

SIMULATOR RANGKAIAN RC DAN DIFFERENTIATOR OPERATIONAL AMPLIFIER

LAPORAN

Laporan pengerjaan tugas besar mata kuliah Pemecahan Masalah dengan C (EL2008)

oleh

Fazha Ivanda 13218008

Apria Wati 13218028

Matthew Terrence A. H. 13218038

Martinus William Hartono 13218044



**PROGRAM STUDI TEKNIK ELEKTRO
INSTITUT TEKNOLOGI BANDUNG**

2020

DAFTAR ISI

1.	DESKRIPSI PERMASALAHAN	3
2.	DESAIN RANGKAIAN	3
2.1	RANGKAIAN RC TIPE 1.....	3
2.2	RANGKAIAN RC TIPE 2.....	4
2.3	RANGKAIAN RC TIPE 3.....	4
2.4	RANGKAIAN RC TIPE 4.....	5
2.5	RANGKAIAN RC TIPE 5.....	5
2.6	RANGKAIAN DIFFERENSIATOR	6
3.	ANALISIS	6
3.1.	RANGKAIAN RC TIPE 1.....	6
3.2.	RANGKAIAN RC TIPE 2.....	7
3.3.	RANGKAIAN RC TIPE 3.....	8
3.4.	RANGKAIAN RC TIPE 4.....	8
3.5.	RANGKAIAN RC TIPE 5.....	9
3.6.	RANGKAIAN DIFFERENSIATOR	10
3.7.	TKINTER GRAPHICAL USER INTERFACE	12
3.8.	COMMAND LINE ARGUMENT	12
3.9.	PLOTTING GRAFIK DENGAN MATPLOTLIB	13
4.	FLOWCHART DAN DATA-FLOW DIAGRAM	13
5.	TES UNIT DAN FUNGSIONAL	28
5.1	UNIT TEST	29
5.1.1	Function: derive	29
5.1.2	Function: deriveTrig	31
5.1.3	Function: triangularFunc	34
5.1.4	Function: squareFunc	36
5.1.5	Procedure: printToFile	38
5.1.6	Function: rangkaian1 (voltage)	40
5.1.7	Function: rangkaian1 (current)	41
5.1.8	Function: rangkaian2 (voltage).....	42
5.1.9	Function: rangkaian2 (current)	44
5.1.10	Function: rangkaian3 (voltage)	45
5.1.11	Function: rangkaian3 (current)	46
5.1.12	Function: rangkaian4 (voltage)	47

5.1.13	Function: rangkaian4 (current)	48
5.1.14	Function: rangkaian5 (voltage)	49
5.1.15	Function: rangkaian5 (current)	50
5.2	<i>FUNCTIONAL TEST</i>	52
5.2.1	GUI to <i>Command Line Argument</i>	53
5.2.2	Run Program dengan <i>Command Line Argument</i>	55
5.2.3	CSV to Plot.....	58
6.	IMPLEMENTASI.....	60
7.	PENGUJIAN KESELURUHAN.....	79
7.1	GRAPHICAL USER INTERFACE.....	79
7.2	PENGUJIAN RANGKAIAN RC	81
7.3	PENGUJIAN RANGKAIAN DIFFERENTIATOR	85
8.	ANALISIS PENGUJIAN.....	89
8.1	ANALISIS GRAPHICAL USER INTERFACE	89
8.2	ANALISIS KETEPATAN SIMULASI RANGKAIAN RC	91
8.3	ANALISIS KETEPATAN SIMULASI RANGKAIAN DIFFERENTIATOR	95
9.	KESIMPULAN DAN <i>LESSON LEARNED</i>	100
10.	PEMBAGIAN TUGAS	101
11.	DAFTAR PUSTAKA.....	102

1. DESKRIPSI PERMASALAHAN

Pada tugas besar Pemecahan Masalah dengan C ini, kelompok diberi tugas untuk melakukan eksplorasi rangkaian dengan menggunakan *nodal analysis*. Simulator tersebut akan terdiri dari beberapa bagian besar yaitu *graphical user interface* (GUI), algoritma perhitungan dengan bahasa C, dan plot output dengan menggunakan *matplotlib python plotting library*. Program secara umum akan menerima input dari *user* berupa nilai komponen dan akan memberi output berupa kurva tegangan setiap waktu pada *node* yang telah ditentukan oleh kelompok serta kurva arus setiap waktu pada *branch* yang telah ditentukan oleh kelompok.

Program yang dirancang oleh kelompok secara umum terdiri dari dua struktur utama, yaitu *interface* dan *plotting* dengan menggunakan bahasa Python dan kode program perhitungan analisis rangkaian dengan menggunakan bahasa C. Secara umum, simulator yang dibuat akan menerima input dari *user* pada *interface* yang dibuat dengan menggunakan Tkinter pada bahasa Python 3.7. Setelah user memberikan input pada interface, data tersebut akan diolah menjadi *command line argument* untuk menjalankan *executable file* dari perhitungan dengan menggunakan bahasa C. Output dari perhitungan adalah file csv yang berisi waktu dan tegangan atau waktu dan arus setiap waktu dari simulasi rangkaian yang telah dilakukan. Kemudian, data csv tersebut akan diplot dengan menggunakan library matplotlib pada bahasa Python.

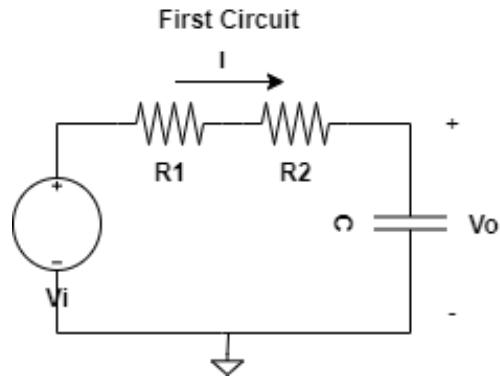
2. DESAIN RANGKAIAN

Dalam menyelesaikan permasalahan yang diberikan, akan dilakukan beberapa pembatasan permasalahan yang telah dilakukan oleh kelompok:

- Rangkaian yang disimulasikan oleh kelompok merupakan rangkaian RC (rangkaian yang terdiri dari kapasitor dan resistor) dengan sebuah sumber tegangan.
- Analisis yang dilakukan merupakan analisis *transient*.
- Jenis rangkaian yang disimulasikan oleh kelompok merupakan rangkaian differensiator dengan menggunakan *operational amplifier* yang terdiri sebuah kapasitor dan sebuah resistor *feedback* serta resistor *load* sebagai output yang besarnya sama dengan resistor *feedback*.
- Rangkaian differensiator akan terhubung dengan sebuah sumber tegangan yang mengeluarkan tegangan berupa sinyal periodik (*sinus*, *cosinus*, *triangular*, *square wave*) dan sinyal output yang ditinjau merupakan sinyal tegangan output op-amp.
- *User* akan menginput nilai resistansi, kapasitansi, jenis sinyal tegangan, amplitudo, dan periode dari sinyal.
- Proses perhitungan akan dilakukan dengan metode *iterasi* dengan jumlah iterasi 1000 buah untuk rangkaian RC dan 20000 buah untuk rangkaian differentiator.

2.1 RANGKAIAN RC TIPE 1

Rangkaian RC tipe 1 ini merupakan rangkaian yang tersusun dari sebuah sumber tegangan DC, dua buah resistor dan sebuah kapasitor yang disusun secara seri. Berikut ini merupakan rangkaian yang akan dianalisis:



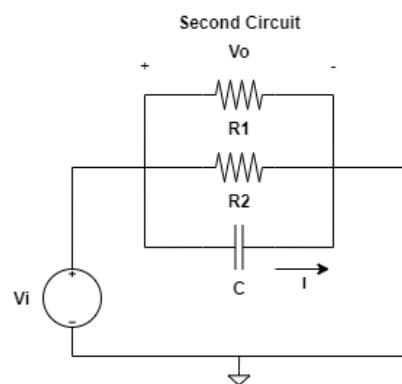
Gambar 2.1 Rangkaian RC Tipe 1

Nilai komponen yang akan diinput oleh user pada rangkaian tipe 1, 2, 3, 4, dan 5 adalah:

- Tegangan sumber V_i
- Resistansi R_1
- Resistansi R_2
- Kapasitansi C

2.2 RANGKAIAN RC TIPE 2

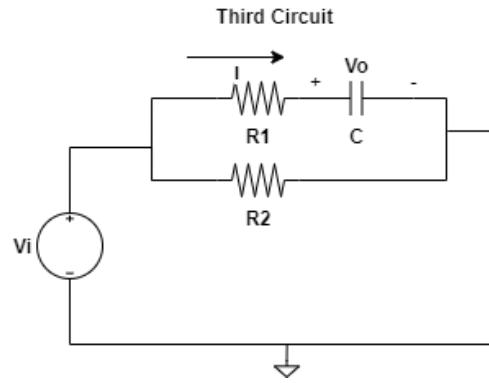
Rangkaian RC tipe 2 merupakan rangkaian yang terdiri dari dua resistor dan sebuah kapasitor yang disusun secara pararel. Berikut ini merupakan rangkaian yang akan dianalisis:



Gambar 2.2 Rangkaian RC Tipe 2

2.3 RANGKAIAN RC TIPE 3

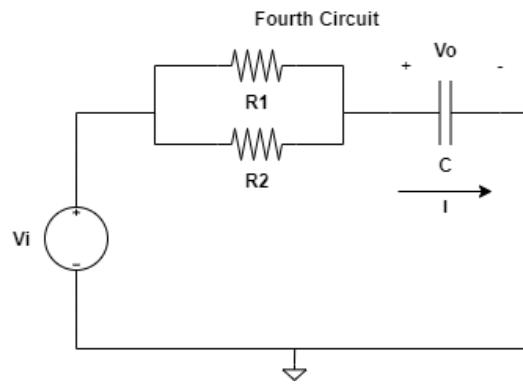
Rangkaian RC tipe 3 merupakan rangkaian yang terdiri dari sebuah resistor yang disusun seri dengan kapasitor, kemudian disusun secara pararel dengan resistor lain. Berikut ini merupakan rangkaian RC tipe 3 yang dianalisis:



Gambar 2.3 Rangkaian RC Tipe 3

2.4 RANGKAIAN RC TIPE 4

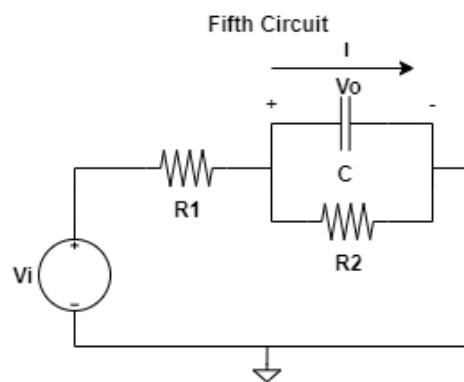
Rangkaian RC tipe 4 merupakan rangkaian dua buah resistor yang disusun pararel dan kemudian disusun secara seri dengan sebuah kapasitor dan sumber tegangan. Berikut ini merupakan rangkaian yang akan digunakan:



Gambar 2.4 Rangkaian RC Tipe 4

2.5 RANGKAIAN RC TIPE 5

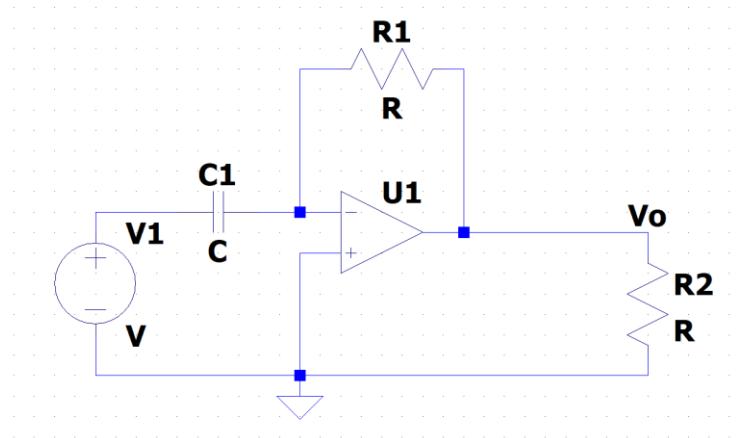
Rangkaian RC tipe 5 merupakan rangakain sebuah resistor yang disusun secara seri dengan resistor yang disusun secara pararel dengan sebuah kapasitor dan sumber tegangan. Berikut ini merupakan rangakain tipe 5 yang akan digunakan:



Gambar 2.5 Rangkaian RC Tipe 5

2.6 RANGKAIAN DIFFERENSIATOR

Selain kelima rangkaian RC, rangkaian juga dapat disimulasi dengan menggunakan aplikasi simulator yang dibuat adalah rangkaian differensiator op-amp orde 1 dengan kapasitor dan sebuah resistor *feedback*. Berikut ini merupakan rangkaian yang akan digunakan:



Gambar 2.6 Rangkaian Differentiator

Pada gambar 2.1, terdapat rangkaian differensiator yang digunakan sebagai analisis. Dalam melakukan analisis rangkaian tersebut, komponen yang digunakan diasumsikan ideal. Op-amp tersebut ideal sehingga tidak ada arus yang masuk pada input inverting maupun noninverting dari opamp. Hal ini menyebabkan tidak ada perbedaan tegangan antara kedua buah input opamp.

Pada rangkaian ini, *user* akan memberikan input beberapa variabel. Beberapa variabel yang akan diinput oleh user adalah:

- Resistansi resistor (Ω)
- Kapasitansi kapastor (F)
- Jenis gelombang tegangan input (*sinus, cosinus, triangular, square wave*)
- Periode gelombang input (s)
- Amplitudo gelombang input (Vp)

3. ANALISIS

Pada bagian ketiga dari laporan ini akan dijelaskan penurunan rumus dan pendekatan yang digunakan pada algoritma program.

3.1. RANGKAIAN RC TIPE 1

Bila ditinjau rangkaian pada gambar 2.1 dapat diturunkan sebuah pendekatan yaitu dengan meninjau *mesh* pada rangkaian, diperoleh:

$$\begin{aligned} \sum_1^n Vn &= 0 \\ -Vs + V_{R1} + V_{R2} + Vo &= 0 \\ Vo &= Vs - V_{R1} - V_{R2} \\ Vo = Vs - i(R1 + R2) &\quad (\text{pers. 1}) \\ i = C \frac{dv}{dt} &\quad (\text{pers. 2}) \end{aligned}$$

Pada kapasitor yang diberi tegangan DC, arus yang mengalir pada kapasitor dapat didekati dengan hubungan pada persamaan 2 dan karena resistor dan kapasitor disusun seri, arus di kedua komponen tersebut sama sehingga:

$$Vo = Vs - (R1 + R2)C \frac{dv}{dt}$$

Dengan τ merupakan konstanta waktu yang bernilai RC , maka:

$$\begin{aligned} \tau &= (R1 + R2)C \\ Vo = Vs - \tau \frac{dv}{dt} &\quad (\text{pers. 3}) \end{aligned}$$

dv atau perubahan nilai tegangan dapat didekati dengan :

$$dv = Vo(t) - Vo(t - 1) \quad (\text{pers. 4})$$

Dengan mensubstitusikan pers. 4 ke pers. 3, didapat:

$$\begin{aligned} Vo(t) &= Vs - \tau \frac{Vo(t) - Vo(t - 1)}{dt} \\ Vo(t)dt &= Vs - \tau[Vo(t) - Vo(t - 1)] \\ Vo(t)[dt + \tau] &= Vsdt - Vo(t - 1)\tau \\ Vo(t) = \frac{Vsdt + Vo(t - 1)\tau}{dt + \tau} &\quad (\text{pers. 5}) \end{aligned}$$

Persamaan Tegangan Output pada Rangkaian RC Tipe 1

$$I = \frac{Vi - Vo}{R1 + R2}$$

Persamaan Arus Output pada Rangkaian RC Tipe 1

3.2. RANGKAIAN RC TIPE 2

Bila ditinjau rangkaian pada gambar 2.2, terlihat bahwa tegangan output yang diamati pada rangkaian tersebut untuk setiap waktu akan selalu bernilai sama dengan nilai tegangan inputnya, hal tersebut juga menyebabkan arus yang mengalir pada *branch* yang telah ditentukan bernilai nol atau dengan kata lain berupa open circuit. Berikut ini kondisi tersebut dalam bentuk persamaan:

$$Vo = Vi$$

Persamaan Hubungan Tegangan Output dengan Tegangan Input pada Rangkaian Tipe 2

$$I_o = 0$$

Persamaan Arus Output pada Rangkaian Tipe 2

3.3. RANGKAIAN RC TIPE 3

Rangkaian RC Tipe 3 merupakan rangkaian paralel antara kapasitor C dan resistor R1 yang disusun seri dengan resistor R2 serta sumber tegangan Vin. Output rangkaian diambil dari tegangan pada kapasitor(Vc). Persamaan tegangan output adalah sebagai berikut:

Arus pada kapasitor memenuhi:

$$I_c = C \frac{d V_c}{dt}$$

Dengan menggunakan analisis nodal, arus yang melalui kapasitor adalah:

$$I_c = \frac{V_{R1} - V_c}{R1}$$

Karena rangkaian bersifat paralel, VR1 akan sama dengan Vin, sehingga

$$I_c = \frac{V_{in} - V_c}{R1}$$

Persamaan Arus Output pada Rangkaian Tipe 3

Maka, Tegangan keluaran rangkaian adalah :

$$\begin{aligned} I_c &= C \frac{d V_c}{dt} \\ I_c &= \frac{V_{in} - V_c}{R1} \\ C \frac{d V_c}{dt} &= \frac{V_{in} - V_c}{R1} \\ d V_c &= \frac{(V_{in} - V_c)}{R1 \cdot C} dt \\ d V_c &= \frac{(V_{in} - V_c)}{R1 \cdot C} dt \\ d V_c &= V_c(t) - V_c(t - 1) \\ \frac{V_c(t) - V_c(t - 1)}{dt} &= \frac{V_{in} - V_c(t)}{R1 \cdot C} \\ V_c(t) &= \frac{V_c(t - 1)(R1 \cdot C) + V_{in} * dt}{R1 \cdot C + dt} \end{aligned}$$

Persamaan Tegangan Output pada Rangkaian Tipe 3

Sehingga, $V_o(t) = V_c(t) = \frac{V_c(t-1)(R1.C)+V_{in}*dt}{R1.C+dt}$. Tegangan pada kapasitor akan

bernilai maksimal ketika tegangannya sama dengan tegangan pada R2 yaitu sama dengan Vin karena rangkaian bersifat paralel.

3.4. RANGKAIAN RC TIPE 4

Pada rangkaian tipe 4, dua buah resistor disusun paralel dan diserikan dengan sebuah kapasitor dan sumber tegangan DC. Berikut ini terdapat penurunan rumus pendekatan tegangan output dan arus output pada rangkaian RC tipe 4:

$$Ic = C \frac{dVc}{dt}$$

Dilakukan KVL pada rangkaian:

$$-Vi + (R_1//R_2) Ic + Vc = 0$$

$$(R_1//R_2) Ic = Vi - Vc$$

$$Ic = \frac{Vi - Vc}{R_1//R_2}$$

Persamaan Arus Output pada Rangkaian 4

Samakan kedua persamaan diatas:

$$C \frac{dVc}{dt} = \frac{Vi - Vc}{R_1//R_2}$$

$$\frac{dVc}{dt} = \frac{Vi - Vc}{C(R_1//R_2)}$$

Dapat diketahui bahwa $Vc = Vo$

$$\frac{dVo}{dt} = \frac{Vi - Vo}{C(R_1//R_2)}$$

Dengan dVo dapat didekati dengan $Vo(t) - Vo(t-1)$

$$\frac{Vo(t) - Vo(t-1)}{dt} = \frac{Vi - Vo(t)}{C(R_1//R_2)}$$

$$C(R_1//R_2) [Vo(t) - Vo(t-1)] = dt (Vi - Vo(t))$$

$$Vo(t)[C(R_1//R_2) + dt] = Vi \cdot dt + Vo(t-1)[C(R_1//R_2)]$$

$$Vo(t) = \frac{Vi \cdot dt + Vo(t-1)[C(R_1//R_2)]}{C(R_1//R_2) + dt}$$

Persamaan Tegangan Output pada Rangkaian RC Tipe 4

3.5. RANGKAIAN RC TIPE 5

Jika ditinjau rangkaian RC Tipe 5 yang terdapat pada gambar 2.5, tegangan output dari rangkaian tersebut merupakan tegangan pada kapasitor dan arus output merupakan arus pada *branch* yang terdapat komponen kapasitor. Berikut ini terdapat penurunan persamaan tegangan dan arus output pada rangkaian tipe 5, dengan *nodal analysis*:

$$\frac{Vo - Vi}{R1} + \frac{Vo}{R2} + C \frac{dVi}{dt} = 0$$

$$\frac{Vo - Vi}{R1} + \frac{Vo}{R2} + C \frac{Vo(i) - Vo(i-1)}{dt} = 0$$

$$Vo \left(\frac{1}{R1} + \frac{1}{R2} + \frac{C}{dt} \right) = C \frac{Vo(i-1)}{dt} + \frac{Vi}{R1}$$

$$Vo = \frac{\left(C \frac{Vo(i-1)}{dt} + \frac{Vi}{R1} \right)}{\frac{1}{R1} + \frac{1}{R2} + \frac{C}{dt}}$$

Persamaan Tegangan Output Rangkaian Tipe 5

$$I_o = C \frac{dV_o}{dt}$$

Persamaan Arus Output Rangkaian 5

3.6. RANGKAIAN DIFFERENSIATOR

Jika ditinjau rangkaian yang terdapat pada gambar 2.6, dapat diturunkan sebuah pendekatan yang akan mempermudah analisis untuk dilakukan. Seperti yang telah dipaparkan pada bagian 2.1, tidak ada arus yang masuk pada input *inverting* dan *non-inverting* pada opamp, hal tersebut menyebabkan arus yang mengalir pada kapasitor sama dengan arus yang mengalir pada resistor. Berikut ini hubungan antara tegangan input dengan output pada rangkaian tersebut:

$$\begin{aligned} i_r &= i_c \\ \frac{0 - V_o}{R} &= C \frac{d(V_i - 0)}{dt} \\ V_o &= -RC \frac{dV_i}{dt} \end{aligned}$$

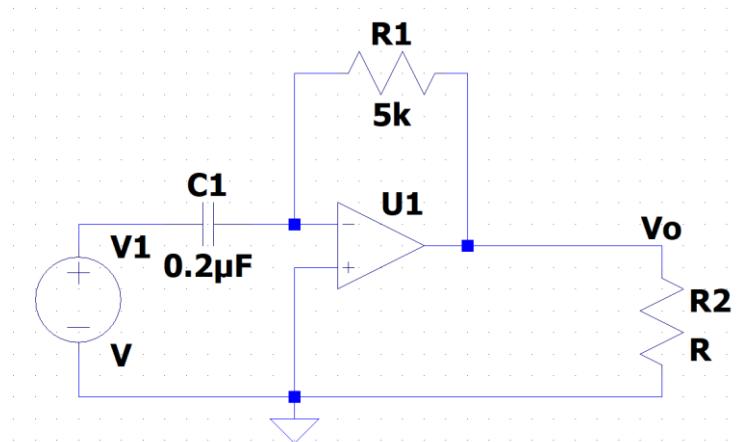
Persamaan Tegangan Output pada Rangkaian Differensiator

$$I_o = -C \frac{dV_i}{dt}$$

Persamaan Arus Output pada Rangkaian Differensiator

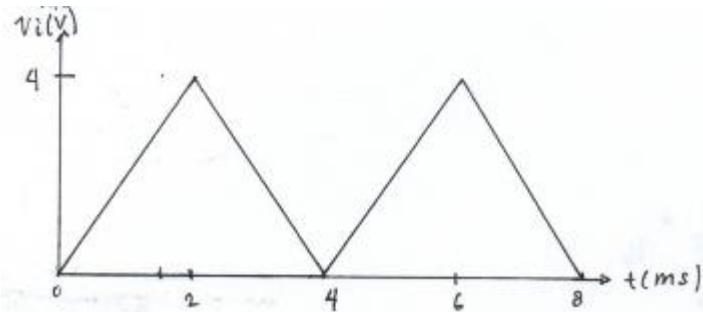
Pada persamaan di atas, terlihat bahwa sinyal tegangan output dari rangkaian tersebut merupakan diferensiasi dari tegangan input dan dikali oleh konstanta $-RC$. Berikut ini terdapat contoh kasus perhitungan rangkaian:

Misalkan terdapat rangkaian differensiator dengan resistor sebesar $5\text{k}\Omega$ dan kapasitor sebesar $0,2\mu\text{F}$ dan diberi input berupa sinyal segitiga dengan periode 4ms dan tegangan puncak 4V. Berdasarkan pendekatan rumus yang telah dibuat, dapat ditentukan sinyal tegangan output dengan:



Gambar 3.6.1 Contoh Kasus Rangkaian Differensiator

Model sinyal tegangan input rangkaian:



Gambar 3.6.2 Sinyal Input Rangkaian

Konstanta waktu rangkaian dapat ditentukan dengan rumus:

$$-RC = -5 \times 10^3 \times 2 \times 10^{-7} = 10^{-3}s$$

Sinyal input dapat dinyatakan dengan:

$$Vi = 2000t, \quad 0ms < t < 2ms$$

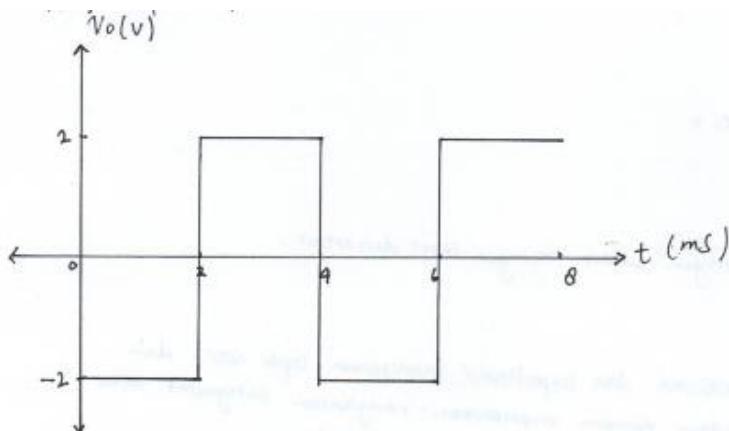
$$Vi = 8 - 2000t, \quad 2ms < t < 4ms$$

Jika ditinjau masing-masing selang, dapat ditentukan nilai tegangan output yaitu:

$$Vo = -10^{-3} \times 2000 = 2V \text{ untuk selang } 0 < t < 2ms$$

$$Vo = -10^{-3} \times -2000 = -2V \text{ untuk selang } 2ms < t < 4ms$$

Nilai tersebut akan berulang dengan periode T=4ms, sehingga didapat sinyal tegangan output yaitu:



Gambar 3.6.3 Sinyal Output Rangkaian

Jika dilihat dari pendekatan rumus hubungan sinyal tegangan output dengan tegangan input, dapat dilihat bahwa algoritma program membutuhkan beberapa fungsi dan prosedur yang mempermudah penggeraan aplikasi simulator. Untuk rangkaian differensiator, terlihat bahwa dibutuhkan *function* untuk diferensiasi fungsi, serta diperlukan *function* untuk mengetahui nilai fungsi setiap waktu.

Untuk mencari nilai turunan dari suatu fungsi, akan digunakan definisi dari turunan sebagai berikut:

$$f'(a) = \lim_{x \rightarrow 0} \frac{f(a+h) - f(a)}{h}$$

Persamaan Definisi Turunan

h tersebut bernilai sangat kecil mendekati nol. Selain itu, secara fisis, nilai turunan suatu fungsi pada suatu titik menyatakan gradien atau kemiringan suatu fungsi pada titik tersebut.

3.7. TKINTER GRAPHICAL USER INTERFACE

Dalam merancang antarmuka (*interface*) dari simulator yang dirancang, digunakan library tkinter dari bahasa Python 3.7. Berikut ini terdapat *mockup* GUI rancangan kelompok:

The mockup GUI consists of two main sections: "Rangkian RC" and "Rangkaian Differentiator".

Rangkian RC:

- Top section: "NAMA SIMULATOR" input field.
- Middle section: "Penjelasan cara melakukan simulasi" text area.
- Bottom section:
 - Two rows of five buttons each, labeled "Gambar rangkaian".
 - Below the buttons:
 - "Jenis Rangkaian" dropdown menu.
 - "Variabel" dropdown menu.
 - "Vi" dropdown menu.
 - "R1" dropdown menu.
 - "R2" dropdown menu.
 - "C" dropdown menu.
 - "Submit Data" button.

Rangkaian Differentiator:

- Top section: "Jenis Gelombang" dropdown menu.
- Middle section:
 - "Variabel" dropdown menu.
 - "A" dropdown menu.
 - "T" dropdown menu.
 - "R" dropdown menu.
 - "C" dropdown menu.
- "Submit Data" button.

Gambar 3.7 Mockup GUI Simulator

3.8. COMMAND LINE ARGUMENT

Command Line Argument pada bahasa C merupakan metode *parsing* data pada *command prompt* dan jika dilakukan, akan menjalankan file *.exe* dari program C yang telah dibuat.

Metode dipilih untuk digunakan karena dapat memudahkan dalam merancang program lintas bahasa pemrograman (dalam kasus ini bahasa Python untuk mengode *interface* dan bahasa C untuk mengode perhitungan). Dalam melakukan *command line argument*, terdapat beberapa aturan struktur pemanggilan yang harus disesuaikan pada program C, maupun pada penulisan argument pada *command prompt*.

Pada saat melakukan penulisan argument pada *command prompt*, terdapat struktur umum yang harus diperhatikan yaitu:

Struktur Penulisan Command Line Argument

```
>> filename arg1 arg2 arg3 arg4 ...
```

Contoh:

```
>> rangkaianrc 5 1000 2000 1e-6
```

Selain itu, program C harus dirancang sedemikian rupa sehingga pada fungsi main mempunyai dua buah parameter yaitu argc, dan *argv[], [4], seperti berikut:

Struktur Penulisan Fungsi Main pada C

```
int main(int argc, char *argv[])
```

Jika ditinjau pada struktur penulisan command line argument, terlihat bahwa pada contoh tersebut, program menerima lima buah argument yaitu: `rangkaianrc`, `5`, `1000`, `2000`, `1e-6`. Argumen pertama merupakan argumen yang menyatakan nama file *executable* program dan diassign ke variabel argv[0]. Argumen selanjutnya akan diassign ke dalam argv[1], argv[2], dan seterusnya. Satu hal yang perlu diperhatikan dalam melakukan *invoke command line argument* adalah semua argument bertipe data string, sehingga untuk melakukan perhitungan dalam program, harus ada *assignment* yang dilakukan dengan menggunakan fungsi `strtod()` yang tersedia pada library stdlib.h. Setelah file *exe* program dijalankan, semua program C yang digunakan pada simulator ini akan memberikan keluaran berupa file .csv yang unik untuk setiap program, file tersebut berisi waktu dan tegangan/arus pada node/branch yang telah ditentukan. File tersebut selanjutnya akan diplot dengan menggunakan bahasa Python.

3.9. PLOTTING GRAFIK DENGAN MATPLOTLIB

Seperti yang telah dijelaskan pada bagian 3.8, file .csv yang dihasilkan oleh program akan diplot dengan menggunakan bahasa Python yaitu dengan memanfaatkan library matplotlib. Grafik akan diplot dan akan ditampilkan pada *window* baru simulator.

4. FLOWCHART DAN DATA-FLOW DIAGRAM

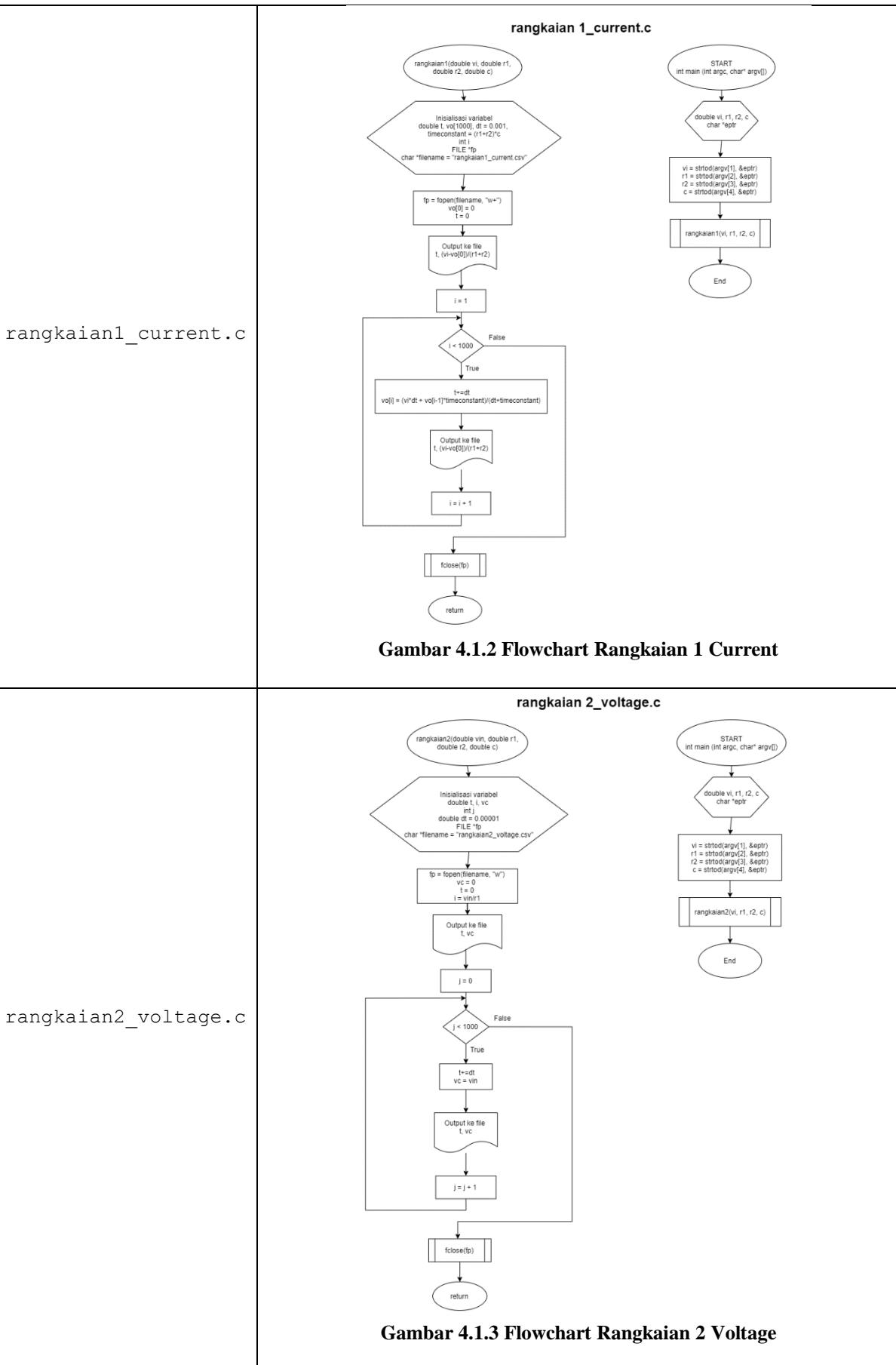
Pada bagian ini, terdapat flowchart algoritma program perhitungan menggunakan bahasa C. Secara umum, pengelompokan flowchart akan dibagi menjadi dua garis besar, yaitu untuk program perhitungan rangkaian RC (tipe 1-5) dan program perhitungan rangkaian

differentiator menggunakan op-amp. Jalannya data secara keseluruhan dari input user pada GUI hingga output plot terdapat pada *data flow diagram*.

Tabel 4.1 Flowchart Program Perhitungan Rangkaian RC

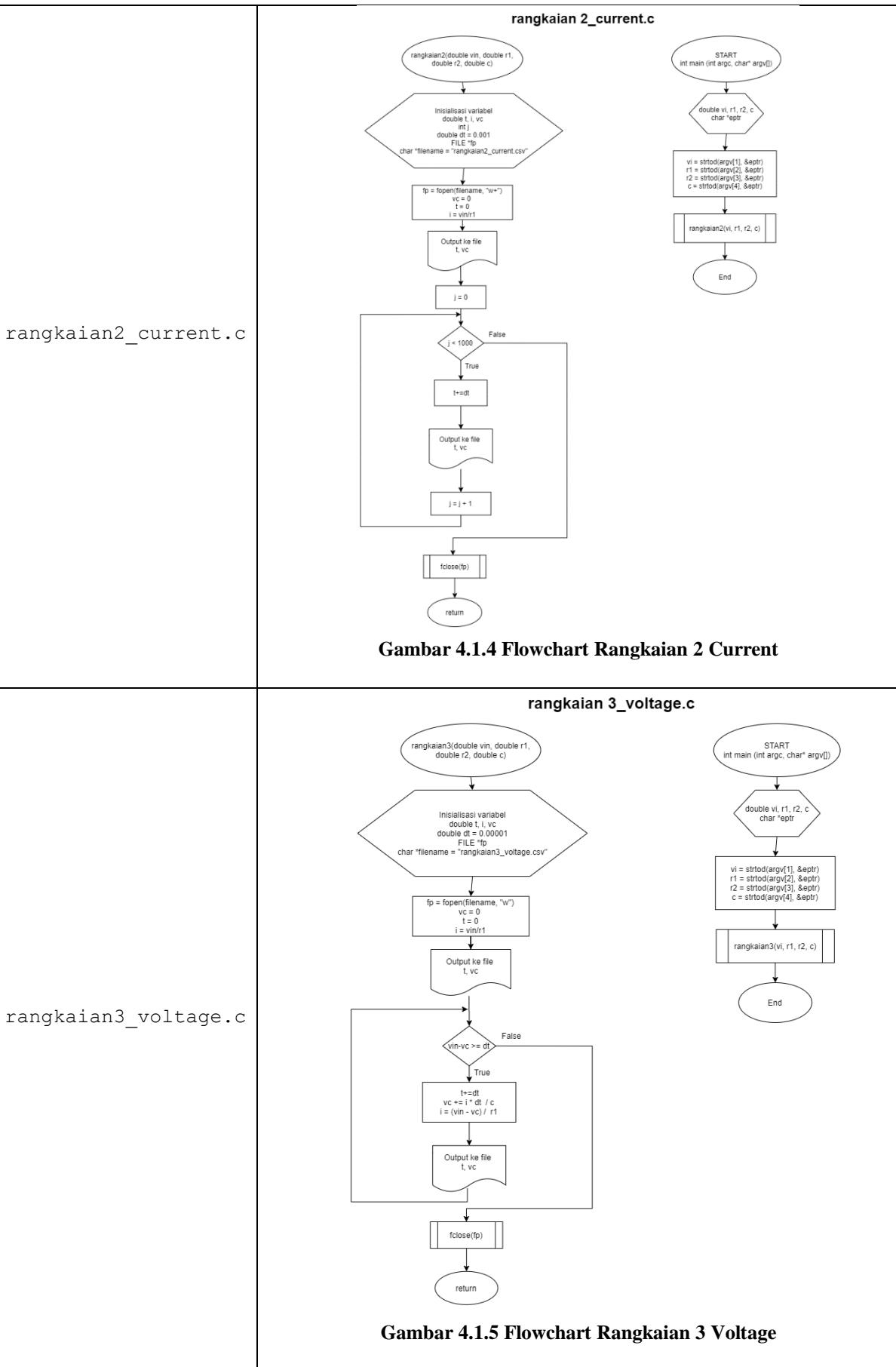
Keterangan	Flowchart
rangkaian1_voltage.c	<pre> graph TD START([START int main (int argc, char* argv[])]) --> INI{Inisialisasi variabel double t, vo[1000], dt = 0.001, timeconstant = (r1+r2)*c int i FILE *fp char *filename = "rangkaian1_voltage.csv"} INI --> FILE{fp = fopen(filename, "w+"); vo[0] = 0 t = 0} FILE --> OUT1{Output ke file t, vo[0]} OUT1 --> I1{i = 1} I1 --> DECISION{i < 1000} DECISION -- True --> CALCULATE{t+=dt vo[i] = (vi*dt + vo[i-1]*timeconstant)/(dt+timeconstant)} CALCULATE --> OUT2{Output ke file t, vo[i]} OUT2 --> I2{i = i + 1} I2 --> DECISION DECISION -- False --> CLOSE{fclose(fp)} CLOSE --> RETURN{return} </pre>

Gambar 4.1.1 Flowchart Rangkaian 1 Voltage



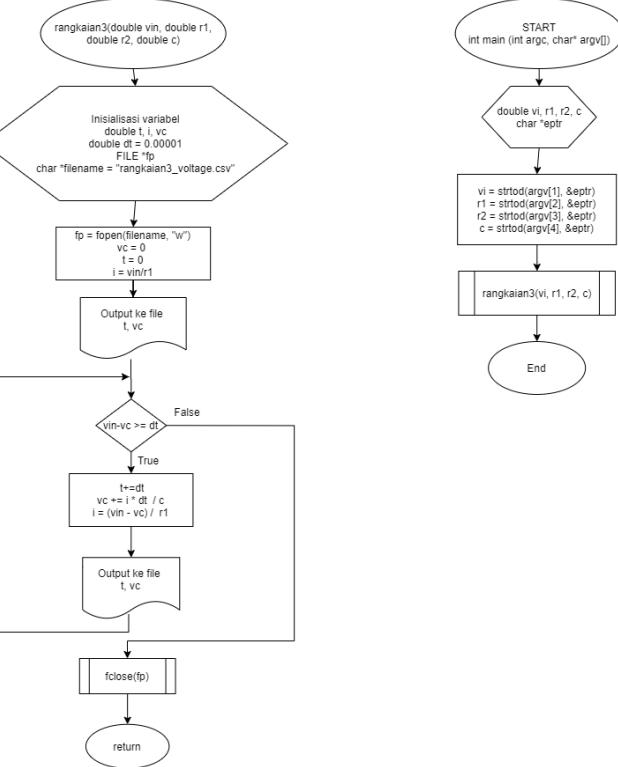
Gambar 4.1.2 Flowchart Rangkaian 1 Current

rangkaian2_voltage.c

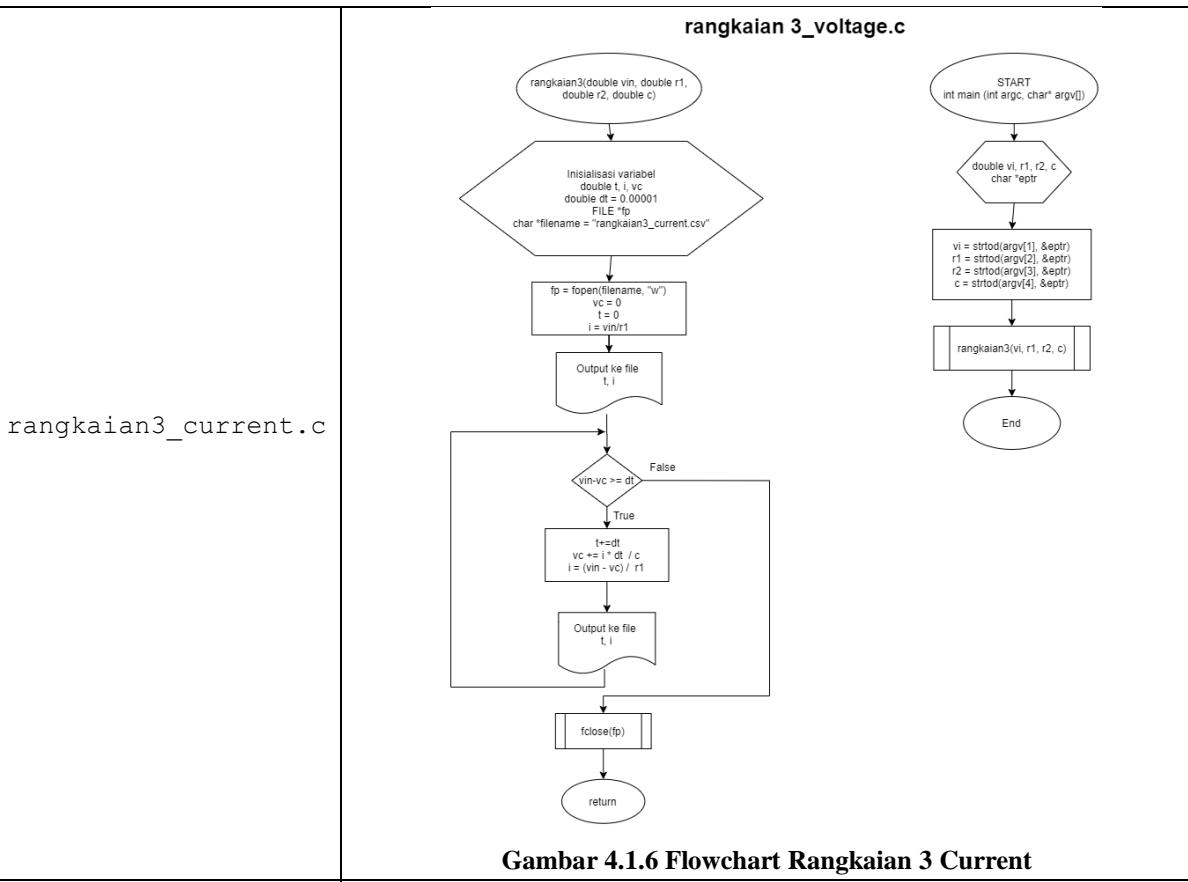


Gambar 4.1.4 Flowchart Rangkaian 2 Current

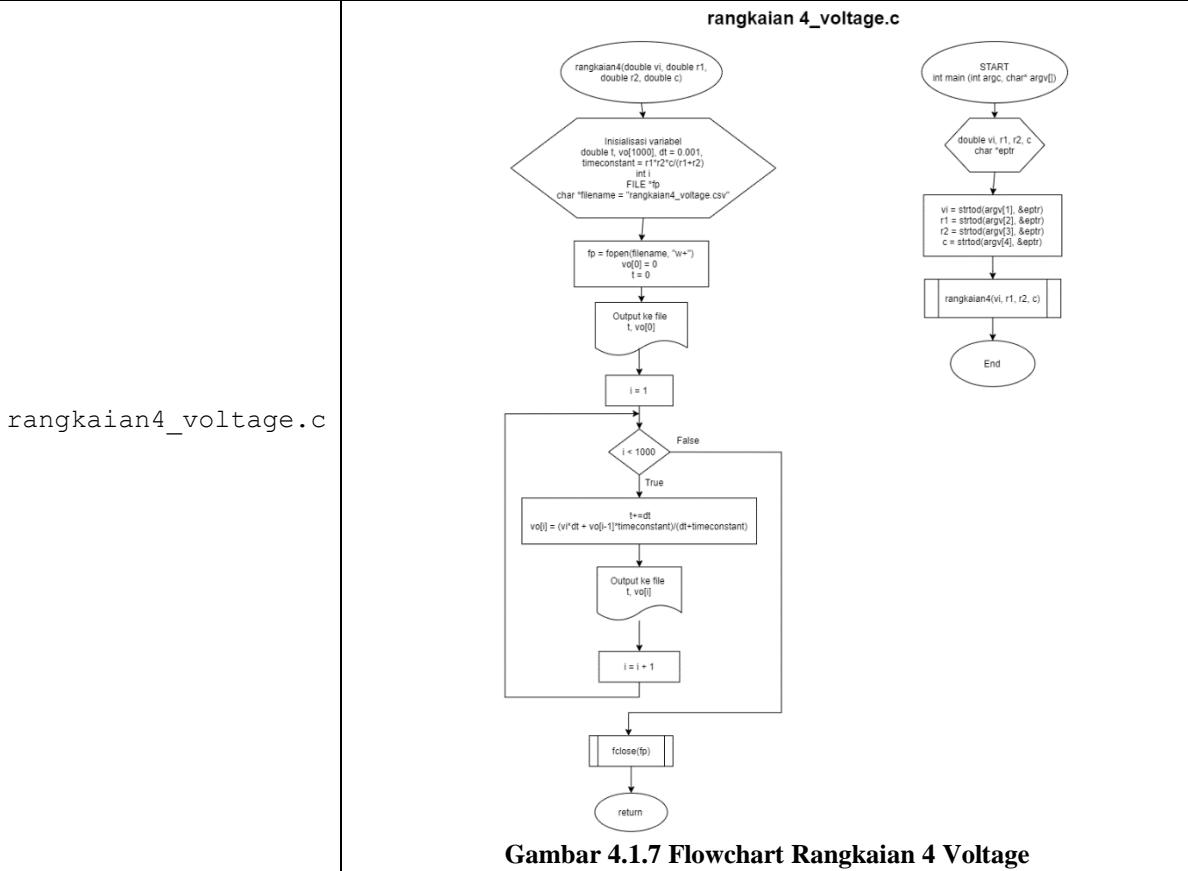
rangkaian3_voltage.c



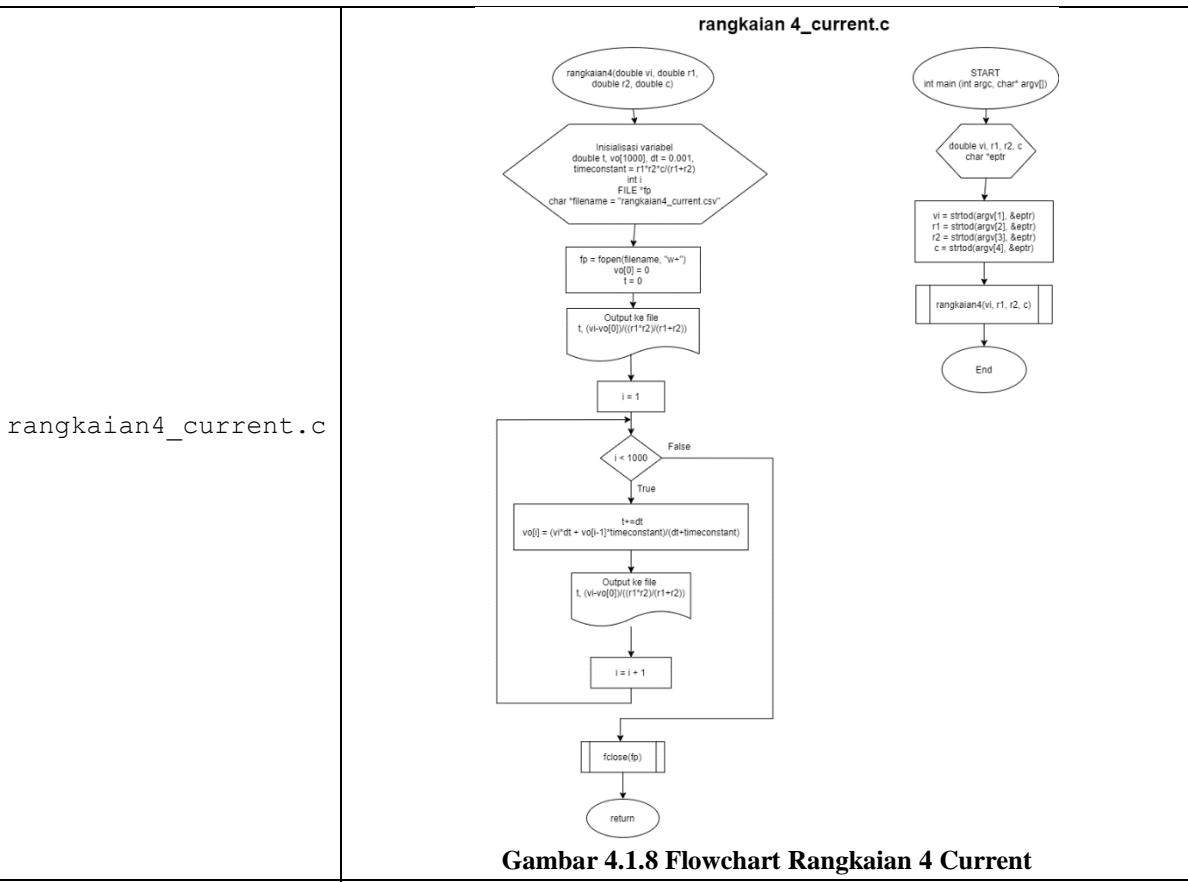
Gambar 4.1.5 Flowchart Rangkaian 3 Voltage



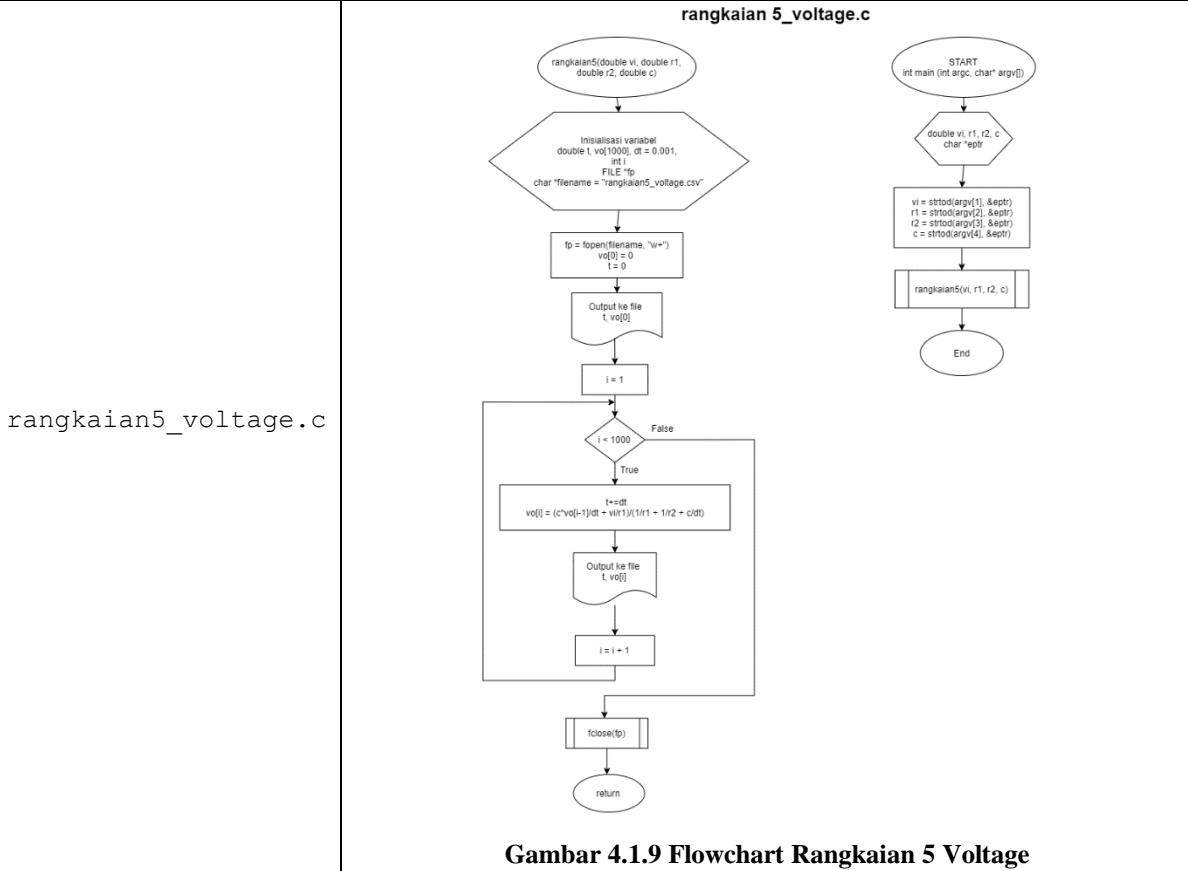
Gambar 4.1.6 Flowchart Rangkaian 3 Current



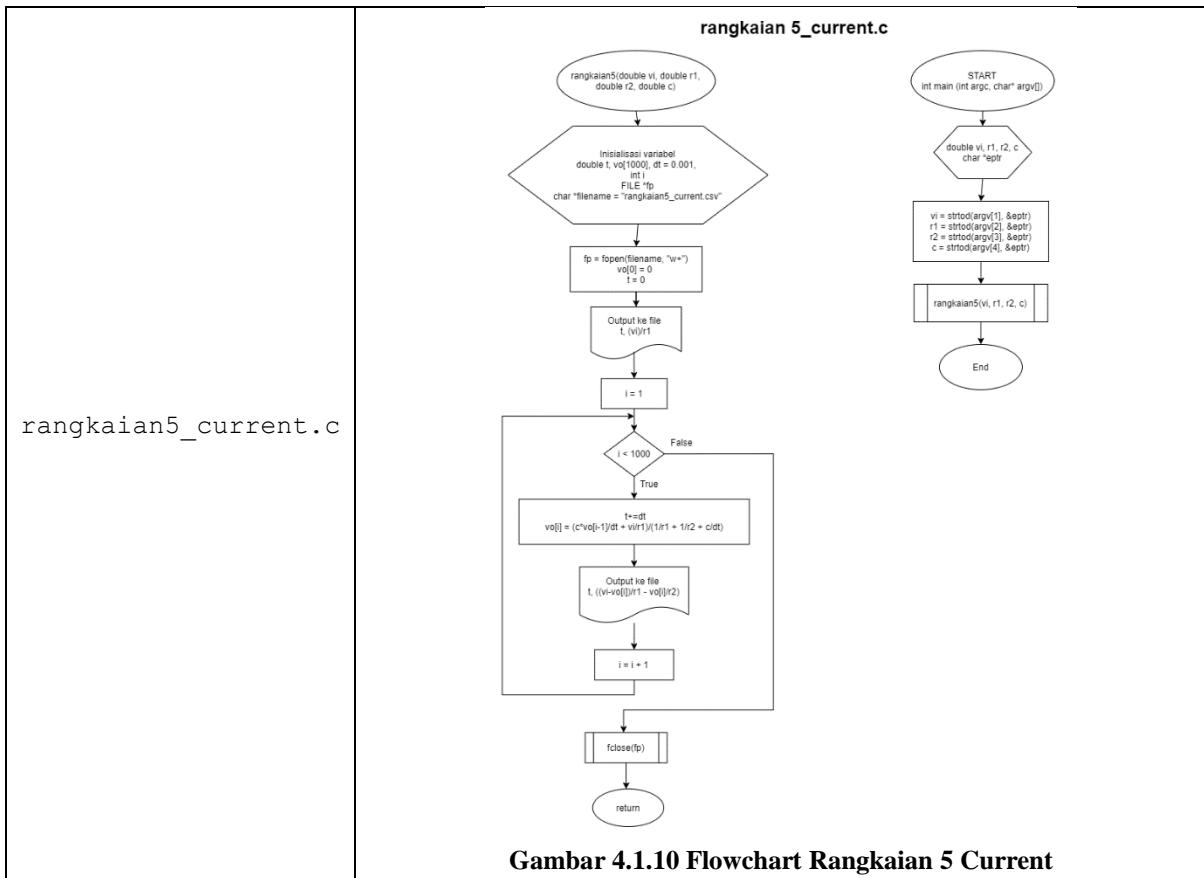
Gambar 4.1.7 Flowchart Rangkaian 4 Voltage



Gambar 4.1.8 Flowchart Rangkaian 4 Current



Gambar 4.1.9 Flowchart Rangkaian 5 Voltage



Gambar 4.1.10 Flowchart Rangkaian 5 Current

Tabel 4.2 Flowchart Program Perhitungan Rangkaian Differentiator

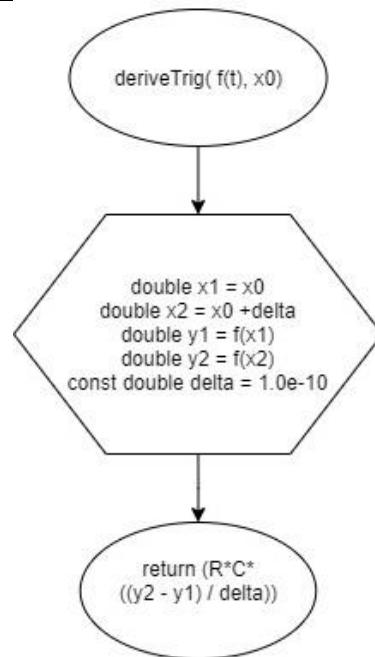
<i>Function/Procedure</i>	<i>Flowchart</i>
<pre>double derive(double (*f)(double t, double T, double A), double x0, double R, double C)</pre>	<p>The flowchart for the <code>derive</code> function starts with an input call to <code>derive(f(t,T,A), x0, R, C)</code>. This leads to a decision diamond where the following calculations are performed:</p> <pre> double x1 = x0 double x2 = x0 + delta double y1 = f(x1, T, A) double y2 = f(x2, T, A) const double delta = 1.0e-10 </pre> <p>After the calculations, the result is returned as:</p> <pre>return (R*C*((y2 - y1) / delta))</pre>

Gambar 4.2.1 Flowchart 1

```

double
deriveTrig(double
    (*f) (double t),
    double x0)

```

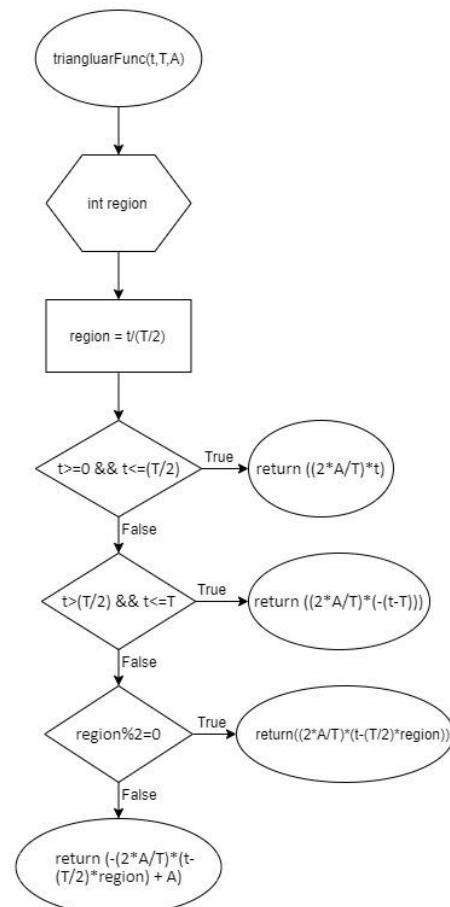


Gambar 4.2.2 Flowchart 2

```

double
triangluarFunc(double
    t, double T, double
    A)

```

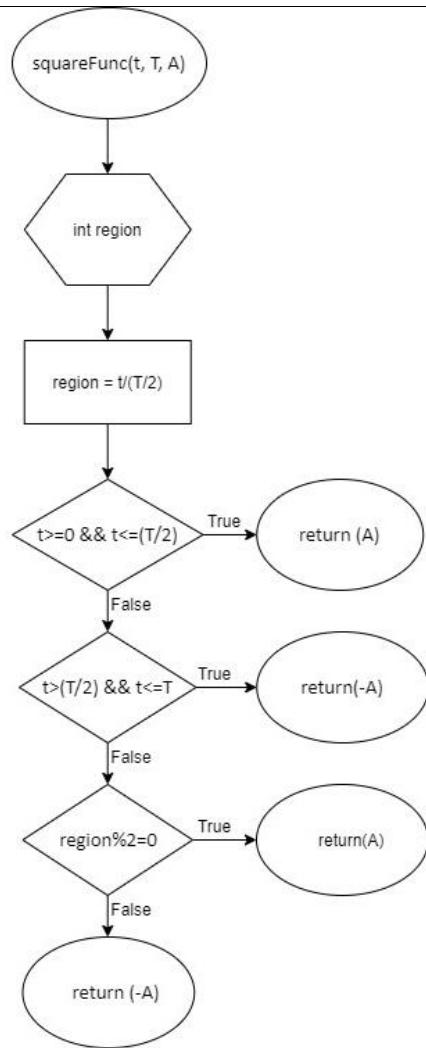


Gambar 4.2.3 Flowchart 3

```

double
squareFunc(double t,
double T, double A)

```

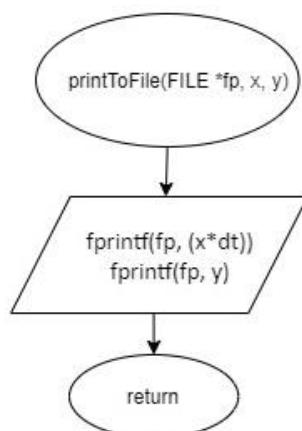


Gambar 4.2.4 Flowchart 4

```

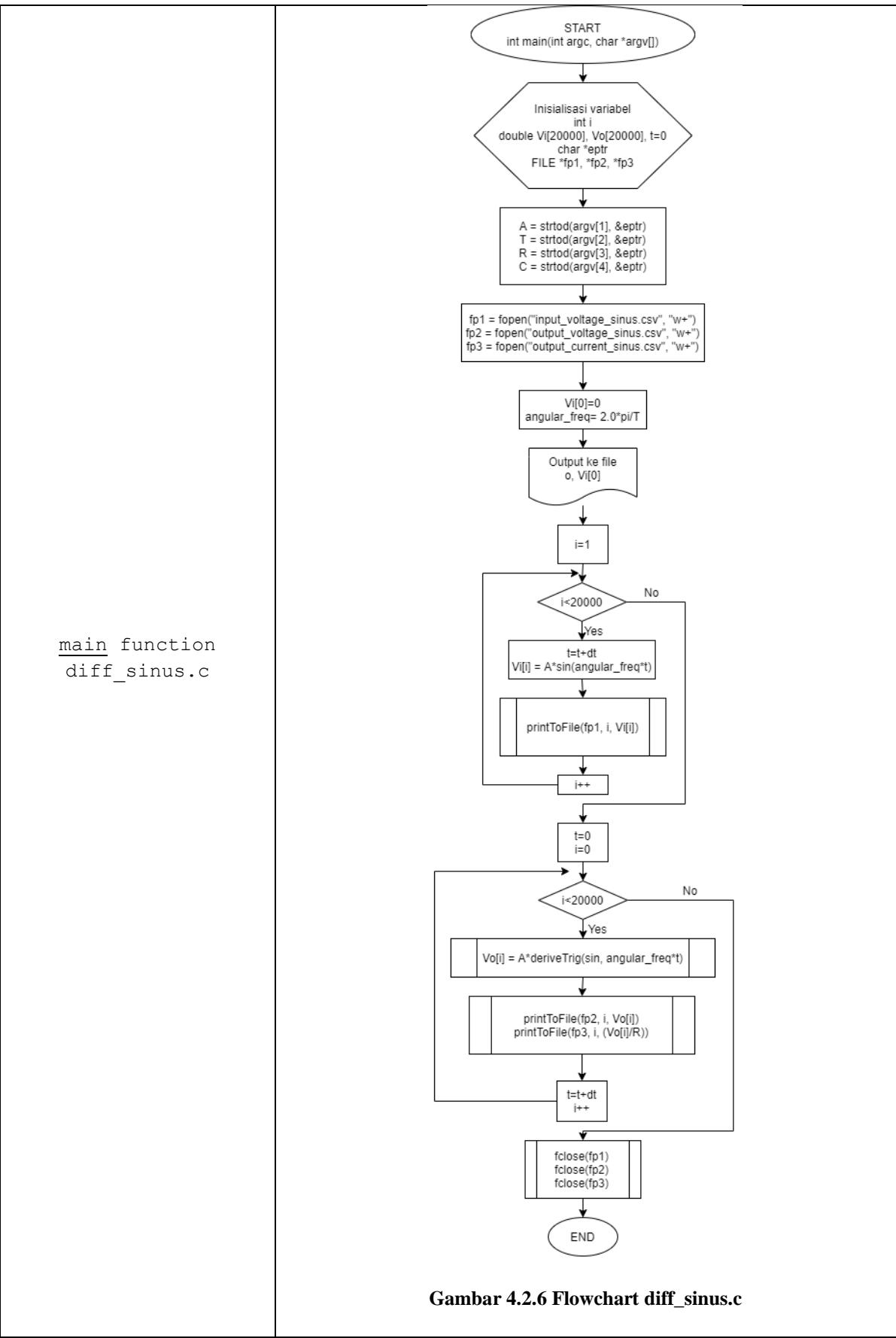
void printToFile(FILE *fp, int x, double y)

```



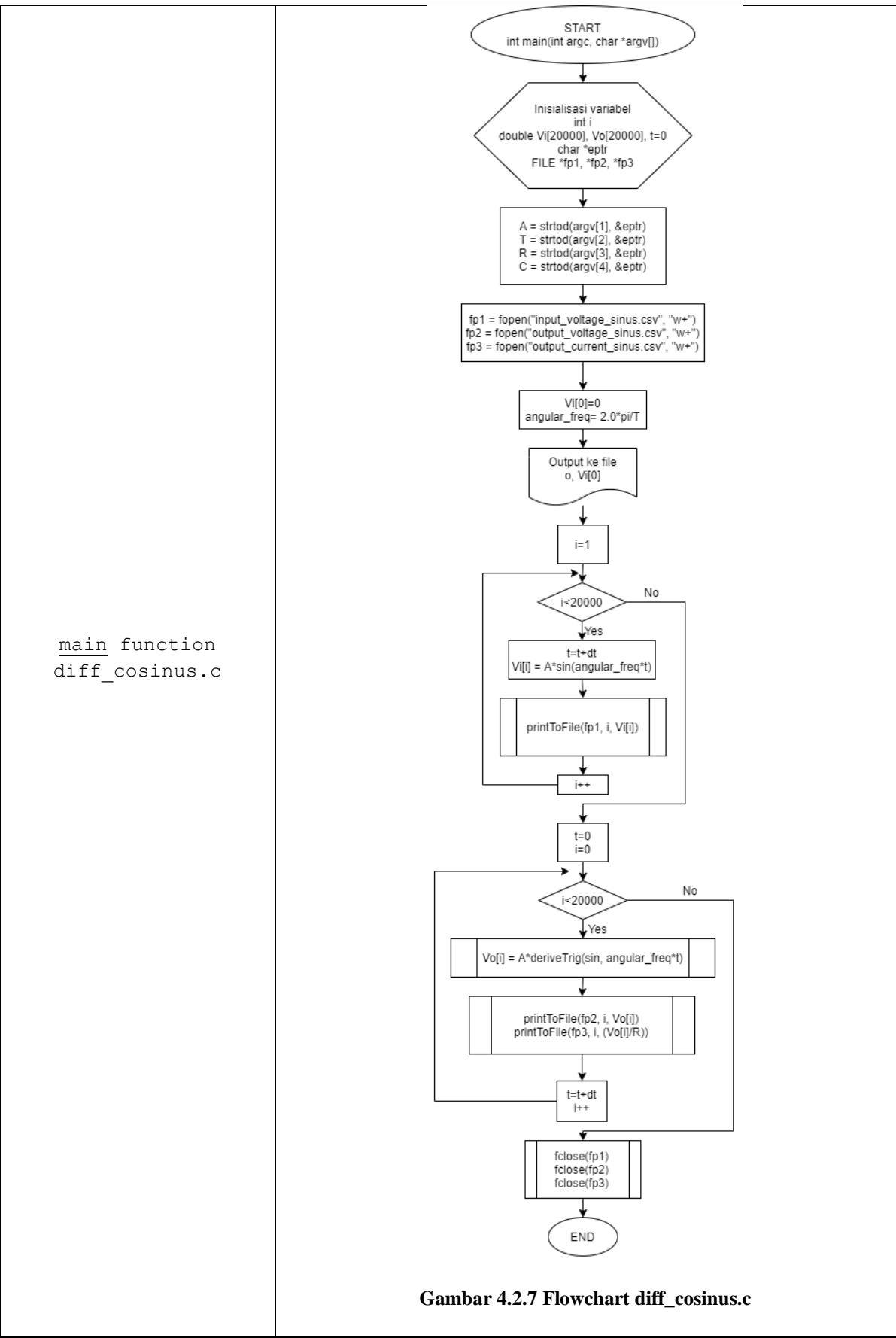
Gambar 4.2.5 Flowchart 5

main function
diff_sinus.c



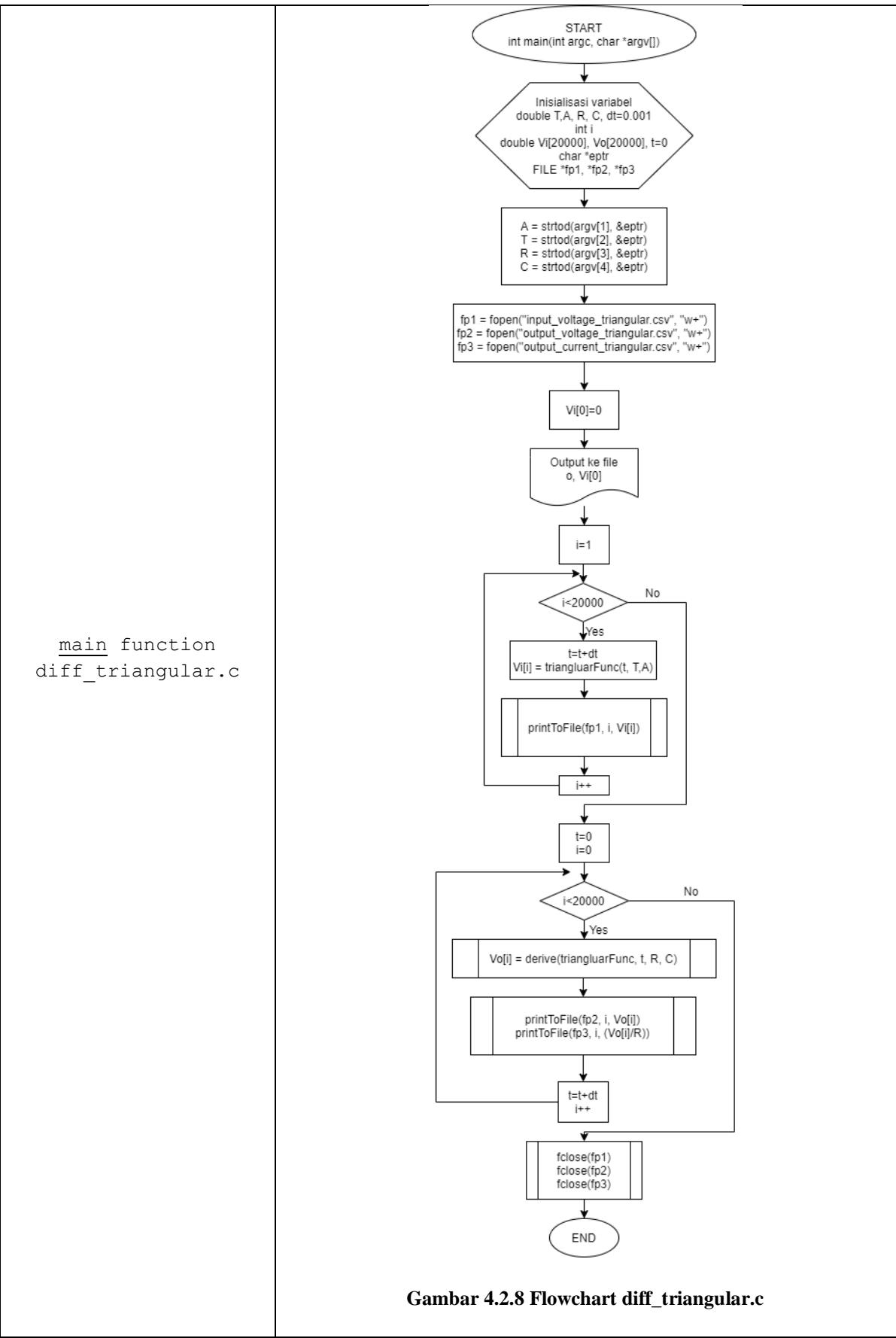
Gambar 4.2.6 Flowchart diff_sinus.c

main function
diff_cosinus.c



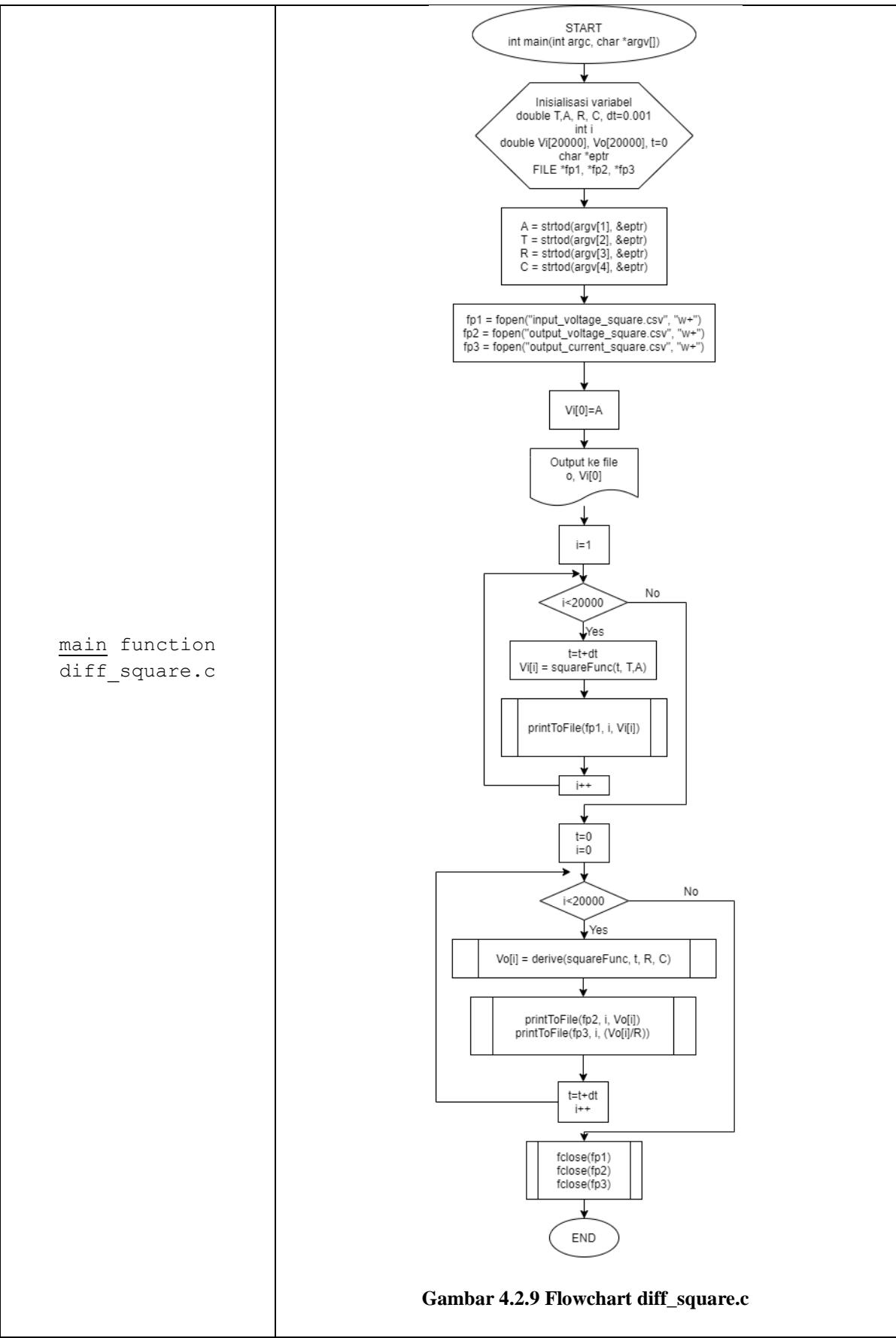
Gambar 4.2.7 Flowchart `diff_cosinus.c`

main function
diff_triangular.c



Gambar 4.2.8 Flowchart diff_triangular.c

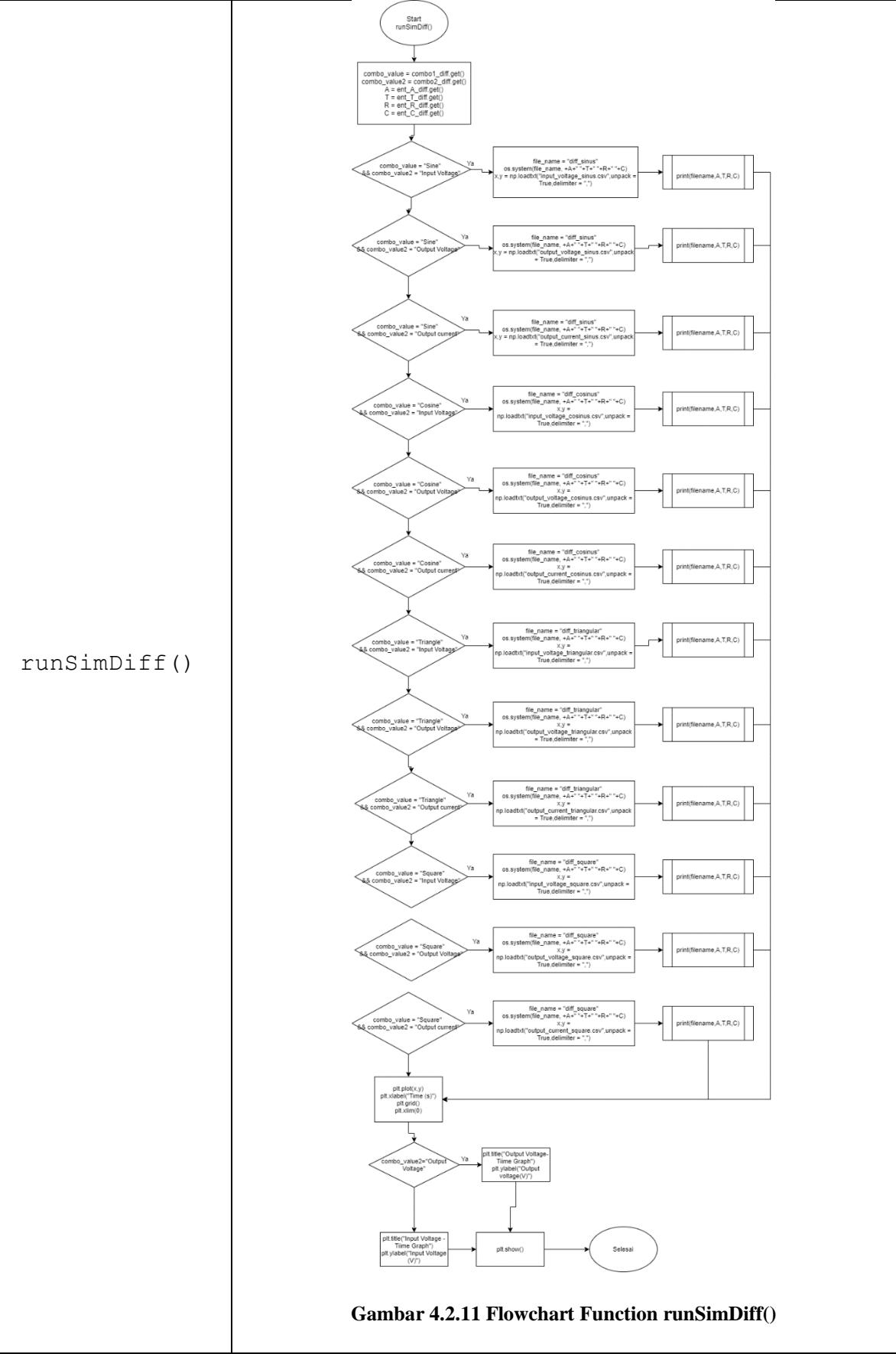
main function
diff_square.c



Gambar 4.2.9 Flowchart diff_square.c

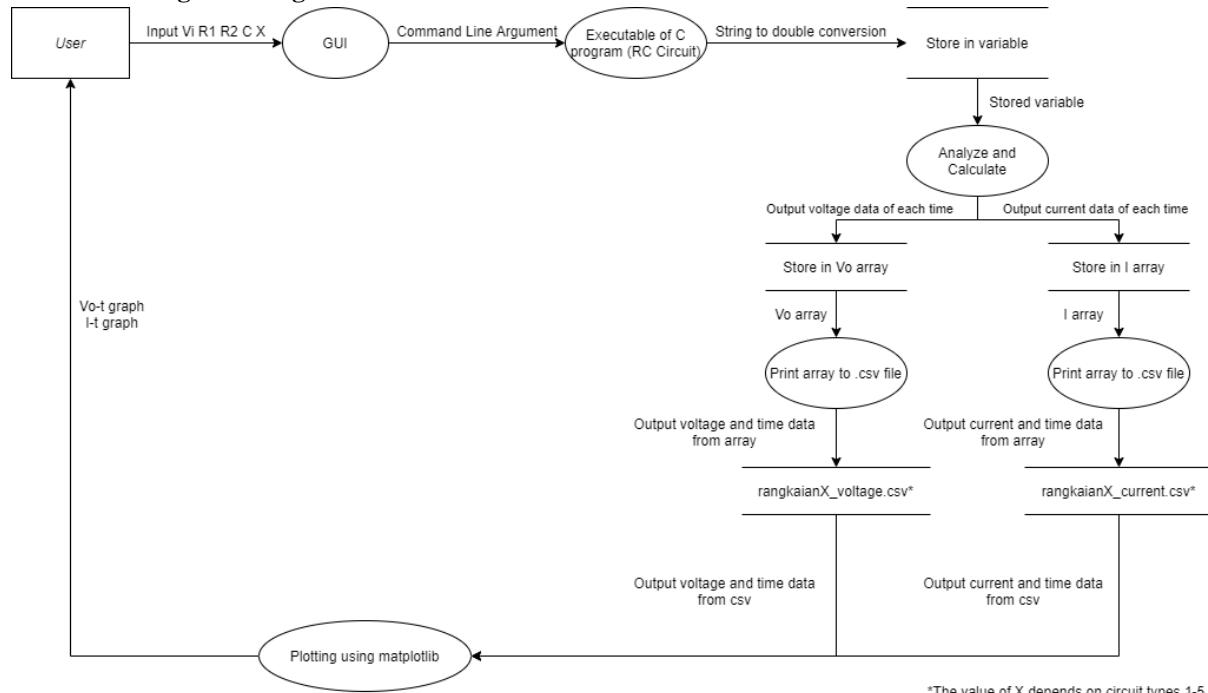
Function/Procedure	Flowchart
runSimRC()	<pre> graph TD Start((Start runSimRC)) --> Init[combo_value = combo1_RC.get() combo_value2 = combo2_RC.get() vi = emt_v1_RC.get() r1 = emt_r1_RC.get() r2 = emt_r2_RC.get() c = emt_c_RC.get()] Init --> Cond1{combo_value == "1st Circuit" && combo_value2 == "Voltage"} Cond1 -- Ya --> File1["file_name = \"rangkaian1_voltage\"\nosystem(file_name, +vi+\"*r1*\"+r2+\"*c\")\nx,y = np.loadtxt(\"rangkaian1_voltage.csv\", unpack = True, delimiter = \",\")"] File1 --> Print1["print(filename,vi,r1,r2,c)"] Cond1 -- Nb --> Cond2{combo_value == "1st Circuit" && combo_value2 == "Current"} Cond2 -- Ya --> File2["file_name = \"rangkaian1_current\"\nosystem(file_name, +vi+\"*r1*\"+r2+\"*c\")\nx,y = np.loadtxt(\"rangkaian1_current.csv\", unpack = True, delimiter = \",\")"] File2 --> Print2["print(filename,vi,r1,r2,c)"] Cond2 -- Nb --> Cond3{combo_value == "2nd Circuit" && combo_value2 == "Voltage"} Cond3 -- Ya --> File3["file_name = \"rangkaian2_voltage\"\nosystem(file_name, +vi+\"*r1*\"+r2+\"*c\")\nx,y = np.loadtxt(\"rangkaian2_voltage.csv\", unpack = True, delimiter = \",\")"] File3 --> Print3["print(filename,vi,r1,r2,c)"] Cond3 -- Nb --> Cond4{combo_value == "2nd Circuit" && combo_value2 == "Current"} Cond4 -- Ya --> File4["file_name = \"rangkaian2_current\"\nosystem(file_name, +vi+\"*r1*\"+r2+\"*c\")\nx,y = np.loadtxt(\"rangkaian2_current.csv\", unpack = True, delimiter = \",\")"] File4 --> Print4["print(filename,vi,r1,r2,c)"] Cond4 -- Nb --> Cond5{combo_value == "3rd Circuit" && combo_value2 == "Voltage"} Cond5 -- Ya --> File5["file_name = \"rangkaian3_voltage\"\nosystem(file_name, +vi+\"*r1*\"+r2+\"*c\")\nx,y = np.loadtxt(\"rangkaian3_voltage.csv\", unpack = True, delimiter = \",\")"] File5 --> Print5["print(filename,vi,r1,r2,c)"] Cond5 -- Nb --> Cond6{combo_value == "3rd Circuit" && combo_value2 == "Current"} Cond6 -- Ya --> File6["file_name = \"rangkaian3_current\"\nosystem(file_name, +vi+\"*r1*\"+r2+\"*c\")\nx,y = np.loadtxt(\"rangkaian3_current.csv\", unpack = True, delimiter = \",\")"] File6 --> Print6["print(filename,vi,r1,r2,c)"] Cond6 -- Nb --> Cond7{combo_value == "4th Circuit" && combo_value2 == "Voltage"} Cond7 -- Ya --> File7["file_name = \"rangkaian4_voltage\"\nosystem(file_name, +vi+\"*r1*\"+r2+\"*c\")\nx,y = np.loadtxt(\"rangkaian4_voltage.csv\", unpack = True, delimiter = \",\")"] File7 --> Print7["print(filename,vi,r1,r2,c)"] Cond7 -- Nb --> Cond8{combo_value == "4th Circuit" && combo_value2 == "Current"} Cond8 -- Ya --> File8["file_name = \"rangkaian4_current\"\nosystem(file_name, +vi+\"*r1*\"+r2+\"*c\")\nx,y = np.loadtxt(\"rangkaian4_current.csv\", unpack = True, delimiter = \",\")"] File8 --> Print8["print(filename,vi,r1,r2,c)"] Cond8 -- Nb --> Cond9{combo_value == "5th Circuit" && combo_value2 == "Voltage"} Cond9 -- Ya --> File9["file_name = \"rangkaian5_voltage\"\nosystem(file_name, +vi+\"*r1*\"+r2+\"*c\")\nx,y = np.loadtxt(\"rangkaian5_voltage.csv\", unpack = True, delimiter = \",\")"] File9 --> Print9["print(filename,vi,r1,r2,c)"] Cond9 -- Nb --> Cond10{combo_value == "5th Circuit" && combo_value2 == "Current"} Cond10 -- Ya --> File10["file_name = \"rangkaian5_current\"\nosystem(file_name, +vi+\"*r1*\"+r2+\"*c\")\nx,y = np.loadtxt(\"rangkaian5_current.csv\", unpack = True, delimiter = \",\")"] File10 --> Print10["print(filename,vi,r1,r2,c)"] Print10 --> Plot1["plt plot(y)\nplt xlabel('Time (s)')\nplt grid()\nplt xlim(0)\nplt ylim(0)"] Plot1 --> Cond11{combo_value2 == "Voltage"} Cond11 -- Ya --> Plot2["plt title('Output Voltage- Time Graph')\nplt ylabel('Output voltage(V)')"] Plot2 --> Plot3["plt title('Output Current- Time Graph')\nplt ylabel('Output Current(A)')"] Plot3 --> Plot4["plt show()"] Plot4 --> Selesai([Selesai]) </pre>

Gambar 4.2.10 Flowchart Function runSimRC()



Gambar 4.2.11 Flowchart Function runSimDiff()

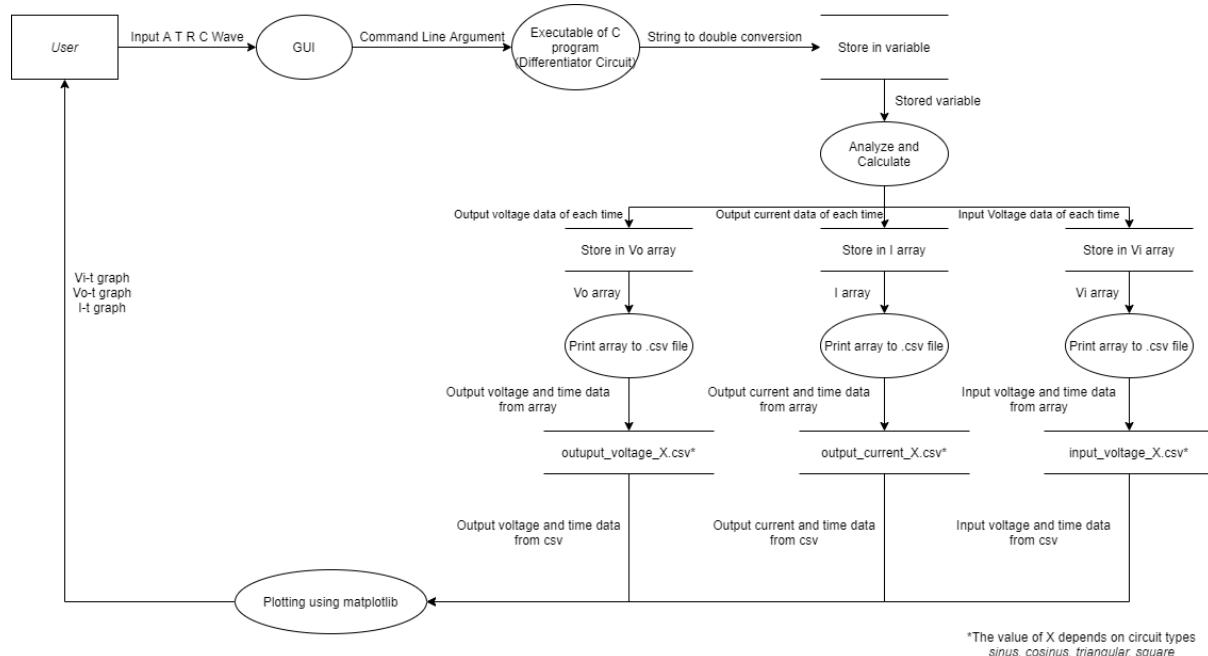
Data-Flow Diagram Rangkaian RC



*The value of X depends on circuit types 1-5

Gambar 4.3.1 Data Flow Diagram Rangkaian RC

Data-Flow Diagram Rangkaian Differentiator



*The value of X depends on circuit types
sinus, cosinus, triangular, square

Gambar 4.3.2 Data Flow Diagram Rangkaian Differentiator

5. TES UNIT DAN FUNGSIONAL

Pada bagian ini akan dilakukan *software testing* pada program yang telah dirancang. Akan dilakukan dua macam test, yaitu *unit testing* dan *functional testing*. *Unit testing* dilakukan untuk memastikan output yang dihasilkan oleh setiap blok program terkecil (fungsi dan prosedur) dapat sesuai dengan yang telah direncanakan. *Functional testing* merupakan

pengujian yang dilakukan pada gabungan beberapa unit/block (fungsi dan prosedur) dari program.

5.1 UNIT TEST

5.1.1 Function: derive

Desain Test Unit derive

```
#include<stdio.h>
#include<stdlib.h>

double t=0, dt=0.01, T, A, Vi[1000], Vo[1000], R=5000, C=0.2e-6;

double triangluarFunc(double t, double T, double A)           //triangle
{
    int region;
    region=t/(T/2);

    if(t>=0 && t<=(T/2))
        return ((2*A/T)*t);
    else if(t>(T/2) && t<=T)
        return ((2*A/T)*(-(t-T)));
    else
    {
        if(region%2==0)
            return((2*A/T)*(t-(T/2)*region));
        else if(region%2==1)
            return (-(2*A/T)*(t-(T/2)*region) + A);
    };
};

double derive(double (*f)(double t, double A), double x0,
double R, double C)
{
    const double delta = 1.0e-10; // or similar
    double x1 = x0;
    double x2 = x0 + delta;
    double y1 = f(x1,T,A);
    double y2 = f(x2,T,A);
    return R*C*((y2 - y1) / delta);
};

void printToFile(FILE *fp, int x, double y)
{
    fprintf(fp, "%f,", (x*dt));
    fprintf(fp, "%f\n", y);
};

int main(void)
{
    int i;
    FILE *fp, *fp2;
    fp = fopen("triangular_test.csv", "w+");
    fp2 = fopen("triangular_derive_test.csv", "w+");

    printf("Input T=");
    scanf("%lf", &T);
    printf("Input A=");
    scanf("%lf", &A);

    for(i=0;i<1000;i++)
    {
        Vi[i]=triangluarFunc(t, T, A);
        Vo[i]=derive(f, t, A);
        printToFile(fp, i, Vo[i]);
    }
}
```

```

        printToFile(fp, i, Vi[i]);
        t = t+dt;
    };

    t=0;
    for(i=0;i<1000;i++)
    {
        Vo[i] = derive(triangluarFunc, t, R, C);
        printToFile(fp2, i, Vo[i]);
        t=t+dt;
    };

    printf("End of program\n");
    return 0;
}

```

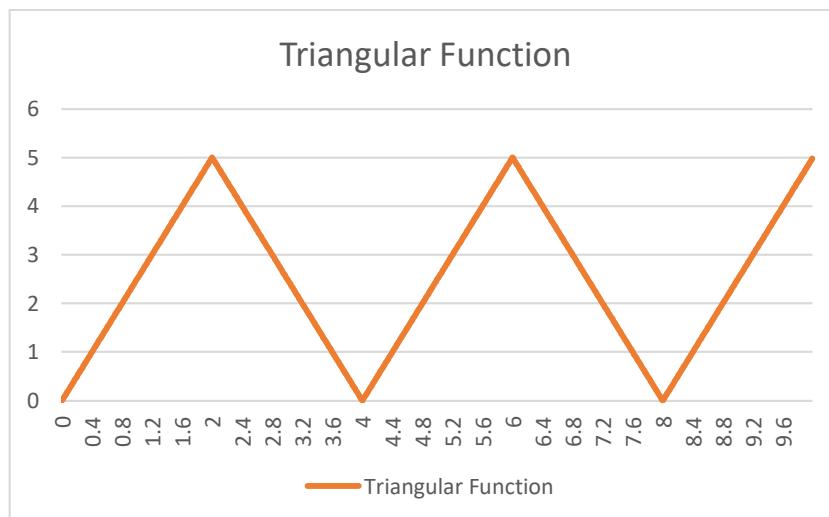
Kode program di atas merupakan kode program yang digunakan untuk melakukan unit test pada blok fungsi *derive*. Fungsi tersebut merupakan fungsi yang digunakan untuk mencari nilai turunan fungsi pada waktu tertentu. Untuk melakukan *unit test* pada blok ini, akan dilakukan iterasi sebanyak 1000 kali untuk mencari nilai tegangan input dan tegangan output yang merupakan turunan dari tegangan input, kemudian, data tegangan setiap waktu pada file tersebut akan diplot pada grafik untuk memverifikasi kebenaran dari *function* ini. Sebagai contoh, pada test case yang dilakukan, digunakan sinyal segitiga sebagai input dan didapat hasil output berupa sinyal kotak. Berdasarkan unit test yang telah dilakukan, dapat disimpulkan bahwa blok *derive* dapat beroperasi dengan baik.

```

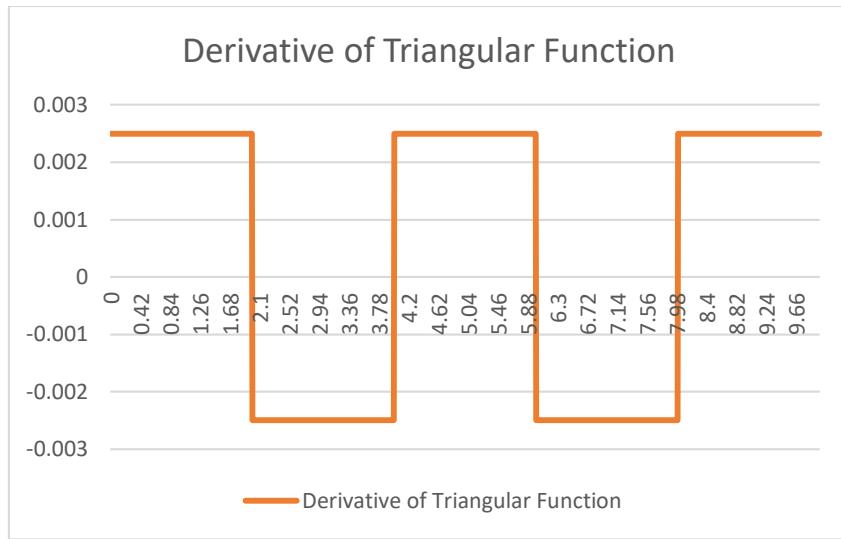
Input T=4
Input A=5
End of program

```

Gambar 5.1.1.1 Tampilan Command Prompt Unit Test *derivate*



Gambar 5.1.1.2 Tampilan Gelombang Input Berupa Gelombang Segitiga



Gambar 5.1.1.3 Tampilan Gelombang Output Berupa Gelombang Kotak

5.1.2 Function: `deriveTrig`

Desain Test Unit `deriveTrig`

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define pi 3.14159265359

double t=0, dt=0.01, T, A, Vi[1000], Vo[1000], R=5000, C=0.2e-6,
angular_freq;

double deriveTrig(double (*f)(double t), double x0)
{
    const double delta = 1.0e-10;
    double x1 = x0;
    double x2 = x0 +delta;
    double y1 = f(x1);
    double y2 = f(x2);
    return R*C*((y2 - y1) / delta);
};

void printToFile(FILE *fp, int x, double y)
{
    fprintf(fp, "%f,", (x*dt));
    fprintf(fp, "%f\n", y);
};

int main(void)
{
    int i;

    FILE *fp, *fp2;
    fp = fopen("sinusoidal.csv", "w+");
    fp2 = fopen("sinusoidal_derive_test.csv", "w+");

    printf("Input T=");
    scanf("%lf", &T);
    printf("Input A=");
    scanf("%lf", &A);
```

```

angular_freq = 2*pi/T;

for(i=0;i<1000;i++)
{
    Vi[i]=sin(angular_freq*t);
    printToFile(fp, i, Vi[i]);
    t = t+dt;
}

t=0Kode di atas
for(i=0;i<1000;i++)
{
    Vo[i] = deriveTrig(sin, t);
    printToFile(fp2, i, Vo[i]);
    t=t+dt;
}

printf("End of program\n");
return 0;
}

```

Kode di atas merupakan kode yang digunakan untuk melakukan *unit test* pada blok fungsi *deriveTrig*. Fungsi ini merupakan fungsi yang digunakan untuk mencari nilai turunan dari fungsi trigonometri. Fungsi trigonometri yang dapat digunakan untuk menjadi sinyal tegangan input pada simulator ini adalah fungsi sinus dan cosinus. Untuk melakukan *unit test* pada blok ini, akan dilakukan iterasi sebanyak 1000 kali untuk mencari nilai tegangan input dan tegangan output yang merupakan turunan dari tegangan input. Hasil yang didapat dimunculkan pada file eksternal .csv dan kemudian diplot sehingga dapat diverifikasi kebenaran dari hasil yang didapat. Pada gambar di bawah ini terdapat contoh test case dari unit test ini, blok tersebut diberi input berupa sinyal sinus dan output blok tersebut didapati berupa gelombang cosinus. Berdasarkan unit test yang telah dilakukan, dapat disimpulkan bahwa block *deriveTrig* dapat beroperasi dengan baik.

```

Input T=4
Input A=5
End of program

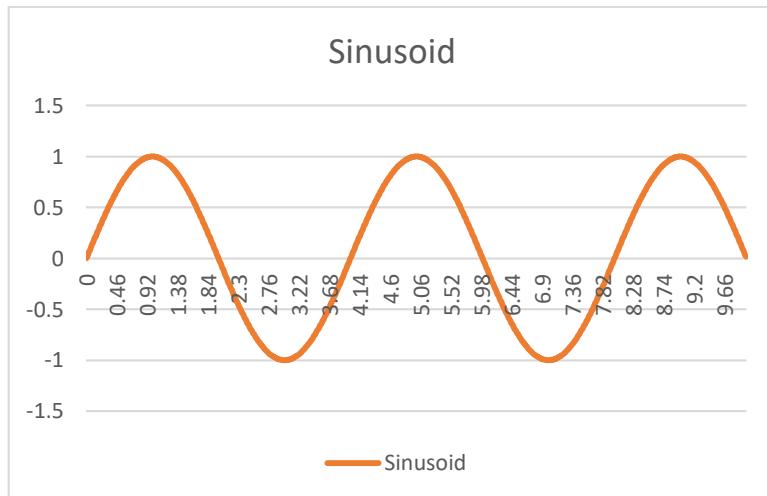
Process returned 0 (0x0)  execution time : 0.614 s
Press any key to continue.

```

Gambar 5.1.2.1 Tampilan Command Prompt Unit Test *deriveTrig*

	A	B
1	0	0
2	0.01	0.015707
3	0.02	0.031411
4	0.03	0.047106
5	0.04	0.062791
6	0.05	0.078459
7	0.06	0.094108
8	0.07	0.109734
9	0.08	0.125333
10	0.09	0.140901

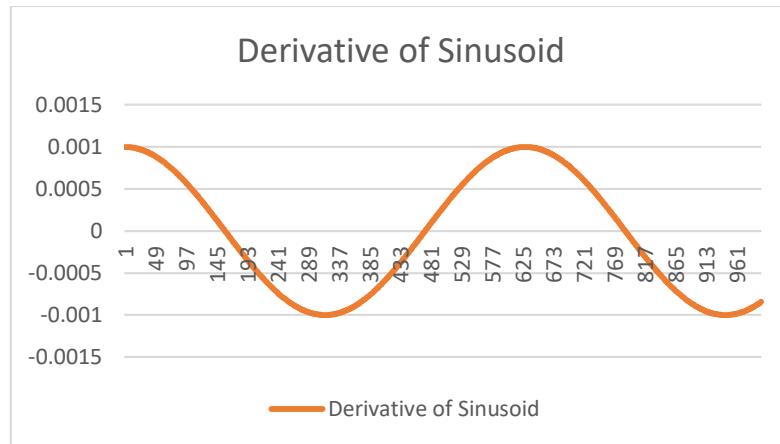
Gambar 5.1.2.2 Data Tegangan Input Hasil Unit Test Fungsi deriveTrig



Gambar 5.1.2.3 Tampilan Gelombang Input Berupa Gelombang Sinus

	A	B
1	0	0.001
2	0.01	0.001
3	0.02	0.001
4	0.03	0.001
5	0.04	0.000999
6	0.05	0.000999
7	0.06	0.000998
8	0.07	0.000998
9	0.08	0.000997
10	0.09	0.000996

Gambar 5.1.2.4 Data Tegangan Output Hasil Unit Test Fungsi deriveTrig



Gambar 5.1.2.4 Tampilan Gelombang Output Berupa Gelombang Cosinus

5.1.3 Function: triangularFunc

Desain Test Unit *triangularFunc*

```
#include<stdio.h>
#include<stdlib.h>

double t=0, dt=0.01, T, A, Vi[1000];

double triangluarFunc(double t, double T, double A)           //triangle
{
    int region;
    region=t/(T/2);

    if(t>=0 && t<=(T/2))
        return ((2*A/T)*t);
    else if(t>(T/2) && t<=T)
        return ((2*A/T)*(-(t-T)));
    else
    {
        if(region%2==0)
            return((2*A/T)*(t-(T/2)*region));
        else if(region%2==1)
            return (-(2*A/T)*(t-(T/2)*region) + A);
    };
};

void printToFile(FILE *fp, int x, double y)
{
    fprintf(fp, "%f,", (x*dt));
    fprintf(fp, "%f\n", y);
};

int main(void)
{
    int i;
    FILE *fp;
    fp = fopen("triangular_test.csv", "w+");

    printf("Input T="); scanf("%lf", &T);
    printf("Input A="); scanf("%lf", &A);

    for(i=0;i<1000;i++)

```

```

    {
        Vi[i]=triangularFunc(t, T, A);
        printf("t=% .2f | Vi=% .1f\n", t, Vi[i]);
        printToFile(fp, i, Vi[i]);
        t = t+dt;
    };

    printf("End of program\n");
    return 0;
}

```

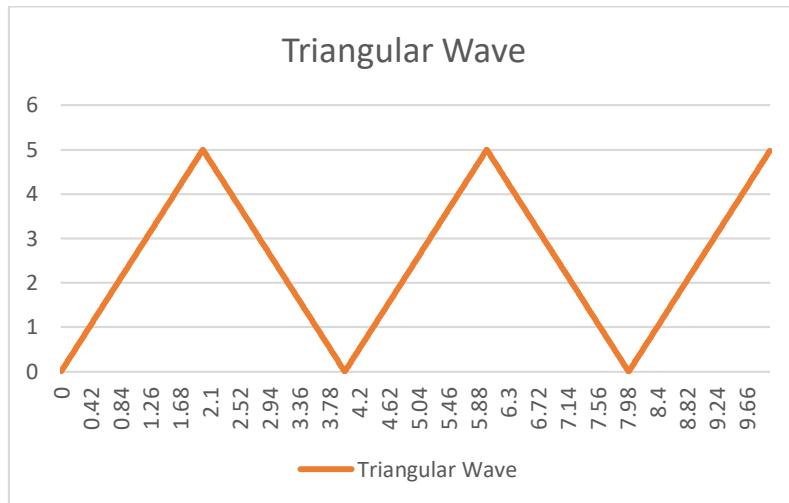
Kode program di atas merupakan kode program yang digunakan untuk melakukan *unit test* pada block fungsi *triangularFunc*. Terdapat fungsi berparameter waktu t, periode T, dan amplitudo gelombang A. Fungsi tersebut akan mengembalikan nilai sinyal segitiga setiap waktu. Untuk mengetes jalannya fungsi, akan dilakukan proses iterasi sebanyak 1000 kali untuk mendapatkan nilai tegangan setiap waktu dari sinyal tersebut dan hasil iterasi tersebut akan dimunculkan pada *command prompt* serta file eksternal .csv untuk kemudian dilakukan plotting grafik agar bentuk fungsi dapat diverifikasi.

Input T=4	
Input A=5	
t=0.00	Vi=0.0
t=0.01	Vi=0.0
t=0.02	Vi=0.1
t=0.03	Vi=0.1
t=0.04	Vi=0.1
t=0.05	Vi=0.1
t=0.06	Vi=0.2
t=0.07	Vi=0.2
t=0.08	Vi=0.2
t=0.09	Vi=0.2
t=0.10	Vi=0.2
t=0.11	Vi=0.3
t=0.12	Vi=0.3
t=0.13	Vi=0.3

Gambar 5.1.4.1 Tampilan Command Prompt Unit Test triangularFunc

	A	B
1	0	0
2	0.01	0.025
3	0.02	0.05
4	0.03	0.075
5	0.04	0.1
6	0.05	0.125
7	0.06	0.15
8	0.07	0.175
9	0.08	0.2
10	0.09	0.225

Gambar 5.1.4.2 Data Excel Hasil Unit Test triangularFunc



Gambar 5.1.4.3 Plot Grafik Fungsi Segitiga

5.1.4 Function: squareFunc

Desain Test Unit *squareFunc*

```
#include<stdio.h>
#include<stdlib.h>

double t=0, dt=0.01, T, A, Vi[1000];

double squareFunc(double t, double T, double A) //kotak
{
    int region;
    region=t/(T/2);

    if(t>=0 && t<=(T/2))
        return A;
    else if(t>(T/2) && t<=T)
        return -A;
    else
    {
        if(region%2==0)
            return A;
        else
            return -A;
    }
}
```

```

        return A;
    else if(region%2==1)
        return -A;
    };
};

void printToFile(FILE *fp, int x, double y)
{
    fprintf(fp, "%f, ", (x*dt));
    fprintf(fp, "%f\n", y);
};

int main(void)
{
    int i;
    FILE *fp;
    fp = fopen("square_test.csv", "w+");

    printf("Input T="); scanf("%lf", &T);
    printf("Input A="); scanf("%lf", &A);

    for(i=0;i<1000;i++)
    {
        Vi[i]=squareFunc(t, T, A);
        printf("t=% .2f | Vi=% .1f\n", t, Vi[i]);
        printToFile(fp, i, Vi[i]);
        t = t+dt;
    };

    printf("End of program\n");
    return 0;
}

```

Kode di atas merupakan kode yang digunakan untuk melakukan *unit test* pada fungsi *squareFunc*. Fungsi tersebut merupakan fungsi yang mempunyai tiga buah parameter, yaitu t sebagai waktu, periode T, dan amplitudo A. Fungsi tersebut akan memunculkan sinyal kotak dengan amplitudo A dan periode T dan akan mengembalikan nilai pada t tertentu. Untuk melakukan *unit test*, dilakukan assignment nilai pada array Vi untuk 1000 iterasi. Nilai tegangan setiap waktu akan ditampilkan pada *command prompt* dan file .csv sehingga mempermudah dilakukannya *debugging*. Berdasarkan *unit test* yang telah dilakukan, didapat bahwa block fungsi *squareFunc* telah beroperasi dengan baik.

```

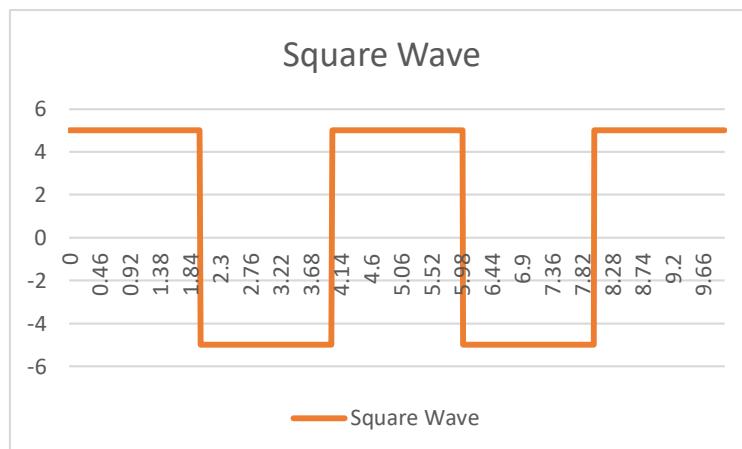
Input T=4
Input A=5
t=0.00 | Vi=5.0
t=0.01 | Vi=5.0
t=0.02 | Vi=5.0
t=0.03 | Vi=5.0
t=0.04 | Vi=5.0
t=0.05 | Vi=5.0
t=0.06 | Vi=5.0
t=0.07 | Vi=5.0
t=0.08 | Vi=5.0
t=0.09 | Vi=5.0
t=0.10 | Vi=5.0
t=0.11 | Vi=5.0
t=0.12 | Vi=5.0
t=0.13 | Vi=5.0
t=0.14 | Vi=5.0
t=0.15 | Vi=5.0

```

Gambar 5.1.4.1 Tampilan Command Prompt Unit Test squareFunc

	A	B
1	0	5
2	0.01	5
3	0.02	5
4	0.03	5
5	0.04	5
6	0.05	5
7	0.06	5
8	0.07	5
9	0.08	5
10	0.09	5

Gambar 5.1.4.2 Data Excel Hasil Unit Test squareFunc



Gambar 5.1.4.3 Square Wave Hasil Unit Test

5.1.5 Procedure: printToFile

Desain Test Unit Fungsi *printToFile*

```

#include<stdio.h>
#include<stdlib.h>

double dt=0.01;

void printToFile(FILE *fp, int x, double y)
{
    fprintf(fp, "%f,", (x*dt));
    fprintf(fp, "%f\n", y);
}

int main(void)
{
    FILE *fp;
    int Vo[10], i;

    fp = fopen("tes_file.csv", "w+");

    for(i=0;i<10;i++)
    {
        Vo[i]=rand();
        printToFile(fp, i, Vo[i]);
    };

    printf("End of Program\n");

    return 0;
}

```

Kode program di atas merupakan kode program yang digunakan untuk melakukan *unit test* dari fungsi *printToFile*. Fungsi tersebut mempunyai tiga buah parameter yaitu, pointer to file, x, dan y. Nilai x dan y merupakan nilai yang akan dicetak pada file .csv. Sebagai *final state*, akan didapat File yang berisi nilai x dan y. Untuk mengetes input program sebagai array, akan digunakan *dummy variable* Vo yang merupakan *array of integer* dengan nilai yang dimunculkan dengan fungsi *rand()* dan pada akhir program. Secara umum hasil unit test pada block ini menunjukkan bahwa fungsi *printToFile* dapat mengeksekusi perintah yang diberikan dengan baik. Berikut ini terdapat *testing* yang telah dilakukan:

```

End of Program

Process returned 0 (0x0)  execution time : 0.040 s
Press any key to continue.

```

Gambar 5.1.5.1 Run Unit Test *printToFile* pada Command Prompt

	A	B
1	0	41
2	0.01	18467
3	0.02	6334
4	0.03	26500
5	0.04	19169
6	0.05	15724
7	0.06	11478
8	0.07	29358
9	0.08	26962
10	0.09	24464

Gambar 5.1.5.2 Excel File Unit Test

5.1.6 Function: rangkaian1 (voltage)

Pada bagian 5.1.6 dan seterusnya, akan dilakukan *unit test* pada program perhitungan tegangan dan arus output pada rangkaian RC. Secara umum, unit test yang akan dilakukan adalah untuk mengecek apakah fungsi yang berkaitan dapat memenuhi spesifikasi input dan output yang telah dirancang. Unit test akan dilakukan pada perhitungan tegangan output dan arus output pada rangkaian RC tipe 1-5.

Untuk melakukan unit test tersebut, dilakukan desain unit test berupa pemanggilan fungsi yang bersangkutan pada fungsi main. Untuk mengetahui apakah proses berjalan dengan baik, pada akhir fungsi main, akan terdapat sebuah perintah output yang akan mengeluarkan string “End of program” pada *command prompt* jika proses sebelumnya berjalan dengan baik. Selain itu, akan dicek juga apakah nilai tegangan terhadap waktu, serta arus terhadap waktu telah berhasil diprint ke file eksternal dengan ekstensi .csv.

Desain Test Unit Fungsi *rangkaian1* (voltage)

```
//RANGKAIAN 1 - VOLTAGE

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

void rangkaian1(double vi,double r1,double r2,double c)
{
    double t, vo[1000], dt=0.001, timeconstant=(r1+r2)*c;
    int i;
    FILE *fp;
    char *filename = "rangkaian1_voltage.csv";
    fp = fopen(filename, "w+");
    vo[0]=0;
    t=0;
    fprintf(fp, "%f,", t);
    fprintf(fp, "%f\n", vo[0]);
    for(i=1;i<1000;i++)
    {
        t+=dt;
        vo[i] = (vi*dt + vo[i-1]*timeconstant)/(dt+timeconstant);
        fprintf(fp, "%f,", t);
        fprintf(fp, "%f\n", vo[i]);
    }
}
```

```

        }
        fclose(fp);
    }

int main(int argc,char* argv[])
{
    double vi, r1, r2, c;
    char *eptr;

    rangkaian1(5, 1000, 1000, 0.0001);
    printf("End of program.");

    return 0;
}

```

```

End of program.
Process returned 0 (0x0)  execution time : 0.033 s
Press any key to continue.

```

Gambar 5.1.6.1 Run Unit Test 1-1

	A	B
1	0	0
2	0.001	0.024876
3	0.002	0.049627
4	0.003	0.074256
5	0.004	0.098762
6	0.005	0.123147
7	0.006	0.14741
8	0.007	0.171552
9	0.008	0.195574

Gambar 5.1.6.2 File Eksternal Unit Test 1-1

5.1.7 Function: rangkaian1 (current)

Desain Test Unit Fungsi *rangkaian1* (current)

```

//RANGKAIAN 1 - CURRENT

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

void rangkaian1(double vi,double r1,double r2,double c)
{
    double t, vo[1000], dt=0.001, timeconstant=(r1+r2)*c;
    int i;
    FILE *fp;
    char *filename = "rangkaian1_current.csv";
    fp = fopen(filename, "w+");

```

```

vo[0]=0;
t=0;
fprintf(fp, "%f,", t);
fprintf(fp, "%f\n", (vi-vo[0])/(r1+r2));
for(i=1;i<1000;i++)
{
    t+=dt;
    vo[i] = (vi*dt + vo[i-1]*timeconstant)/(dt+timeconstant);
    fprintf(fp, "%f,", t);
    fprintf(fp, "%f\n", (vi-vo[i])/(r1+r2));
}
fclose(fp);
}

int main(int argc,char* argv[])
{
    double vi, r1, r2, c;

    rangkaian1(5, 1000, 1000, 0.0001);
    printf("End of Program");

    return 0;
}

```

End of Program
Process returned 0 (0x0) execution time : 0.031 s
Press any key to continue.

Gambar 5.1.7.1 Run Unit Test 1-2

	A	B
1	0	0.0025
2	0.001	0.002488
3	0.002	0.002475
4	0.003	0.002463
5	0.004	0.002451
6	0.005	0.002438
7	0.006	0.002426
8	0.007	0.002414
9	0.008	0.002402

Gambar 5.1.7.2 File Eksternal Unit Test 1-2

5.1.8 Function: rangkaian2 (voltage)

Desain Test Unit Fungsi *rangkaian2* (voltage)

```

//RANGKAIAN 2 - VOLTAGE

#include <stdio.h>
#include <stdlib.h>

```

```

void rangkaian2 (double vin,double r1,double r2,double c){
    double t,i,vc;
    int j;
    double dt = 0.00001;
    FILE *fp;
    char *filename = "rangkaian2_voltage.csv";
    fp = fopen(filename,"w");
    vc = 0;
    t = 0 ;
    i = vin/r1;
    fprintf(fp, "%lf,", t);
    fprintf(fp, "%lf\n", vc);
    for (j=0;j<1000;j++) {
        vc = vin;
        t += dt;
        fprintf(fp, "%lf,", t);
        fprintf(fp, "%lf\n", vc);
    }
    fclose(fp);
}

int main(int argc,char* argv[])
{
    double vi, r1, r2, c;
    char *eptr;

    rangkaian2(5, 1000, 1000, 0.0001);
    printf("End of program");

    return 0;
}

```

```

End of program
Process returned 0 (0x0)   execution time : 0.078 s
Press any key to continue.
-
```

Gambar 5.1.8.1 Run Unit Test 2-1

	A	B
1	0	0
2	0.00001	5
3	0.00002	5
4	0.00003	5
5	0.00004	5
6	0.00005	5
7	0.00006	5
8	0.00007	5
9	0.00008	5

Gambar 5.1.8.2 File Eksternal Unit Test 2-1

5.1.9 Function: rangkaian2 (current)

Desain Test Unit Fungsi *rangkaian2* (current)

```
//RANGKAIAN 2 - CURRENT

#include <stdio.h>
#include <stdlib.h>

void rangkaian2a (double vin,double r1,double r2,double c){
    double t,i,vc;
    int j;
    double dt = 0.001;
    FILE *fp;
    char *filename = "rangkaian2_current.csv";
    fp = fopen(filename,"w+");
    vc = 0;
    t = 0 ;
    i = vin/r1;
    fprintf(fp, "%f, ", t);
    fprintf(fp, "%f\n", vc);
    for (j=0;j<1000;j++) {
        t += dt;
        fprintf(fp, "%f, ", t);
        fprintf(fp, "%f\n", vc);
    }
    fclose(fp);
}

int main(int argc,char* argv[])
{
    double vi, r1, r2, c;

    rangkaian2a(5, 1000, 1000, 0.0001);
    printf("End of program");

    return 0;
}
```

```
End of program
Process returned 0 (0x0)  execution time : 0.029 s
Press any key to continue.
```

Gambar 5.1.9.1 Run Unit Test 2-2

	A	B
1	0	0
2	0.001	0
3	0.002	0
4	0.003	0
5	0.004	0
6	0.005	0
7	0.006	0
8	0.007	0
9	0.008	0

Gambar 5.1.9.2 File Eksternal Unit Test 2-2

5.1.10 Function: rangkaian3 (voltage)

Desain Test Unit Fungsi *rangkaian3* (voltage)

```
//RANGKAIAN 3 - VOLTAGE

#include <stdio.h>
#include <stdlib.h>

void rangkaian2b (double vin,double r1,double r2,double c){
    double i,vc,t;
    double dt = 0.00001;
    FILE *fp;
    char *filename = "rangkaian3_voltage.csv";
    fp = fopen(filename,"w");
    vc = 0;
    t = 0;
    i = vin/r1;
    fprintf(fp, "%lf,", t);
    fprintf(fp, "%lf\n", vc);
    while (vin - vc >= dt){
        t += dt;
        vc += i * dt / c;
        i = (vin - vc) / r1;
        fprintf(fp, "%lf,", t);
        fprintf(fp, "%lf\n", vc);
    }
    fclose(fp);
}

int main(int argc,char* argv[])
{
    double vi, r1, r2, c;

    rangkaian2b(5, 1000, 1000, 0.0001);
    printf("End of program");

    return 0;
}
```

```

End of program
Process returned 0 (0x0)    execution time : 0.206 s
Press any key to continue.
-
```

Gambar 5.1.10.1 Run Unit Test 3-1

	A	B
1	0	0
2	0.00001	0.0005
3	0.00002	0.001
4	0.00003	0.0015
5	0.00004	0.002
6	0.00005	0.0025
7	0.00006	0.002999
8	0.00007	0.003499
9	0.00008	0.003999
10	0.00009	0.004498

Gambar 5.1.10.2 File Eksternal Unit Test 3-1

5.1.11 Function: rangkaian3 (current)

Desain Test Unit Fungsi *rangkaian3* (current)

```

//RANGKAIAN 3 - CURRENT

#include <stdio.h>
#include <stdlib.h>

void rangkaian2b (double vin,double r1,double r2,double c){
    double i,vc,t;
    double dt = 0.00001;
    FILE *fp;
    char *filename = "rangkaian3_current.csv";
    fp = fopen(filename,"w");
    vc = 0;
    t = 0;
    i = vin/r1;
    fprintf(fp, "%lf,", t);
    fprintf(fp, "%lf\n", i);
    while (vin - vc >= dt){
        t += dt;
        vc += i * dt / c;
        i = (vin - vc) / r1;
        fprintf(fp, "%lf,", t);
        fprintf(fp, "%lf\n", i);
    }
    fclose(fp);
}

int main(int argc,char* argv[])

```

```

{
    double vi, r1, r2, c;

    rangkaian2b(5, 1000, 1000, 0.0001);
    printf("End of program");

    return 0;
}

```

```

End of program
Process returned 0 (0x0)   execution time : 0.210 s
Press any key to continue.

```

Gambar 5.1.11.1 Run Unit Test 3-2

	A	B
1	0	0.005
2	0.00001	0.004999
3	0.00002	0.004999
4	0.00003	0.004999
5	0.00004	0.004998
6	0.00005	0.004998
7	0.00006	0.004997
8	0.00007	0.004997
9	0.00008	0.004996

Gambar 5.1.11.2 File Eksternal Unit Test 3-2

5.1.12 Function: rangkaian4 (voltage)

Desain Test Unit Fungsi *rangkaian4* (voltage)

```

//RANGKAIAN 4 - VOLTAGE

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

void rangkaian4(double vi,double r1,double r2,double c)
{
    double t, vo[1000], dt=0.001, timeconstant=r1*r2*c/(r1+r2);
    int i;
    FILE *fp;
    char *filename = "rangkaian4_voltage.csv";
    fp = fopen(filename, "w+");
    vo[0]=0;
    t=0;
    fprintf(fp, "%f,", t);
    fprintf(fp, "%f\n", vo[0]);
    for(i=1;i<1000;i++)
    {
        t+=dt;

```

```

        vo[i] = (vi*dt + vo[i-1]*timeconstant)/(dt+timeconstant);
        fprintf(fp, "%f", t);
        fprintf(fp, "%f\n", vo[i]);
    }
}

int main(int argc,char* argv[])
{
    double vi, r1, r2, c;

    rangkaian4(5, 1000, 1000, 0.0001);
    printf("End of program");

    return 0;
}

```

```

End of program
Process returned 0 (0x0)   execution time : 0.032 s
Press any key to continue.

```

Gambar 5.1.12.1 Run Unit Test 4-1

	A	B
1	0	0
2	0.001	0.098039
3	0.002	0.194156
4	0.003	0.288388
5	0.004	0.380773
6	0.005	0.471346
7	0.006	0.560143
8	0.007	0.647199
9	0.008	0.732548

Gambar 5.1.13.2 File Eksternal Unit Test 4-1

5.1.13 Function: rangkaian4 (current)

Desain Test Unit Fungsi *rangkaian4* (current)

```

//RANGKAIAN 4 - CURRENT

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

void rangkaian4(double vi,double r1,double r2,double c)
{
    double t, vo[1000], dt=0.001, timeconstant=r1*r2*c/(r1+r2);
    int i;
    FILE *fp;
    char *filename = "rangkaian4_current.csv";
    fp = fopen(filename, "w+");

```

```

vo[0]=0;
t=0;
fprintf(fp, "%f,", t);
fprintf(fp, "%f\n", (vi-vo[0])/((r1*r2)/(r1+r2)));
for(i=1;i<1000;i++)
{
    t+=dt;
    vo[i] = (vi*dt + vo[i-1]*timeconstant)/(dt+timeconstant);
    fprintf(fp, "%f,", t);
    fprintf(fp, "%f\n", (vi-vo[i])/((r1*r2)/(r1+r2)));
}
}

int main(int argc,char* argv[])
{
    double vi, r1, r2, c;

    rangkaian4(5, 1000, 1000, 0.0001);
    printf("End of program");

    return 0;
}

```

```

End of program
Process returned 0 (0x0)   execution time : 0.026 s
Press any key to continue.

```

Gambar 5.1.13.1 Run Unit Test 4-2

	A	B
1	0	0.01
2	0.001	0.009804
3	0.002	0.009612
4	0.003	0.009423
5	0.004	0.009238
6	0.005	0.009057
7	0.006	0.00888
8	0.007	0.008706

Gambar 5.1.13.2 File Eksternal Unit Test 4-2

5.1.14 Function: rangkaian5 (voltage)

Desain Test Unit Fungsi *rangkaian5* (voltage)

```

//RANGKAIAN 5 - VOLTAGE

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

void rangkaian5(double vi,double r1,double r2,double c)
{

```

```

double t, vo[1000], dt=0.001;
int i;
FILE *fp;
char *filename = "rangkaian5_voltage.csv";
fp = fopen(filename, "w+");
vo[0]=0;
t=0;
fprintf(fp, "%f,", t);
fprintf(fp, "%f\n", vo[0]);
for(i=1;i<1000;i++)
{
    t+=dt;
    vo[i] = (c*vo[i-1]/dt + vi/r1)/(1/r1 + 1/r2 + c/dt);
    fprintf(fp, "%f,", t);
    fprintf(fp, "%f\n", vo[i]);
}
}

int main(int argc,char* argv[])
{
    double vi, r1, r2, c;

    rangkaian5(5, 1000, 1000, 0.0001);
    printf("End of program");

    return 0;
}

```

```

End of program
Process returned 0 (0x0)   execution time : 0.120 s
Press any key to continue.
-
```

Gambar 5.1.14.1 Run Unit Test 5-1

	A	B
1	0	0
2	0.001	0.04902
3	0.002	0.097078
4	0.003	0.144194
5	0.004	0.190386
6	0.005	0.235673
7	0.006	0.280072
8	0.007	0.3236
9	0.008	0.366274

Gambar 5.1.14.2 File Eksternal Unit Test 5-1

5.1.15 Function: rangkaian5 (current)

Desain Test Unit Fungsi *rangkaian5* (current)

```
//RANGKAIAN 5 - CURRENT
```

```

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

void rangkaian5(double vi,double r1,double r2,double c)
{
    double t, vo[1000], dt=0.001;
    int i;
    FILE *fp;
    char *filename = "rangkaian5_current.csv";
    fp = fopen(filename, "w+");
    vo[0]=0;
    t=0;
    fprintf(fp, "%f,", t);
    fprintf(fp, "%f\n", (vi)/r1);
    for(i=1;i<1000;i++)
    {
        t+=dt;
        vo[i] = (c*vo[i-1]/dt + vi/r1)/(1/r1 + 1/r2 + c/dt);
        fprintf(fp, "%f,", t);
        fprintf(fp, "%f\n", ((vi-vo[i])/r1 - vo[i]/r2));
    }
}

int main(int argc,char* argv[])
{
    double vi, r1, r2, c;

    rangkaian5(5, 1000, 1000, 0.0001);
    printf("End of program");

    return 0;
}

```

End of program
Process returned 0 (0x0) execution time : 0.030 s
Press any key to continue.

Gambar 5.1.15.1 Run Unit Test 5-2

	A	B
1	0	0.005
2	0.001	0.004902
3	0.002	0.004806
4	0.003	0.004712
5	0.004	0.004619
6	0.005	0.004529
7	0.006	0.00444
8	0.007	0.004353

Gambar 5.1.15.2 File Eksternal Unit Test 5-2

Berdasarkan *unit test* yang telah dilakukan pada semua fungsi perhitungan menggunakan bahasa C, diperoleh simpulan hasil unit test:

Tabel 5.1 Hasil Unit Test Program

No	Nama Fungsi	Run	CSV File	Ketepatan
1	derive	✓	✓	✓
2	deriveTrig	✓	✓	✓
3	triangularFunc	✓	✓	✓
4	squareFunc	✓	✓	✓
5	printToFile	✓	✓	✓
6	rangkaian1 (voltage)	✓	✓	✓
7	rangkaian1 (current)	✓	✓	✓
8	rangkaian2 (voltage)	✓	✓	✓
9	rangkaian2 (current)	✓	✓	✓
10	rangkaian3 (voltage)	✓	✓	✓
11	rangkaian3 (current)	✓	✓	✓
12	rangkaian4 (voltage)	✓	✓	✓
13	rangkaian4 (current)	✓	✓	✓
14	rangkaian5 (voltage)	✓	✓	✓
15	rangkaian5 (current)	✓	✓	✓

Berdasarkan unit test yang telah dilakukan, didapat simpulan bahwa semua unit program dapat berjalan dengan baik dan dapat memenuhi spesifikasi input dan output yang diharapkan.

5.2 FUNCTIONAL TEST

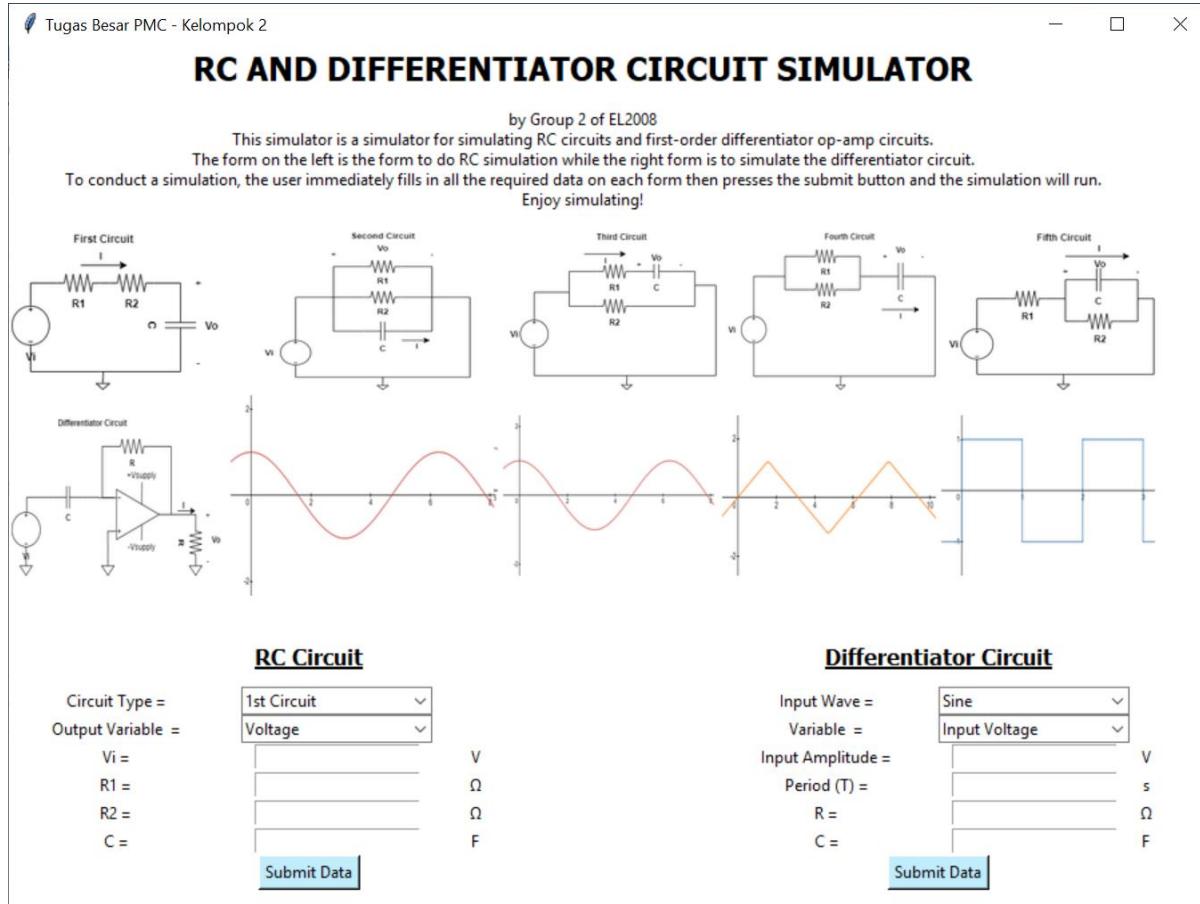
Setelah kelompok melakukan unit test pada setiap fungsi yang digunakan dalam program perhitungan yang telah dilakukan, akan dilakukan *functional test* untuk mengetahui integrasi antar blok program. Berdasarkan data-flow diagram yang telah dibuat pada bagian 4, terlihat bahwa perpindahan data program yang paling vital terjadi pada:

- *Graphical User Interface to command line argument*
- Run program dengan *command line argument*

- CSV to plot

5.2.1 GUI to Command Line Argument

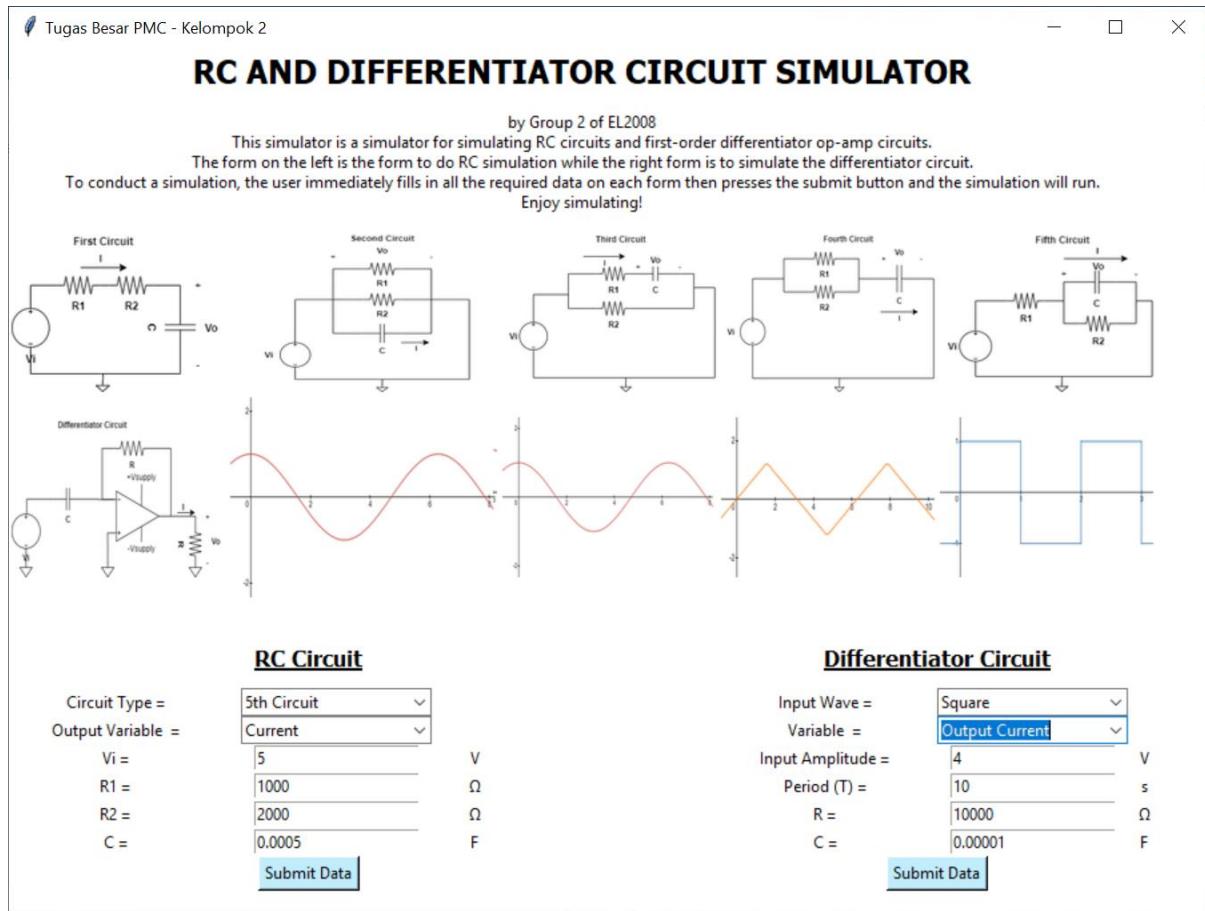
Langkah pertama ketika *user* menjalankan aplikasi simulator yang dirancang oleh kelompok, *user* akan diminta untuk menginput nilai variabel yang akan digunakan dalam perhitungan. Berikut ini terdapat *graphical user interface* yang dirancang oleh kelompok dengan menggunakan *tkinter* pada bahasa Python (penjelasan detail GUI terdapat pada bagian 8):



Gambar 5.2.1.1 Tampilan GUI Simulator

Pada gambar 5.2.1.1, terdapat tampilan GUI simulator yang telah dirancang oleh kelompok. GUI tersebut akan tampilan pada *window* baru. Pada bagian *functional testing*, akan difokuskan pada *form* untuk menginput data. Terdapat dua buah form, form sebelah kiri merupakan form untuk menginput data yang dibutuhkan untuk melakukan simulasi rangkaian RC, sedangkan form sebelah kanan merupakan form untuk melakukan simulasi rangkaian differentiator.

Berdasarkan *data-flow diagram* yang telah dibuat pada bagian 4, terlihat bahwa input GUI merupakan input user dan akan memberikan output data berupa *command line argument* yang akan meng-*invoke* jalannya file *executable* program C yang telah dibuat. Untuk mengecek keberjalan proses ini, kelompok akan melakukan *functional testing* dengan memberikan input pada kedua form dan mengecek apakah jika button ditekan, akan muncul *command line argument* pada *command prompt*. Berikut ini terdapat hasil *functional test* yang telah dilakukan:



Gambar 5.2.1.2 Tampilan Input GUI pada Functional Test

```
C:\Windows\System32\cmd.exe - python top_simulator.py
Microsoft Windows [Version 10.0.17763.1158]
(c) 2018 Microsoft Corporation. All rights reserved.

D:\MARTINUS\SEKOLAH\INSTITUT TEKNOLOGI BANDUNG\Semester 4\Pemecahan Masalah dengan C\Tugas Besar\Testing Python>python top_simulator.py
rangkaian1_voltage 5 1000 1000 0.0001
rangkaian1_current 5 1000 1000 0.0001
rangkaian2_voltage 5 10000 2000 0.0001
rangkaian2_current 5 10000 2000 0.0001
rangkaian3_voltage 5 10000 2000 0.0001
rangkaian3_current 5 10000 2000 0.0001
rangkaian4_voltage 5 10000 2000 0.0001
rangkaian4_current 5 10000 2000 0.0001
rangkaian5_voltage 5 1000 2000 0.0001
rangkaian5_current 5 1000 2000 0.0005
diff_sinus 4 5 10000 0.00001
diff_sinus 4 5 10000 0.00001
diff_sinus 4 5 10000 0.00001
diff_cosinus 4 5 10000 0.00001
diff_cosinus 4 5 10000 0.00001
diff_cosinus 4 5 10000 0.00001
diff_triangular 4 10 10000 0.00001
diff_triangular 4 10 10000 0.00001
diff_triangular 4 10 10000 0.00001
diff_square 4 10 10000 0.00001
diff_square 4 10 10000 0.00001
diff_square 4 10 10000 0.00001
```

Gambar 5.2.1.3 Tampilan Command Prompt Functional Test

Pada gambar 5.2.1.2 di atas, terdapat tampilan GUI ketika user menginput data yang dibutuhkan untuk melakukan simulasi, baik pada rangkaian RC maupun rangkaian differentiator. Pada form rangkaian differentiator, user akan menginput jenis rangkaian, variabel yang akan diplot, resistansi R1, R2, dan kapasitansi kapasitor C. Sedangkan, untuk rangkaian differentiator, user akan menginput jenis rangkaian berupa jenis gelombang input, variabel yang akan diamati, amplitudo gelombang, periode, resistansi R, dan kapasitansi C.

Berikut ini terdapat struktur penulisan *command line argument* pada simulasi rangkaian RC dan rangkaian differentiator:

Command Line Argument Rangkaian RC

```
>> filename_RC Vi R1 R2 C
```

Command Line Argument Rangkaian Differentiator

```
>> filename_diff A T R C
```

Jika ditinjau gambar 5.2.1.3, terdapat semua kemungkinan pilihan input pada simulator yang telah dirancang. Terlihat bahwa kaidah penulisan *command line argument* untuk rangkaian RC dan rangkaian differentiator terpenuhi dengan baik. Berdasarkan *functional test* yang telah dilakukan dapat disimpulkan bahwa GUI *to command line argument* dapat berfungsi dengan baik.

5.2.2 Run Program dengan *Command Line Argument*

Setelah data diinput oleh *user* dan berhasil untuk dinyatakan dalam sebagai *command line argument*, data berupa *command line argument* tersebut selanjutnya akan digunakan untuk meng-*invoke* file *executable* program C. Untuk melakukan *functional test* pada bagian ini, dapat dilakukan dengan mencoba untuk melakukan *testrun* pada file *executable* dari masing-masing program C. Karena unit test telah dilakukan dan didapat bahwa setiap blok program dapat berjalan dengan baik, keberhasilan *functional test* ini dapat dilihat bahwa jika *command line argument* di-*invoke* pada *command prompt*, akan muncul file baru yang merupakan file hasil *run* program C dengan parameter variabel yang telah diinput pada argument. Sebelum melakukan *testrun*, akan dibahas terlebih dahulu mengenai *assignment* variabel dari *command line argument* ke variabel dalam program C dengan kondisi program C telah ter-*compile*.

Jika ditinjau penulisan struktur command line argument untuk rangkaian RC di bawah ini:

Contoh Command Line Argument untuk Rangkaian RC

```
>> rangkaian1_voltage 5 1000 2000 1e-6
```

Pada *command line argument* di atas, terdapat argumen untuk memanggil rangkaian1_voltage.exe. Terdapat 5 buah argument yang dilempar pada program yaitu rangkaian1_voltage, 5, 1000, 2000, 1e-6. Perlu diperhatikan bahwa semua argumen bertipe data *string* sehingga untuk melakukan perhitungan, perlu dilakukan konversi dari *string* ke *double* dengan menggunakan fungsi *strtod()* yang terdapat pada library stdlib.h. Berikut ini contoh konversi dan *assignment* dari *command line argument* ke variabel dalam program:

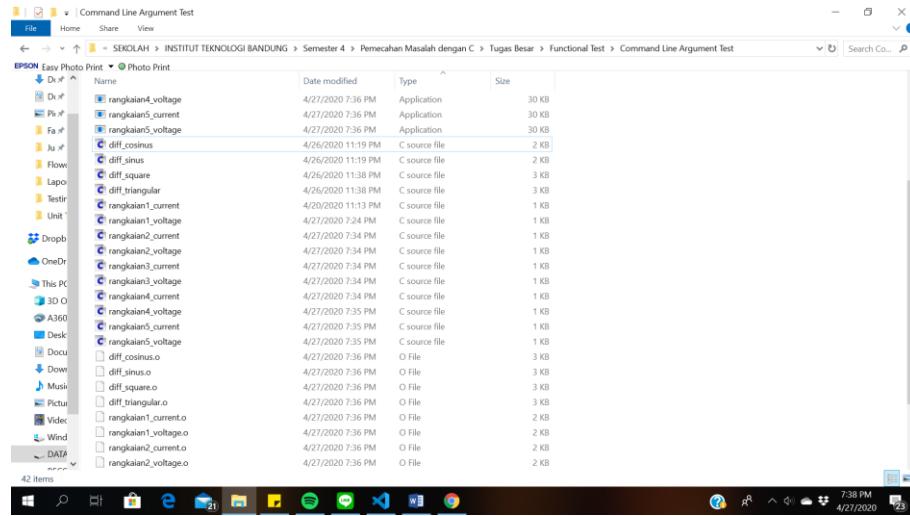
Potongan Kode Program untuk Melakukan Konversi

```
double vi, r1, r2, c;
char *eptr;
vi = strtod(argv[1], &eptr);
r1 = strtod(argv[2], &eptr);
r2 = strtod(argv[3], &eptr);
c = strtod(argv[4], &eptr);
```

Pada potongan kode program di atas, dilakukan *assignment* variabel dari argumen yang diberikan dari *command line argument*. vi merupakan argumen dengan index 1 yang bernilai

5, r1 argumen index 2 dengan nilai 1000, dan seterusnya. Setelah proses *assignment* dilakukan, akan dilakukan proses selanjutnya yaitu perhitungan dan memindahkan *array* hasil perhitungan ke dalam file .csv.

Untuk mengecek apakah setiap program telah dapat dipanggil dengan *command line argument*, semua program C akan di-copy ke folder baru dan dilakukan kompilasi di sana. Kemudian satu per satu program akan di-*invoke* melalui *command prompt* dengan menggunakan *command line argument*. Berikut ini tangkapan layar hasil pengujian:



Gambar 5.2.2.1 Tangkapan Layar Folder Kondisi Awal

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.17763.1158]
(c) 2018 Microsoft Corporation. All rights reserved.

D:\MARTINUS\SEKOLAH\INSTITUT TEKNOLOGI BANDUNG\Semester 4\Pemecahan Masalah dengan C\Tugas Besar\Functional Test\Command Line Argument Test>diff_cosinus 4 5 1000 0.0001
D:\MARTINUS\SEKOLAH\INSTITUT TEKNOLOGI BANDUNG\Semester 4\Pemecahan Masalah dengan C\Tugas Besar\Functional Test\Command Line Argument Test>diff_sinus 4 5 1000 0.0001
D:\MARTINUS\SEKOLAH\INSTITUT TEKNOLOGI BANDUNG\Semester 4\Pemecahan Masalah dengan C\Tugas Besar\Functional Test\Command Line Argument Test>diff_triangular 4 5 1000 0.0001
D:\MARTINUS\SEKOLAH\INSTITUT TEKNOLOGI BANDUNG\Semester 4\Pemecahan Masalah dengan C\Tugas Besar\Functional Test\Command Line Argument Test>diff_square 4 5 1000 0.0001
D:\MARTINUS\SEKOLAH\INSTITUT TEKNOLOGI BANDUNG\Semester 4\Pemecahan Masalah dengan C\Tugas Besar\Functional Test\Command Line Argument Test>
```

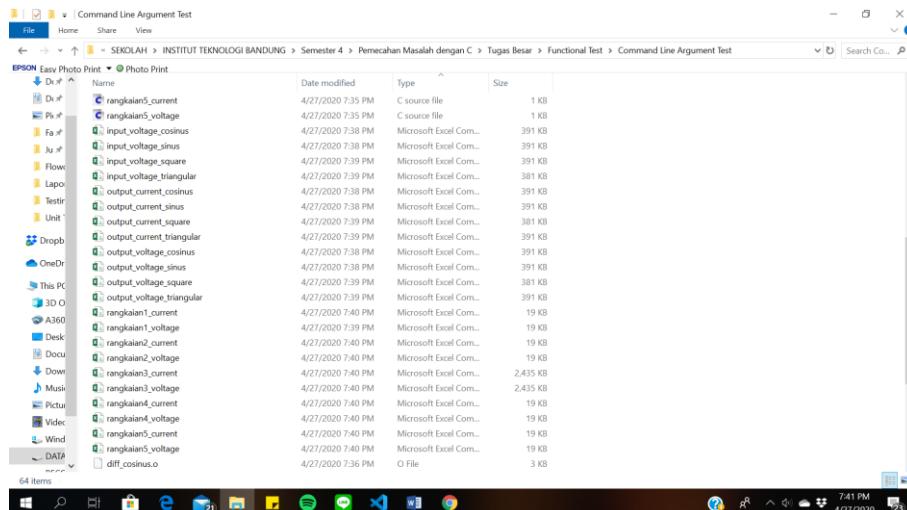
Gambar 5.2.2.2 Tangkapan Layar Command Prompt 1

```

C:\Windows\System32\cmd.exe
Line Argument Test>diff_sinus 4 5 1000 0.0001
D:\MARTINUS\SEKOLAH\INSTITUT TEKNOLOGI BANDUNG\Semester 4\Pemecahan Masalah dengan C\Tugas Besar\Functional Test\Command Line Argument Test>diff_triangular 4 5 1000 0.0001
D:\MARTINUS\SEKOLAH\INSTITUT TEKNOLOGI BANDUNG\Semester 4\Pemecahan Masalah dengan C\Tugas Besar\Functional Test\Command Line Argument Test>diff_square 4 5 1000 0.0001
D:\MARTINUS\SEKOLAH\INSTITUT TEKNOLOGI BANDUNG\Semester 4\Pemecahan Masalah dengan C\Tugas Besar\Functional Test\Command Line Argument Test>rangkaian1_voltage 5 1000 1000 0.0001
D:\MARTINUS\SEKOLAH\INSTITUT TEKNOLOGI BANDUNG\Semester 4\Pemecahan Masalah dengan C\Tugas Besar\Functional Test\Command Line Argument Test>rangkaian1_current 5 1000 1000 0.0001
D:\MARTINUS\SEKOLAH\INSTITUT TEKNOLOGI BANDUNG\Semester 4\Pemecahan Masalah dengan C\Tugas Besar\Functional Test\Command Line Argument Test>rangkaian2_voltage 5 1000 1000 0.0001
D:\MARTINUS\SEKOLAH\INSTITUT TEKNOLOGI BANDUNG\Semester 4\Pemecahan Masalah dengan C\Tugas Besar\Functional Test\Command Line Argument Test>rangkaian2_current 5 1000 1000 0.0001
D:\MARTINUS\SEKOLAH\INSTITUT TEKNOLOGI BANDUNG\Semester 4\Pemecahan Masalah dengan C\Tugas Besar\Functional Test\Command Line Argument Test>rangkaian3_voltage 5 1000 1000 0.0001
D:\MARTINUS\SEKOLAH\INSTITUT TEKNOLOGI BANDUNG\Semester 4\Pemecahan Masalah dengan C\Tugas Besar\Functional Test\Command Line Argument Test>rangkaian3_current 5 1000 1000 0.0001
D:\MARTINUS\SEKOLAH\INSTITUT TEKNOLOGI BANDUNG\Semester 4\Pemecahan Masalah dengan C\Tugas Besar\Functional Test\Command Line Argument Test>rangkaian4_voltage 5 1000 1000 0.0001
D:\MARTINUS\SEKOLAH\INSTITUT TEKNOLOGI BANDUNG\Semester 4\Pemecahan Masalah dengan C\Tugas Besar\Functional Test\Command Line Argument Test>rangkaian4_current 5 1000 1000 0.0001
D:\MARTINUS\SEKOLAH\INSTITUT TEKNOLOGI BANDUNG\Semester 4\Pemecahan Masalah dengan C\Tugas Besar\Functional Test\Command Line Argument Test>rangkaian5_voltage 5 1000 1000 0.0001
D:\MARTINUS\SEKOLAH\INSTITUT TEKNOLOGI BANDUNG\Semester 4\Pemecahan Masalah dengan C\Tugas Besar\Functional Test\Command Line Argument Test>rangkaian5_current 5 1000 1000 0.0001

```

Gambar 5.2.2.3 Tangkapan Layar Command Prompt 2



Gambar 5.2.2.4 Tangkapan Layar Folder Final

Pada gambar 5.2.2.1, terdapat gambar tangkapan layar yang menunjukkan kondisi awal folder (ketika program C telah ter-*compile* dan belum dilakukan pemanggilan *command line argument*). Gambar 5.2.2.2 dan 5.2.2.3 merupakan tangkapan layar yang menunjukkan pemanggilan masing-masing *executable* file dengan menggunakan *command line argument*. Kondisi akhir dari folder ditunjukkan pada gambar 5.2.2.4 yang menunjukkan adanya file .csv baru yang terbentuk dari *run* program. Berdasarkan hasil yang didapat, dapat disimpulkan bahwa bagian tersebut dapat secara fungsional bekerja dengan baik.

5.2.3 CSV to Plot

Setelah file csv dari program terbentuk, proses terakhir yang harus dilakukan adalah untuk melakukan *plotting* file data V-t atau I-t yang terdapat pada file csv dengan menggunakan library matplotlib pyplot serta numpy pada bahasa Python. Versi Python yang digunakan dalam pengerjaan tugas besar ini merupakan Python 3.7.

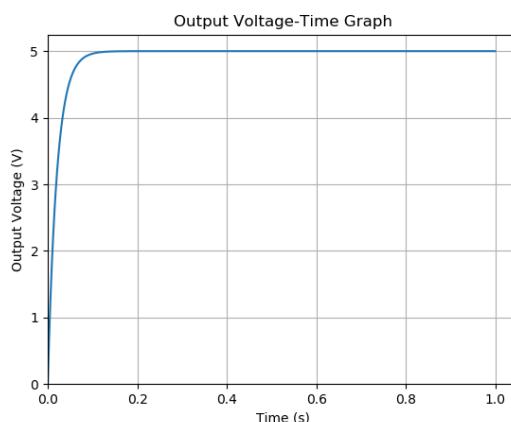
Dalam melakukan *functional test* pada bagian plotting grafik dari file .csv dengan menggunakan matplotlib, testing didesain sedemikian rupa sehingga telah terdapat file .csv awal yang telah ter-generate dari bagian 5.2.2 dan akan digunakan bahasa Python untuk melakukan plotting grafik sebagai berikut:

Potongan Kode Program Plotting Grafik

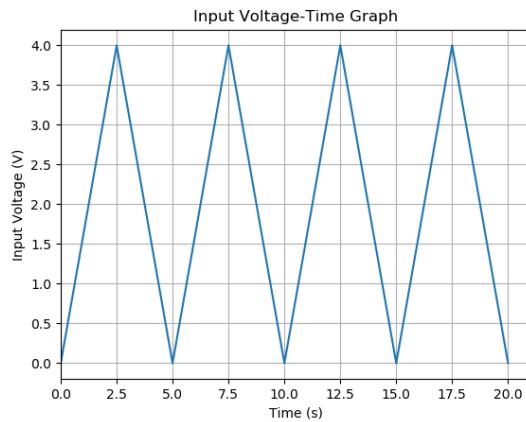
```
import numpy as np
import matplotlib.pyplot as plt

x, y = np.loadtxt("output_voltage_triangular.csv", delimiter=",", unpack=True,
plt.plot(x,y)
plt.xlabel("Time (s)")
plt.ylabel("Voltage (V)")
plt.grid()
plt.xlim(0)
plt.show()
```

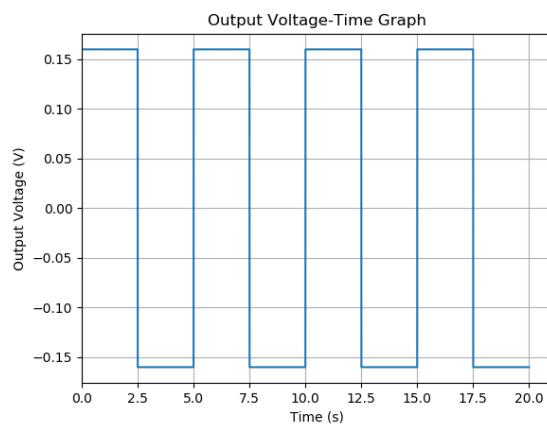
Potongan kode program di atas merupakan potongan kode program untuk melakukan plotting grafik dari file csv. Pada line 1 dan 2, terdapat deklarasi import library yang akan digunakan untuk melakukan plotting grafik. Pada line 3, terdapat proses assignment variabel x dan y yang masing-masing menyatakan waktu dan tegangan atau arus (pada kasus di atas merupakan tegangan). Line 4 dan seterusnya merupakan kode untuk memunculkan grafik serta mengatur tampilan grafik (label sumbu x dan y). Berikut ini terdapat grafik hasil *test run* program:



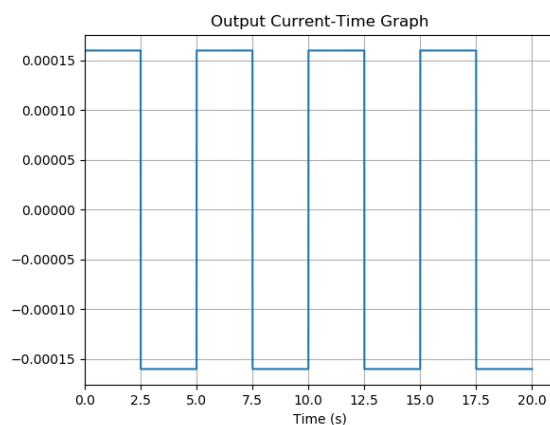
Gambar 5.2.3.1 Hasil Test Run Rangkaian RC 1



Gambar 5.2.3.1 Tegangan Input Rangkaian Differentiator dengan Input Triangular



Gambar 5.2.3.2 Tegangan Output Rangkaian Differentiator dengan Input Triangular



Gambar 5.2.3.3 Arus Output Rangkaian Differentiator dengan Input Triangular

Berdasarkan *functional test CSV to Plot* terlihat bahwa plot yang bersesuaian, baik untuk rangkaian RC maupun rangkaian differentiator dapat muncul dengan baik. Sehingga, berdasarkan testing yang telah dilakukan proses *plotting* grafik dapat berfungsi dengan baik.

Tabel 5.2 Hasil Functional Test Program

No	Process	Hasil Functional Test
1	GUI to command line argument	✓
2	Run program dengan command line argument	✓
3	CSV to plot	✓

6. IMPLEMENTASI

Pada bagian ini terdapat implementasi kode program yang telah dirancang oleh kelompok. Secara umum terdapat dua jenis kode program, yang pertama adalah yang menggunakan bahasa Python dan yang kedua adalah bahasa C. Bahasa Python digunakan untuk mengatur *interface* dan *plotting grafik* dari simulator sedangkan C digunakan untuk melakukan perhitungan pada simulator yang dibuat. Berikut ini terdapat implementasi kode program:

Implementasi Graphical User Interface Program (top_simulator.py)

```
# PROGRAM SIMULATOR RANGKAIAN RC DAN DIFFERENTIATOR OLEH KELOMPOK 2
# Fazha Ivanda / 13218008
# Apria Wati / 13218028
# Matthew Terrence A. H. / 13218038
# Martinus William Hartono / 13218044

import tkinter as tk
import os
import numpy as np
import matplotlib.pyplot as plt
from tkinter import ttk
from PIL import Image, ImageTk

#FUNCTION
def runSimRC():
    combo_value = combo1_RC.get()
    combo_value2 = combo2_RC.get()
    vi = ent_vi_RC.get()
    r1 = ent_r1_RC.get()
    r2 = ent_r2_RC.get()
    c = ent_c_RC.get()

    if (combo_value == "1st Circuit" and combo_value2 == "Voltage"):
        file_name = "rangkaian1_voltage"
        os.system(file_name +vi+" "+r1+" "+r2+" "+c)
        print(file_name, vi, r1, r2, c) #debug
        x, y = np.loadtxt("rangkaian1_voltage.csv", unpack=True, delimiter=",")
    elif (combo_value == "1st Circuit" and combo_value2 == "Current"):
        file_name = "rangkaian1_current"
        os.system(file_name +vi+" "+r1+" "+r2+" "+c)
        print(file_name, vi, r1, r2, c) #debug
        x, y = np.loadtxt("rangkaian1_current.csv", unpack=True, delimiter=",")
    elif (combo_value == "2nd Circuit" and combo_value2 == "Voltage"):
        file_name = "rangkaian2_voltage"
        os.system(file_name +vi+" "+r1+" "+r2+" "+c)
        print(file_name, vi, r1, r2, c) #debug
        x, y = np.loadtxt("rangkaian2_voltage.csv", unpack=True, delimiter=",")
    elif (combo_value == "2nd Circuit" and combo_value2 == "Current"):
        file_name = "rangkaian2_current"
        os.system(file_name +vi+" "+r1+" "+r2+" "+c)
        print(file_name, vi, r1, r2, c) #debug
```

```

x, y = np.loadtxt("rangkaian2_current.csv", unpack=True, delimiter=",")
elif (combo_value == "3rd Circuit" and combo_value2 == "Voltage"):
    file_name = "rangkaian3_voltage"
    os.system(file_name +vi+ "+r1+" "+r2+" "+c)
    print(file_name, vi, r1, r2, c) #debug
    x, y = np.loadtxt("rangkaian3_voltage.csv", unpack=True, delimiter=",")
elif (combo_value == "3rd Circuit" and combo_value2 == "Current"):
    file_name = "rangkaian3_current"
    os.system(file_name +vi+ "+r1+" "+r2+" "+c)
    print(file_name, vi, r1, r2, c) #debug
    x, y = np.loadtxt("rangkaian3_current.csv", unpack=True, delimiter=",")
elif (combo_value == "4th Circuit" and combo_value2 == "Voltage"):
    file_name = "rangkaian4_voltage"
    os.system(file_name +vi+ "+r1+" "+r2+" "+c)
    print(file_name, vi, r1, r2, c) #debug
    x, y = np.loadtxt("rangkaian4_voltage.csv", unpack=True, delimiter=",")
elif (combo_value == "4th Circuit" and combo_value2 == "Current"):
    file_name = "rangkaian4_current"
    os.system(file_name +vi+ "+r1+" "+r2+" "+c)
    print(file_name, vi, r1, r2, c) #debug
    x, y = np.loadtxt("rangkaian4_current.csv", unpack=True, delimiter=",")
elif (combo_value == "5th Circuit" and combo_value2 == "Voltage"):
    file_name = "rangkaian5_voltage"
    os.system(file_name +vi+ "+r1+" "+r2+" "+c)
    print(file_name, vi, r1, r2, c) #debug
    x, y = np.loadtxt("rangkaian5_voltage.csv", unpack=True, delimiter=",")
elif (combo_value == "5th Circuit" and combo_value2 == "Current"):
    file_name = "rangkaian5_current"
    os.system(file_name +vi+ "+r1+" "+r2+" "+c)
    print(file_name, vi, r1, r2, c) #debug
    x, y = np.loadtxt("rangkaian5_current.csv", unpack=True, delimiter=",")

plt.plot(x,y)
plt.xlabel("Time (s)")
plt.grid()
plt.xlim(0)
plt.ylim(0)

if(combo_value2 == "Voltage"):
    plt.title("Output Voltage-Time Graph")
    plt.ylabel("Output Voltage (V)")
else:
    plt.title("Output Current-Time Graph")
    plt.ylabel("Output Current (A)")

plt.show()

def runSimDiff():
    combo_value = combo1_diff.get()
    combo_value2 = combo2_diff.get()
    A = ent_A_diff.get()
    T = ent_T_diff.get()
    R = ent_R_diff.get()
    C = ent_C_diff.get()

    if (combo_value == "Sine" and combo_value2 == "Input Voltage"):
        file_name = "diff_sinus"
        os.system(file_name +A+" "+T+" "+R+" "+C)
        print(file_name, A, T, R, C) #debug
        x, y = np.loadtxt("input_voltage_sinus.csv", unpack=True, delimiter=",")
    elif (combo_value == "Sine" and combo_value2 == "Output Voltage"):
        file_name = "diff_sinus"
        os.system(file_name +A+" "+T+" "+R+" "+C)
        print(file_name, A, T, R, C) #debug
        x, y = np.loadtxt("output_voltage_sinus.csv", unpack=True,
delimiter=",")
    elif (combo_value == "Sine" and combo_value2 == "Output Current"):
        file_name = "diff_sinus"

```

```

os.system(file_name +A+ " " +T+ " " +R+ " " +C)
print(file_name, A, T, R, C) #debug
x, y = np.loadtxt("output_current_sinus.csv", unpack=True,
delimiter=", ")
elif (combo_value == "Cosine" and combo_value2 == "Input Voltage"):
    file_name = "diff_cosinus"
    os.system(file_name +A+ " " +T+ " " +R+ " " +C)
    print(file_name, A, T, R, C) #debug
    x, y = np.loadtxt("input_voltage_cosinus.csv", unpack=True,
delimiter=", ")
elif (combo_value == "Cosine" and combo_value2 == "Output Voltage"):
    file_name = "diff_cosinus"
    os.system(file_name +A+ " " +T+ " " +R+ " " +C)
    print(file_name, A, T, R, C) #debug
    x, y = np.loadtxt("output_voltage_cosinus.csv", unpack=True,
delimiter=", ")
elif (combo_value == "Cosine" and combo_value2 == "Output Current"):
    file_name = "diff_cosinus"
    os.system(file_name +A+ " " +T+ " " +R+ " " +C)
    print(file_name, A, T, R, C) #debug
    x, y = np.loadtxt("output_current_cosinus.csv", unpack=True,
delimiter=", ")
elif (combo_value == "Triangle" and combo_value2 == "Input Voltage"):
    file_name = "diff_triangular"
    os.system(file_name +A+ " " +T+ " " +R+ " " +C)
    print(file_name, A, T, R, C) #debug
    x, y = np.loadtxt("input_voltage_triangular.csv", unpack=True,
delimiter=", ")
elif (combo_value == "Triangle" and combo_value2 == "Output Voltage"):
    file_name = "diff_triangular"
    os.system(file_name +A+ " " +T+ " " +R+ " " +C)
    print(file_name, A, T, R, C) #debug
    x, y = np.loadtxt("output_voltage_triangular.csv", unpack=True,
delimiter=", ")
elif (combo_value == "Triangle" and combo_value2 == "Output Current"):
    file_name = "diff_triangular"
    os.system(file_name +A+ " " +T+ " " +R+ " " +C)
    print(file_name, A, T, R, C) #debug
    x, y = np.loadtxt("output_current_triangular.csv", unpack=True,
delimiter=", ")
elif (combo_value == "Square" and combo_value2 == "Input Voltage"):
    file_name = "diff_square"
    os.system(file_name +A+ " " +T+ " " +R+ " " +C)
    print(file_name, A, T, R, C) #debug
    x, y = np.loadtxt("input_voltage_square.csv", unpack=True,
delimiter=", ")
elif (combo_value == "Square" and combo_value2 == "Output Voltage"):
    file_name = "diff_square"
    os.system(file_name +A+ " " +T+ " " +R+ " " +C)
    print(file_name, A, T, R, C) #debug
    x, y = np.loadtxt("output_voltage_square.csv", unpack=True,
delimiter=", ")
elif (combo_value == "Square" and combo_value2 == "Output Current"):
    file_name = "diff_square"
    os.system(file_name +A+ " " +T+ " " +R+ " " +C)
    print(file_name, A, T, R, C) #debug
    x, y = np.loadtxt("output_current_square.csv", unpack=True,
delimiter=", ")

plt.plot(x,y)
plt.xlabel("Time (s)")
plt.grid()
plt.xlim(0)

if (combo_value2 == "Output Voltage"):
    plt.title("Output Voltage-Time Graph")
    plt.ylabel("Output Voltage (V)")
elif (combo_value2 == "Input Voltage"):
```

```

plt.title("Input Voltage-Time Graph")
plt.ylabel("Input Voltage (V)")
else:
    plt.title("Output Current-Time Graph")
    plt.ylabel("Output Current (A)")

plt.show()

#MAIN
window = tk.Tk()
window.title("Tugas Besar PMC - Kelompok 2")
window.geometry("900x650")
window.resizable(width=True, height=True)

lbl_judul = tk.Label(text="RC AND DIFFERENTIATOR CIRCUIT SIMULATOR", font="Tahoma 18 bold", bg="white")
lbl_judul.grid(row=0, columnspan=10)

lbl_penjelasan = tk.Label(text="by Group 2 of EL2008\nThis simulator is a simulator for simulating RC circuits and first-order differentiator op-amp circuits.\n\nThe form on the left is the form to do RC simulation while the right form is to simulate the differentiator circuit.\n\nTo conduct a simulation, the user immediately fills in all the required data on each form then presses the submit button and the simulation will run.\n\nEnjoy simulating!", bg="white")
lbl_penjelasan.grid(columnspan=10, row=1, pady=10)

photo1 = tk.PhotoImage(file="rsz_rangkaian1.png")
pict_circuit1 = tk.Label(window, image=photo1, bg="white")
pict_circuit1.grid(row=2, column=0, columnspan=2)

photo2 = tk.PhotoImage(file="rsz_rangkaian2.png")
pict_circuit2 = tk.Label(window, image=photo2, bg="white")
pict_circuit2.grid(row=2, column=2, columnspan=2)

photo3 = tk.PhotoImage(file="rsz_rangkaian3.png")
pict_circuit3 = tk.Label(window, image=photo3, bg="white")
pict_circuit3.grid(row=2, column=4, columnspan=2)

photo4 = tk.PhotoImage(file="rsz_rangkaian4.png")
pict_circuit4 = tk.Label(window, image=photo4, bg="white")
pict_circuit4.grid(row=2, column=6, columnspan=2)

photo5 = tk.PhotoImage(file="rsz_rangkaian5.png")
pict_circuit5 = tk.Label(window, image=photo5, bg="white")
pict_circuit5.grid(row=2, column=8, columnspan=2)

photo6 = tk.PhotoImage(file="rsz_differentiator.png")
pict_circuit6 = tk.Label(window, image=photo6, bg="white")
pict_circuit6.grid(row=3, column=0, columnspan=2)

photo7 = tk.PhotoImage(file="rsz_sinus.png")
pict_sin = tk.Label(window, image=photo7, bg="white")
pict_sin.grid(row=3, column=2, columnspan=2)

photo8 = tk.PhotoImage(file="rsz_cosinus.png")
pict_cos = tk.Label(window, image=photo8, bg="white")
pict_cos.grid(row=3, column=4, columnspan=2)

photo9 = tk.PhotoImage(file="rsz_triangle.png")
pict_triangle = tk.Label(window, image=photo9, bg="white")
pict_triangle.grid(row=3, column=6, columnspan=2)

photo10 = tk.PhotoImage(file="rsz_square.png")
pict_square = tk.Label(window, image=photo10, bg="white")
pict_square.grid(row=3, column=8, columnspan=2)

#Gap
lbl_gap = tk.Label(bg="white")

```

```

lbl_gap.grid(column=0, row=4, columnspan=10, pady=0)

#RC Circuit
lbl_headingRC = tk.Label(text="RC Circuit", font="Tahoma 12 bold underline",
bg="white")
lbl_headingRC.grid(column=0, row=5, columnspan=5, pady=10)

#Combobox 1 RC
lbl_combo1_RC = tk.Label(text="Circuit Type = ", bg="white")
lbl_combo1_RC.grid(column=0, row=6, columnspan=2)
combo1_RC = ttk.Combobox(window, values=["1st Circuit", "2nd Circuit", "3rd
Circuit", "4th Circuit", "5th Circuit"])
combo1_RC.grid(column=2, row=6)
combo1_RC.current(0)

#Combobox 2 RC
lbl_combo2_RC = tk.Label(text="Output Variable = ", bg="white")
lbl_combo2_RC.grid(column=0, row=7, columnspan=2)
combo2_RC = ttk.Combobox(window, values=["Voltage", "Current"])
combo2_RC.grid(column=2, row=7)
combo2_RC.current(0)

#Entry vi
lbl_vi_RC = tk.Label(text="Vi = ", bg="white")
lbl_vi_RC.grid(column=0, row=8, columnspan=2)
ent_vi_RC = tk.Entry(width=20)
ent_vi_RC.grid(column=2, row=8)
lbl_vi2_RC = tk.Label(text=" V", bg="white")
lbl_vi2_RC.grid(column=3, row=8)

#Entry R1
lbl_r1_RC = tk.Label(text="R1 = ", bg="white")
lbl_r1_RC.grid(column=0, row=9, columnspan=2)
ent_r1_RC = tk.Entry(width=20)
ent_r1_RC.grid(column=2, row=9)
lbl_r12_RC = tk.Label(text=" Ω", bg="white")
lbl_r12_RC.grid(column=3, row=9)

#Entry R2
lbl_r2_RC = tk.Label(text="R2 = ", bg="white")
lbl_r2_RC.grid(column=0, row=10, columnspan=2)
ent_r2_RC = tk.Entry(width=20)
ent_r2_RC.grid(column=2, row=10)
lbl_r22_RC = tk.Label(text=" Ω", bg="white")
lbl_r22_RC.grid(column=3, row=10)

#Entry C
lbl_c_RC = tk.Label(text="C = ", bg="white")
lbl_c_RC.grid(column=0, row=11, columnspan=2)
ent_c_RC = tk.Entry(width=20)
ent_c_RC.grid(column=2, row=11)
lbl_c2_RC = tk.Label(text=" F", bg="white")
lbl_c2_RC.grid(column=3, row=11)

#Submit RC
btn_submit_RC = tk.Button(text="Submit Data", bg='#C2EDFD', command=runSimRC)
btn_submit_RC.grid(column=0, row=12, columnspan=5)

#Gap
lbl_gap2 = tk.Label(bg="white")
lbl_gap2.grid(column=4, row=6, columnspan=2, rowspan=7)

#Differentiator
lbl_headingDiff = tk.Label(text="Differentiator Circuit", font="Tahoma 12 bold
underline", bg="white")
lbl_headingDiff.grid(column=6, row=5, columnspan=5, pady=10)

#Combobox 1 Diff

```

```

lbl_combo1_diff = tk.Label(text="Input Wave = ", bg="white")
lbl_combo1_diff.grid(column=6, row=6, columnspan=2)
combo1_diff = ttk.Combobox(window, values=["Sine", "Cosine", "Triangle",
"Square"])
combo1_diff.grid(column=8, row=6)
combo1_diff.current(0)

#Combobox 2 Diff
lbl_combo2_diff = tk.Label(text="Variable = ", bg="white")
lbl_combo2_diff.grid(column=6, row=7, columnspan=2)
combo2_diff = ttk.Combobox(window, values=["Input Voltage", "Output Voltage",
"Output Current"])
combo2_diff.grid(column=8, row=7)
combo2_diff.current(0)

#Entry A
lbl_A_diff = tk.Label(text="Input Amplitude = ", bg="white")
lbl_A_diff.grid(column=6, row=8, columnspan=2)
ent_A_diff = tk.Entry(width=20)
ent_A_diff.grid(column=8, row=8)
lbl_A2_diff = tk.Label(text=" V", bg="white")
lbl_A2_diff.grid(column=9, row=8)

#Entry T
lbl_T_diff = tk.Label(text="Period (T) = ", bg="white")
lbl_T_diff.grid(column=6, row=9, columnspan=2)
ent_T_diff = tk.Entry(width=20)
ent_T_diff.grid(column=8, row=9)
lbl_T2_diff = tk.Label(text=" s", bg="white")
lbl_T2_diff.grid(column=9, row=9)

#Entry R Diff
lbl_R_diff = tk.Label(text="R = ", bg="white")
lbl_R_diff.grid(column=6, row=10, columnspan=2)
ent_R_diff = tk.Entry(width=20)
ent_R_diff.grid(column=8, row=10)
lbl_R2_diff = tk.Label(text=" Ω", bg="white")
lbl_R2_diff.grid(column=9, row=10)

#Entry C Diff
lbl_C_diff = tk.Label(text="C = ", bg="white")
lbl_C_diff.grid(column=6, row=11, columnspan=2)
ent_C_diff = tk.Entry(width=20)
ent_C_diff.grid(column=8, row=11)
lbl_C2_diff = tk.Label(text=" F", bg="white")
lbl_C2_diff.grid(column=9, row=11)

#Submit Diff
btn_submit_diff = tk.Button(text="Submit Data", bg="#C2EDFD", command=runSimDiff)
btn_submit_diff.grid(column=6, row=12, columnspan=5)

window.configure(bg="white")
window.mainloop()

```

Implementasi rangkaian1_voltage.c

```

//RANGKAIAN 1 - VOLTAGE

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

void rangkaian1(double vi,double r1,double r2,double c)
{
    double t, vo[1000], dt=0.001, timeconstant=(r1+r2)*c;
    int i;

```

```

FILE *fp;
char *filename = "rangkaian1_voltage.csv";
fp = fopen(filename, "w+");
vo[0]=0;
t=0;
fprintf(fp, "%f,", t);
fprintf(fp, "%f\n", vo[0]);
for(i=1;i<1000;i++)
{
    t+=dt;
    vo[i] = (vi*dt + vo[i-1]*timeconstant)/(dt+timeconstant);
    fprintf(fp, "%f,", t);
    fprintf(fp, "%f\n", vo[i]);
}
fclose(fp);
}

int main(int argc,char* argv[])
{
    double vi, r1, r2, c;
    char *eptr;
    vi = strtod(argv[1], &eptr);
    r1 = strtod(argv[2], &eptr);
    r2 = strtod(argv[3], &eptr);
    c = strtod(argv[4], &eptr);

    rangkaian1(vi, r1, r2, c);

    return 0;
}

```

Implementasi rangkaian1_current.c

```

//RANGKAIAN 1 - CURRENT

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

void rangkaian1(double vi,double r1,double r2,double c)
{
    double t, vo[1000], dt=0.001, timeconstant=(r1+r2)*c;
    int i;
    FILE *fp;
    char *filename = "rangkaian1_current.csv";
    fp = fopen(filename, "w+");
    vo[0]=0;
    t=0;
    fprintf(fp, "%f,", t);
    fprintf(fp, "%f\n", (vi-vo[0])/(r1+r2));
    for(i=1;i<1000;i++)
    {
        t+=dt;
        vo[i] = (vi*dt + vo[i-1]*timeconstant)/(dt+timeconstant);
        fprintf(fp, "%f,", t);
        fprintf(fp, "%f\n", (vi-vo[i])/(r1+r2));
    }
    fclose(fp);
}

```

```

int main(int argc,char* argv[])
{
    double vi, r1, r2, c;
    char *eptr;
    vi = strtod(argv[1], &eptr);
    r1 = strtod(argv[2], &eptr);
    r2 = strtod(argv[3], &eptr);
    c = strtod(argv[4], &eptr);

    rangkaian1(vi, r1, r2, c);

    return 0;
}

```

Implementasi rangkaian2_voltage.c

```

//RANGKAIAN 2 - VOLTAGE

#include <stdio.h>
#include <stdlib.h>

void rangkaian2 (double vin,double r1,double r2,double c){
    double t,i,vc;
    int j;
    double dt = 0.00001;
    FILE *fp;
    char *filename = "rangkaian2_voltage.csv";
    fp = fopen(filename,"w");
    vc = 0;
    t = 0 ;
    i = vin/r1;
    fprintf(fp, "%lf,", t);
    fprintf(fp, "%lf\n", vc);
    for (j=0;j<1000;j++) {
        vc = vin;
        t += dt;
        fprintf(fp, "%lf,", t);
        fprintf(fp, "%lf\n", vc);
    }
    fclose(fp);
}

int main(int argc,char* argv[])
{
    double vi, r1, r2, c;
    char *eptr;
    vi = strtod(argv[1], &eptr);
    r1 = strtod(argv[2], &eptr);
    r2 = strtod(argv[3], &eptr);
    c = strtod(argv[4], &eptr);

    rangkaian2(vi, r1, r2, c);

    return 0;
}

```

Implementasi rangkaian2_current.c

```
//RANGKAIAN 2 - CURRENT
```

```

#include <stdio.h>
#include <stdlib.h>

void rangkaian2 (double vin,double r1,double r2,double c){
    double t,i,vc;
    int j;
    double dt = 0.001;
    FILE *fp;
    char *filename = "rangkaian2_current.csv";
    fp = fopen(filename,"w+");
    vc = 0;
    t = 0 ;
    i = vin/r1;
    fprintf(fp, "%f,", t);
    fprintf(fp, "%f\n", vc);
    for (j=0;j<1000;j++) {
        t += dt;
        fprintf(fp, "%f,", t);
        fprintf(fp, "%f\n", vc);
    }
    fclose(fp);
}

int main(int argc,char* argv[])
{
    double vi, r1, r2, c;
    char *eptr;
    vi = strtod(argv[1], &eptr);
    r1 = strtod(argv[2], &eptr);
    r2 = strtod(argv[3], &eptr);
    c = strtod(argv[4], &eptr);

    rangkaian2(vi, r1, r2, c);

    return 0;
} //RANGKAIAN 3 - VOLTAGE

#include <stdio.h>
#include <stdlib.h>

void rangkaian3 (double vin,double r1,double r2,double c){
    double i,vc,t;
    double dt = 0.00001;
    FILE *fp;
    char *filename = "rangkaian3_voltage.csv";
    fp = fopen(filename,"w");
    vc = 0;
    t = 0;
    i = vin/r1;
    fprintf(fp, "%lf,", t);
    fprintf(fp, "%lf\n", vc);
    while (vin - vc >= dt){
        t += dt;
        vc += i * dt / c;
        i = (vin - vc) / r1;
        fprintf(fp, "%lf,", t);
        fprintf(fp, "%lf\n", vc);
    }
    fclose(fp);
}

```

```

}

int main(int argc,char* argv[])
{
    double vi, r1, r2, c;
    char *eptr;
    vi = strtod(argv[1], &eptr);
    r1 = strtod(argv[2], &eptr);
    r2 = strtod(argv[3], &eptr);
    c = strtod(argv[4], &eptr);

    rangkaian3(vi, r1, r2, c);

    return 0;
}

```

Implementasi rangkaian3_voltage.c

```

//RANGKAIAN 3 - VOLTAGE

#include <stdio.h>
#include <stdlib.h>

void rangkaian3 (double vin,double r1,double r2,double c){
    double i,vc,t;
    double dt = 0.00001;
    FILE *fp;
    char *filename = "rangkaian3_voltage.csv";
    fp = fopen(filename,"w");
    vc = 0;
    t = 0;
    i = vin/r1;
    fprintf(fp, "%lf,", t);
    fprintf(fp, "%lf\n", vc);
    while (vin - vc >= dt){
        t += dt;
        vc += i * dt / c;
        i = (vin - vc) / r1;
        fprintf(fp, "%lf,", t);
        fprintf(fp, "%lf\n", vc);
    }
    fclose(fp);
}

int main(int argc,char* argv[])
{
    double vi, r1, r2, c;
    char *eptr;
    vi = strtod(argv[1], &eptr);
    r1 = strtod(argv[2], &eptr);
    r2 = strtod(argv[3], &eptr);
    c = strtod(argv[4], &eptr);

    rangkaian3(vi, r1, r2, c);

    return 0;
}

```

Implementasi rangkaian3_current.c

```

//RANGKAIAN 3 - CURRENT

#include <stdio.h>
#include <stdlib.h>

void rangkaian3 (double vin,double r1,double r2,double c){
    double i,vc,t;
    double dt = 0.00001;
    FILE *fp;
    char *filename = "rangkaian3_current.csv";
    fp = fopen(filename,"w");
    vc = 0;
    t = 0;
    i = vin/r1;
    fprintf(fp, "%lf,", t);
    fprintf(fp, "%lf\n", i);
    while (vin - vc >= dt){
        t += dt;
        vc += i * dt / c;
        i = (vin - vc) / r1;
        fprintf(fp, "%lf,", t);
        fprintf(fp, "%lf\n", i);
    }
    fclose(fp);
}

int main(int argc,char* argv[])
{
    double vi, r1, r2, c;
    char *eptr;
    vi = strtod(argv[1], &eptr);
    r1 = strtod(argv[2], &eptr);
    r2 = strtod(argv[3], &eptr);
    c = strtod(argv[4], &eptr);

    rangkaian3(vi, r1, r2, c);

    return 0;
}

```

Implementasi rangkaian4_voltage.c

```

//RANGKAIAN 4 - VOLTAGE

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

void rangkaian4(double vi,double r1,double r2,double c)
{
    double t, vo[1000], dt=0.001, timeconstant=r1*r2*c/(r1+r2);
    int i;
    FILE *fp;
    char *filename = "rangkaian4_voltage.csv";
    fp = fopen(filename, "w+");
    vo[0]=0;
    t=0;
    fprintf(fp, "%f,", t);
    fprintf(fp, "%f\n", vo[0]);
    for(i=1;i<1000;i++)

```

```

    {
        t+=dt;
        vo[i] = (vi*dt + vo[i-1]*timeconstant)/(dt+timeconstant);
        fprintf(fp, "%f,", t);
        fprintf(fp, "%f\n", vo[i]);
    }
}

int main(int argc,char* argv[])
{
    double vi, r1, r2, c;
    char *eptr;
    vi = strtod(argv[1], &eptr);
    r1 = strtod(argv[2], &eptr);
    r2 = strtod(argv[3], &eptr);
    c = strtod(argv[4], &eptr);

    rangkaian4(vi, r1, r2, c);

    return 0;
}

```

Implementasi rangkaian4_current.c

```

//RANGKAIAN 4 - CURRENT

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

void rangkaian4(double vi,double r1,double r2,double c)
{
    double t, vo[1000], dt=0.001, timeconstant=r1*r2*c/(r1+r2);
    int i;
    FILE *fp;
    char *filename = "rangkaian4_current.csv";
    fp = fopen(filename, "w+");
    vo[0]=0;
    t=0;
    fprintf(fp, "%f,", t);
    fprintf(fp, "%f\n", (vi-vo[0])/((r1*r2)/(r1+r2)));
    for(i=1;i<1000;i++)
    {
        t+=dt;
        vo[i] = (vi*dt + vo[i-1]*timeconstant)/(dt+timeconstant);
        fprintf(fp, "%f,", t);
        fprintf(fp, "%f\n", (vi-vo[i])/((r1*r2)/(r1+r2)));
    }
}

int main(int argc,char* argv[])
{
    double vi, r1, r2, c;
    char *eptr;
    vi = strtod(argv[1], &eptr);
    r1 = strtod(argv[2], &eptr);
    r2 = strtod(argv[3], &eptr);
    c = strtod(argv[4], &eptr);

    rangkaian4(vi, r1, r2, c);
}

```

```

    return 0;
}

```

Implementasi rangkaian5_voltage.c

```

//RANGKAIAN 5 - VOLTAGE

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

void rangkaian5(double vi,double r1,double r2,double c)
{
    double t, vo[1000], dt=0.001;
    int i;
    FILE *fp;
    char *filename = "rangkaian5_voltage.csv";
    fp = fopen(filename, "w+");
    vo[0]=0;
    t=0;
    fprintf(fp, "%f,", t);
    fprintf(fp, "%f\n", vo[0]);
    for(i=1;i<1000;i++)
    {
        t+=dt;
        vo[i] = (c*vo[i-1]/dt + vi/r1)/(1/r1 + 1/r2 + c/dt);
        fprintf(fp, "%f,", t);
        fprintf(fp, "%f\n", vo[i]);
    }
}

int main(int argc,char* argv[])
{
    double vi, r1, r2, c;
    char *eptr;
    vi = strtod(argv[1], &eptr);
    r1 = strtod(argv[2], &eptr);
    r2 = strtod(argv[3], &eptr);
    c = strtod(argv[4], &eptr);

    rangkaian5(vi, r1, r2, c);

    return 0;
}

```

Implementasi rangkaian5_current.c

```

//RANGKAIAN 5 - CURRENT

#include <stdio.h>
#include <math.h>
#include <stdlib.h>

void rangkaian5(double vi,double r1,double r2,double c)
{
    double t, vo[1000], dt=0.001;
    int i;
    FILE *fp;
    char *filename = "rangkaian5_current.csv";

```

```

fp = fopen(filename, "w+");
vo[0]=0;
t=0;
fprintf(fp, "%f,", t);
fprintf(fp, "%f\n", (vi)/r1);
for(i=1;i<1000;i++)
{
    t+=dt;
    vo[i] = (c*vo[i-1]/dt + vi/r1)/(1/r1 + 1/r2 + c/dt);
    fprintf(fp, "%f,", t);
    fprintf(fp, "%f\n", ((vi-vo[i])/r1 - vo[i]/r2));
}
}

int main(int argc,char* argv[])
{
    double vi, r1, r2, c;
    char *eptr;
    vi = strtod(argv[1], &eptr);
    r1 = strtod(argv[2], &eptr);
    r2 = strtod(argv[3], &eptr);
    c = strtod(argv[4], &eptr);

    rangkaian5(vi, r1, r2, c);

    return 0;
}

```

Implementasi diff_sinus.c

```

//DIFFERENTIATOR SIMULATOR FOR SINUSOIDAL WAVES INPUT
/*
    Program akan menerima input berupa command line argument yang
    diinvoke oleh file top_simulator.py
    Output program adalah file input_voltage.sinus.csv,
    output_voltage.sinus.csv, dan output_current.sinus.csv
*/
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define pi 3.14159265359

double T,A, R, C, angular_freq, dt=0.001;

double deriveTrig(double (*f)(double t), double x0)
{
    const double delta = 1.0e-10;
    double x1 = x0;
    double x2 = x0 +delta;
    double y1 = f(x1);
    double y2 = f(x2);
    return ((y2 - y1) / delta);
};

void printToFile(FILE *fp, int x, double y)
{
    fprintf(fp, "%f,", (x*dt));
    fprintf(fp, "%f\n", y);
};

```

```

int main(int argc, char *argv[])
{
    int i;
    double Vi[20000], Vo[20000], t=0;

    char *eptr;
    A = strtod(argv[1], &eptr);
    T = strtod(argv[2], &eptr);
    R = strtod(argv[3], &eptr);
    C = strtod(argv[4], &eptr);

    FILE *fp1, *fp2, *fp3;
    fp1 = fopen("input_voltage_sinus.csv", "w+");
    fp2 = fopen("output_voltage_sinus.csv", "w+");
    fp3 = fopen("output_current_sinus.csv", "w+");

    Vi[0]=0;
    angular_freq= 2.0*pi/T;
    fprintf(fp1, "%f,", 0);
    fprintf(fp1, "%f\n", Vi[0]);

    for(i=1;i<20000;i++)
    {
        t=t+dt;
        Vi[i] = A*sin(angular_freq*t);
        printToFile(fp1, i, Vi[i]);
    };

    t=0;
    for(i=0;i<20000;i++)
    {
        Vo[i] = -A*angular_freq*R*C*deriveTrig(sin, angular_freq*t);
        printToFile(fp2, i, Vo[i]);
        printToFile(fp3, i, (Vo[i]/R));
        t=t+dt;
    };

    fclose(fp1);
    fclose(fp2);
    fclose(fp3);

    return 0;
}

```

Implementasi diff_cosinus.c

```

//DIFFERENTIATOR SIMULATOR FOR COSINUSOIDAL WAVES INPUT
/*
    Program akan menerima input berupa command line argument yang
    diinvoke oleh file top_simulator.py
    Output program adalah file input_voltage_cosinus.csv,
    output_voltage_cosinus.csv, dan output_current_cosinus.csv
*/
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define pi 3.14159265359

```

```

double T,A, R, C, angular_freq, dt=0.001;

double deriveTrig(double (*f)(double t), double x0)
{
    const double delta = 1.0e-10;
    double x1 = x0;
    double x2 = x0 +delta;
    double y1 = f(x1);
    double y2 = f(x2);
    return ((y2 - y1) / delta);
};

void printToFile(FILE *fp, int x, double y)
{
    fprintf(fp, "%f,", (x*dt));
    fprintf(fp, "%f\n", y);
};

int main(int argc, char *argv[])
{
    int i;
    double Vi[20000], Vo[20000], t=0;

    char *eptr;
    A = strtod(argv[1], &eptr);
    T = strtod(argv[2], &eptr);
    R = strtod(argv[3], &eptr);
    C = strtod(argv[4], &eptr);

    FILE *fp1, *fp2, *fp3;
    fp1 = fopen("input_voltage_cosinus.csv", "w+");
    fp2 = fopen("output_voltage_cosinus.csv", "w+");
    fp3 = fopen("output_current_cosinus.csv", "w+");

    Vi[0]=0;
    angular_freq= 2.0*pi/T;
    fprintf(fp1, "%f,", 0);
    fprintf(fp1, "%f\n", Vi[0]);

    for(i=1;i<20000;i++)
    {
        t=t+dt;
        Vi[i] = A*cos(angular_freq*t);
        printToFile(fp1, i, Vi[i]);
    };

    t=0;
    for(i=0;i<20000;i++)
    {
        Vo[i] = -A*angular_freq*R*C*deriveTrig(cos, angular_freq*t);
        printToFile(fp2, i, Vo[i]);
        printToFile(fp3, i, (Vo[i]/R));
        t=t+dt;
    };

    fclose(fp1);
    fclose(fp2);
    fclose(fp3);

    return 0;
};

```

Implementasi diff_triangular.c

```
//DIFFERENTIATOR SIMULATOR FOR TRIANGULAR WAVES INPUT
/*
    Program akan menerima input berupa command line argument yang
    diinvoke oleh file top_simulator.py
    Output program adalah file input_voltage_triangular.csv,
    output_voltage_triangular.csv, dan output_current_triangular.csv
*/
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define pi 3.14159265359

double T,A, R, C, dt=0.001;

double derive(double (*f)(double t, double T, double A), double x0,
double R, double C)
{
    const double delta = 1.0e-10; // or similar
    double x1 = x0;
    double x2 = x0 + delta;
    double y1 = f(x1,T,A);
    double y2 = f(x2,T,A);
    return -R*C*((y2 - y1) / delta);
};

double triangluarFunc(double t, double T, double A)           //triangle
{
    int region;
    region=t/(T/2);

    if(t>=0 && t<=(T/2))
        return ((2*A/T)*t);
    else if(t>(T/2) && t<=T)
        return ((2*A/T)*(-(t-T)));
    else
    {
        if(region%2==0)
            return((2*A/T)*(t-(T/2)*region));
        else if(region%2==1)
            return (-(2*A/T)*(t-(T/2)*region) + A);
    };
};

void printToFile(FILE *fp, int x, double y)
{
    fprintf(fp, "%f,", (x*dt));
    fprintf(fp, "%f\n", y);
};

int main(int argc, char *argv[])
{
    int i;
    double Vi[20000], Vo[20000], t=0;

    char *eptr;
    A = strtod(argv[1], &eptr);
    T = strtod(argv[2], &eptr);
    R = strtod(argv[3], &eptr);
    C = strtod(argv[4], &eptr);
```

```

FILE *fp1, *fp2, *fp3;
fp1 = fopen("input_voltage_triangular.csv", "w+");
fp2 = fopen("output_voltage_triangular.csv", "w+");
fp3 = fopen("output_current_triangular.csv", "w+");

Vi[0]=0;
fprintf(fp1, "%f,", 0);
fprintf(fp1, "%f\n", Vi[0]);

for(i=1;i<20000;i++)
{
    t=t+dt;
    Vi[i] = triangluarFunc(t, T,A);
    printToFile(fp1, i, Vi[i]);
}

t=0;
for(i=0;i<20000;i++)
{
    Vo[i] = derive(triangluarFunc, t, R, C);
    printToFile(fp2, i, Vo[i]);
    printToFile(fp3, i, (Vo[i]/R));
    t=t+dt;
}

fclose(fp1);
fclose(fp2);
fclose(fp3);

return 0;
}

```

Implementasi diff_square.c

```

//DIFFERENTIATOR SIMULATOR FOR SQUARE WAVES INPUT
/*
    Program akan menerima input berupa command line argument yang
diinvoke oleh file top_simulator.py
    Output program adalah file input_voltage_square.csv,
output_voltage_square.csv, dan output_current_square.csv
*/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define pi 3.14159265359

double T,A, R, C, dt=0.001;

double derive(double (*f)(double t, double A), double x0,
double R, double C)
{
    const double delta = 1.0e-10; // or similar
    double x1 = x0;
    double x2 = x0 + delta;
    double y1 = f(x1,T,A);
    double y2 = f(x2,T,A);
    return -R*C*((y2 - y1) / delta);
}

```

```

double squareFunc(double t, double T, double A)           //kotak
{
    int region;
    region=t/(T/2);

    if(t>=0 && t<=(T/2))
        return A;
    else if(t>(T/2) && t<=T)
        return -A;
    else
    {
        if(region%2==0)
            return A;
        else if(region%2==1)
            return -A;
    };
};

void printToFile(FILE *fp, int x, double y)
{
    fprintf(fp, "%f,", (x*dt));
    fprintf(fp, "%f\n", y);
};

int main(int argc, char *argv[])
{
    int i;
    double Vi[20000], Vo[20000], t=0;

    char *eptr;
    A = strtod(argv[1], &eptr);
    T = strtod(argv[2], &eptr);
    R = strtod(argv[3], &eptr);
    C = strtod(argv[4], &eptr);

    FILE *fp1, *fp2, *fp3;
    fp1 = fopen("input_voltage_square.csv", "w+");
    fp2 = fopen("output_voltage_square.csv", "w+");
    fp3 = fopen("output_current_square.csv", "w+");

    Vi[0]=A;
    fprintf(fp1, "%f,", 0);
    fprintf(fp1, "%f\n", Vi[0]);

    for(i=1;i<20000;i++)
    {
        t=t+dt;
        Vi[i] = squareFunc(t, T,A);
        printToFile(fp1, i, Vi[i]);
    };

    t=0;
    for(i=0;i<20000;i++)
    {
        Vo[i] = derive(squareFunc, t, R, C);
        printToFile(fp2, i, Vo[i]);
        printToFile(fp3, i, (Vo[i]/R));
        t=t+dt;
    };
}

```

```

fclose(fp1);
fclose(fp2);
fclose(fp3);

return 0;
}

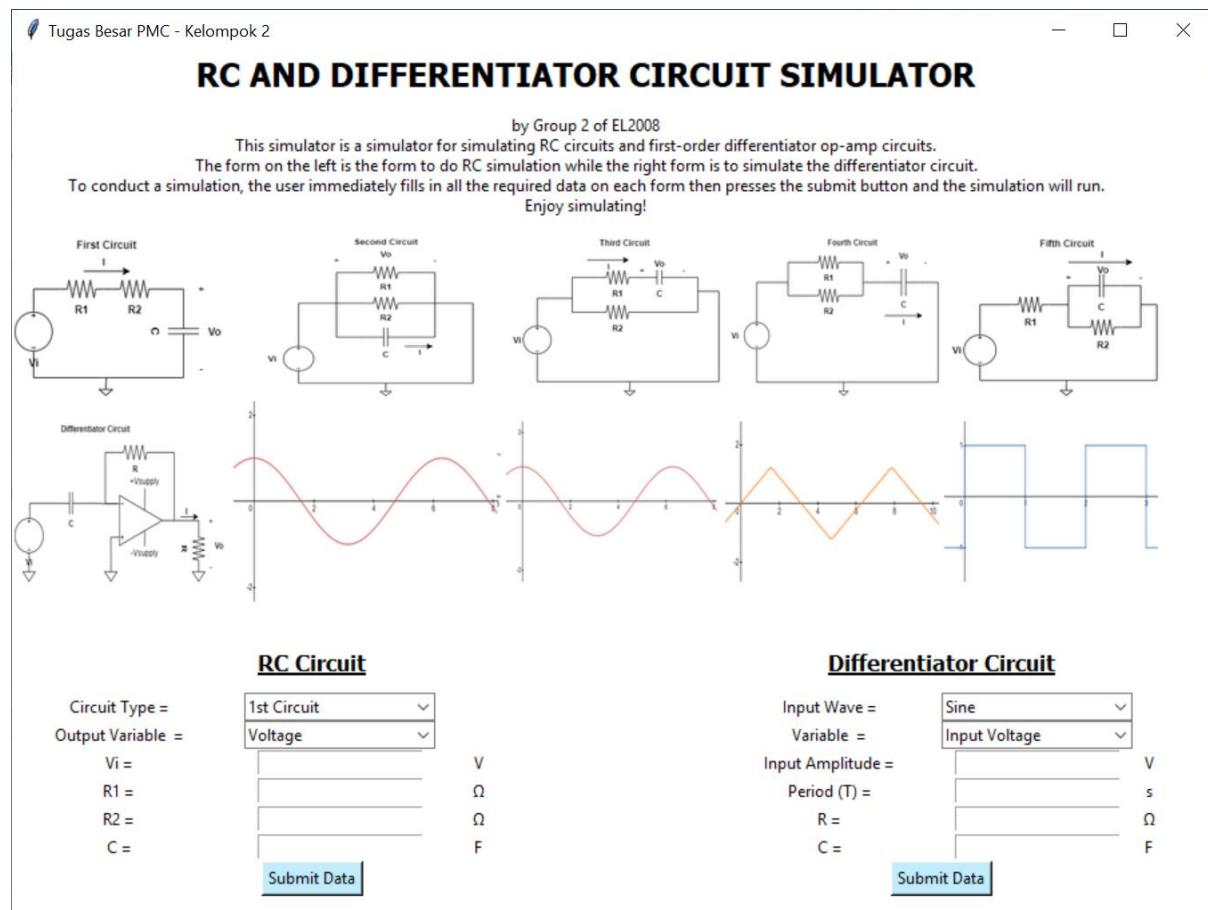
```

7. PENGUJIAN KESELURUHAN

Setelah melakukan *unit test* dan *functional test*, akan dilakukan pengujian secara menyeluruh untuk simulator yang telah dirancang oleh kelompok. Untuk mempermudah pengujian akan dilakukan secara berurutan. Pertama, akan dilakukan pengujian untuk melihat apakah semua fitur pada GUI telah muncul dengan baik, kedua adalah untuk melihat apakah semua rangkaian RC dapat disimulasikan dengan baik, dan ketiga adalah untuk menguji rangkaian differentiator.

7.1 GRAPHICAL USER INTERFACE

Berikut ini terdapat hasil pengujian *graphical interface*:



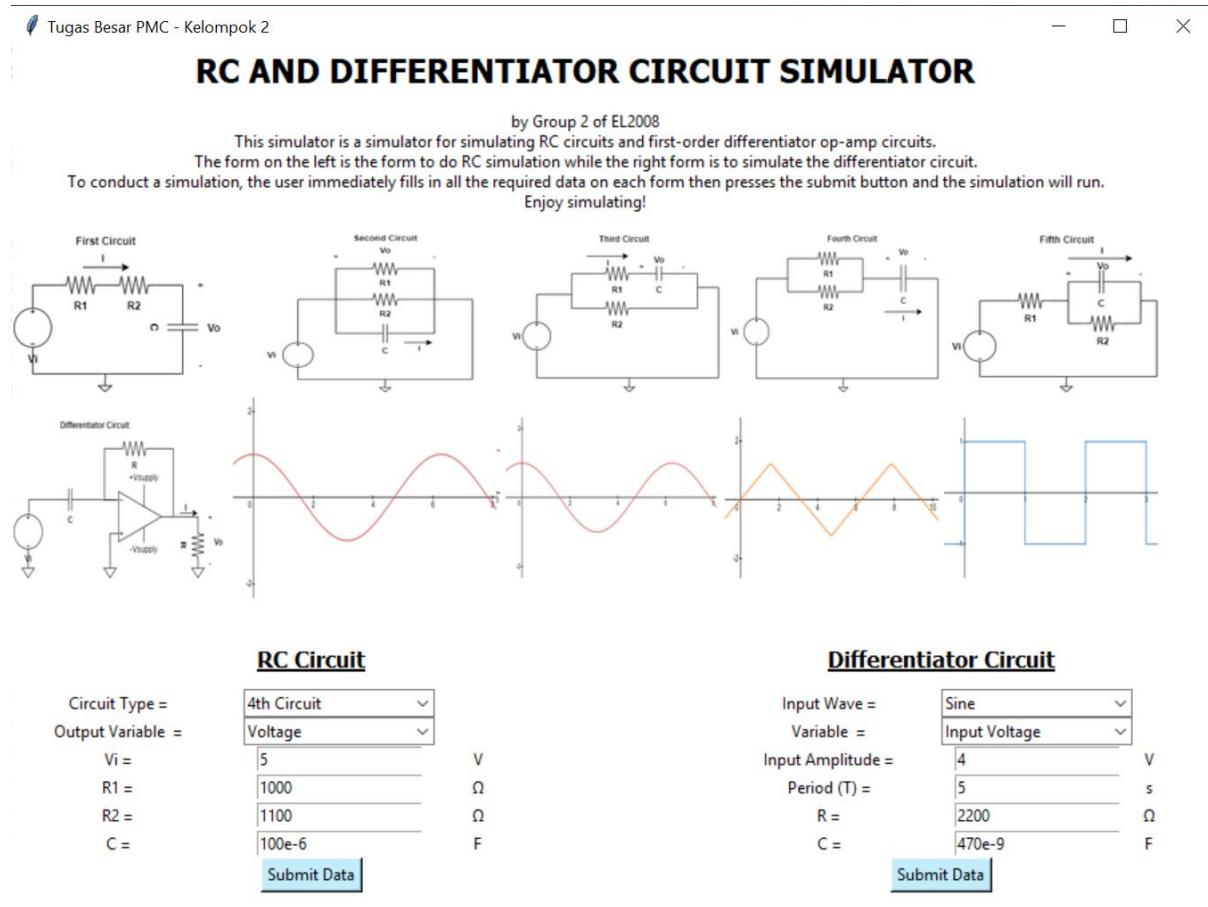
Gambar 7.1.1 Tampilan Graphical User Interface

Tabel 7.1 Tabel Tampilan Combobox pada GUI

Keterangan	Tampilan
<u>Combobox RC 1</u> Untuk memilih jenis rangkaian	<p style="text-align: center;">RC Circuit</p> <p>Circuit Type = <input type="button" value="1st Circuit"/> <input type="button" value="2nd Circuit"/> <input type="button" value="3rd Circuit"/> <input type="button" value="4th Circuit"/> <input type="button" value="5th Circuit"/></p> <p>Output Variable = <input type="button" value="Vi"/> <input type="button" value="R1"/> <input type="button" value="R2"/> <input type="button" value="C"/></p> <p>Vi = <input type="text"/> R1 = <input type="text"/> R2 = <input type="text"/> C = <input type="text"/></p> <p><input type="button" value="Submit Data"/></p>
	Gambar 7.1.2 Tampilan Combobox RC 1
<u>Combobox RC 2</u> Untuk memilih variabel output yang ditampilkan	<p style="text-align: center;">RC Circuit</p> <p>Circuit Type = <input type="button" value="1st Circuit"/> <input type="button" value="Voltage"/> <input type="button" value="Current"/></p> <p>Output Variable = <input type="button" value="Voltage"/> <input type="button" value="Current"/></p> <p>Vi = <input type="text"/> R1 = <input type="text"/> R2 = <input type="text"/> C = <input type="text"/></p> <p><input type="button" value="Submit Data"/></p>
	Gambar 7.1.3 Tampilan Combobox RC 2
<u>Combobox Differentiator 1</u> Untuk memilih jenis gelombang input	<p style="text-align: center;">Differentiator Circuit</p> <p>Input Wave = <input type="button" value="Sine"/> <input type="button" value="Cosine"/> <input type="button" value="Triangle"/> <input type="button" value="Square"/></p> <p>Variable = <input type="text"/> Input Amplitude = <input type="text"/> Period (T) = <input type="text"/> R = <input type="text"/> C = <input type="text"/></p> <p><input type="button" value="Submit Data"/></p>
	Gambar 7.1.4 Tampilan Combobox Differentiator 1
<u>Combobox Differentiator 2</u> Untuk memilih jenis variabel yang akan ditampilkan	<p style="text-align: center;">Differentiator Circuit</p> <p>Input Wave = <input type="button" value="Sine"/> <input type="button" value="Input Voltage"/> <input type="button" value="Output Voltage"/> <input type="button" value="Output Current"/></p> <p>Variable = <input type="text"/> Input Amplitude = <input type="text"/> Period (T) = <input type="text"/> R = <input type="text"/> C = <input type="text"/></p> <p><input type="button" value="Submit Data"/></p>
	Gambar 7.1.5 Tampilan Combobox Differentiator 2

7.2 PENGUJIAN RANGKAIAN RC

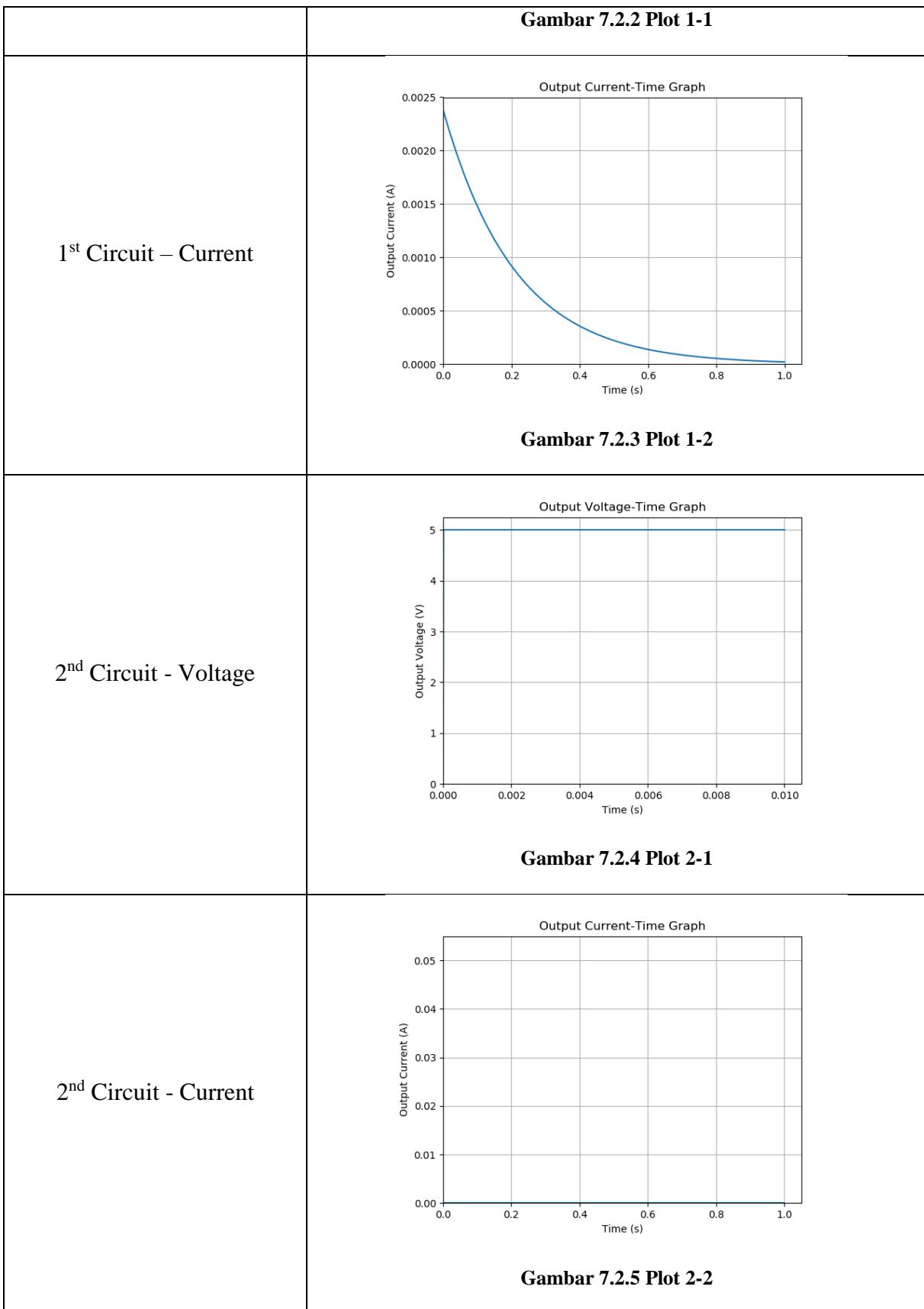
Untuk menguji jalannya simulasi rangkaian RC, akan digunakan nilai komponen yang sama untuk setiap komponen yaitu $V_i=5V$, $R_1=1000$, $R_2=1100$, $C=100\mu F$. Berikut ini terdapat hasil pengujian yang telah dilakukan:



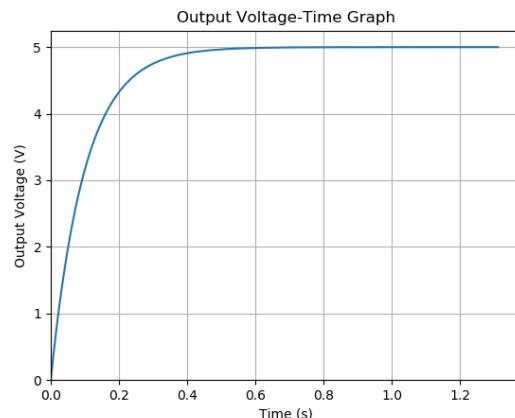
Gambar 7.2.1 Tampilan GUI Pengujian Rangkaian RC

Tabel 7.2 Tabel Pengujian Rangkaian RC

Keterangan Pilihan Combobox 1-2	Tampilan Plot
1 st Circuit - Voltage	<p style="text-align: center;">Output Voltage-Time Graph</p>

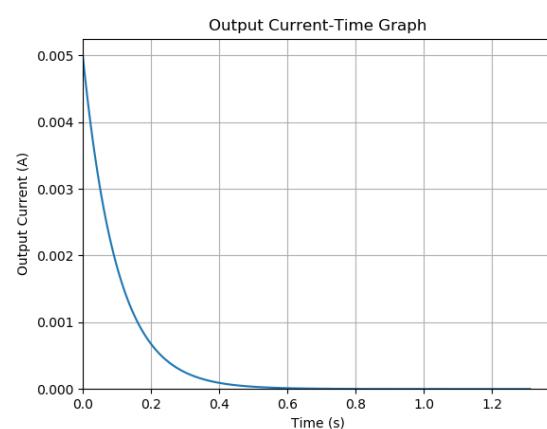


3rd Circuit - Voltage



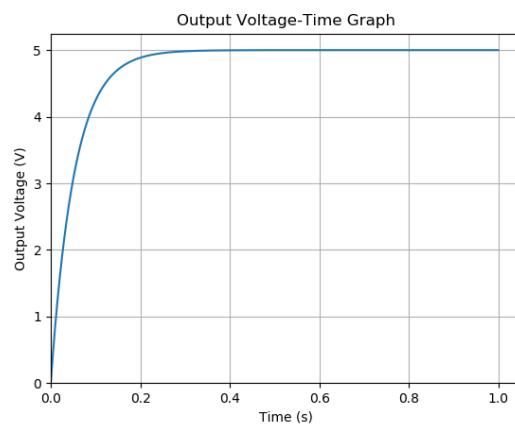
Gambar 7.2.6 Plot 3-1

3rd Circuit – Current



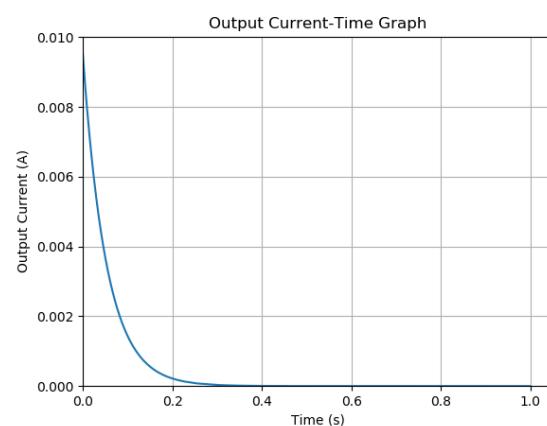
Gambar 7.2.7 Plot 3-2

4th Circuit - Voltage



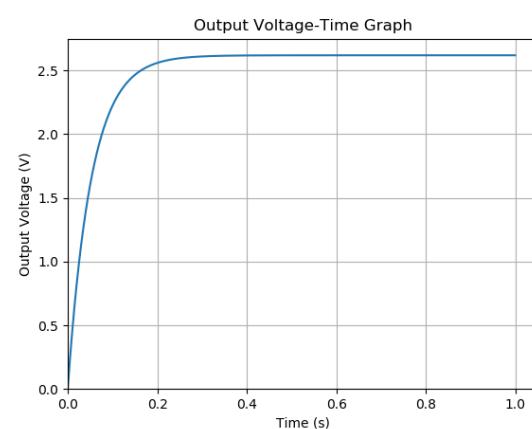
Gambar 7.2.8 Plot 4-1

4th Circuit - Current



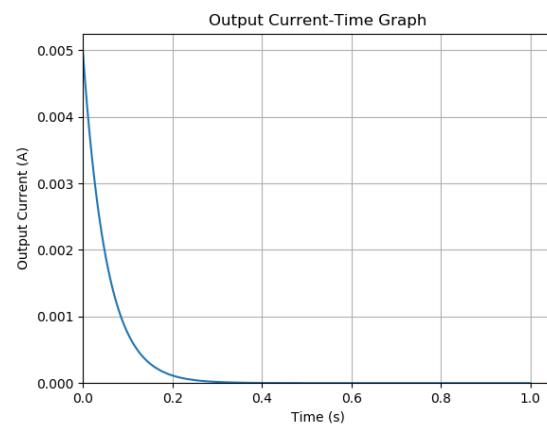
Gambar 7.2.9 Plot 4-2

5th Circuit - Voltage



Gambar 7.2.10 Plot 5-1

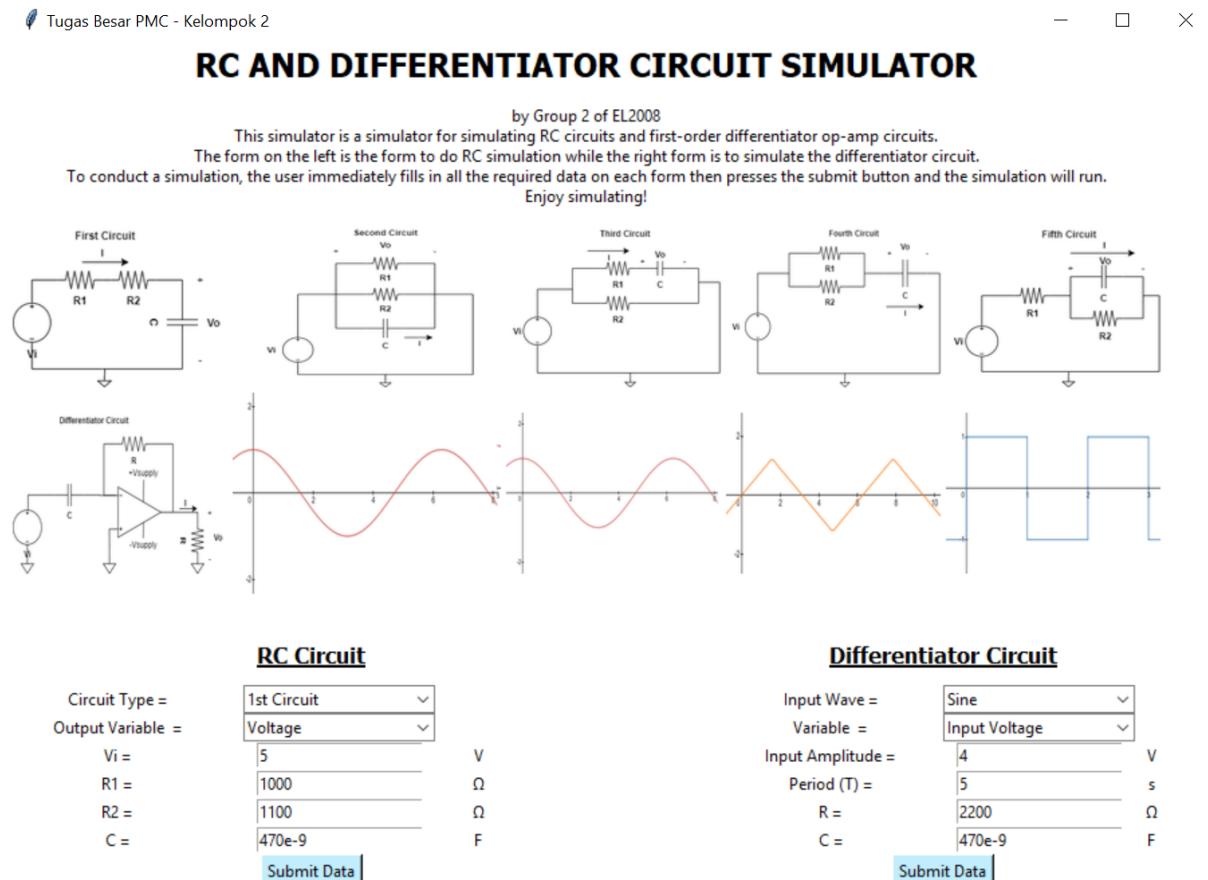
5th Circuit - Current



Gambar 7.2.11 Plot 5-2

7.3 PENGUJIAN RANGKAIAN DIFFERENTIATOR

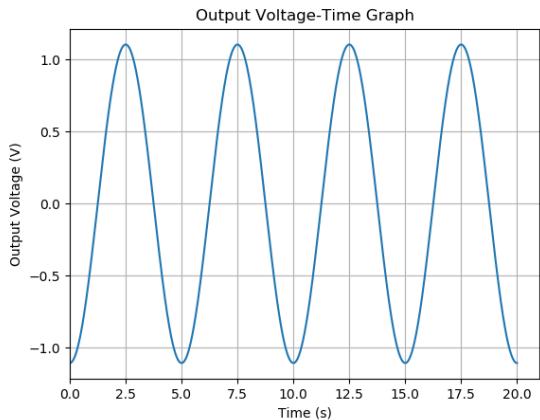
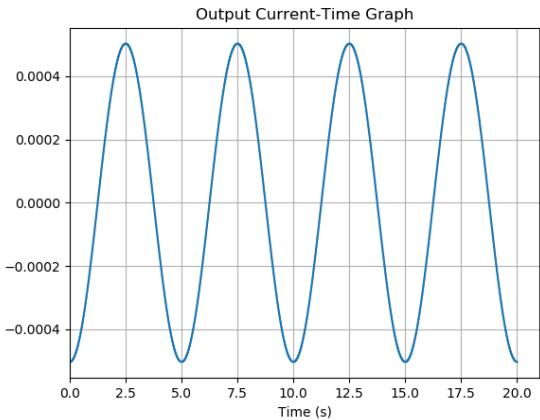
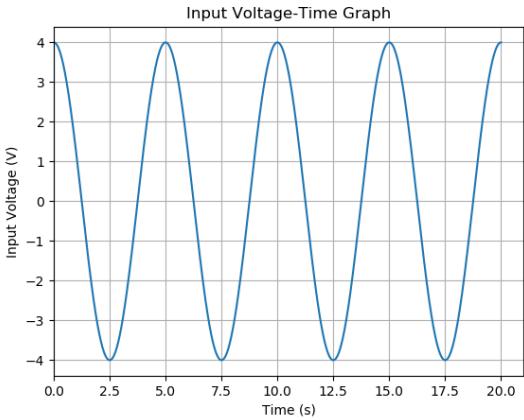
Untuk menguji jalannya simulasi rangkaian Differentiator, akan digunakan nilai komponen yang sama untuk setiap komponen yaitu $A=4V$, $T=5s$ $R2=2200$, $C=100\mu F$. Berikut ini terdapat hasil pengujian yang telah dilakukan:



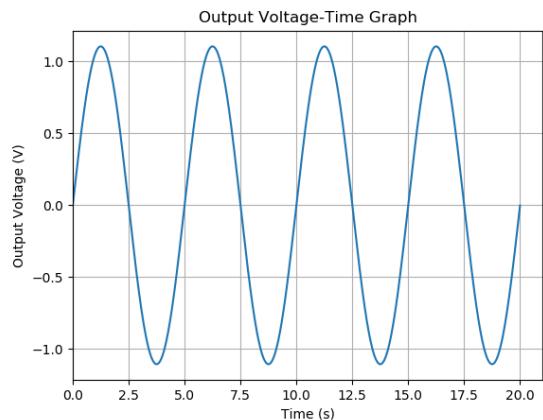
Gambar 7.3.1 Tampilan GUI Pengujian Rangkaian Differentiator

Tabel 7.3 Tabel Pengujian Rangkaian Differentiator

Keterangan Pilihan Combobox 1-2	Tampilan Plot
Sine – Input Voltage	<p style="text-align: center;">Input Voltage-Time Graph</p>

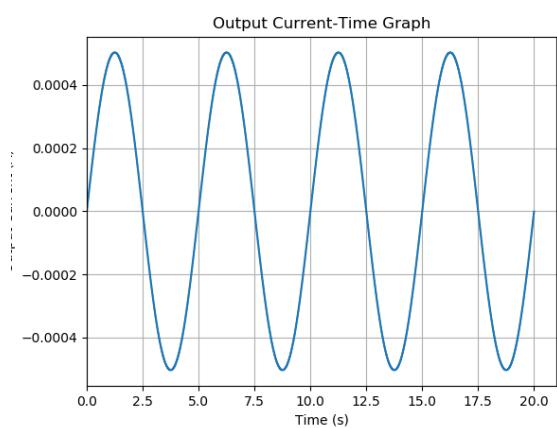
	Gambar 7.3.2 Plot 1-1																				
Sine – Output Voltage	 <p style="text-align: center;">Output Voltage-Time Graph</p> <p>The graph shows a sine wave oscillating between -1.0 and 1.0 V over a 20-second period. The x-axis is labeled "Time (s)" and ranges from 0.0 to 20.0 with major grid lines every 2.5 units. The y-axis is labeled "Output Voltage (V)" and ranges from -1.0 to 1.0 with major grid lines every 0.5 units.</p> <table border="1"> <caption>Data for Output Voltage-Time Graph</caption> <thead> <tr> <th>Time (s)</th> <th>Output Voltage (V)</th> </tr> </thead> <tbody> <tr><td>0.0</td><td>-1.0</td></tr> <tr><td>2.5</td><td>1.0</td></tr> <tr><td>5.0</td><td>-1.0</td></tr> <tr><td>7.5</td><td>1.0</td></tr> <tr><td>10.0</td><td>-1.0</td></tr> <tr><td>12.5</td><td>1.0</td></tr> <tr><td>15.0</td><td>-1.0</td></tr> <tr><td>17.5</td><td>1.0</td></tr> <tr><td>20.0</td><td>-1.0</td></tr> </tbody> </table>	Time (s)	Output Voltage (V)	0.0	-1.0	2.5	1.0	5.0	-1.0	7.5	1.0	10.0	-1.0	12.5	1.0	15.0	-1.0	17.5	1.0	20.0	-1.0
Time (s)	Output Voltage (V)																				
0.0	-1.0																				
2.5	1.0																				
5.0	-1.0																				
7.5	1.0																				
10.0	-1.0																				
12.5	1.0																				
15.0	-1.0																				
17.5	1.0																				
20.0	-1.0																				
Sine – Output Current	 <p style="text-align: center;">Output Current-Time Graph</p> <p>The graph shows a sine wave oscillating between -0.0004 and 0.0004 A over a 20-second period. The x-axis is labeled "Time (s)" and ranges from 0.0 to 20.0 with major grid lines every 2.5 units. The y-axis is labeled "Output Current (A)" and ranges from -0.0004 to 0.0004 with major grid lines every 0.0002 units.</p> <table border="1"> <caption>Data for Output Current-Time Graph</caption> <thead> <tr> <th>Time (s)</th> <th>Output Current (A)</th> </tr> </thead> <tbody> <tr><td>0.0</td><td>-0.0004</td></tr> <tr><td>2.5</td><td>0.0004</td></tr> <tr><td>5.0</td><td>-0.0004</td></tr> <tr><td>7.5</td><td>0.0004</td></tr> <tr><td>10.0</td><td>-0.0004</td></tr> <tr><td>12.5</td><td>0.0004</td></tr> <tr><td>15.0</td><td>-0.0004</td></tr> <tr><td>17.5</td><td>0.0004</td></tr> <tr><td>20.0</td><td>-0.0004</td></tr> </tbody> </table>	Time (s)	Output Current (A)	0.0	-0.0004	2.5	0.0004	5.0	-0.0004	7.5	0.0004	10.0	-0.0004	12.5	0.0004	15.0	-0.0004	17.5	0.0004	20.0	-0.0004
Time (s)	Output Current (A)																				
0.0	-0.0004																				
2.5	0.0004																				
5.0	-0.0004																				
7.5	0.0004																				
10.0	-0.0004																				
12.5	0.0004																				
15.0	-0.0004																				
17.5	0.0004																				
20.0	-0.0004																				
Cosine – Input Voltage	 <p style="text-align: center;">Input Voltage-Time Graph</p> <p>The graph shows a cosine wave oscillating between -4 and 4 V over a 20-second period. The x-axis is labeled "Time (s)" and ranges from 0.0 to 20.0 with major grid lines every 2.5 units. The y-axis is labeled "Input Voltage (V)" and ranges from -4 to 4 with major grid lines every 1 unit.</p> <table border="1"> <caption>Data for Input Voltage-Time Graph</caption> <thead> <tr> <th>Time (s)</th> <th>Input Voltage (V)</th> </tr> </thead> <tbody> <tr><td>0.0</td><td>4.0</td></tr> <tr><td>2.5</td><td>-4.0</td></tr> <tr><td>5.0</td><td>4.0</td></tr> <tr><td>7.5</td><td>-4.0</td></tr> <tr><td>10.0</td><td>4.0</td></tr> <tr><td>12.5</td><td>-4.0</td></tr> <tr><td>15.0</td><td>4.0</td></tr> <tr><td>17.5</td><td>-4.0</td></tr> <tr><td>20.0</td><td>4.0</td></tr> </tbody> </table>	Time (s)	Input Voltage (V)	0.0	4.0	2.5	-4.0	5.0	4.0	7.5	-4.0	10.0	4.0	12.5	-4.0	15.0	4.0	17.5	-4.0	20.0	4.0
Time (s)	Input Voltage (V)																				
0.0	4.0																				
2.5	-4.0																				
5.0	4.0																				
7.5	-4.0																				
10.0	4.0																				
12.5	-4.0																				
15.0	4.0																				
17.5	-4.0																				
20.0	4.0																				
	Gambar 7.3.5 Plot 2-1																				

Cosine – Output Voltage



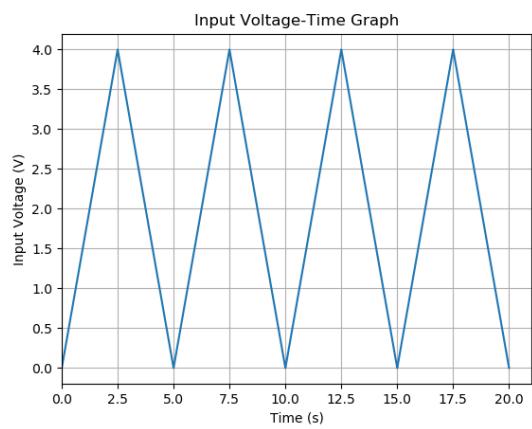
Gambar 7.3.6 Plot 2-2

Cosine – Output Current



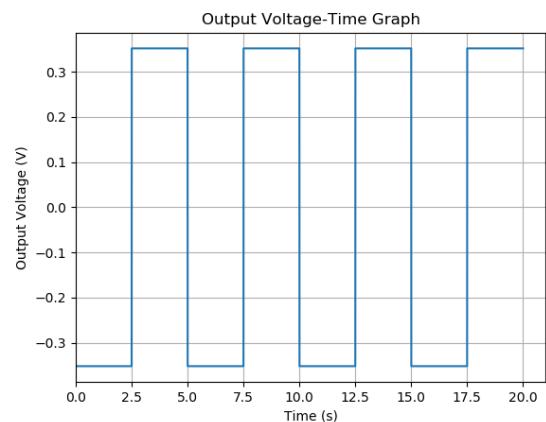
Gambar 7.3.7 Plot 2-3

Triangle – Input Voltage



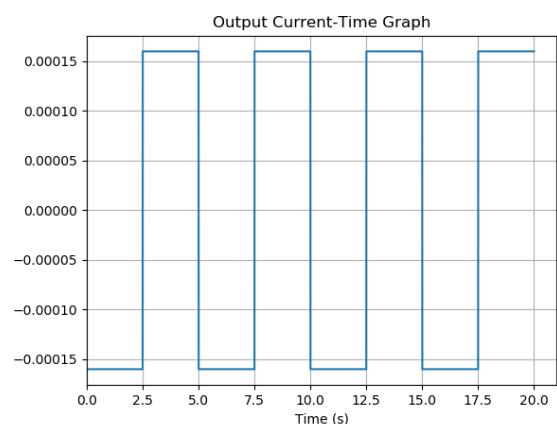
Gambar 7.3.8 Plot 3-1

Triangle – Output Voltage



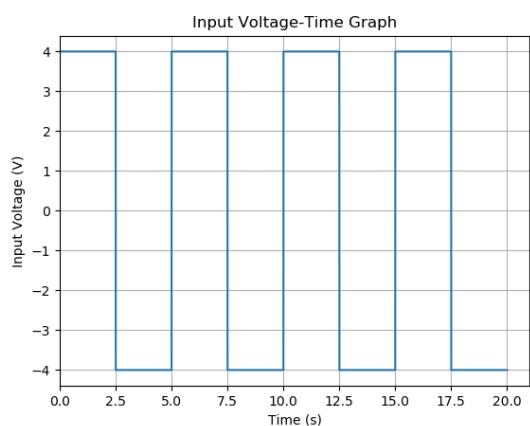
Gambar 7.3.9 Plot 3-2

Triangle – Output Current



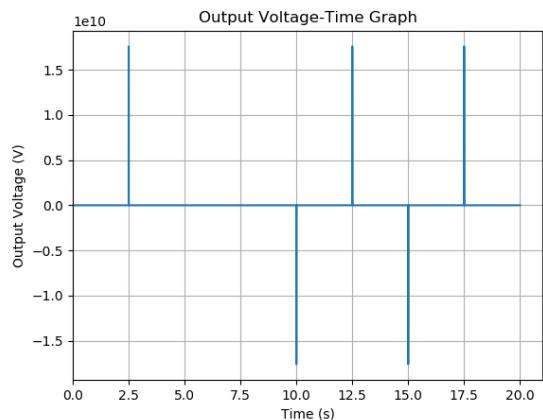
Gambar 7.3.10 Plot 3-3

Square – Input Voltage



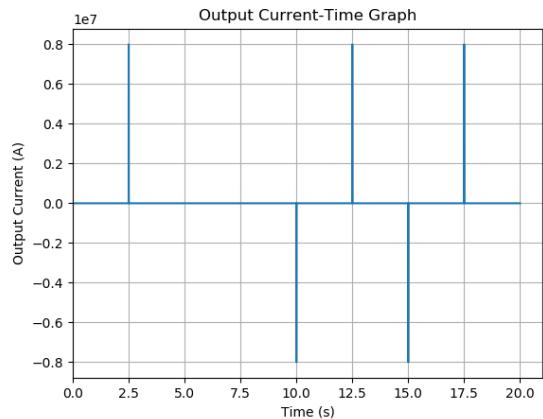
Gambar 7.3.11 Plot 4-1

Square – Output Voltage



Gambar 7.3.12 Plot 4-2

Square – Output Current



Gambar 7.3.13 Plot 4-3

8. ANALISIS PENGUJIAN

Berdasarkan pengujian secara keseluruhan yang telah dilakukan pada bagian 7, akan dilakukan analisis secara lebih detail. Analisis pada bagian ini dibagi menjadi 3 buah bagian yaitu analisis fitur GUI, analisis ketepatan simulasi rangkaian RC, dan *differentiator*.

8.1 ANALISIS GRAPHICAL USER INTERFACE

Berikut ini terdapat tampilan *graphical user interface* yang telah dirancang oleh kelompok:

RC AND DIFFERENTIATOR CIRCUIT SIMULATOR

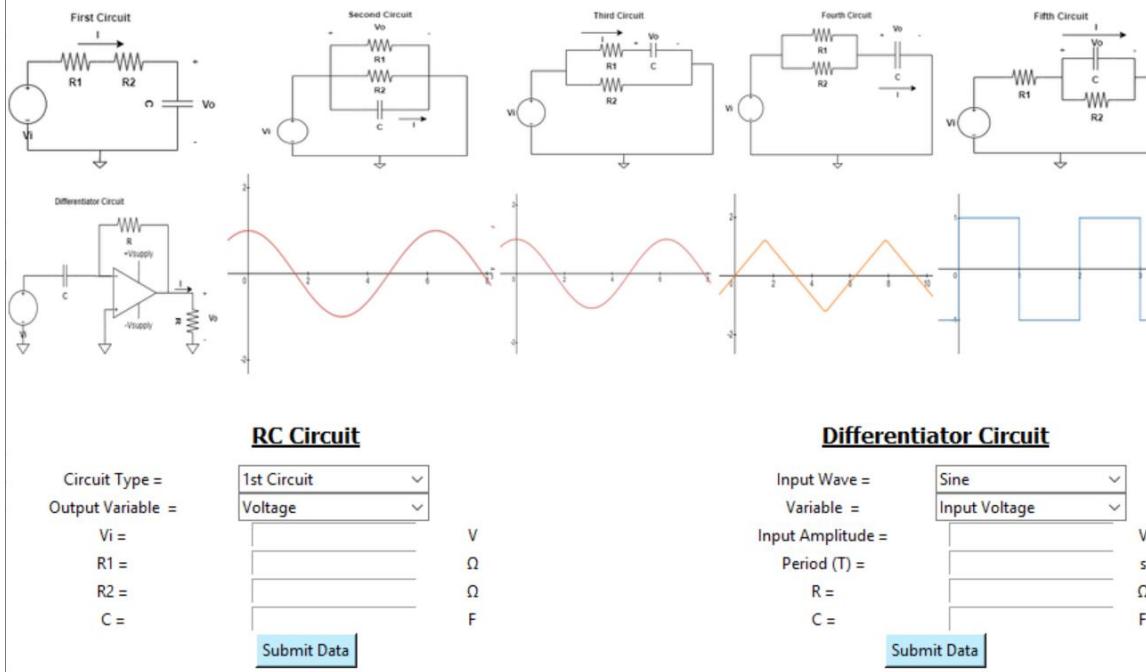
by Group 2 of EL2008

This simulator is a simulator for simulating RC circuits and first-order differentiator op-amp circuits.

The form on the left is the form to do RC simulation while the right form is to simulate the differentiator circuit.

To conduct a simulation, the user immediately fills in all the required data on each form then presses the submit button and the simulation will run.

Enjoy simulating!



Gambar 8.1.1 Tampilan Graphical User Interface

Secara umum, dalam membuat GUI tersebut, digunakan bahasa Python 3.7 dan library tkinter. Library tersebut dipilih untuk digunakan karena memudahkan kelompok untuk melakukan perancangan GUI. Pada GUI tersebut, masing-masing komponen (misalkan gambar, *button*, *entry box*) di atur penempatannya dengan menggunakan fungsi dari library tersebut yaitu *grid*. Pada bagian atas GUI, terdapat judul simulator beserta keterangan cara melakukan simulasi pada simulator ini. Di bawahnya, terdapat enam jenis rangkaian (lima jenis rangkaian RC dan sebuah differentiator), serta terdapat empat jenis gelombang input yang dapat dipilih sebagai sinyal tegangan input pada rangkaian differentiator. Di bagian bawah GUI, terdapat dua buah *form*, form pada sebelah kiri merupakan form untuk melakukan simulasi rangkaian RC sedangkan form kanan merupakan form yang diisi jika *user* ingin melakukan simulasi rangkaian differentiator. Jika *user* telah selesai mengisi nilai komponen dan keterangan yang terdapat pada form, simulasi dapat dijalankan dengan menekan *button* bertuliskan “Submit Data.”

RC Circuit

Circuit Type =

Output Variable =

$V_i =$ V

$R_1 =$ Ω

$R_2 =$ Ω

$C =$ F

Gambar 8.1.2 Form Simulasi Rangkaian RC

Differentiator Circuit

Input Wave =	<input type="button" value="Sine"/>
Variable =	<input type="button" value="Input Voltage"/>
Input Amplitude =	<input style="width: 50px; height: 25px; border: 1px solid black;" type="text"/> V
Period (T) =	<input style="width: 50px; height: 25px; border: 1px solid black;" type="text"/> s
R =	<input style="width: 50px; height: 25px; border: 1px solid black;" type="text"/> Ω
C =	<input style="width: 50px; height: 25px; border: 1px solid black;" type="text"/> F
<input type="button" value="Submit Data"/>	

Gambar 8.1.3 Form Simulasi Rangkaian Differentiator

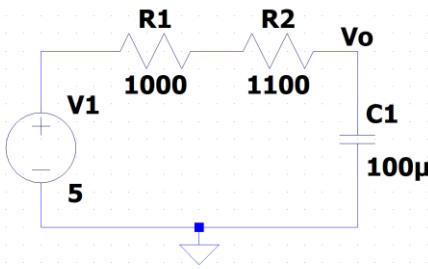
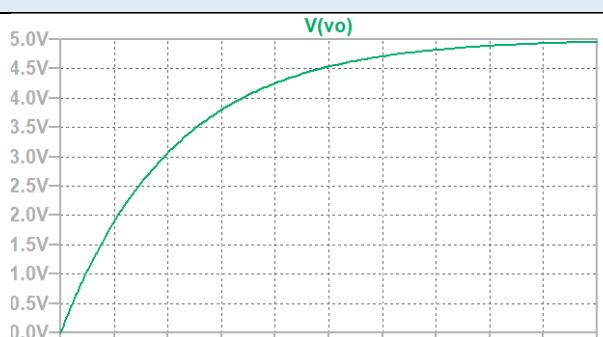
Pada gambar 8.1.2, terdapat bagian dari GUI berupa form yang diisi jika *user* ingin melakukan simulasi rangkaian RC. Terdapat 5 buah data yang akan diinput oleh *user* pada 5 buah widget yang terdapat pada form tersebut. Terdapat dua buah *widget* berupa *combobox* untuk menginput tipe rangkaian dan variabel output yang akan disimulasikan serta 3 buah *entrybox* untuk menginput V_i , R_1 , R_2 , dan C . Pilihan pada *combobox* terdapat pada bagian 7.1.

Analog dengan form simulator rangkaian RC, gambar 8.1.3 merupakan form yang digunakan *user* untuk menginput data yang diperlukan untuk melakukan simulasi rangkaian *differentiator*. Terdapat dua buah *combobox* yang berfungsi untuk menginput jenis sinyal input dan variabel yang akan diamati. Terdapat empat buah *entrybox* yang digunakan untuk menginput amplitudo sinyal, perioda, nilai resistansi R , dan kapasitansi C . Sama seperti form rangkaian RC, *value* dari *combobox* terdapat pada bagian 7.1.

8.2 ANALISIS KETEPATAN SIMULASI RANGKAIAN RC

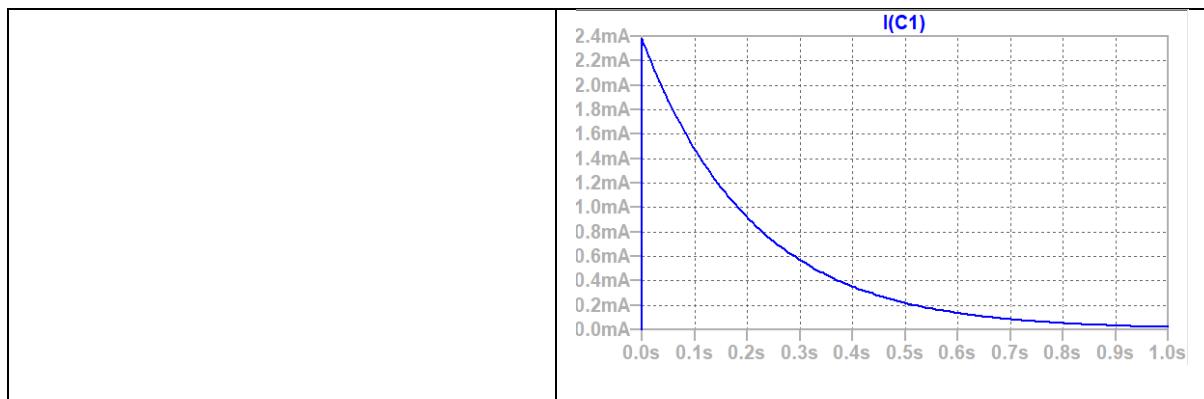
Pada bagian 8.2 ini, akan dilakukan dibandingkan hasil simulasi pada kelima jenis rangkaian RC yang dapat disimulasikan pada simulator yang telah dirancang dengan simulator referensi. Simulator referensi yang digunakan adalah LTspice XVII dan merupakan simulator rangkaian yang umum untuk digunakan. Untuk membandingkan hasil yang diperoleh oleh simulator yang telah dibuat dengan simulator referensi, akan digunakan nilai komponen yang sama dengan nilai yang digunakan pada simulator rancangan kelompok, yaitu $V_i=5V$, $R_1=1000\Omega$, $R_2=1100\Omega$, $C=100\mu F$. Simulasi yang dilakukan adalah analisis transien dengan *stop time* 1s. Berikut ini terdapat rangkaian dan hasil simulasi dengan menggunakan LTSpice:

Tabel 8.2 Referensi Simulasi dengan LTSpice

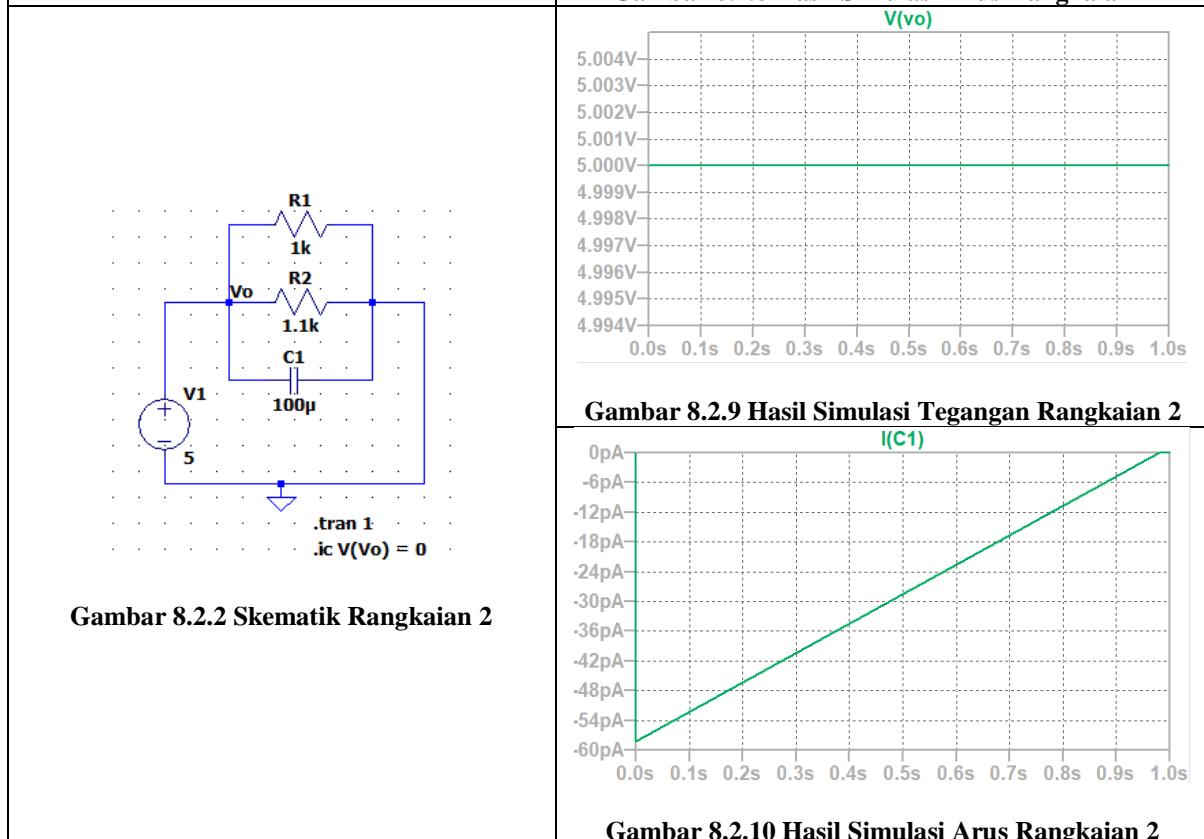
Skematik Rangkaian	Hasil Simulasi
 $.tran 1$ $.ic V(Vo) = 0$	

Gambar 8.2.1 Skematik Rangkaian 1

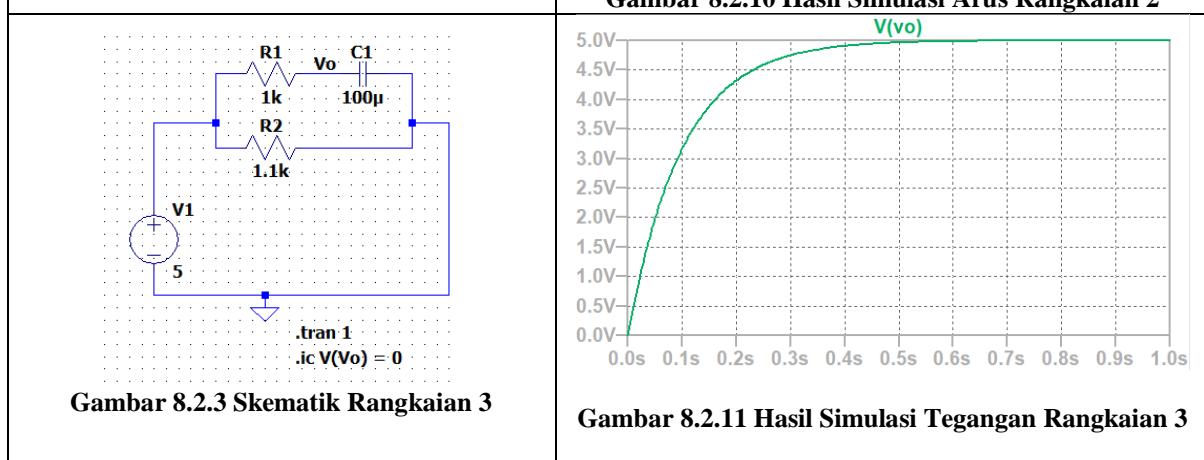
Gambar 8.2.6 Hasil Simulasi Tegangan Rangkaian 1



Gambar 8.2.8 Hasil Simulasi Arus Rangkaian 1

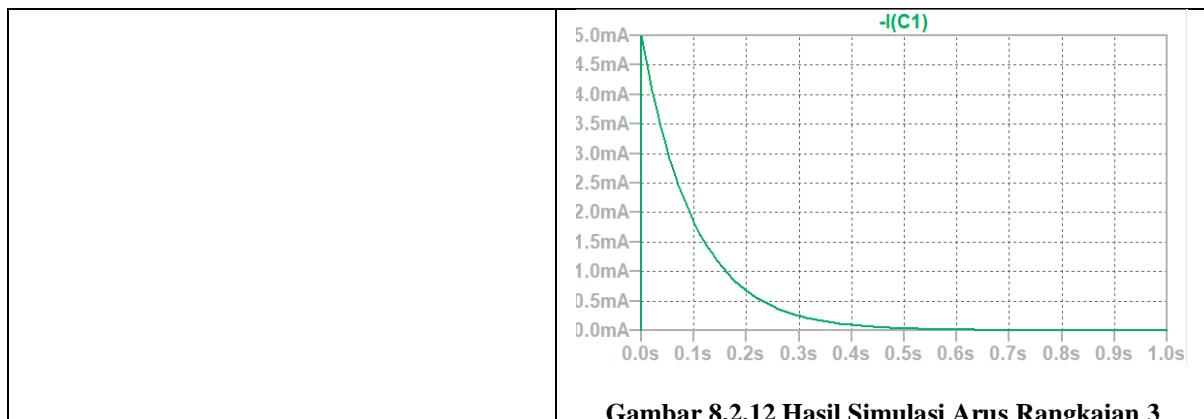


Gambar 8.2.9 Hasil Simulasi Tegangan Rangkaian 2

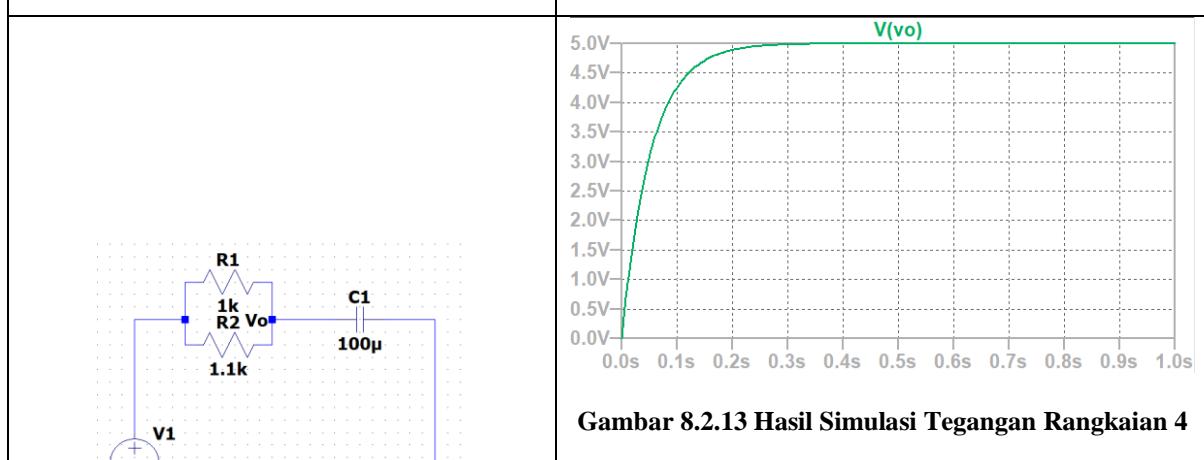


Gambar 8.2.10 Hasil Simulasi Arus Rangkaian 2

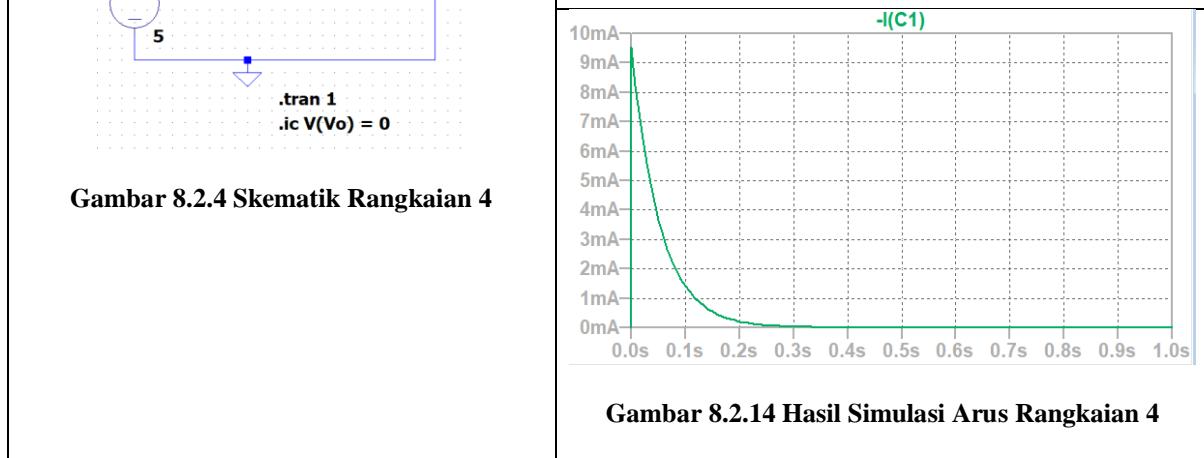
Gambar 8.2.11 Hasil Simulasi Tegangan Rangkaian 3



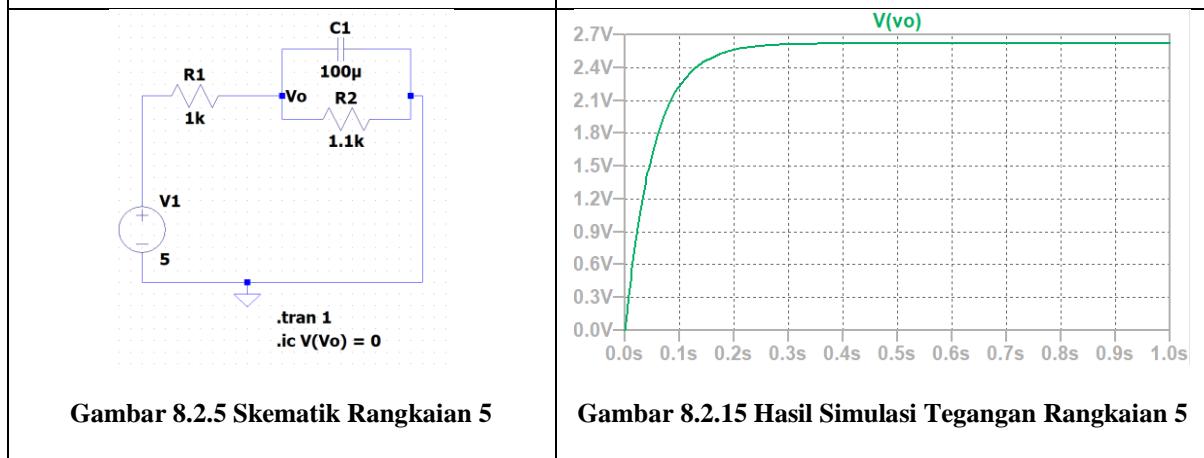
Gambar 8.2.12 Hasil Simulasi Arus Rangkaian 3



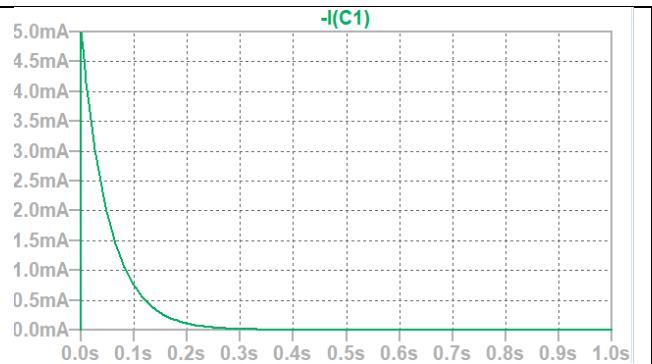
Gambar 8.2.13 Hasil Simulasi Tegangan Rangkaian 4



Gambar 8.2.14 Hasil Simulasi Arus Rangkaian 4



Gambar 8.2.15 Hasil Simulasi Tegangan Rangkaian 5



Gambar 8.2.16 Hasil Simulasi Arus Rangkaian 5

Pada tabel 8.2 di atas, terdapat hasil simulasi lima jenis rangkaian RC yang telah dirancang oleh kelompok. Bila dibandingkan hasil simulasi dengan menggunakan simulator yang dirancang oleh kelompok (tabel 7.2) dengan hasil yang didapat dari simulator referensi LTSpice XVII pada tabel 8.2, terlihat bahwa secara umum hasil yang diperoleh sesuai.

Untuk memverifikasi kebenaran hasil simulasi rangkaian secara praktis, akan dibandingkan hasil yang diperoleh pada hasil simulasi tegangan output untuk rangkaian 1 dan 4. Secara umum kedua rangkaian tersebut merupakan rangkaian yang serupa, yang berbeda dari kedua rangkaian tersebut hanya pada rangkaian 1, dua buah resistor disusun secara seri, sedangkan pada rangkaian 4, kedua buah resistor tersebut disusun secara pararel dan kemudian diserikan dengan sebuah kapasitor. Perlu diingat bahwa semua simulasi rangkaian RC menggunakan nilai komponen yang sama. Untuk memverifikasi hasil yang diperoleh dari simulasi yang dirancang, salah satu hal yang dapat ditinjau adalah dari segi konstanta waktu. Pada rangkaian RC, konstanta waktu didefinisikan sebagai waktu yang diperlukan suatu sinyal untuk berkurang atau bertambah nilai sebanyak 36,79% dari nilai. Secara matematis didekati dengan $\tau = RC$. Pada rangkaian 1 $\tau = (R_1 + R_2) * C$ sedangkan pada rangkaian 4, $\tau = (R_1 // R_2) * C$. Tentu dengan nilai R_1 , R_2 , dan C yang sama, terlihat bahwa konstanta waktu rangkaian 1 akan lebih besar daripada rangkaian 4, dan diperoleh bahwa tegangan kapasitor pada rangkaian 1 akan lebih lama untuk mencapai keadaan tunak daripada rangkaian 4, dengan tegangan tunak yang sama yaitu sebesar 5V. Hal ini, secara cepat menunjukkan bahwa simulasi rangkaian berjalan dengan baik dan sesuai dengan referensi.

Jika dilihat dari kurva arus-waktu secara umum, terlihat bahwa pada $t=0$ (awal simulasi), arus output rangkaian mempunyai nilai, namun pada akhir simulasi, nilai arus output pada rangkaian bernilai 0A. Hal ini dipahami karena kapasitor pada akhir simulasi dilakukan, berada pada keadaan tunak dan pada kondisi DC, saat kapasitor berada pada keadaan tunak, akan berlaku sebagai *open circuit* (rangkaian terbuka) dan hal tersebut ditunjukkan dari nilai arus pada *branch* kapasitor berada akan bernilai 0A.

Jika ditinjau rangkaian kedua, terlihat bahwa pada kasus ini, nilai tegangan output yang merupakan tegangan jepit pada kapasitor akan selalu bernilai sama dengan nilai tegangan input, sehingga pada semua waktu, nilai tegangannya akan sama dengan 5V. Jika dilihat hasil simulasi pada simulator referensi, terlihat bahwa arus output pada rangkaian 2 mempunyai nilai

sangat kecil pada waktu yang sangat singkat dan pada akhirnya akan bernilai nol. Hal ini dapat didekati dengan nilai arus yang bernilai nol untuk setiap waktu.

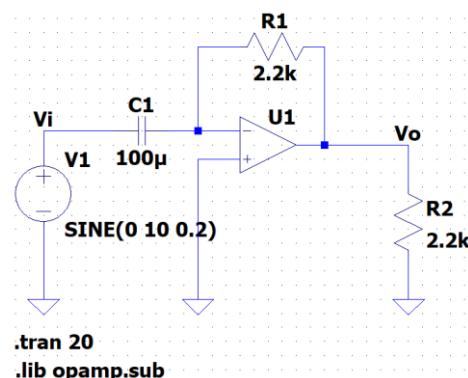
Pada rangkaian lima, terjadi suatu hal yang menarik. Pada hasil simulasi oleh simulator rancangan kelompok yang terdapat pada tabel 7.2, terlihat bahwa kurva tegangan output kapasitor akan berawal dari nilai 0V dan berakhir pada asimtot 2.62V. Hal ini terjadi karena kapasitor disusun secara pararel dengan resistor 2 yang diserikan dengan resistor 1. Pada awalnya, kapasitor telah ter-*discharge* dengan sempurna sehingga nilai tegangan bernilai nol dan pada kondisi tunak, nilai tegangan kapasitor akan bernilai sama dengan nilai tegangan pada R2 yaitu dengan prinsip pembagi tegangan:

$$V_o = \frac{1.1}{1 + 1.1} \times 5 = 2.62V$$

Kurva arus-waktu dari rangkaian akan mempunyai nilai pada awal simulasi berjalan dan akan mempunyai nilai akhir 0A yang dipahami bahwa pada saat kapasitor berada pada keadaan tunak (*fully charged*), kapasitor akan berlaku sebagai *open circuit*. Berdasarkan hasil analisis yang telah dilakukan, dapat disimpulkan bahwa simulator rangkaian RC yang dirancang oleh kelompok dapat berfungsi dengan baik.

8.3 ANALISIS KETEPATAN SIMULASI RANGKAIAN DIFFERENTIATOR

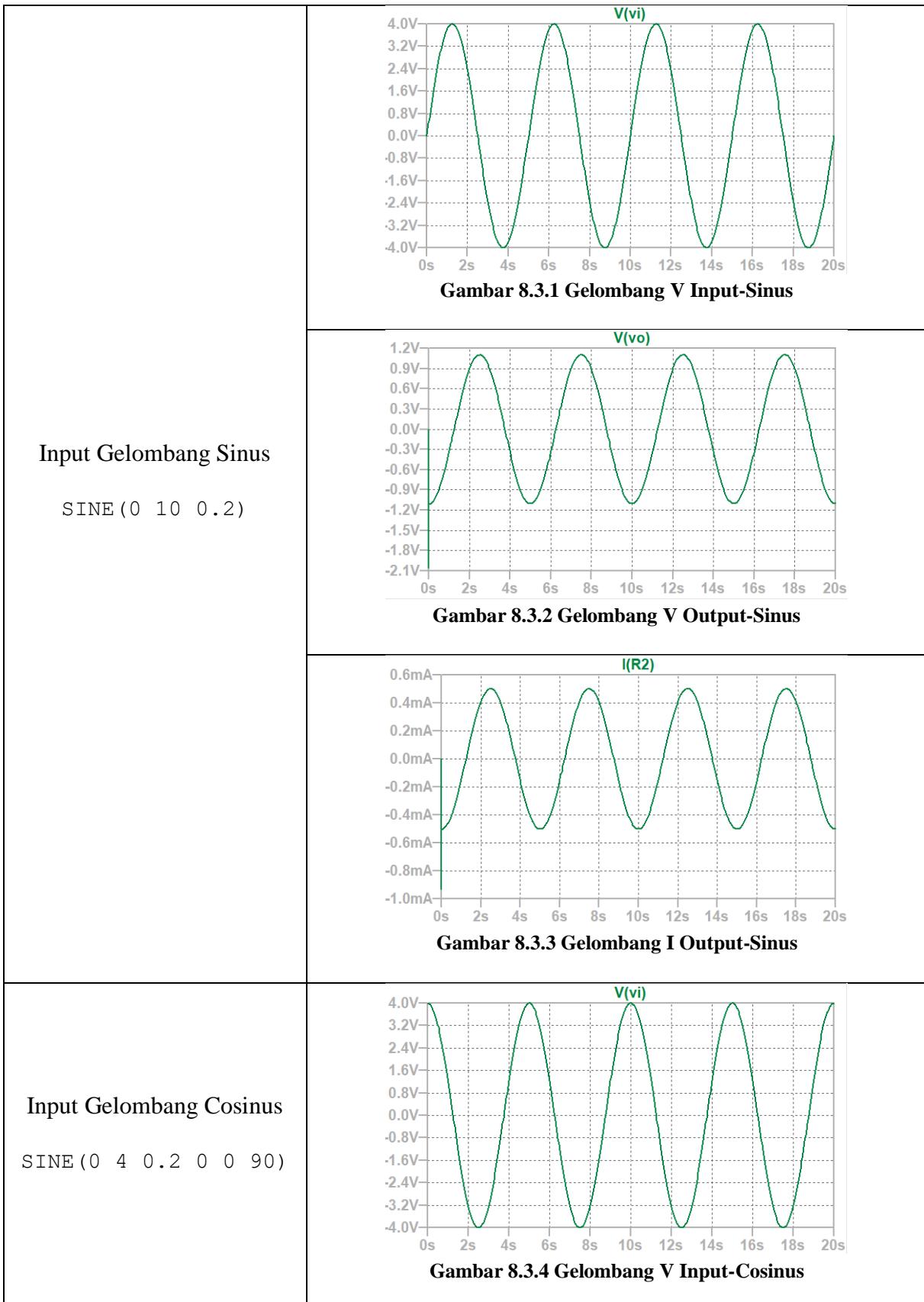
Pada bagian 8.3 ini, akan dibandingkan hasil simulasi rangkaian *differentiator* yang telah dirancang oleh kelompok dengan hasil simulasi referensi yang berasal dari simulator LTSpice XVII. Dengan metode yang sama dengan bagian 8.2, yaitu analisis ketepatan simulasi rangkaian RC, akan digunakan nilai komponen yang sama dengan komponen simulasi rangkaian kelompok, yaitu dengan $A=4V$, $T=5s$, $R=2200\Omega$, $C=100\mu F$ dan simulasi dilakukan selama 20s. Berikut ini terdapat skematik rangkaian yang digunakan untuk simulasi pada LTSpice XVII dan hasil yang diperoleh untuk setiap jenis gelombang input:

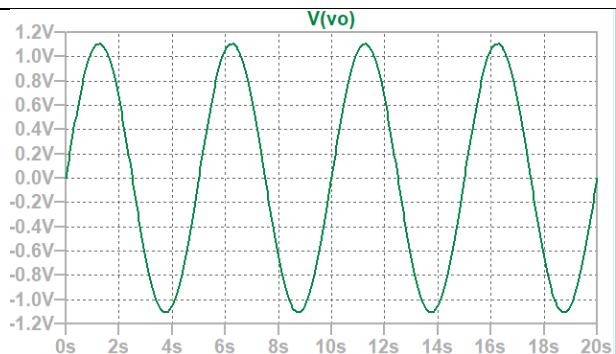


Gambar 8.3 Skematic Rangkaian Differentiator

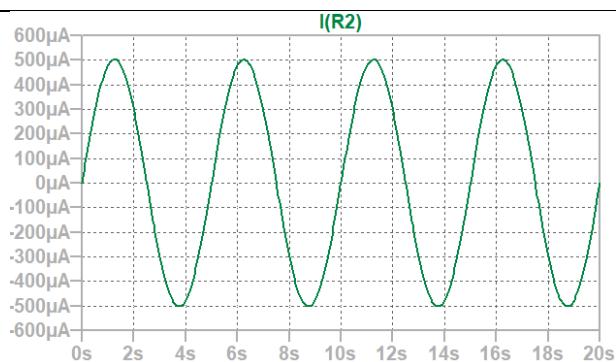
Tabel 8.3 Hasil Simulasi Referensi Rangkaian Differentiator

Jenis Gelombang Input	Hasil Simulasi

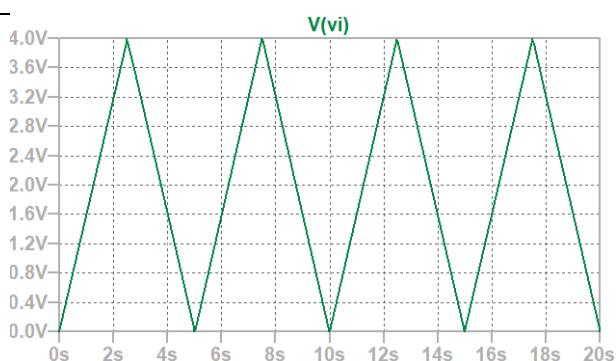




Gambar 8.3.5 Gelombang V Output-Cosinus



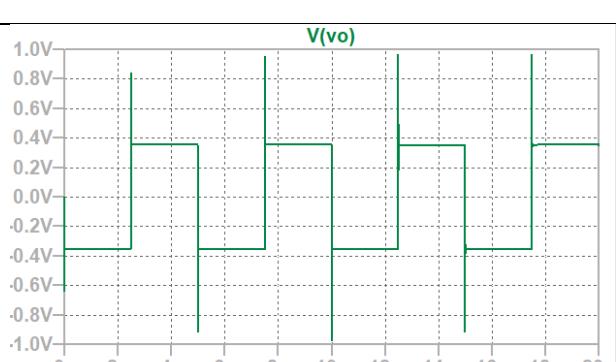
Gambar 8.3.6 Gelombang I Output-Cosinus



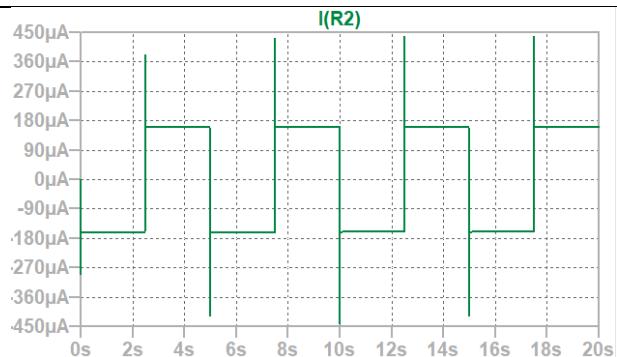
Gambar 8.3.7 Gelombang V Input-Triangular

Input Gelombang Segitiga

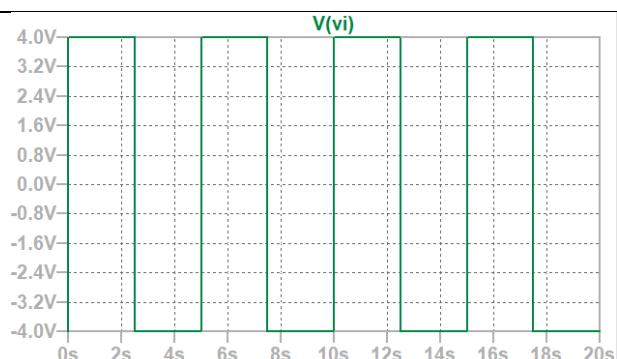
```
PULSE(0 4 0 2.5 2.5
      0 5 1000)
```



Gambar 8.3.8 Gelombang V Output-Triangular



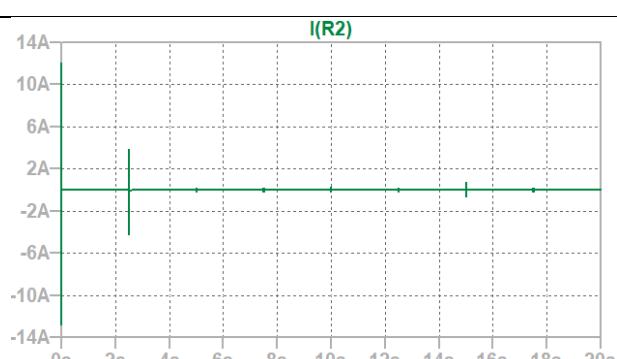
Gambar 8.3.9 Gelombang I Output -Triangular



Gambar 8.3.10 Gelombang V Input-Square



Gambar 8.3.11 Gelombang V Output-Square



Gambar 8.3.12 Gelombang I Output-Square

Input Gelombang Kotak

PULSE (-4 4 1p 1p 1p
2.5 5)

Pada tabel 8.3, terdapat hasil referensi hasil simulasi yang telah dilakukan oleh kelompok dengan menggunakan simulator LTSpice XVII pada rangkaian *differentiator* dengan skematik ditunjukkan pada gambar 8.3. Terdapat empat buah variasi gelombang input yang akan

digunakan sebagai sinyal tegangan input rangkaian, yaitu sinyal sinus, cosinus, segitiga, dan sinyal kotak. Pada masing-masing variasi sinyal input rangkaian, akan diamati beberapa variabel seperti tegangan input, tegangan output, dan arus output dari rangkaian differentiator. Hasil simulasi yang diperoleh oleh simulator yang dirancang oleh kelompok pada tabel 7.3 akan dibandingkan dengan hasil simulasi yang diperoleh dari simulator referensi yang terdapat pada tabel 8.3.

Gelombang input pertama yang disimulasikan adalah gelombang sinus dengan spesifikasi $A=4V$, $T=5s$ dan disimulasikan dengan rangkaian yang mempunyai nilai komponen $R=2200\Omega$, $C=100\mu F$. Sebagai referensi, digunakan persamaan berikut yang merupakan persamaan hubungan tegangan output dengan tegangan input:

$$Vo = -RC \frac{dVi}{dt}$$

Persamaan Tegangan Output pada Rangkaian Differentiator

Berdasarkan persamaan di atas, bila rangkaian diberi input berupa gelombang sinus diperoleh:

$$Vo = -RC(\omega A \cos(\omega t)) = -\omega R C A \cos(\omega t)$$

$$Vo(t) = -1.11 \cos(\omega t)$$

Untuk memverifikasi grafik tegangan output pada hasil simulasi, misalkan disubtitusikan $t=0$ pada persamaan, akan dihasilkan $Vo(0)=-1.11V$. Bila ditinjau hasil yang diperoleh pada tabel 7.3, dapat disimpulkan bahwa hasil simulasi yang diperoleh oleh kelompok memberikan hasil yang tepat. Hal tersebut juga ditunjukkan dari kesesuaian data hasil simulasi simulator rancangan kelompok dengan hasil yang diperoleh pada LTSpice, baik dari tegangan input, tegangan output, maupun arus output pada sinyal input sinus.

Untuk sinyal input berupa sinyal cosinus, dengan pendekatan yang sama dengan sinyal input berupa sinyal sinus diperoleh:

$$Vo(t) = 1.11 \sin(\omega t)$$

Hal tersebut menunjukkan hasil yang sesuai dengan hasil yang diperoleh dari simulator rancangan kelompok dan referensi dari LTSpice.

Jika ditinjau kondisi saat sinyal tegangan input berupa sinyal segitiga (*triangular wave*), terlihat bahwa sinyal tegangan output yang diperoleh merupakan sinyal kotak (*square wave*). Hal ini merupakan hasil yang sesuai dengan harapan. Turunan terhadap waktu dari suatu fungsi dengan parameter waktu secara matematis merupakan gradien/kemiringan sesaat fungsi tersebut. Untuk menganalisa hasil tersebut, dapat ditinjau dari keadaan saat sinyal segitiga tersebut naik (*rising*) dan turun (*falling*). Ketika sinyal tersebut naik, sinyal tersebut mempunyai gradien bernilai positif sebesar $4/2.5=1.6V/s$ dan saat turun mempunyai gradien sebesar $-1.6V/s$. Berdasarkan persamaan tegangan output, diperoleh:

$$Vo = - - 1 \times 2200 \times 100 \times 10^{-6} \times 1.6 = -0.352V \text{ untuk } 0 < t < \frac{T}{2}$$

dan

$$Vo = 0.352V \text{ untuk } \frac{T}{2} < t < T$$

$$\text{dengan } T = 5n, n = 1, 2, 3, \dots$$

Hasil tersebut sesuai dengan hasil yang diperoleh dari simulator rancangan kelompok sehingga dapat disimpulkan bahwa simulator rangkaian *differentiator* untuk input sinyal segitiga dapat berjalan dengan baik.

Selain input sinyal sinus, cosinus, dan segitiga, simulator yang dirancang juga dapat mensimulasikan sinyal input berupa sinyal kotak. Dengan pendekatan yang serupa dengan pendekatan pada sinyal input segitiga, turunan terhadap waktu sebuah fungsi dengan parameter waktu dapat diartikan sebagai gradien sesaat dari fungsi tersebut. Sinyal kotak pada rentang tertentu mempunyai gradien bernilai 0, artinya nilai sinyal tersebut akan bernilai sama untuk rentang tertentu ($T/2$). Namun pada keadaan transisi ($T/2, T, 3T/2, \dots$), nilai sinyal tersebut berubah secara instan dari $A \rightarrow -A$, begitu pula sebaliknya dalam waktu yang sangat singkat. Perubahan ini mempunyai waktu yang sangat singkat sehingga gradien fungsi pada nilai tersebut bernilai mendekati negatif infinit, begitu pula sebaliknya. Hal tersebut terlihat pada gambar berupa *impulse* pada waktu yang sangat singkat seperti ditunjukkan pada hasil yang diperoleh pada tabel 7.3 dan 8.3. Hal tersebut membuktikan bahwa tegangan output hasil simulasi rangkaian *differentiator* yang dirancang oleh kelompok dapat berfungsi dengan baik. Bila ditinjau semua arus output dari rangkaian, nilainya akan bernilai Vo/R untuk setiap waktu. Hal tersebut terbukti dari hasil simulasi rancangan kelompok yang sesuai dengan hasil referensi.

Berdasarkan analisis keseluruhan yang telah dilakukan, seluruh fitur simulator yang dirancang oleh kelompok dapat berfungsi dengan baik dan menghasilkan plot yang *valid* dengan referensi yaitu LTSpice. Maka dari itu, dapat disimpulkan bahwa simulator yang dirancang oleh kelompok merupakan simulator yang berhasil.

9. KESIMPULAN DAN LESSON LEARNED

Berdasarkan perancangan dan hasil simulasi yang telah dilakukan, diperoleh beberapa kesimpulan:

1. Aplikasi simulator rangkaian dapat dirancang dan diimplementasikan dengan baik oleh kelompok.
2. Simulator yang dirancang dapat melakukan simulasi rangkaian RC sebanyak 5 jenis rangkaian RC.
3. Simulator yang dirancang dapat melakukan simulasi rangkaian *differentiator* order 1 dengan menggunakan op-amp.
4. Dari hasil *unit test* yang telah dilakukan, semua *function*, *procedure*, dan *block program* yang digunakan pada simulator secara keseluruhan dapat berfungsi dengan baik.

5. Dari hasil *functional test*, yaitu *GUI to command line, run program via command line argument*, dan *plotting data* dari CSV dengan menggunakan library numpy dan matplotlib dapat berfungsi dengan baik.
6. Dari hasil pengujian secara keseluruhan, dapat disimpulkan bahwa secara keseluruhan, simulator yang dibuat dapat berfungsi dengan baik.
7. Dari hasil analisis dengan membandingkan hasil simulasi aplikasi yang telah dirancang dengan hasil simulasi oleh *LTspice* dapat disimpulkan bahwa semua hasil simulasi sesuai dengan hasil yang didapat dari simulator referensi.

Hal yang dipelajari dari penggerjaan tugas besar ini adalah sebagai *engineer*, harus mampu untuk memecahkan semua permasalahan praktis yang dihadapi. Selain itu, dari *programming* secara umum, kami mendapat konsep *computational thinking* sehingga dalam menghadapi suatu masalah dalam keseharian, kami dapat memandang masalah tersebut dan menyelesaiakannya secara bertahap hingga *problem* tersebut dapat diatasi dengan efektif dan efisien. Penggerjaan secara berkelompok memberikan kemudahan bahwa sebuah masalah kompleks dapat dipecah menjadi masalah yang lebih kecil dan jika masalah kecil tersebut terselesaikan, masalah yang besar pun akan teratasi, konsep ini dikenal dengan *divide and conquer*.

10. PEMBAGIAN TUGAS

Tabel Kontribusi Anggota

Penjelasan	Berkas Terkait	Programmer	Tester
Program simulator rangkaian RC Tipe 1	rangkaian1_voltage.c rangkaian1_current.c	Matthew Terence 13218038	
Program simulator rangkaian RC Tipe 2 dan 3	rangkaian2_voltage.c rangkaian2_current.c rangkaian3_voltage.c rangkaian3_current.c	Fazha Ivanda 13218008	
Program simulator rangkaian RC Tipe 4	rangkaian4_voltage.c rangkaian4_current.c	Apria Wati 13218028	
Program simulator rangkaian RC Tipe 5	rangkaian5_voltage.c rangkaian5_current.c		
Program simulator rangkaian <i>differentiator</i> untuk input gelombang sinus	diff_sinus.c		Seluruh anggota kelompok
Program simulator rangkaian <i>differentiator</i> untuk input gelombang cosinus	diff_cosinus.c	Martinus William Hartono 13218044	
Program simulator rangkaian <i>differentiator</i> untuk input gelombang segitiga	diff_triangular.c		
Program simulator rangkaian <i>differentiator</i> untuk input gelombang kotak	diff_square.c		

<i>Graphical User Interface</i> dan integrasi program secara keseluruhan	top_simulator.py		
--	------------------	--	--

11. DAFTAR PUSTAKA

- [1] https://www.electronics-tutorials.ws/opamp/opamp_7.html, 6 April 2020, 18.45.
- [2] <https://www.onlinemathlearning.com/calculus-derivatives.html>, 6 April 2020, 19.37.
- [3] Charles K. dan Matthew N.O. Sadiku, *Fundamentals of Electric Circuits Fifth edition*, McGraw-Hill Companies, USA, 2013.
- [4] https://www.tutorialspoint.com/cprogramming/c_command_line_arguments.htm, 26 April 2020, 10:04.