Computational Problem 3

Table of Contents

Setup

```
close all, clear all, clc
format long, format compact
fs = 16;
set(0,'defaulttextfontsize',fs);
set(0,'defaultaxesfontsize',fs);
```

EXERCISE 1

(a)

Since the initial value problem is:

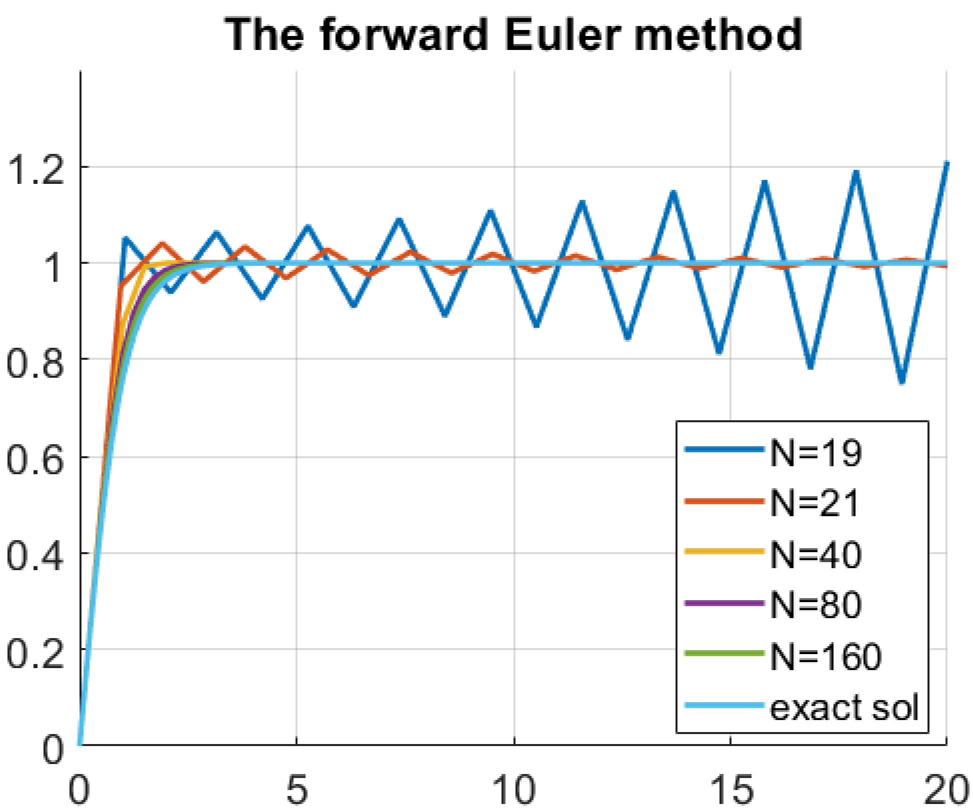
$$\begin{cases} y'(t) = 1 - y^2, \quad t \in (0, 20), \\ y(0) = 0. \end{cases}$$

Thus, we can get that the forward Euler method is:

$$\begin{cases} u_0 = y_0, \\ u_{n+1} = u_n + h_n f(t_n, u_n). \end{cases}$$

```
figure
grid on
hold on

exact_f = @(t) (exp(1).^(2*t) - 1) / (exp(1).^(2*t) + 1);

f = @(t,y) 1 - y.^2;

N_list = [19, 21, 40, 80, 160];

for i = 1:length(N_list)
```

```
    N = N_list(i);
    y0 = 0;
    tmax = 20;
    [t_fe, u_fe] = feuler(f,[0,tmax],y0,N);

    range = linspace(0, 20, N + 1);
    plot(range, u_fe,'LineWidth', 2);
end

exact_sol = zeros(1, 160 + 1);

for n = 1:161
    exact_sol(n) = exact_f((n - 1) * (20 / 160));
end

range = linspace(0, 20, 160 + 1);
plot(range, exact_sol,'LineWidth', 2);
legend({'N=19', 'N=21', 'N=40', 'N=80', 'N=160', 'exact
 sol'}, 'Location', 'Best');
title('The forward Euler method')
```



According to the figure above, we can see that the error between numerical solutions and the exact solution becomes smaller as N incresaes.

```
e_fe = zeros(1, length(N_list));
e_fe_end = zeros(1, length(N_list));
```

```
for i = 1:length(N_list)
    N = N_list(i);
    y0 = 0;
    tmax = 20;
    [t_fe, u_fe] = feuler(f,[0,tmax],y0,N);


    u_sol = zeros(1, N + 1);

    for n = 1:N + 1
        u_sol(n) = exact_f((n - 1) * (20 / N));
    end

    e_fe(i) = max(abs(u_sol - u_fe));
    e_fe_end(i) = abs(u_sol(N + 1) - u_fe(N + 1));
end

e_fe
e_fe_end
```

```
e_fe =
  Columns 1 through 3
   0.269804222128289   0.211521891180364   0.113405844044235
  Columns 4 through 5
   0.049976606700864   0.023966059963128
e_fe_end =
  Columns 1 through 3
   0.211064865800020   0.006209752178513                   0
  Columns 4 through 5
                   0   0.000000000000000
```

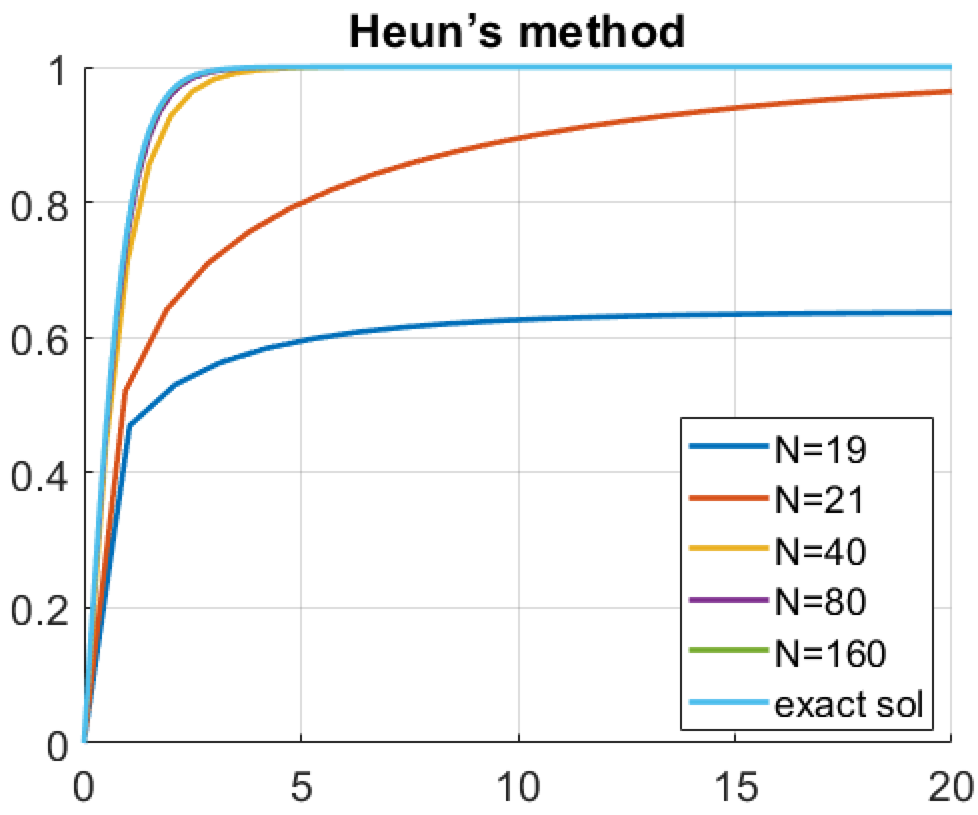
(b)

Now, we using Heun s method to compute numerical solutions to the initial value problem.

```
figure
grid on
hold on

for i = 1:length(N_list)
    N = N_list(i);
    y0 = 0;
    tmax = 20;
    [t_he, u_he] = heun(f,[0,tmax],y0,N);

    range = linspace(0, 20, N + 1);
    plot(range, u_he,'LineWidth', 2);
end

exact_sol = zeros(1, 160 + 1);

for n = 1:161
    exact_sol(n) = exact_f((n - 1) * (20 / 160));
```

```
end

range = linspace(0, 20, 160 + 1);
plot(range, exact_sol,'LineWidth', 2);
legend({'N=19', 'N=21', 'N=40', 'N=80', 'N=160', 'exact
 sol'}, 'Location', 'Best');
title('Heun’s method')


e_heun = zeros(1, length(N_list));
e_heun_end = zeros(1, length(N_list));

for i = 1:length(N_list)
    N = N_list(i);
    y0 = 0;
    tmax = 20;
    [t_he, u_he] = heun(f,[0,tmax],y0,N);


    u_sol = zeros(1, N + 1);

    for n = 1:N + 1
        u_sol(n) = exact_f((n - 1) * (20 / N));
    end

    e_heun(i) = max(abs(u_sol - u_he));
    e_heun_end(i) = abs(u_sol(N + 1) - u_he(N + 1));
end

e_heun
e_heun_end
```

```
e_heun =
  Columns 1 through 3
   0.440640166919561   0.315594896763482   0.049328827646571
  Columns 4 through 5
   0.009671464283940   0.002131963986823
e_heun_end =
  Columns 1 through 3
   0.363356646248982   0.035955392303006   0.000000000001049
  Columns 4 through 5
   0.000000000000000   0.000000000000000
```

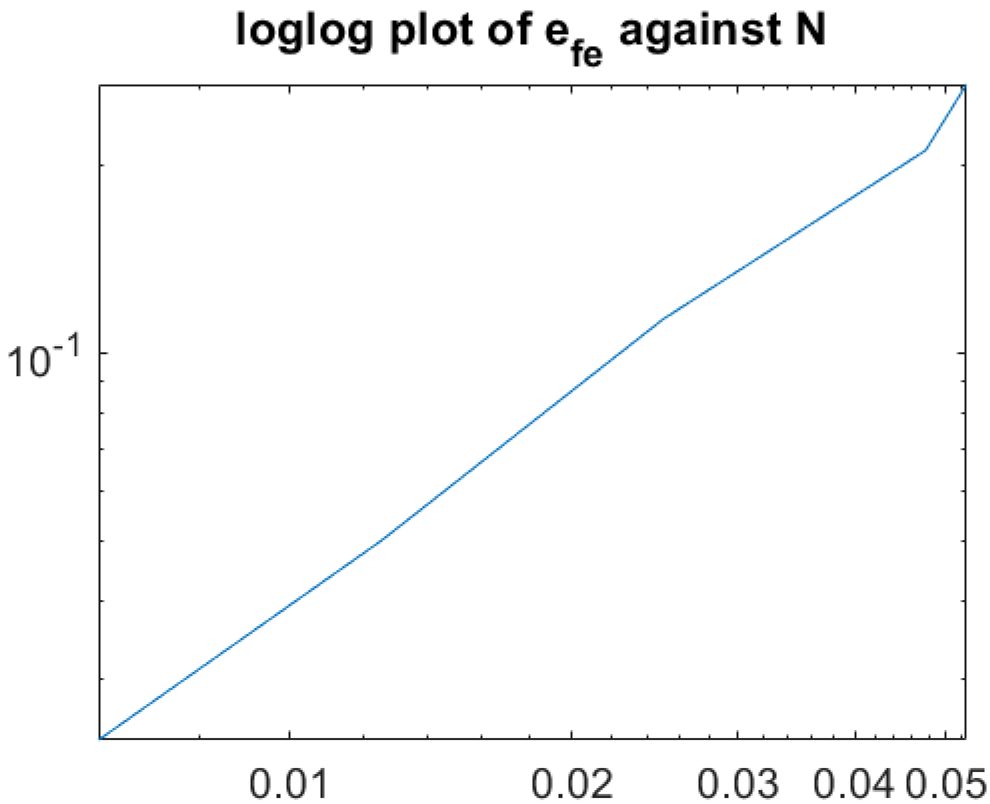


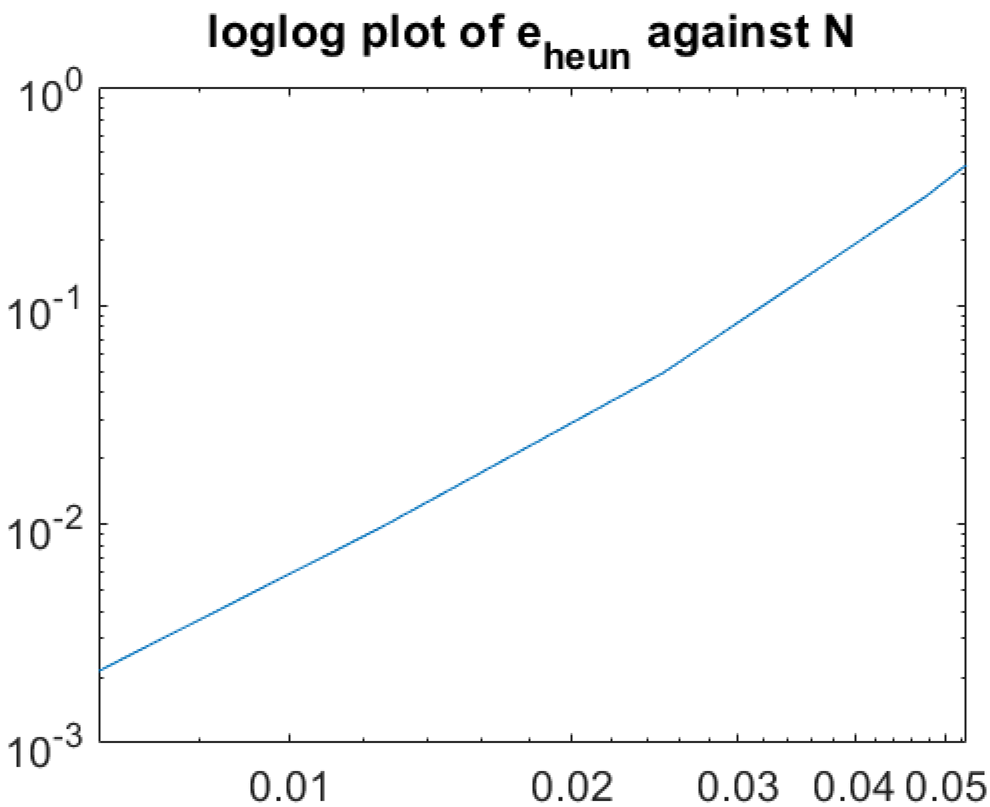
(c)

```
hvect = 1./N_list;

figure
loglog(hvect, e_fe)
title('loglog plot of e_{fe} against N')


syms p
syms c
cond1 = p * log(hvect(1)) + c == log(e_fe(1));
cond2 = p * log(hvect(5)) + c == log(e_fe(5));
conds = [cond1 cond2];

r = solve(conds, p, c);
vpa(r.p)



figure
loglog(hvect, e_heun)
title('loglog plot of e_{heun} against N')
```

```
syms p
syms c
cond1 = p * log(hvect(1)) + c == log(e_heun(1));
cond2 = p * log(hvect(5)) + c == log(e_heun(5));
conds = [cond1 cond2];

r = solve(conds, p, c);
vpa(r.p)
```

```
ans =
1.1362549156789549235527967182 51
ans =
2.50204055881693776308099363211 86
```





In the two loglog figures above, each figure show a straight line with slope p and offset logC.

According to the calculation results, we can know that for the forward Euler method, the slope p = 1.136, so the convergence order of the forward Euler method is 1. Moreover, for the Heun's method, we calculated the slope p = 2.502, so the convergence order of the Heun's method is 2.

Therefore, the results of our actual calculation agree with the theory from lectures.

(d)

According to the figures we draw above, not every approximation reproduce the same asymptotic behaviour. Obviously, the value of y(20) is approximated well for both methods when N = 40, 80, 160, and the y(20) is approximated not well in the early stage for both methods when N = 21. However, in the case of N = 19, the approximation is bad.

Yes, because when N = 19 (h = 20 / 19 > 1), the asymptotic behaviour cannot approximate y(20); but when N = 21 (h = 20 / 21 < 1), the asymptotic behaviour approximate y(20) relative well. So, the methods are stable when critical value of h = 1.

For the forward Euler method, the lack of stability for N = 19 is reflected in the approximation shows fluctuation, and the fluctuation increases as t -> 20.

For the Heun's method, the lack of stability for N = 19 is reflected in the approximation cannot approximate y(20), and its value reached 0.64 instead of 1 when t = 20.

Published with MATLAB® R2022b