APC 523 HW1

Xiaohan Li

1.

1. Assume $x = \pm \left(\sum_{p=1}^{\infty} b_p 2^{-p} \right) \cdot 2^e$

① if $b_{p+1} = 1$, then $rd(x) = \pm \left(\sum_{l=1}^{p} b_p 2^{-p} + 2^{-(p+1)} \right) \cdot 2^e$

then $\max |x - rd(x)| = \left(\sum_{l=p+2}^{\infty} 2^{-l} \right) \cdot 2^e = 2^{-(p+1)} \cdot 2^e$

$\therefore \max \left| \frac{x - rd(x)}{x} \right| \leq \frac{2^{-(p+1)} \cdot 2^e}{2^{-1} \cdot 2^e} = 2^{-p}$

② if $b_{p+1} = 0$, then $rd(x) = \pm \left(\sum_{l=1}^{p} b_p 2^{-p} + 0 \right) \cdot 2^e$

then $\max |x - rd(x)| = \left(\sum_{l=p+2}^{\infty} 2^{-l} \right) \cdot 2^e = 2^{-(p+1)} \cdot 2^e$

$\therefore \max \left| \frac{x - rd(x)}{x} \right| \leq \frac{2^{-(p+1)} \cdot 2^e}{2^{-1} \cdot 2^e} = 2^{-p}$

In conclusion, $\left| \frac{x - rd(x)}{x} \right| \leq 2^{-p}$

2.

(a) Based on my Matlab code, the answer is 244.71.

(b) Based on my Matlab code, when k=17, the value of $e^{5.5}$ converges to five significant figures 244.71.

Using exp function, $e^{5.5}$ =244.691932264220.

Therefore, the relative error is

$$\varepsilon = \left| \frac{e^{5.5} - 244.71}{e^{5.5}} \right| = 7.38387065418834 \times 10^{-5}$$

(c) The calculation result is when k=17, the answer converges to 244.70, which is a little different from part(b). In this case, relative error is

$$\varepsilon = \left|\frac{e^{5.5} - 244.70}{e^{5.5}}\right| = 3.29709921571637 \times 10^{-5}$$

(d)

(i) k=25, answer= 0.0038363, $\varepsilon = 0.0612883402547713$.

(ii) k=19, answer= 0.0040000, $\varepsilon = 0.0212322709431183$.

(iii) k=17, answer=0, $\varepsilon = 1$

(iv) k=18, answer=0.0100, $\varepsilon = 1.44691932264220$.

From the calculation result, we can see that for negative argument, method (iii) converges most quickly. Method (ii) has the lowest error.

For positive argument, we can see that adding from left and adding from right have the same convergence rate. However, adding from right to the left introduces lower error.

(e)

Algorithm:

Fist compute $e^{5.5}$ and adding from right to left. Then use $1/e^{5.5}$ to calculate the answer.

Results:

k=17, answer=0. 0.0040866, $\varepsilon = 4.19496090368344 \times 10^{-5}$.

3. (a) ① $fl(x \cdot x) = x^2(1+\varepsilon)$

$fl(x \cdot fl(x^2)) = fl(x \cdot x^2(1+\varepsilon)) = x \cdot x^2(1+\varepsilon)^2 = x^3(1+\varepsilon)^2 \approx x^3(1+2\varepsilon)$

$fl(x \cdot fl(x^3)) = fl(x \cdot x^3(1+2\varepsilon)) = x^4(1+\varepsilon)(1+2\varepsilon) \approx x^4(1+3\varepsilon)$

$\vdots$

Repeat the multiplication for n times, we can get

$fl(x \cdot fl(x^{n-1})) = fl(x \cdot x^{n-1}[1+(n-2)\varepsilon]) = x^n[1+(n-2)\varepsilon]\cdot(1+\varepsilon)$

$\approx x^n(1+(n-1)\varepsilon)$

② $fl(\ln x) = (\ln x)(1+\varepsilon)$

$fl(n\ln x) = n\cdot(\ln x)(1+\varepsilon)\cdot(1+\varepsilon) = n\ln x\cdot(1+\varepsilon)^2 \approx n\ln x\cdot(1+2\varepsilon)$

$fl[e^{fl(n\ln x)}] = fl[e^{n\ln x\cdot(1+2\varepsilon)}] = fl[e^{n\ln x}\cdot e^{2\varepsilon}] = x^n\cdot e^{2\varepsilon}(1+\varepsilon)$

$\approx x^n\cdot(1+2\varepsilon)(1+\varepsilon)$

$\approx x^n(1+3\varepsilon)$

Comparing ① and ②, we can see that when $n-1<3 \Rightarrow n<4$, exponentiating via repeated multiplication is more accurate than the log-exponential method.

when $n-1>3 \Rightarrow n>4$, repeated multiplication is less accurate than log-exponential method.

when $n=4$, two methods are comparable for accuracy

(b) (i) $x^{a(1+\varepsilon_a)} = x^a x^{a\varepsilon_a} = x^a \underline{e^{a\varepsilon_a \ln x}} \approx x^a [1 + a\varepsilon_a \ln x]$

Taylor expansion

$\because e^{a\varepsilon_a \ln x} = 1 + a\varepsilon_a \ln x + \dots$

(ii) $[x(1+\varepsilon_x)]^a = x^a (1+\varepsilon_x)^a \approx x^a \cdot (1 + a\varepsilon_x)$

For (i), if $x \to 0$ or $x \to \infty$, or $|a|$ is large,

the propagated error is substantial.

For (ii), if $|a|$ is large, the propagated error is substantial.

4. (a) $(\mathrm{cond} f)(x) = \left| \frac{x f'(x)}{f(x)} \right|$

$f'(x) = e^{-x}$

$\Rightarrow (\mathrm{cond} f)(x) = \left| \frac{x e^{-x}}{1 - e^{-x}} \right| = \left| \frac{x}{e^x - 1} \right| = \frac{x}{e^x - 1}$

when $x \in [0,1]$

Taylor expansion: $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$

$\therefore$ for $x \in [0,1]$, $e^x - 1 \geq x$

$\therefore (\mathrm{cond} f)(x) \leq 1$

(b) $f_A(x) = [1 - e^{-x}(1+\varepsilon_1)](1+\varepsilon_2)$, ε_1 is introduced by exp

ε_2 is introduced by subtraction

$f(x_A) = 1 - e^{-x_A}$

Assume $f_A(x) = f(x_A)$,

then $|f_A(x) - f(x)| = |f(x_A) - f(x)|$

$\therefore \left|[1 - e^{-x}(1+\varepsilon_1)](1+\varepsilon_2) - (1-e^{-x})\right| \approx |f'(x)(x_A - x)|$

$\therefore \left|\varepsilon_2(1-e^{-x}) - \varepsilon_1 e^{-x}\right| \approx |f'(x)(x_A - x)|$

$\because x \in [0,1] \Rightarrow 1 - e^{-x} \geqslant 0$

Assume $|\varepsilon_2| = |\varepsilon_1| = eps$

$\therefore \left|\varepsilon_2(1-e^{-x}) - \varepsilon_1 e^{-x}\right| \leq \left|eps(1-e^{-x}) + eps\, e^{-x}\right| = |eps|$

$\therefore |f'(x)(x_A - x)| \leq |eps| \Rightarrow |x - x_A| \leq \left|\frac{eps}{f'(x)}\right|$

$\therefore \frac{1}{eps}\left|\frac{x - x_A}{x}\right| \leq \frac{1}{eps}\left|\frac{eps}{f'(x)\,x}\right| = \left|\frac{e^x}{x}\right| = \frac{e^x}{x}$

$\therefore$ the least upper bound of $\frac{1}{eps}\left|\frac{x-x_A}{x}\right| = \frac{e^x}{x}$

$\therefore$ By definition, $(\text{cond}\,A)(x) = \frac{e^x}{x}$

taylor expansion: $\frac{e^x}{x} = \frac{1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots}{x} = \frac{1}{x} + 1 + \frac{x}{2!} + \frac{x^2}{3!} + \cdots$

x

$\therefore (\text{cond}\,A)(x) = \frac{e^x}{x} > 1$ everywhere on $[0,1]$

(c) As shown in following figures, A becomes progressively more ill-condition when x is small. The reason is that when x is small, e^{-x} approaches 1 so that 1-e^{-x} introduces large error.

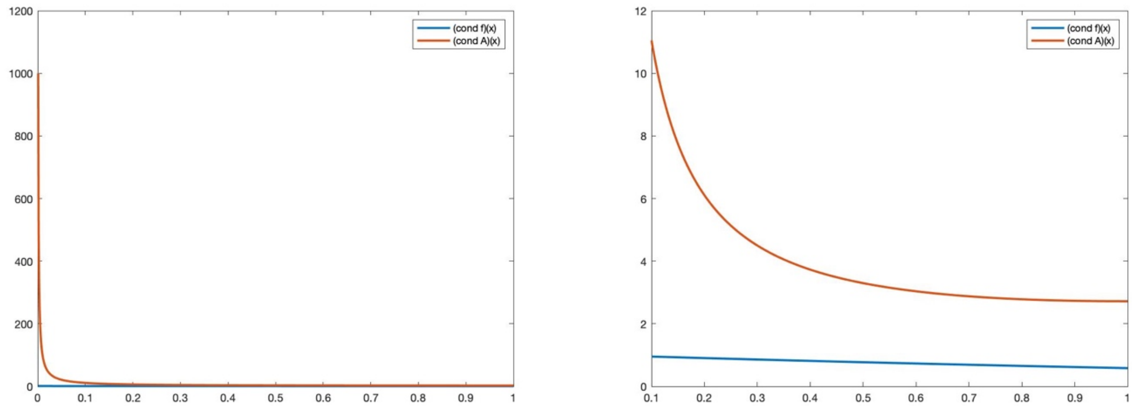

Figure 1: (cond f)(x) and (cond A)(x) for $x \in [0,1]$ (left) and $x \in [0.1,1]$

(d) At most b bit of significance loss, $\left|\frac{x-y}{x}\right| \geq 2^{-b}$

Here $x=1$; $y = e^{-x}$, $x \in [0,1]$ $\Rightarrow$ $1-e^{-x} \geq 2^{-b}$ $\Rightarrow$ $x \geq -\ln(1-2^{-b})$

When $b=1$, $x \geq 0.6931$, $x_{least} = 0.6931$

When $b=2$, $x \geq 0.2877$, $x_{least} = 0.2877$

When $b=3$, $x \geq 0.1335$, $x_{least} = 0.1335$

When $b=4$, $x \geq 0.0645$, $x_{least} = 0.0645$

(e) upper bound on the relative error is

$$\varepsilon_{max} = \frac{eps}{2^{-b}} = 2^b \, eps$$

when $b=1$, $\varepsilon_{max} = 2^1 \cdot eps = 2^{-51}$

when $b=2$, $\varepsilon_{max} = 2^2 \cdot eps = 2^{-50}$

when $b=3$, $\varepsilon_{max} = 2^3 \cdot eps = 2^{-49}$

when $b=4$, $\varepsilon_{max} = 2^4 \cdot eps = 2^{-48}$

(f) We can use Taylor expansion to solve this problem.

$$f(x) = 1 - e^{-x} = 1 - \left(1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots\right)$$

$$\therefore \; f(x) = x - \frac{x^2}{2!} + \frac{x^3}{3!} - \frac{x^4}{4!} + \cdots \quad ①$$

We can pick enough terms in equation ① to make sure convergence when $x \to 0$, using equation ① to calculate $f(x)$.

5. The code is shown as follows. (Matlab code)

```
fid = fopen('Limits.txt','wt');
fprintf(fid,'n\t(1+1/n)^n\n');
for i = 1:500
   n_1 = 10^(i-1);
   n_2 = 10 * n_1;
   e_1 = (1 + 1/n_1)^n_1;%the first e value in the ith trial
   e_2 = (1 + 1/n_2)^n_2;% the second e value in the ith trial
   e_diff = round(e_1-e_2,13,'significant');%calculate the difference between success e value

   if e_diff == 0%see whether they agree to 12 significant figures
      fprintf(fid,'%d\t%.12f\n',i-1,e_1);%print the  intermediate value in ith trial
      fprintf(fid,'%d\t%.12f\n',i,e_2);%print the  final value in ith trial
```

```
        break;
    else
        fprintf(fid,'%d\t%.12f\n',i-1,e_1);% print the first intermediate value in ith trial
    end

end
fprintf(fid,'final value %.12f\n',e_2);%printf final convergened value e_2
fprintf(fid,'nstop %d\n',i);%nstops equals final value of i
```

The output is shown as follows.

n $(1+\frac{1}{n})^n$

0 2.000000000000

1 2.593742460100

2 2.704813829422

3 2.716923932236

4 2.718145926825

5 2.718268237192

6 2.718280469096

7 2.718281694132

8 2.718281798347

9 2.718282052012

10 2.718282053235

11 2.718282053357

12 2.718523496037

13 2.716110034087

14 2.716110034087

15 3.035035206549

16 1.000000000000

17 1.000000000000

final value 1.000000000000

nstop 17

The result converges to 1.000000000000 and $n_{stop} = 17$. Theoretically, the limit of convergence should be $e \approx 2.718281828$. Reasons why the code converged to 1 are

(1) with n increases, the relative error of $(1+\frac{1}{n})^n$ increases with n;

(2) $eps = 2^{52} < 10^{16}$. Therefore, by computation $1 + \frac{1}{n} = 1, when\ n = 10^{16},\ which\ lead\ to\ (1 + \frac{1}{n})^n = 1^n = 1.$

6. The result is shown in the following figures.

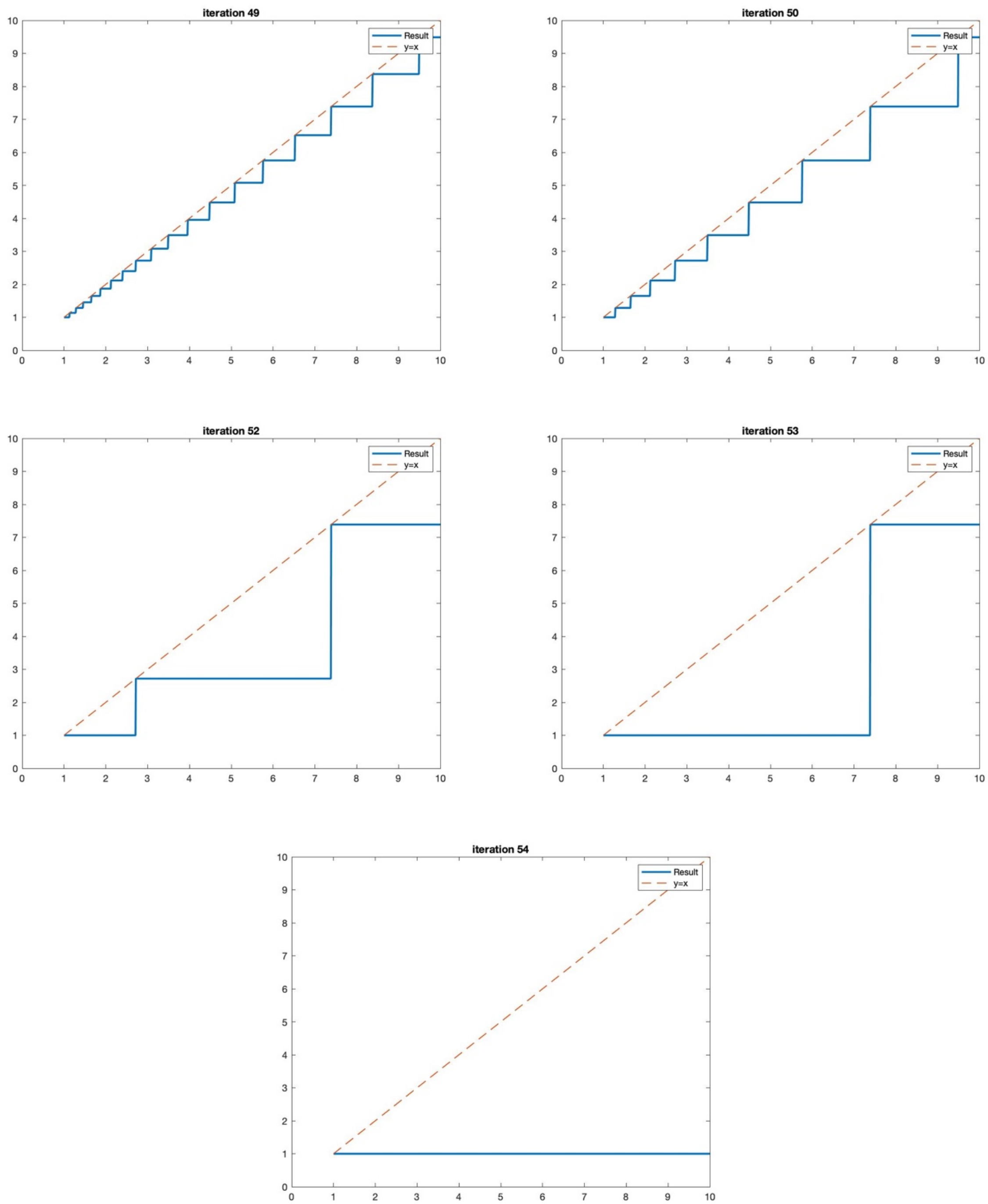

Figure 2: Results of calculation for 49, 50, 52, 53, 54 iteration times

Take 52 iterations for example, we can see that there are 3 points recovered at $x = 1$ *and close to* e, e^2.

The reason is that $eps = 2^{-52}$. For square-root 52 times, denote y_1 is the answer after 52 times iteration. For square 52 times, denote y_2 is the answer after 52 times iteration. Then

$$y_1 = ((x^{\frac{1}{2}})^{\frac{1}{2}} \ldots\ldots)^{\frac{1}{2}} = x^{2^{-52}} = e^{2^{-52}lnx} = 1 + 2^{-52}lnx = 1 + eps * lnx$$

Since $lnx \in [0, ln10]$, therefore

when $0 \le lnx < 1, y_1 = 1 + 0 * eps = 1$ (at x=1, y_1 calculated is the same as the true value); then $y_2 = {y_1}^{2^{52}} = 1$. Therefore, point at x=1 is recovered.

when $1 \le lnx < 2, y_1 = 1 + 1 * eps = 1 + eps$(at x=e, y_1 calculated is the same as the true value); then $y_2 = {y_1}^{2^{52}} = (1 + eps)^{2^{52}}$. Therefore, point at x=e is recovered.

when $2 \le lnx < ln10, y_1 = 1 + 2 * eps = 1 + 2eps$ (at x=e^2, y_1 calculated is the same as the true value); then $y_2 = {y_1}^{2^{52}} = (1 + 2eps)^{2^{52}}$. Therefore, point at $x = e^2$is recovered.

For other cases, using the similar method, we can get

$$y_1 = 1 + eps * 2^{-(n-52)}lnx,$$

here n is the iteration time for square-root.

Therefore, we can get that for $x \in [0,10]$

(1) For iteration 54 times, there are 1 point recovered at $x = 1$, because $2^{-(n-52)}lnx$ has only one integer value at x=1.

(2) For iteration 53 times, there are 2 points recovered at $x = 1, e^2$, because $2^{-(n-52)}lnx$ has two integer values at $x = 1, e^2$.

(3) For iteration 50 times, there are 10 points recovered because $2^{-(n-52)}lnx$ has 10 integer values.

(4) For iteration 49 times, there are 19 points recovered because $2^{-(n-52)}lnx$ has 19 integer values.

7. (a) Calculated by Matlab, we can get that

$$\begin{aligned} w(x) = x^{20} - 210\,x^{19} + 20615x^{18} - 1256850\,x^{17} + 53327946x^{16} - 1672280820x^{15} + 40171771630x^{14} \\ - 756111184500x^{13} + 11310276995381x^{12} - 135585182899530x^{11} \\ + 1307535010540395x^{10} \quad - 10142299865511450x^{9} + 63030812099294896x^{8} \\ - 311333643161390640x^{7} + \ 1206647803780373360x^{6} \ - 3599979517947607200x^{5} \\ + 8037811822645051776x^{4} - 12870931245150988800x^{3} + 13803759753640704000x^{2} \\ - 8752948036761600000x + 2432902008176640000 \end{aligned}$$

(b)

Using Matlab roots() function to solve the polynomial, we ca get the solution near 21 is 19.9998.

I write Newton-Raphson iteration code as follows. From my code, I find that although the initial guess is 21, really near 20. But the final solution value is 1.

```
coef=[2.43290200817664e+18,-8.75294803676160e+18,1.38037597536407e+19,-
1.28709312451510e+19,8.03781182264505e+18,-3.59997951794761e+18,1.20664780378037e+18,-
3.11333643161391e+17,6.30308120992949e+16,-1.01422998655115e+16,1.30753501054040e+15,-
135585182899530,11310276995381.0,-756111184500.000,40171771630.0000,-1672280820.00000,53327946,-
1256850,20615,-210,1];
len=length(coef); %number of coefficients of the given polynomial.
x=21; % initial guess(This can be set to any value as the preference)
xold=0;
error=1;
while(abs(error)>0.00001) %answer is correct up to 0.00001.
 y=0;
 ydif=0;
  xold=x;%record old value
    for n=0:len-1
    y=y+coef(len-n)*x^n; %calculating the polynomial value
    ydif=ydif+n*coef(len-n)*x^(n-1); %calculating the derivative value
    end
    error=y/ydif;
    x=x-y/ydif; %newly approximated value for the newtown rapson method
end
disp(x);
```

(c) Using Matlab roots() function to solve the polynomial, the result for different δ is shown in the following table.

δ	Largest root
10^{-2}	38.478184
10^{-4}	28.400212
10^{-6}	23.149016
10^{-8}	20.647583

From the calculation result, we can see that the largest root increase with δ increases.

(d)

Initially, roots 16 and 17 are 17.0254271462374 and 15.9462867166080.

If set $a_{19} = -210 - 2^{-23}$, roots 16 and 17 become 16.7307448797927 + 2.81262489672198i and 16.7307448797927 - 2.81262489672198i.

We can see that small perturbation makes roots 16 and 17 become conjugate complex numbers from real numbers.

(e)

7. (e) (i) $\Gamma_{kl} = \left| \frac{\varepsilon_k}{\varepsilon_l} \right|$, Here ε_l is the perturbation of a_l

ε_k is the resulted perturbation on Ω_k

If we make a perturbation on a_l, and then we get the resulted solution

$$\underline{\text{perturbation on } a_l} \Downarrow a_l(1+\varepsilon_l) \qquad \underline{\text{resulted solution}} \Downarrow \Omega_k(1+\varepsilon_k)$$

then $$a_0 + a_1\Omega_k(1+\varepsilon_k) + \cdots + a_l(1+\varepsilon_l)[\Omega_k(1+\varepsilon_k)]^l + \cdots + a_n[\Omega_k(1+\varepsilon_k)]^n = 0$$

$\Downarrow$ neglect higher order of ε_k

$$\underbrace{a_0 + a_1\Omega_k + a_2\Omega_k^2 + \cdots + a_l\Omega_k^l + \cdots + a_n\Omega_k^n}_{=0} + a_1\Omega_k\varepsilon_k + a_2\Omega_k^2 \cdot 2\varepsilon_k + \cdots + a_l\Omega_k^l(l\varepsilon_k + \varepsilon_l) \cdots + a_n\Omega_k^n n\varepsilon_k = 0$$

$\Downarrow$

$$(a_1\Omega_k + 2a_2\Omega_k^2 + \cdots + l a_l\Omega_k^l + \cdots + n a_n\Omega_k^n)\varepsilon_k + a_l\Omega_k^l\varepsilon_l = 0$$

therefore, $$P'(\Omega_k)\cdot\Omega_k\cdot\varepsilon_k + a_l\Omega_k^l\varepsilon_l = 0$$

$\Downarrow$

$$\left|\frac{\varepsilon_k}{\varepsilon_l}\right| = \left|\frac{a_l\Omega_k^l}{P'(\Omega_k)\cdot\Omega_k}\right| = \left|\frac{a_l\Omega_k^{l-1}}{P'(\Omega_k)}\right|$$

$$\therefore (\mathrm{cond}\,\Omega_k)(\vec{a}) \equiv \sum_{l=0}^{n-1}\left|\frac{a_l\Omega_k^{l-1}}{P'(\Omega_k)}\right|$$

(ii) the condition numbers for the roots r=14,16,17,20 are shown in the following table.

r	Condition number
14	53951642837965.9
16	35404232858970.3
17	18130215927254.7
20	137803429015.878

From the calculation result, we can see that r=14,16,17,20 is really bad-conditioned. And that is why small perturbations of coefficient in part (c) and part (d) can lead to large variant in the roots.

(iii) I don't think there is a sufficient clever algorithm help us here because the condition number in (ii) is too large. The value of the condition number calculated in (ii) is independent of algorithm and only depend on the form of the polynomial.

8.

8. (a) $y_n = \dfrac{e - y_{n+1}}{n+1} = \dfrac{e}{n+1} - \dfrac{y_{n+1}}{n+1}$

$\Downarrow$

$y_{n-1} = \dfrac{e}{n} - \dfrac{y_n}{n}$

$y_{n-2} = \dfrac{e}{n-1} - \dfrac{y_{n-1}}{n-1} = \dfrac{e}{n-1} - \dfrac{e - y_n}{n(n-1)} = \dfrac{y_n}{n(n-1)} + \dfrac{e}{n}$

$y_{n-3} = \dfrac{e - y_{n-2}}{n-2} = \dfrac{e}{n-2} - \dfrac{\left(\frac{e}{n-1} - \frac{e-y_n}{n(n-1)}\right)}{n-2}$

$= -\dfrac{y_n}{n(n-1)(n-2)} + \dfrac{(n-1)e}{n(n-2)}$

$\vdots$

$|y_{n-k}| = \left|\dfrac{y_n}{n\cdot(n-1)\cdots(n-k+1)}\right| + C(n)$, $C(n)$ is a value relavent to n.

$\therefore$ For error propagation,

$\left|y_{n-k}(1+\varepsilon_k)\right| = \left|\dfrac{y_n(1+\varepsilon_n)}{n\cdot(n-1)\cdots(n-k+1)}\right|$

$\Downarrow$

$\left|y_{n-k}\,\varepsilon_k\right| = \left|\dfrac{y_n\,\varepsilon_n}{n\cdot(n-1)\cdots(n-k+1)}\right|$

$\therefore \left|\dfrac{\varepsilon_k}{\varepsilon_n}\right| = \left|\dfrac{y_n\cdot k!}{y_{n-k}\cdot n!}\right| \leqslant \dfrac{k!}{n!}$ (since $y_n \geqslant y_{n-k}$ if $k \geq 0$)

$\therefore (\operatorname{cond} g_k)(y_n) = \left|\dfrac{y_n\cdot k!}{y_{n-k}\cdot n!}\right| \leqslant \dfrac{k!}{n!}$, the upper limit is $\dfrac{k!}{n!}$

(b) Assume $|\varepsilon_k| = \frac{k!}{N!}|\varepsilon_N|$

When $\varepsilon_N = 100\%$, $\frac{k!}{N!} = |\varepsilon_k|$

if the target relative error in y_k is ε, (assume ε is a positive number)

$$|\varepsilon_k| \le \varepsilon \Rightarrow \frac{k!}{N!} \le \varepsilon \Rightarrow N! \geq \frac{k!}{\varepsilon}$$

$\therefore$ the minimum value of N should satisfy

$$N_{min}! = \frac{k!}{\varepsilon}$$

(c) $\varepsilon = eps = 2^{-52}$, $N_{min}! = \frac{20!}{\varepsilon} \Rightarrow N_{min} = 32$

$\therefore$ if $N = 32$, we can achieve this error in y_{20}.

(d) Using Matlab, the value of y_{20} is 0.123803830762570.

Using Mathematica, the integration value is 0.12380383076256994869.

Compare those result, we can see that those results agree really well. And backward method is well enough to calculate the integral.

The Matlab code is shown as follows

```
y = 0; %N=32, y=0 as the start point
e = exp(1);
for i= 1:12
   N = 32-i;
   y=(e-y)/(N+1);%calculate yN
   disp(y);
end
```