

## UNIT 13

### SYMPY DAN SCIPY

#### 1.1 Tujuan:

- 1.1.1 Mahasiswa dapat melakukan data visualisasi pada Python.
- 1.1.2 Mahasiswa dapat melakukan manajemen serta memasukkana data dengan menggunakan Pandas.
- 1.1.3 Mahasiswa dapat melakukan proses import library pada Python
- 1.1.4 Mahasiswa mampu melakukan analisis data dengan menggunakan metode statistik dengan Seaborn

#### 1.2 Dasar Teori

SymPy adalah *library* Python untuk simbol matematika yang bertujuan untuk menjadi *computer algebra system* (CAS) berfitur lengkap dengan menggunakan kode sesederhana mungkin agar dapat dipahami dan mudah dikembangkan. SymPy ditulis seluruhnya dengan Python.

SciPy adalah kumpulan algoritma matematika yang dibangun dari *library* NumPy. SciPy menambahkan kekuatan yang signifikan ke sesi Python interaktif dengan menyediakan pengguna dengan perintah dan kelas tingkat tinggi untuk memanipulasi dan memvisualisasikan data. Dengan SciPy, sesi Python interaktif menjadi pemrosesan data dan sistem prototipe lingkungan yang menyaingi sistem, seperti MATLAB, IDL, Octave, R-Lab, dan SciLab.

Manfaat tambahan dari mendasarkan SciPy pada Python adalah bahwa ini juga membuat bahasa pemrograman yang kuat tersedia untuk digunakan dalam mengembangkan program canggih dan aplikasi khusus. Aplikasi ilmiah yang menggunakan SciPy mendapat manfaat dari pengembangan modul tambahan di berbagai relung lanskap perangkat lunak oleh pengembang di seluruh dunia. Semuanya, mulai dari pemrograman paralel hingga subrutin dan kelas web dan basis data telah tersedia untuk pemrogram Python. Semua kekuatan ini tersedia selain perpustakaan matematika di SciPy.

#### 1.3 Komputasi Simbolik dengan SymPy

Komputasi simbolik berkaitan dengan perhitungan objek matematika secara simbolis. Ini berarti bahwa objek matematika direpresentasikan secara tepat, bukan aproksimasi, dan ekspresi matematika dengan menggunakan variabel dan dalam bentuk simbolis.

Dapat dilihat dibawah dengan memperbandingkan komputasi dengan menggunakan *library* NumPy atau Math dibanding dengan menggunakan *library* SymPy:

```

1 # Komputasi Simbolik menggunakan SymPy
2 import sympy as sym
3 import math
4 import numpy as np
5
6 print(f"Menggunakan Numpy akar dari 3 adalah : {np.sqrt(3)}\n")
7 print(f"Menggunakan Math akar dari 3 adalah : {math.sqrt(3)}\n")
8 print(f"Menggunakan SymPy akar dari 3 adalah : ")
9 sym.sqrt(3)

```

Menggunakan Numpy akar dari 3 adalah : 1.7320508075688772

Menggunakan Math akar dari 3 adalah : 1.7320508075688772

Menggunakan SymPy akar dari 3 adalah :  $\sqrt{3}$

SymPy dapat mengubah suatu variabel kedalam bentuk simbolik matematis dengan menggunakan fungsi :

```
variabel = sym.symbol('simbol')
```

Variabel akan digunakan untuk menyimpan bentuk simbolik dari fungsi dengan mendefinisikan simbol dengan mengisi informasi pada bagian tutup kurung () fungsi.

```

1 import sympy as sym
2 x = sym.symbols('x')
3 y = sym.symbols('y')
4
5 fungsi = y**2 * x + x**2 + 2*x*y
6 fungsi

```

$x^2 + xy^2 + 2xy$

#### 1.4 Penjabaran Aljabar pada SymPy

SymPy menyediakan fungsi penjabaran secara aljabar pada variabel simbolik dengan menggunakan fungsi :

```
Variabel_penjabaran = sym.expand(variabel_sebelum)
```

Fungsi diatas akan melakukan penjabaran secara simbolik dengan memasukkan variabel yang ingin dijabarkan pada variabel baru pada fungsi tanda kurung ().

```

1 fungsi_dikali_x = fungsi*x
2 fungsi_dikali_x

 $x(x^2 + xy^2 + 2xy)$ 

1 fungsi_penjabaran = sym.expand(fungsi_dikali_x)
2 fungsi_penjabaran

 $x^3 + x^2y^2 + 2x^2y$ 

```

### 1.5 Menyederhanakan Fungsi pada SymPy

SymPy menyediakan fungsi penyederhanaan secara aljabar pada variabel simbolik dengan menggunakan fungsi :

```
Variabel_penjabaran = sym.simplify(variabel_sebelum)
```

Fungsi diatas akan melakukan simplifikasi secara simbolik dengan memasukkan variabel yang ingin dijabarkan pada variabel baru pada fungsi tanda kurung ()

```

1 fungsi_penjabaran = sym.expand(fungsi_dikali_x)
2 fungsi_penjabaran

 $x^3 + x^2y^2 + 2x^2y$ 

1 fungsi_penyederhanaan = sym.simplify(fungsi_penjabaran)
2 fungsi_penyederhanaan

 $x^2(x + y^2 + 2y)$ 

```

### 1.6 Membuat Persamaan Menggunakan SymPy

SymPy menyediakan fungsi untuk membuat suatu persamaan yang dapat diekspresikan melalui matematis simbolik, fungsi ini diawali dengan membuat komputasi simbolik dari masing-masing variabel yang ingin digunakan, kemudian dilanjutkan dengan membuat  $y = f(x)$  yang nantinya akan di masukkan kedalam fungsi , menggunakan fungsi :

```
Variabel = Eq(y, angka)
```

Dengan nilai pada bagian kurung menyatakan nilai fungsi pada ruas kanan dan nilai 'angka' untuk menyatakan nilai pada ruas kiri persamaan.

```

1 # Membuat persamaan  $x^2 - 1 = 0$ 
2 import sympy as sp
3
4 x = sym.symbols('x')
5 y = sym.symbols('y')
6
7 y = x**2 - 1
8 y = Eq(y, 0)
9 y

```

$$x^2 - 1 = 0$$

### 1.7 Menyelesaikan Persamaan Pada Fungsi dengan SymPy

Menyelsaikan persamaan yang telah di deklarasikan dengan menggunakan fungsi :

`solve(y)`

```

1 # Membuat persamaan  $x^2 - 1 = 0$ 
2 import sympy as sp
3
4 x = sym.symbols('x')
5 y = sym.symbols('y')
6
7 y = x**2 - 1
8 y = Eq(y, 0)
9 y

```

$$x^2 - 1 = 0$$

```

1 solve(y)

```

$$[-1, 1]$$

### 1.8 Melakukan Derivasi pada SymPy

SymPy menyediakan fungsi yang dapat digunakan untuk melakukan proses derivasi. Derivasi dilakukan dengan menyatakan nilai fungsi dan variabel simboliknya, fungsi :

`Variabel = Derivative(y, x, angka)`

Parameter pada fungsi diatas pada tanda kurung ( ) menyatakan fungsi yang ingin diturunkan, parameter kedua menyatakan terhadap

variabel apa fungsi diturunkan, dan parameter terakhir menyatakan banyak proses turunan pada fungsi.

```
1 import sympy as sym
2
3 x = sym.symbols('x')
4 y = sym.symbols('y')
5
6 y = sym.cos(x)**2 - 2*sym.cos(x) - 3
7 y
```

$$\cos^2(x) - 2\cos(x) - 3$$

```
1 y = Derivative(y, x, 1)
2 y
```

$$\frac{d}{dx} (\cos^2(x) - 2\cos(x) - 3)$$

```
1 y.doit()
```

$$-2\sin(x)\cos(x) + 2\sin(x)$$

Atau juga bisa langsung menghitung derivasinya dengan menggunakan fungsi :

`diff(fungsi_sebelum, turunan_terhadap_variabel, banyak_turunan)`

```
1 import sympy as sym
2
3 x = sym.symbols('x')
4 y = sym.symbols('y')
5
6 y = sym.cos(x)**2 - 2*sym.cos(x) - 3
7 y
```

$$\cos^2(x) - 2\cos(x) - 3$$

```
1 y = y.diff(x, 1)
2 y
```

$$-2\sin(x)\cos(x) + 2\sin(x)$$

## 1.9 Melakukan Integral dengan SymPy

SymPy menyediakan fungsi yang dapat digunakan untuk melakukan proses Integrasi. Integrasi dilakukan dengan menyatakan nilai fungsi dan variabel simboliknya, fungsi :

`Variabel = Integral(y, x, angka)`

Parameter pada fungsi diatas pada tanda kurung () menyatakan fungsi yang ingin diturunkan, parameter kedua menyatakan terhadap variabel apa fungsi diturunkan

```
1 import sympy as sym
2 x = sym.symbols('x')
3 y = sym.symbols('y')
4
5 y = sym.cos(x)**2 - 2*sym.cos(x) - 3
```

---

```
1 y = Integral(y, x)
2 y
```

$$\int (\cos^2(x) - 2 \cos(x) - 3) dx$$


---

```
1 y.doit()
```

$$-\frac{5x}{2} + \frac{\sin(x) \cos(x)}{2} - 2 \sin(x)$$

Apabila proses integrasi memerlukan batas atas integral dan batas bawah integral, dapat menggunakan fungsi berikut, jika batas merupakan tak hingga/ infinity dapat menggunakan notasi (oo) seperti pada contoh.

Variabel = Integral(y, (x, batas\_bawah, batas\_atas))

```
1 y = Integral(y, (x,3, oo))
2 y
```

$$\int_3^{\infty} (\cos^2(x) - 2 \cos(x) - 3) dx$$


---

```
1 y.doit()
```

$$-\infty$$

### 1.10 Mengganti Nilai pada Fungsi

Mengganti nilai pada fungsi dengan variabel lain, variabel dapat diganti dengan variabel lain ataupun angka yang diinginkan, dengan

mengubah dan mendeklarasikan paramater pada fungsi yang terletak di tanda kurung ().

`sym.subs( variabel_sebelum, variabel_sesudah)`

```
1 import sympy as sym
2
3 y = sym.Function('y')
4 x = sym.symbols('x')
5
6 fungsi_cos = sym.cos(x) + 10
7 fungsi_cos

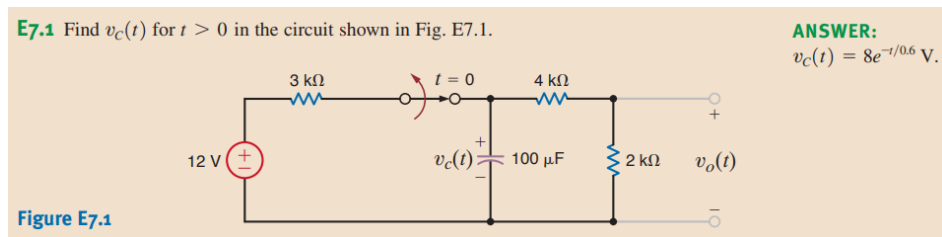
cos(x) + 10

1 fungsi_cos.subs(x, 0)

11
```

### 1.11 Menghitung Persamaan Differensial

SymPy dapat digunakan dalam mennyelesaikan permasalahan pada persamaan differensial, baik persamaan differensial orde pertama maupun orde kedua. Persamaan differensial yang paling sering muncul pada Teknik Elektro adalah persamaan diferensial orde 1 dan orde 2 rangkaian RLC, pada contoh program dibawah akan dihasilkan jawaban dari persamaan differensial pada tegangan kapasitor pada keadaan transien.



Program diawali dengan menghitung nilai tegangan pada kapasitor sebelum saklarnya dibuka, pada keadaan *Steady state* kapasitor akan bersifat seperti *open circuit*, sehingga untuk dapat menghitung tegangan pada kapasitor salah satunya dapat dilakukan dengan menghitung tegangan pada Resistor 4kΩ dan 2kΩ menggunakan Kirchoff Voltage Law (KVL).

```

1 import sympy as sym
2 t = sym.symbols('t')
3 V_sebelum = sym.symbols('Vc(0+)')
4 I_sebelum = sym.symbols('I(0+)')
5 VR1 = sym.symbols('VR1(0+)')
6 VR2 = sym.symbols('VR2(0+)')
7 R1 = sym.symbols('R1')
8 R2 = sym.symbols('R2')
9 R3 = sym.symbols('R3')
10
11 # Menggunakan KVL untuk menghitung arus pada I(0+)
12 I_sebelum = Eq(I_sebelum, 12 * (1/(R1 + R2 + R3)))
13 I_sebelum

```

$$I(0+) = \frac{12}{R_1 + R_2 + R_3}$$

Untuk dapat menghitung tegangan pada Resistor 4kΩ dan 2kΩ menggunakan Kirchoff Voltage Law (KVL), diperlukan nilai arus yang mengalir pada kedua resistor tersebut.

```

1 R1 = 3*(10**3)
2 R2 = 4*(10**3)
3 R3 = 2*(10**3)
4
5 # Menggunakan fungsi untuk mendapat nilai pecahan dari
6 # 12/(3k+4k+2k)
7 I_sebelum = sym.Rational(12, (R1 + R2 + R3))
8 I_sebelum

```

$$\frac{1}{750}$$

```

1 # Melakukan kalkulasi pada Vc(0+)
2 # Vc(0+) = VR1(0+) + VR2(0+)
3
4 V_sebelum = sym.Eq(V_sebelum, VR1 + VR2)
5 V_sebelum

```

$$Vc(0+) = VR1(0+) + VR2(0+)$$

Dan didapatkan nilai tegangan kapasitor pada keadaan *Steady state* adalah sebesar 8 V.

```

1 V_sebelum = I_sebelum * R2 + I_sebelum * R3
2 print(f"Nilai tegangan pada kapasitor saat (t=0) = {V_sebelum} V")

```

Nilai tegangan pada kapasitor saat (t=0) = 8 V

Setelah di dapatkan nilai tegangan kapasitor pada keadaan *Steady state*, maka selanjutnya menghitung tegangan pada keadaan transien, pada analisis sirkuit keadaan transien, kapasitor akan mengeluarkan energi yang disebabkan akibatnya adanya perubahan arus yang memicu perubahan medan elektrik pada kapasitor.



Pada fungsi khusus proses simbolik menggunakan metode kelas Function untuk membedakannya dengan proses simbolik biasa.

Sym.Function('Vc')

```
1 # Melakukan persamaan diferensial orde 1
2 V_sesudah = sym.Function('Vc')
3 t = sym.symbols('t')
4 c = sym.symbols('C')
5
6 persamaan = c * V_sesudah(t).diff(t) + V_sesudah(t) * 1/(R2 + R3)
7 persamaan = Eq(persamaan, 0)
8 persamaan
```

$$C \frac{d}{dt} V_c(t) + \frac{V_c(t)}{R_2 + R_3} = 0$$

Masukkan nilai dari masing-masing kapasitor dan resistor pada persamaan, dan lakukan penyederhanaan untuk mempermudah proses kalkulasi dan komputasi.

```
1 R1 = 3*(10**3)
2 R2 = 4*(10**3)
3 R3 = 2*(10**3)
4 c = 100*(10**(-6))
5 persamaan = c * V_sesudah(t).diff(t) + V_sesudah(t) * 1/(R2 + R3)
6 persamaan
```

$$\frac{V_c(t)}{6000} + 0.0001 \frac{d}{dt} V_c(t)$$

```
1 persamaan = persamaan * 10**4
2 persamaan
```

$$\frac{5 V_c(t)}{3} + 1.0 \frac{d}{dt} V_c(t)$$

Kemudian masukkan variabel persamaan yang telah dibuat ke dalam parameter dalam kurung () pada fungsi dan nilai dari tegangan *Steady state* atau *initial value* ke dalam parameter.

```
1 sym.dsolve(persamaan, ics = {V_sesudah(0):V_sebelum})
```

$$V_c(t) = 8e^{-1.66666666666667t}$$

Pada persamaan differensiasi homogen umumnya hanya memerlukan satu *initial value* namun pada persamaan differensiasi non homogen perlu menghitung dan memasukkan dua *initial value* kedalam parameter untuk dapat di kalkulasi.

## 1.12 Transformasi Laplace

Transformasi laplace adalah transformasi yang dapat mengubah suatu fungsi pada domain waktu kedalam kompleks frekuensi domain, transformasi laplace digunakan dalam menyelesaikan banyak permasalahan salah seperti sistem kontrol, percitraan gambar dan analisis sirkuit.

Fungsi diawali dengan mendeklarasikan variabel yang diperlukan kedalam sympy komputasi simbolik serta meuliskan fungsi yang ingin dilakukan proses transformasi. Fungsi laplace transform menggunakan tiga paramer yakni:

```
Sym.laplace_transform(fungsi, domain_asal, domain_tujuan,  
noconds = True)
```

Parameter pertama adalah fungsi yang ingin dilakukan transformasi, parameter kedua adalah domain asal dari fungsi yakni domain waktu yang di deklarasikan sebagai 't' dan domain tujuan yakni domain frekuensi yakni 's'. Parameter terakhir apabila ingin mengamati hasil transformasi fungsi. Fungsi umumnya akan mengembalikan tiga parameter yakni fungsi dan keterangan konvergensi,

```
1 import sympy as sym
2
3 # Melakukan printing yang paling tepat pada environment yang digunakan
4 sym.init_printing()
5 t = sym.symbols('t')
6 a = sym.symbols('a')
7 s = sym.symbols('s')
8
9 fungsi_awal = sym.exp(-a*t)
10 fungsi_awal
```

$$e^{-at}$$

```
1 laplace_dengan_kondisi = sym.laplace_transform(fungsi_awal, t, s)
2 laplace_dengan_kondisi
```

$$\left(\frac{1}{a+s}, 0, \text{True}\right)$$

```
1 hanya_laplace = sym.laplace_transform(fungsi_awal, t, s, noconds = True)
2 hanya_laplace
```

$$\frac{1}{a+s}$$

Transformasi laplace ini dapat digunakan untuk fungsi-fungsi lain dan tidak terbatas untuk fungsi eksponensial, namun untuk transformasi pada derivasi masih perlu dilakukan secara manual.

```

1 import sympy as sym
2
3 # Melakukan printing yang paling tepat pada environment yang digunakan
4 sym.init_printing()
5 t = sym.symbols('t')
6 a = sym.symbols('a')
7 s = sym.symbols('s')
8 v = sym.Function('v')
9 omega = sym.Symbol('omega')
10 cos = sym.cos
11 functions = [1,
12             t,
13             sym.exp(-a*t),
14             t*sym.exp(-a*t),
15             t**2*sym.exp(-a*t),
16             sym.sin(omega*t),
17             cos(omega*t),
18             1 - sym.exp(-a*t),
19             sym.exp(-a*t)*sym.sin(omega*t),
20             sym.exp(-a*t)*sym.cos(omega*t),
21             2*v(t).diff(t) + 8*t
22             ]
23 functions

```

$$\left[ 1, t, e^{-at}, te^{-at}, t^2e^{-at}, \sin(\omega t), \cos(\omega t), 1 - e^{-at}, e^{-at} \sin(\omega t), e^{-at} \cos(\omega t), 8t + 2 \frac{d}{dt} v(t) \right]$$

```

1 fungsi_laplace = []
2 for i in functions:
3     fungsi_laplace.append(sym.laplace_transform(i, t, s, noconds = True))
4 fungsi_laplace

```

$$\left[ \frac{1}{s}, \frac{1}{s^2}, \frac{1}{a+s}, \frac{1}{(a+s)^2}, \frac{2}{(a+s)^3}, \frac{\omega}{\omega^2+s^2}, \frac{s}{\omega^2+s^2}, \frac{a}{s(a+s)}, \frac{\omega}{\omega^2+(a+s)^2}, \frac{a+s}{\omega^2+(a+s)^2}, 2\mathcal{L}_t \left[ \frac{d}{dt} v(t) \right] (s) + \frac{8}{s^2} \right]$$

### 1.13 Inverse Transformasi Laplace

Merupakan kebalikan dari transformasi laplace, yang pada inverse transformasi laplace mengembalikan dari domain frekuensi kedalam domain waktu. Fungsi diawali dengan mendeklarasikan variabel yang diperlukan kedalam sympy komputasi simbolik serta meuliskan fungsi yang ingin dilakukan proses transformasi. Fungsi laplace transform menggunakan tiga paramer yakni:

`Sym.inverse_laplace_transform(fungsi, domain_asal, domain_tujuan, noconds = True)`

Fungsi akan mengembalikan fungsi awal dengan nilai theta merepresentasikan *Heavyside step function* atau *unit step function*, yang mana nilainya bernilai satu.

$$H(x) := \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

as

```

1 fungsi_laplace = []
2 for i in functions:
3     fungsi_laplace.append(sym.laplace_transform(i, t, s, noconds = True))
4 fungsi_laplace


$$\left[ \frac{1}{s}, \frac{1}{s^2}, \frac{1}{a+s}, \frac{1}{(a+s)^2}, \frac{2}{(a+s)^3}, \frac{a}{s(a+s)}, 2\mathcal{L}_t \left[ \frac{d}{dt} v(t) \right] (s) + \frac{8}{s^2} \right]$$


1 inverse_laplace = []
2 for i in fungsi_laplace:
3     inverse_laplace.append(sym.inverse_laplace_transform(i, s, t))
4 inverse_laplace


$$\left[ \theta(t), t\theta(t), e^{-t(\operatorname{re}(a)+i\operatorname{im}(a))}\theta(t), te^{-t(\operatorname{re}(a)+i\operatorname{im}(a))}\theta(t), t^2e^{-t(\operatorname{re}(a)+i\operatorname{im}(a))}\theta(t), -e^{-t\operatorname{re}(a)-it\operatorname{im}(a)}\theta(t) + \theta(t), 8t\theta(t) + 2\mathcal{L}_s^{-1} \left[ \mathcal{L}_t \left[ \frac{d}{dt} v(t) \right] (s) \right] (t) \right]$$

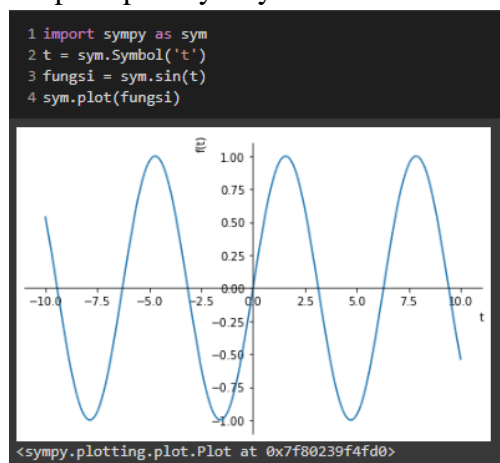

```

## 1.14 Membuat Plot dengan SymPy

SymPy merupakan suatu *library* yang disusun atas subset dari *library* NumPy dan Matplotlib, berbeda dengan Matplotlib, SciPy hanya bisa digunakan dalam memproses fungsi dan kalkulasi dari SymPy, selain itu perbedaan signifikan antara Matplotlib dengan SymPy adalah, SymPy hanya dapat digunakan dalam plotting matematis, berbeda dengan Matplotlib yang dapat melakukan visualisasi gambar.

### 1.14.1 Plot Fungsi 2 Dimensi

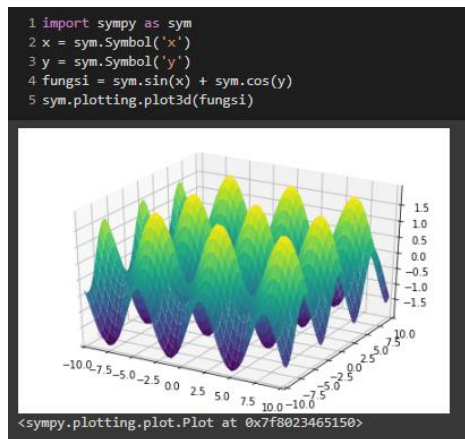
Berbeda dengan menggunakan Matplotlib, fungsi yang digunakan pada plot SymPy lebih sederhana.



### 1.14.2 Plot Fungsi 3 Dimensi

Plot fungsi 3D memiliki pola yang sama dengan plot 2D, diawali dengan mendeklarasikan variabel simbolik 'x' dan 'y' kemudian mendeklarasikan fungsi, dan memasukkan fungsi ke dalam parameter

Sym.plotting.plot3d(fungsi



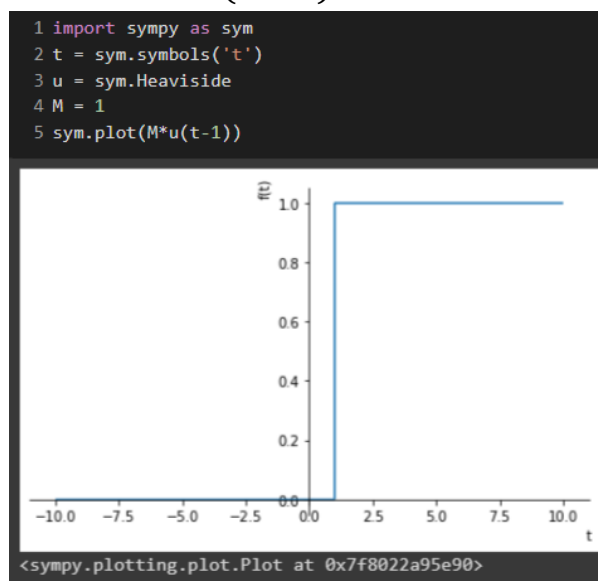
### 1.14.3 Fungsi Unit Step

Fungsi unit step atau fungsi *Heaviside*  $\theta(t)$  adalah fungsi yang seringkali ditemui terutama pada proses *switching* rangkaian sirkuit perubahan nilai tegangan dari keadaan mati ke keadaan menyala. Nilai “M” menyatakan besar dari unit step, dan nilai “a” menyatakan waktu “t” proses *switching*.

$$u(t) = \begin{cases} 0 & \text{Jika } t < a \\ M & \text{Jika } t > a \end{cases}$$

Plot diawali dengan mendeklarasikan variabel simbolik dan fungsi step, kemudian mendeklarasikan nilai “M” dan besar nilai *shifting* kemudian melakukan proses plot, fungsi unit step pada contoh dibawah adalah fungsi yang telah mengalami delayed sebanyak satu detik.

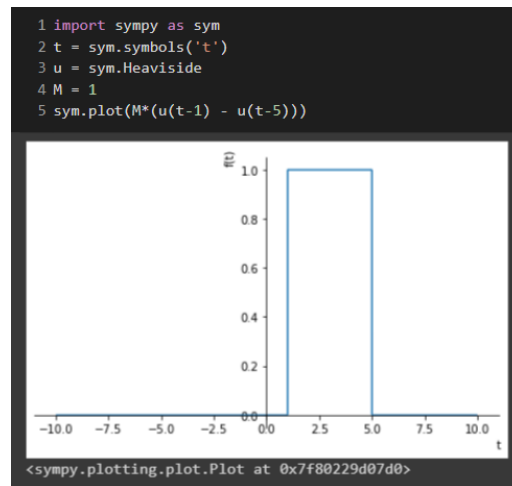
$$1u(t - 1)$$



#### 1.14.4 Fungsi Pulsa Kotak

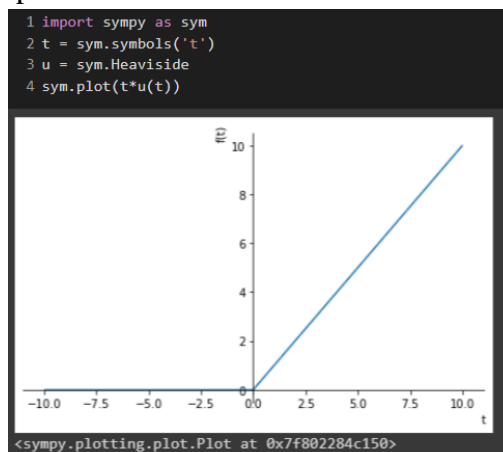
Seringkali tegangan dari rangkaian tertentu menyala pada interval waktu tertentu, yang menyebabkan grafik tegangan sirkuit berbentuk seperti fungsi pulsa kotak. Fungsi pulsa kotak dapat dirumuskan sebagai irisan antara kedua fungsi unit step, contoh dibawah merupakan irisan dari fungsi unit step  $u(t-1)$  dengan fungsi step unit  $u(t-5)$  yang membentuk fungsi pulsa kotak dengan nilai lebar 4, dan tinggi 1.

$$Rect(t) = \begin{cases} 0 & \text{jika } t < t_{awal} \\ M & \text{jika } t_{awal} \leq t < t_{akhir} \\ 0 & \text{jika } t > t_{akhir} \end{cases}$$



#### 1.14.5 Fungsi Ramp

Merupakan sinyal yang meningkat secara linear terhadap waktu, fungsi ramp merupakan fungsi yang terbentuk atas fungsi unit step.

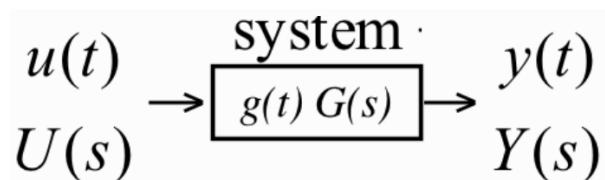


### 1.15 Konvolusi dan Fungsi Transfer

Konvolusi (convolution) adalah sebuah proses dimana citra dimanipulasi dengan menggunakan eksternal mask / subwindows untuk menghasilkan citra yang baru. Respon dari suatu sistem dapat ditemukan dengan melakukan transformasi laplace terhadap input dan sistem, kemudian melakukan perkalian antara input dan sistem untuk ditemukan nilai output yang kemudian dilakukan inverse laplace untuk mendapatkan nilainya pada domain waktu.

*Respon dari keluaran adalah*

$$Y(s) = G(s)U(s)$$



Dengan nilai input, respon dan output setelah dilakukan transformasi laplace adalah:

$$\mathcal{L}\{u(t)\} = U(s), \quad \mathcal{L}\{y(t)\} = Y(s), \quad \mathcal{L}\{g(t)\} = G(s)$$

Untuk mendapatkan nilai dari  $y(t)$ , dilakukan proses konvolusi dan inverse laplace transform dari nilai input sistem respon

$$y(t) = (g * u)(t)$$

Yang didefinisikan pada konvolusi dibawah

$$(g * u)(t) = \int_{-\infty}^{\infty} g(\tau)u(t - \tau)d\tau = \int_{-\infty}^{\infty} u(\tau)g(t - \tau)d\tau$$

Apabila fungsi  $g(t)=0, u(t)=0$  ketika  $t<0$ , maka integral dari konvolusi dapat dirumuskan sebagai berikut:

$$(g * u)(t) = \int_0^t g(\tau)u(t - \tau)d\tau = \int_0^t u(\tau)g(t - \tau)d\tau$$

Pada program dibawah, respon sistem *first-order* dengan input step respon  $U(s)$  dan diketahui nilai  $G(s)$ , dan diminta untuk mencari nilai  $y(t)$ :

### 1.15.1 Melalui Transformasi Laplace

Nilai  $y(t)$  di cari melalui pendekatan laplace, yakni dengan mencari nilai  $Y(s)$  secara langsung, kemudian melakukan inverse laplace terhadap  $Y(s)$  untuk mencari nilai  $y(t)$ .

```
1 import sympy as sym
2 s = sym.Symbol('s')
3 t = sym.Symbol('t')
4 tau = sym.Symbol('tau')
5 # Respon sistem G(s)
6 G_s = 1/(tau*s + 1)
7 G_s
```

$$\frac{1}{s\tau + 1}$$

```
1 # Jika nilai tau diketahui
2 G_s = G_s.subs({tau: 1})
3 G_s
```

$$\frac{1}{s + 1}$$

```
1 # Mencari nilai g(t) dengan
2 # Melakukan inverse laplace terhadap G(s)
3 g_t = sym.inverse_laplace_transform(G_s, s, t)
4 g_t = g_t/sym.Heaviside(t)
5 g_t
6
```

$$e^{-t}$$

```
1 # Nilai input adalah fungsi impuls
2 u_t = sym.Heaviside(t)
3 u_t
```

$$\theta(t)$$

```
1 # Mencari nilai U_s
2 U_s = sym.laplace_transform(u_t, t, s, noconds = True)
3 U_s
```

$$\frac{1}{s}$$

```
1 # Nilai keluaran Y(s) = G(s)U(s)
2 Y_s = G_s * U_s
3 Y_s
```

$$\frac{1}{s(s + 1)}$$

```
1 # Mencari nilai y(t) dengan melakukan
2 # Inverse laplace dari Y(s)
3 y_t = sym.inverse_laplace_transform(Y_s, s, t)
4 y_t
```

$$\theta(t) - e^{-t}\theta(t)$$



### 1.15.2 Melalui Konvolusi

Nilai  $y(t)$  dicari secara langsung melalui konvolusi antara fungsi respon sistem dengan input, dapat dilihat hasil dari nilai  $y(t)$  melalui konvolusi maupun nilai  $y(t)$  melalui laplace bernilai sama.

```
1 # Mencari nilai y(t) = (g * u)(t)
2 y_t_2 = g_t.subs({t: tau})*u_t.subs({t: t-tau})
3 y_t_2
```

$$e^{-\tau}\theta(t-\tau)$$

```
1 konvolusi = sym.Integral(y_t_2, (tau, 0, t))
2 konvolusi
```

$$\int_0^t e^{-\tau}\theta(t-\tau) d\tau$$

```
1 konvolusi.doit()
```

$$-(-1 + e^{-t})\theta(t)$$

### 1.15.3 Diskrit Konvolusi Numerikal

SymPy menyediakan fungsi untuk melakukan proses matematika secara analitik, namun tidak bisa melakukan secara numerikal, untuk dapat melakukan konvolusi secara numerikal diperlukan library SciPy. Library SciPy mampu melakukan konvolusi dengan bantuan library NumPy.

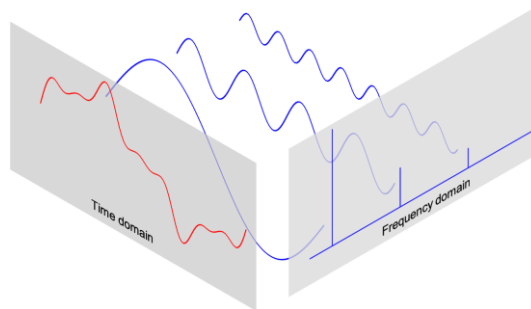
```
1 # y(n) = x(n)*h(n)
2 import scipy.signal as signal
3 import numpy as np
4 x = np.array([1,2,3,4])
5 h = np.array([1,1,-1,1])
6 y = signal.convolve(x, h)
7 print(f"Nilai konvolusi dari y(n) = x(n)*h(n) adalah : \n{y}")
```

Nilai konvolusi dari  $y(n) = x(n)*h(n)$  adalah :

```
[ 1  3  4  6  3 -1  4]
```

## 1.16 Transformasi Fourier dan Fourier Series

Transformasi fourier merupakan metode untuk mengekspresikan fungsi sebagai bentuk penjumlahan dari fungsi periodik. Secara garis besar Transformasi fourier dibagi tiga yakni diskrit fourier transform, fourier series dan transformasi fourier



### 1.16.1 Diskrit Transformasi Fourier

Diskrit transformasi Fourier adalah fungsi Fourier yang direpresentasikan sebagai bilangan diskrit, diskrit transformasi Fourier seringkali digunakan pada komputasi numerik karena memiliki proses komputasi yang cepat pada algoritmanya, yang sering kali disebut sebagai *Fast Fourier Transform*.

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi kn/N} = \sum_{n=0}^{N-1} x_n [\cos(2\pi kn/N) - i \cdot \sin(2\pi kn/N)]$$

Fungsi diskrit paling sederhana adalah fungsi impuls, yang akan dilakukan transformasi Fourier

```
1 from scipy.fft import fft, ifft
2 x = np.array([1.0, 2.0, 1.0, -1.0, 1.5])
3 y = fft(x)
4 y

array([ 4.5+0.j, 2.08155948-1.65109876j,
       -1.83155948+1.60822041j, -1.83155948-1.60822041j,
        2.08155948+1.65109876j])
```

Untuk mengembalikannya ke nilai awal dapat dengan menggunakan inverse diskrit Fourier transform yakni:

```
1 inverse_y = ifft(y)
2 inverse_y

array([ 1.+0.j, 2.+0.j, 1.+0.j, -1.+0.j, 1.5+0.j])
```

### 1.16.2 Fourier Transform

Fourier transform merupakan sinyal yang memiliki nilai waktu kontinu akan tetapi bersifat aperiodik, yang memiliki batas interval dari  $[-\infty, \infty]$ .

$$F(k) = \int_{-\infty}^{\infty} f(x)e^{-2\pi i x k} dx.$$

```
1 import sympy as sym
2
3 a = sym.Symbol('a', positive = True)
4 x = sym.Symbol('x')
5 k = sym.Symbol('k')
6 fungsi = sym.exp(-a*x**2)
7 fungsi


$$e^{-ax^2}$$


1 fungsi_fourier = sym.fourier_transform(fungsi, x, k)
2 fungsi_fourier


$$\frac{\sqrt{\pi} e^{-\frac{k^2}{a}}}{\sqrt{a}}$$

```

Untuk mengembalikan transformasi fourier ke fungsi awal dapat dengan melakukan inverse fourier transform seperti dibawah:

```
1 inverse_fourier = sym.inverse_fourier_transform(fungsi_fourier, k, x)
2 inverse_fourier


$$e^{-ax^2}$$

```

### 1.16.3 Fourier Series

Fourier series merupak sinyal yang memiliki nilai kontinu dan periodik, karena nilainya yang periodik maka fungsi dapat dibatasi ke dalam interval tertentu dari  $[0, T]$ .

$$\hat{x}(f_n) = \frac{1}{T} \int_0^T x(t)e^{-2\pi i f_n t} dt$$

```
1 import sympy as sym
2 t = sym.symbols('t', real=True)
3 k = sym.symbols('k', real=True, positive=True)
4 n = sym.symbols('n', real=True, positive=True)
5 T = sym.symbols('T', real=True, positive=True)
6 fn = n/T
7 x = sym.exp(-k * t)
8 x


$$e^{-kt}$$


1 x_FT = sym.integrate(1/T * x*sym.exp(-2*sym.pi*sym.I*fn*t), (t, 0, T))
2 x_FT


$$-\frac{1}{Tke^{Tke^{2i\pi n}} + 2i\pi ne^{Tke^{2i\pi n}}} + \frac{1}{Tk + 2i\pi n}$$


1 x_FT.simplify()


$$\frac{1 - e^{-Tk - 2i\pi n}}{Tk + 2i\pi n}$$

```

## Analisis Praktikum

1. Carilah Transformasi Laplace dari Fungsi dibawah:

$$f(t) = 3t^2$$

```
import sympy as sym
t = _____
s = _____
fungsi = _____
fungsi_laplace = sym._____(_____, _____, _____, noconds =
True)
fungsi_laplace
```

2. Carilah Transformasi Fourier dari Fungsi dibawah:

$$f(t) = e^{-at}u(t)$$

```
import sympy as sym
t = _____
a = _____
k = _____
fungsi = _____
sym._____(fungsi, t, k)
```

3. Lakukanlah konvolusi numerik  $y(n)$  menggunakan scipy dengan nilai  $x(n)$  dan  $h(n)$  dibawah:

$$x[n] = \{-1, 2, 0, 1\}$$

$$h[n] = \{3, 1, 0, -1\}$$

```
# y(n) = x(n)*h(n)
import scipy.signal as signal
import numpy as np
x = _____
h = _____
y = signal._____(x, h)
print(f"Nilai konvolusi dari y(n) = x(n)*h(n) adalah : \n{y}
")
```

## Tugas Akhir:

(Program beserta gambar keluaran dilampirkan pada saat pengumpulan tugas akhir)

1. Buatlah transformasi fourier dari fungsi:

$$f(t) = e^{-(NIM)t}u(t - NIM),$$

Dengan NIM adalah 2 digit terakhir

Contoh : 2010314069, NIM : 69

$$f(t) = e^{-(69)t}u(t - 69)$$

Output:



A hand-drawn mathematical expression representing the Fourier transform of the given function. It shows the exponential term  $e^{-i2\pi k t - 4761}$  in the numerator and the linear term  $2i\pi k + 69$  in the denominator, separated by a horizontal line.

$$\frac{e^{-i2\pi k t - 4761}}{2i\pi k + 69}$$

2. Buatlah transformasi laplace dari fungsi:

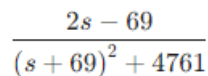
$$f(t) = e^{-(NIM)t}(2 \cos NIMt - 3 \sin NIMt)$$

Dengan NIM adalah 2 digit terakhir

Contoh : 2010314069, NIM : 69

$$f(t) = e^{-(69)t}(2 \cos 69t - 3 \sin 69t)$$

Output:



A hand-drawn mathematical expression representing the Laplace transform of the given function. It shows the linear term  $2s - 69$  in the numerator and the quadratic term  $(s + 69)^2 + 4761$  in the denominator, separated by a horizontal line.

$$\frac{2s - 69}{(s + 69)^2 + 4761}$$