



MATH7003-00: Assignment #2

Sangman Jung

e-mail: sangmanjung@khu.ac.kr

Kyung Hee University – March 25, 2020

Problem. Perform the midpoint method for example 1 on slide 15 and generate the same Table 6.6. Discuss about the result [1].

Example 1. Consider the model problem (6.4,10) with $\lambda = -1$. The numerical results are given in Table 6.6 for $h = 0.25$. The value y_1 was obtained using Euler's method, as in (6.4,6). From the values in the table, the parasitic solution is clearly growing in magnitude. For $x_n = 2.25$, the numerical solution y_n becomes negative, and it alternates in sign with each successive step.

Table 6.6 Example 1 of midpoint method instability

x_n	y_n	$Y(x_n)$	Error
.25	.7500	.7788	.0288
.50	.6250	.6065	-.0185
.75	.4375	.4724	.0349
1.00	.4063	.3679	-.0384
1.25	.2344	.2865	.0521
1.50	.2891	.2231	-.0659
1.75	.0898	.1738	.0839
2.00	.2441	.1353	-.1088
2.25	-.0322	.1054	.1376

Solution. Our initial value problem is as follows.

$$\frac{dy}{dx} = \lambda y \quad \text{where } \lambda = -1, \quad y(0) = 1.$$

In order to compute this equation using midpoint method, we have to calculate the difference equation as 'the numerical scheme'. First we let the step size $h = 0.25$ and $x_i \in [0, 2.25]$. Since the midpoint method is the multistep method, we need to compute y_1 using Euler's method as follows.

$$y_1 = y_0 + hf(x_0, y_0) = y_0 - hy_0, \quad y_0 = y(0), \quad x_0 = 0.$$

Then, we can get the results of midpoint method employing y_0 and y_1 . The numerical scheme of midpoint method in our equation is the following.

$$y_{n+1} = y_{n-1} + 2h\lambda y_n, \quad n \geq 1.$$

In order to investigate the stability of this scheme, we now have to find a **parasitic solution** of this method. By the textbook [1] in our lecture, we can easily obtain the formulas of it as follows.

$$r_0 = h\lambda + \sqrt{1 + h^2\lambda^2}, \quad r_1 = h\lambda - \sqrt{1 + h^2\lambda^2}$$

$$\beta_0 = \frac{y_1 - r_1 y_0}{r_0 - r_1} \text{ and } \beta_1 = \frac{y_0 r_0 - y_1}{r_0 - r_1} \quad \because \beta_0 + \beta_1 = y_0 \text{ and } \beta_0 r_0 + \beta_1 r_1 = y_1.$$

In this, we called $\beta_1 r_1^n$ a **parasitic solution** of the numerical scheme of our problem, since it does not correspond to any solution for the original differential equation $y' = \lambda y$. The results of this problem is shown **Figure 1**. We can see that the new solution $\beta_1 r_1^n$ is a creation of the numerical method: for problem $y' = \lambda y$ with $\lambda < 0$, it will cause the numerical solution to diverge from the true solution as $x_n \rightarrow \infty$. Because of this behavior, we say that the midpoint method is only weakly stable [1].


 명령 창

Table 6.6 Example 1 of midpoint method instability

	$x_{\{n\}}$		$y_{\{n\}}$		$Y(x_{\{n\}})$		Error	
	0.25		0.7500		0.7788		0.0288	
	0.50		0.6250		0.6065		-0.0185	
	0.75		0.4375		0.4724		0.0349	
	1.00		0.4063		0.3679		-0.0384	
	1.25		0.2344		0.2865		0.0521	
	1.50		0.2891		0.2231		-0.0659	
	1.75		0.0898		0.1738		0.0839	
	2.00		0.2441		0.1353		-0.1088	
	2.25		-0.0322		0.1054		0.1376	

*** Parasitic solution of the midpoint method ***

$x_{\{n\}} = 0.00$	and	$\beta_{\{1\}} * r_{\{1\}}^{(1)} = -0.0191$
$x_{\{n\}} = 0.25$	and	$\beta_{\{1\}} * r_{\{1\}}^{(2)} = 0.0245$
$x_{\{n\}} = 0.50$	and	$\beta_{\{1\}} * r_{\{1\}}^{(3)} = -0.0314$
$x_{\{n\}} = 0.75$	and	$\beta_{\{1\}} * r_{\{1\}}^{(4)} = 0.0402$
$x_{\{n\}} = 1.00$	and	$\beta_{\{1\}} * r_{\{1\}}^{(5)} = -0.0515$
$x_{\{n\}} = 1.25$	and	$\beta_{\{1\}} * r_{\{1\}}^{(6)} = 0.0659$
$x_{\{n\}} = 1.50$	and	$\beta_{\{1\}} * r_{\{1\}}^{(7)} = -0.0844$
$x_{\{n\}} = 1.75$	and	$\beta_{\{1\}} * r_{\{1\}}^{(8)} = 0.1081$
$x_{\{n\}} = 2.00$	and	$\beta_{\{1\}} * r_{\{1\}}^{(9)} = -0.1384$
$x_{\{n\}} = 2.25$	and	$\beta_{\{1\}} * r_{\{1\}}^{(10)} = 0.1773$


 >>

Figure 1. Generation of Table 6.6 and each value of the parasitic solution

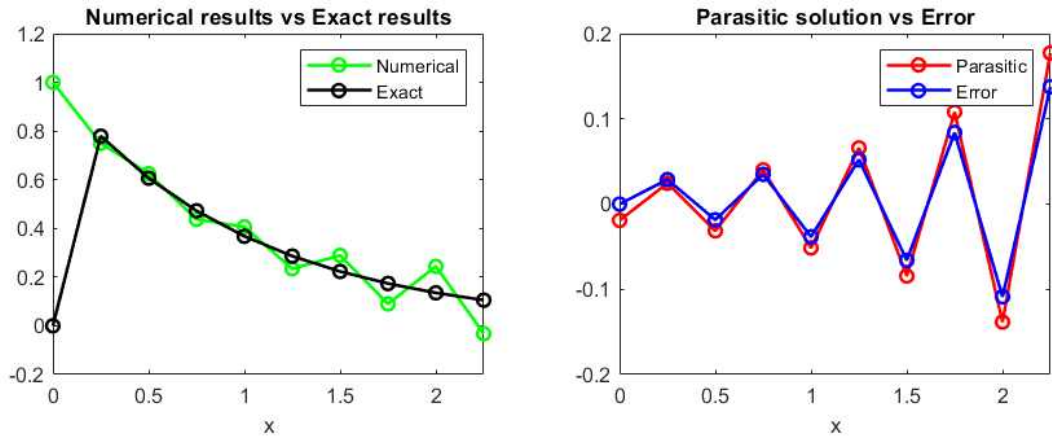


Figure 2. The comparison between numerical and exact result and their parasitic solution and error.

We can see the **Figure 2.** that the error and parasitic solution are diverge and they alternate in sign with each successive step. The **MATLAB code** is attached next paper.

Midpoint_method_HW2.m

```
%% MATH7003-00: Assignment #2, 2019310290 Sangman Jung
clear,clc

% parameters, initial conditions
h = 0.25; % step size
x = 0:0.25:2.25;
lambda = -1; % parameter. the case of lambda < 0
y(1) = 1; % initial condition
y(2) = y(1)+h*lambda*y(1); % obtained using Euler's method

% general solution of midpoint scheme
r_0 = h*lambda + sqrt(1+h^2*lambda^2); % root r0
r_1 = h*lambda - sqrt(1+h^2*lambda^2); % root r1
beta_0 = (y(2)-r_1*y(1))/(r_0-r_1); % coefficient beta0
beta_1 = (y(1)*r_0-y(2))/(r_0-r_1); % coefficient beta1

% Midpoint method for the equation { y' = lambda*y , y(0) = 1 }
fprintf("Table 6.6 Example 1 of midpoint method instability\n");
fprintf("-----\n");
fprintf("| x_{n} | y_{n} | Y(x_{n}) | Error | \n");
fprintf("-----\n");
for n = 2:length(x)
    y(n+1) = y(n-1)+2*h*lambda*y(n); % midpoint method
    Y(n) = exp(-x(n)); % exact solution of y'
    Error(n) = Y(n)-y(n);
    fprintf(' %1.2f %1.4f %1.4f %1.4f \n',...
        [x(n) y(n) Y(n) Error(n)]);
end
fprintf("-----\n");

% Parasitic solution of the midpoint method
fprintf('\n *** Parasitic solution of the midpoint method ***\n');
fprintf("-----\n");
for n = 1:length(x)
    para_sol(n) = beta_1*r_1^n;
    fprintf(' x_{n} = %1.2f and beta_{1}*r_{1}^{%1d} = %1.4f\n', [x(n) n para_sol(n)]);
end
fprintf("-----\n");

% Graphs
subplot(1,2,1)
plot(x,y(1:10),'g-o',x,Y,'k-o','LineWidth',1.5);
xlim([0 x(end)]);ylim([-0.2 1.2]);
xlabel('x');
legend('Numerical','Exact');
title('Numerical results vs Exact results');
tt = get(gca,'title');
tt.FontWeight = 'bold';
subplot(1,2,2)
plot(x,para_sol,'r-o',x>Error,'b-o','LineWidth',1.5);
xlim([0 x(end)]);ylim([-0.2 0.2]);
xlabel('x');
legend('Parasitic','Error');
title('Parasitic solution vs Error');
tt = get(gca,'title');
tt.FontWeight = 'bold';
```

References.

- [1] Atkinson, K. E. (2008). An introduction to numerical analysis. John wiley & sons.
- [2] Shampine, L. F. (2009). Stability of the leapfrog/midpoint method. Applied mathematics and computation, 208(1), 293-298.