

V. Tugas Praktikum

5.1 $f(x) = \sin x$ disekitar $x=0$ dengan tingkat kesalahan $\varepsilon = 0.0000001$

Listing:

```
#include<stdio.h>

#include<math.h>

#define RAD M_PI/180

float faktorial(float i){

    if (i==0)

        return 1;

    else

        return i*faktorial(i-1);

}

main() {

    float  x, hasil=1, toleransi=0.00000001, jumlah = 0, e,i;

    int n;

    printf("Masukkan x :  \n");

    scanf("%f",&x);

    for(i=0;fabs (hasil)>=toleransi;i++)

    {
```

```

hasil=pow(-1,i)*pow(x*RAD,2*i+1)/faktorial(2*i+1);

jumlah=jumlah+hasil;

e=fabs (jumlah-sin(x*RAD));

printf("%.0f \t%.10f \t%.10f \n" ,i ,jumlah, e);

}

}

```

```

C:\Users\MIPA13\Documents\wowow.exe
Masukkan x : 30
0      0.5235987902    0.0235987902
1      0.4996742010    0.0003257990
2      0.5000021450    0.0000021450
3      0.5000000000    0.0000000000
4      0.5000000000    0.0000000000

Process exited after 1.6 seconds with return value 0
Press any key to continue . . .

```

Analisa: Ini adalah program pendekatan sin x. Seperti contoh sin 30 memiliki banyak pendekatan hasil sampai dengan 8 iterasi. Karena tidak dibatasi berapa kali iterasi maksimalnya, maka iterasi akan terus dilakukan sampai batas error yang telah ditentukan. Cara memperbesar tingkat batas error dengan mengganti toleransi nya. Jika diperbesar, maka akan semakin banyak iterasi yang dilakukan

5.2 $f(x) = \ln x$ di sekitar $x=1$ hingga suku ke-8

Listing:

```
#include<stdio.h>

#include<math.h>

main() {

float  x, hasil, toleransi=0.000000001, jumlah = 0, e, n,
jumlahsebelum=0;

printf("Masukkan x :  \n");

scanf("%f",&x);

for (n=1;n<=8;n++)

{

hasil=(pow(-1,n-1)*pow(x-1,n))/n;

jumlah=jumlah+hasil;

e=fabs (jumlah-jumlahsebelum)/jumlah;

printf("%.0f \t%.10f \t%.10f \n" ,n ,jumlah, e);

jumlahsebelum=jumlah;

}

}
```

```
C:\Users\MIPA13\Documents\mantap.exe
Masukkan x :
2
1 1.0000000000 1.0000000000
0.5000000000 1.0000000000
3 0.8333333331 0.4000000358
0.5833333331 0.4285714838
4 0.7933333611 0.2553191185
0.6166666746 0.2782762888
5 0.7595238898 0.1888877614
0.6345238898 0.1969981194
Process exited after 6.788 seconds with return value 0
Press any key to continue . . .
```

Analisa: ini adalah fungsi $\ln x$. Batas iterasi nya hanya sampai 8, maka dari itu jika kita menggunakan kalkulator biasanya tidak akan sama persis dengan hasil ini. Dengan adanya batas, maka pengulangan (iterasi) nya dibatasi hingga batas-n. Error ini dinamakan *round-off error*.

5.3 $f(x) = e^x$ di sekitar $x=0$ dengan $\varepsilon = 0.000000001$

Listing:

```
#include<stdio.h>

#include<math.h>

float faktorial(float i){

if (i==0)

return 1;

else

return i*faktorial(i-1);
```

```

}

main() {

float  x, hasil=1, toleransi=0.00000001, jumlah = 0, e,i;

int n;

printf("Masukkan x :  \n");

scanf("%f",&x);

printf("Iterasi      Hampiran      Error \n");

printf("----- \n");

for(i=0;fabs (hasil)>=toleransi;i++)

{

hasil=pow(x,i)/faktorial(i);

jumlah=jumlah+hasil;

e=fabs (jumlah-exp(x));

printf("%.0f \t%.10f \t%.10f \n" ,i ,jumlah, e);

}

}

```

```

C:\Users\MIPA13\Documents\heuu.exe
Masukkan x =
Iterasi  Hampiran      Error
0 1.0000000000000000 1.7182818651
1 0.0000000000000000 0.7182818651
2 0.0000000000000000 0.2182818653
3 0.0000000000000000 0.0516150812
4 0.0000000000000000 0.0079481365
5 0.0000000000000000 0.0016140916
6 0.0000000000000000 0.0002561654
7 0.0000000000000000 0.0000277391
8 0.0000000000000000 0.0000029436
9 0.0000000000000000 0.0000000825
10 0.0000000000000000 0.0000001559
11 0.0000000000000000 0.0000001559
12 0.0000000000000000 0.0000001559

Process exited after 0.3972 seconds with return value 0
Press any key to continue . . .

```

Analisa: Ini adalah program eksponensial. Memiliki rumus $\frac{x^n}{n!}$. Seperti program pertama, program ini akan melakukan iterasi hingga batas error 10^{-9} . Jika sudah melebihi dari batas, maka operasi program akan dihentikan dan muncul seperti di gambar atas.

VI. Tugas Akhir

6.1 Mengapa metode numeric menjadi penting untuk dipelajari? Padahal hasil dari metode tersebut masih memiliki kesalahan? Jelaskan!

Karena metode numeric dipelajari untuk menghampiri suatu fungsi ke nilai yang sebenarnya. Jadi agar kita tahu seberapa banyak range hasil dari suatu fungsi x

6.2 Hampiri persamaan berikut,

$$f(x) = -0.5x^2 - 0.25x + 1.2$$

Listing:

```
#include<stdio.h>

#include<math.h>


main() {

float x, hasil, nilaisejati, suku0,
suku1,suku2,suku3,suku4,orde0,orde1,orde2,orde3,orde4,e ;

printf("Masukkan x: \n");

scanf("%f",&x);

nilaisejati=-0.5*x*x-0.25*x+1.2;

suku0 = 1.2;

suku1 = -0.25;

suku2 = -1/2;

suku3 = 0;

suku4 = 0;

orde0 = suku0;

orde1 = orde0 + suku1;

orde2 = orde1 + suku2;

orde3 = orde2 + suku3;

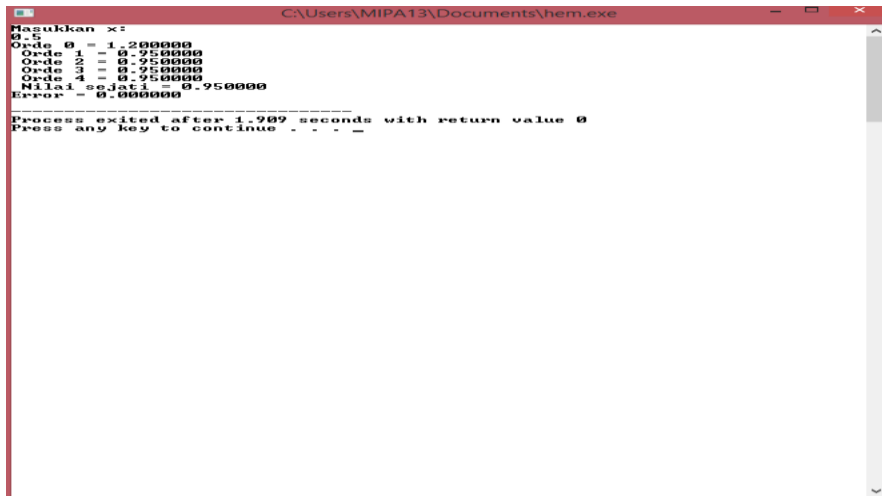
orde4 = orde3 + suku4;

printf("Orde 0 = %f\n Orde 1 = %f\n Orde 2 = %f\n Orde 3 = %f\n Orde
4 = %f\n ", orde0, orde1, orde2, orde3, orde4);

printf("Nilai sejati = %f \n", nilaisejati);
```

```
printf("Error = %f\n", e=fabs(nilaisejati-orde4));

}
```



```

C:\Users\MIPA13\Documents\hem.exe
Masukkan x:
0.5
Orde 0 = 1.200000
Orde 1 = 0.950000
Orde 2 = 0.950000
Orde 3 = 0.950000
Orde 4 = 0.950000
Nilai sejati = 0.950000
Error = 0.000000
=====
Process exited after 1.289 seconds with return value 0
Press any key to continue . . .

```

Analisa:

Ini adalah program perbandingan nilai sejati dengan hampiran. Nilai sejati adalah nilai di mana x dimasukkan langsung ke dalam fungsi. Dan hampiran seperti deret Maclaurin dengan menggunakan rumus seperti deret Maclaurin. Errornya adalah pengurangan dari nilai sejati dikurangi orde paling akhir.

VII. Kesimpulan

7.1. Persamaan matematis dapat diubah ke deret Taylor dengan menggunakan rumus deret.

7.2 Deret Taylor adalah deret untuk menurunkan fungsi ke dalam suatu hampiran hasil nilainya