04
10.6959	1.0000	0.092380	3.04e-04	7.35e-05

\>>

Figure 1. The result of Table 6.9 using MATLAB.

Detrap_run_HW4.m

```
%% Homework # 4, 2019310290 Sangman Jung
clear,clc
% define ODE and it's exact solution
f = @(x,y) (1/(1+x^2)) - 2*y^2; % ODE in the example
Y = @(x) x/(1+x^2); % exact solution of y'
% initial values
y0 = 0; % y(0) = 0
% interval of x
x0 = 0; x_end = 10; % x in [0,10]
% step size h
h_min = 0.001; h_max = 1; h = 0.1;
% error tolerence
epsilon = 0.0005;
% run Detrap
Detrap(Y,f,x0,y0,x_end,epsilon,h,h_min,h_max)
```

Detrap.m

```
function Detrap(Y,f,x0,y0,x_end,epsilon,h,h_min,h_max)
%%% Homework # 3, 2019310290 Sangman Jung

% 1. Remark: The problem being solved is Y'=f(x,Y), Y(x0)=y0, for x0 =< x
% =<x_end, using the method described earlier in the section. The
% approximate solution values are printed at each node point. The error
% parameter epsilon and the stepsize parameters were discussed earlier in
% the section. The variable ier is an error indicator, output when exiting
% the algorithm: ier = 0 means a normal return; ier = 1 means that the
% integration was terminated due to a necessary h < h_min.

% Table 6.9 Example of algorithm Detrap
fprintf('Table 6.9 Example of algorithm Detrap\n');
fprintf('---------------------------------------------------------------------\n');
fprintf('|   x_{n}  |     h    |  y_{n}     |  Y(x_{n})-y_{n}  |    trunc   |\n');
fprintf('---------------------------------------------------------------------\n');

% 2. Initialize:
loop = 1; ier = 0;

% 3. Remark: Choose an initial value of h
while 1
   % 4. Calculate y_{h}(x0+h), y_{h/2}(x0+(h/2)), y_{h/2}(x0+h) using
   % method (6.5.2). In each case, use the Euler predictor (6.5.12) and
   % follow it by two iterations of (6.5.3).
   y_p = y0 + h*f(x0,y0);
   y_p = y0 + (h/2)*(f(x0,y0)+f(x0+h,y_p));
   y_h = y0 + (h/2)*(f(x0,y0)+f(x0+h,y_p)); % y_{h}(x0+h)
   y_p = y0 + (h/2)*f(x0,y0);
   y_p = y0 + (h/4)*(f(x0,y0)+f(x0+h/2,y_p));
   y_h22 = y0 + (h/4)*(f(x0,y0)+f(x0+h/2,y_p)); % y_{h/2}(x0+(h/2))
   y_p = y_h22 + (h/2)*f(x0+h/2,y_h22);
   y_p = y_h22 + (h/4)*(f(x0+h/2,y_h22)+f(x0+h,y_p));
   y_h2 = y_h22 + (h/4)*(f(x0+h/2,y_h22)+f(x0+h,y_p)); % y_{h/2}(x0+h)


   % 5. For the error in y_{h}(x0+h), use
   trunc = (4/3)*(y_h2-y_h);

   % 6. If epsilon*h/4 =< abs(trunc) =< epsilon*h, or if loop = 2, then x1
   % = x0 + h, y1 = y_{h}(x0+h), print x1, y1, and go to step 10.
   if (epsilon*h/4 <= abs(trunc) && abs(trunc) <= epsilon*h) || loop == 2
      x1 = x0 + h;
      y1 = y_h;
      fprintf('|  %1.4f  |  %1.4f  |  %1.6f  |     %1.2e     |  %1.2e  |\n',[x1 h y1 Y(x1)-y1 trunc]);
      break
   end

   % 7. Calculate D_3y = Y^{3}(x0) from (6.6.4). If D_3y ~= 0, then
   D_3y = (f(x0+h,y_h2)-2*f(x0+(h/2),y_h22)+f(x0,y0))/(h^2/4);
   if D_3y == 0
      h = h_max;
      loop = 2;
   else
      h = sqrt((6*epsilon)/abs(D_3y));
   end
```

```
    % 8. If h < h_min, then ier = 2 and exit. If h > h_max, then h = h_max,
    % ier = 1, loop = 2.
    if h < h_min
        ier = 2;
        return
    end
    if h > h_max
        h = h_max;
        ier = 1;
        loop = 2;
    end
    % 9. Go to step 4.
end

% 10. Remark : This portion of the algorithm contains the regular
% predictor-corrctor step with error control.
while 1
    while 1
        % 11. Let x2 = x1 + h, and y2^{0} = y0 + 2*h*f(x1,y1). Iterate
        % (6.5.3) once to obtain y2.
        x2 = x1 + h;
        y_p = y0 + 2*h*f(x1,y1);
        y2 = y1 + (h/2)*(f(x1,y1)+f(x2,y_p));

        % 12. trunc = -(y2 - y2^{0})/6
        trunc = -(y2 - y_p)/6;

        % 13. If abs(trunc) > epsilon*h or abs(trunc) < epsilon*h/4, then
        % go to step 16.
        if abs(trunc) < ((epsilon*h)/4) || epsilon*h < abs(trunc)
            break
        end

        % 14. Print x2, y2.
        if round(x2,4) == 0.2725 || round(x2,4) == 1.9595 || round(x2,4) == 7.6959
            fprintf('-------------------------------------------------------------------------\n');
        end
        fprintf('|  %1.4f  |  %1.4f  |  %1.6f  |     %1.2e     |  %1.2e  |\n',[x2 h y2 Y(x1)-y1 trunc]);

        % 15. x0 = x1, x1 = x2, y0 = y1, y1 = y2. If x1 < x_end, then go to
        % step 11. Otherwise exit.
        x1 = x2; y0 = y1; y1 = y2;
        if ~(x1 < x_end)
            return
        end
    end
    % 16. Remark : Change the stepsize.
    % 17. x0 = x1, y0 = y1, h0 = h, and calculate h using (6.6.8)
    x0 = x1; y0 = y1; h0 = h;
    h = sqrt((epsilon*(h0^3))/(2*abs(trunc)));

    % 18. h = min{h, 2*h0}
    h = min(h,2*h0);
```

```
    % 19. If h < h_min, then ier = 2 and exit. If h > h_max, then ier = 1
    % and h = h_max.
    if h < h_min
        ier = 2;
        return
    end
    if h > h_max
        ier = 1;
        h = h_max;
    end

    % 20. y1^{0} = y0 + h*f(x0,y0), and iterate twice in (6.5.3) to
    % calculate y1. Also, x1 = x0 + h.
    y_p = y0 + h*f(x0,y0);
    y_p = y0 + (h/2)*(f(x0,y0)+f(x0+h,y_p));
    y1 = y0 + (h/2)*(f(x0,y0)+f(x0+h,y_p));
    x1 = x0 + h;

    % 21. Print x1, y1.
    if round(x1,4) == 0.2725 || round(x1,4) == 1.9595 || round(x1,4) == 7.6959
        fprintf('-------------------------------------------------------------------\n');
    end
    fprintf('|  %1.4f  |  %1.4f  |  %1.6f  |     %1.2e     |  %1.2e  |\n',[x1 h y1 Y(x1)-y1 trunc]);

    % 22. If x1 < x_end, then go to step 10. Otherwise, exit.
    if ~(x1 < x_end)
        return
    end
end
```

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.