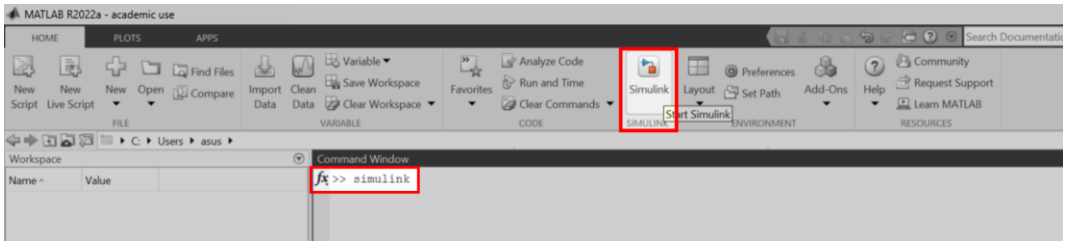
Daftar Isi

MODUL 4 SIMULINK : Modelling & Controlling

Pada modul ini akan dibahas mengenai SIMULINK. SIMULINK adalah sebuah aplikasi pemrograman grafis untuk memodelkan, mensimulasikan, dan menganalisis dinamika sistem. Dalam SIMULINK, model sistem dinamis atau dinamika sistem diwakili oleh diagram blok, dimana setiap blok merepresentasikan sebuah fungsi matematika atau operasi pada sistem tersebut. *User* dapat menghubungkan blok-blok ini bersama untuk membentuk model simulasi sistem yang kompleks. Aplikasi ini sangat populer dalam bidang teknik, terutama untuk mengembangkan sistem kontrol.

BAB 1 Pengenalan SIMULINK

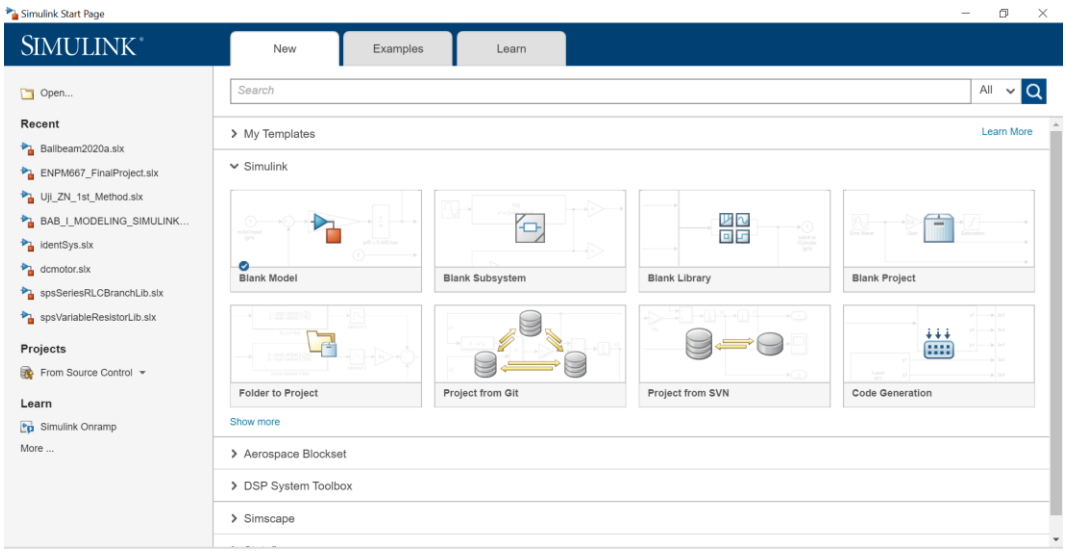
4.1 *Interface* dan *Block*



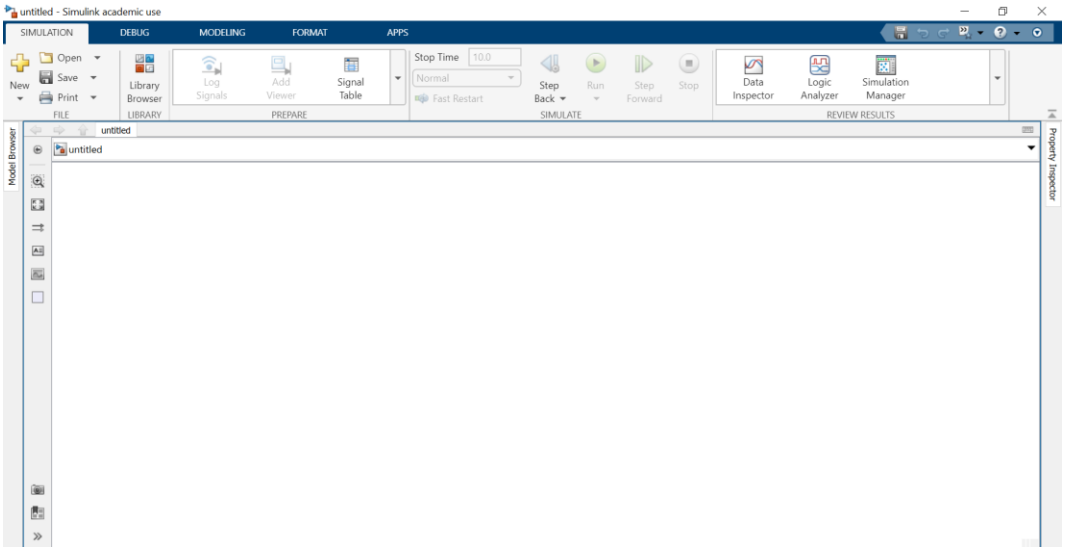
Sebelum mempelajari SIMULINK, perlu diketahui cara membuka SIMULINK itu sendiri. Terdapat dua alternatif cara :

- Dengan melakukan klik pada ikon SIMULINK
- Dengan mengetikkan 'simulink' pada *command window*

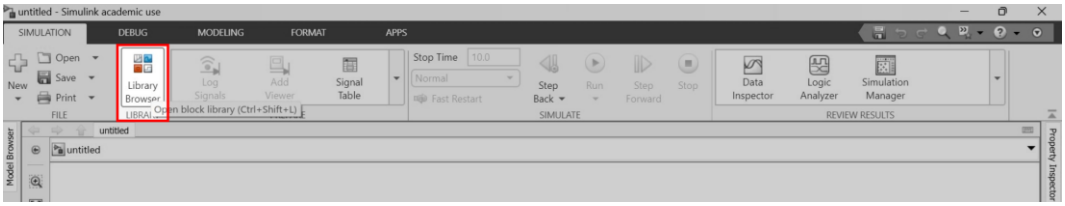
Setelah melakukan langkah sebelumnya, maka akan diarahkan pada '*SIMULINK Start Page*' seperti pada gambar di bawah.

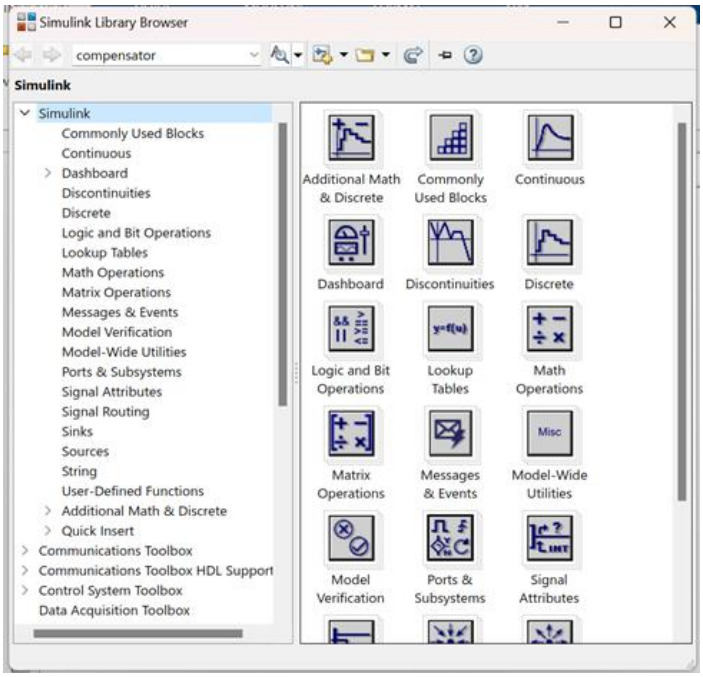


Selanjutnya klik '*Blank Model*' untuk memulai SIMULINK dengan kanvas kosong.

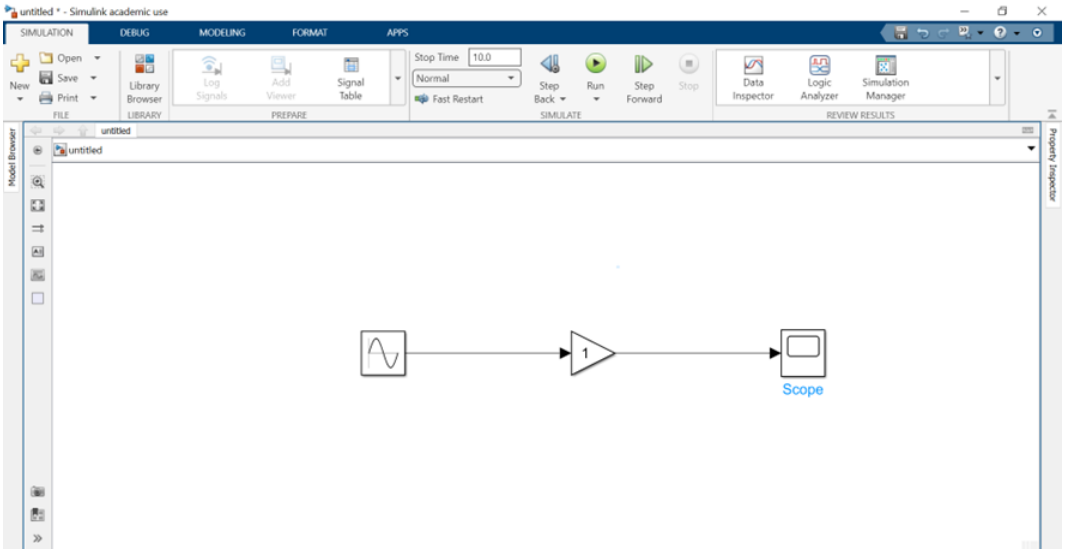


Pada SIMULINK dikenal istilah '*block*' yaitu elemen dasar yang digunakan untuk membangun model. Kumpulan *block* ini dapat ditemukan pada '*Library Browser*'. *Library Browser* dapat diakses dengan klik pada *icon* di *toolbar*.



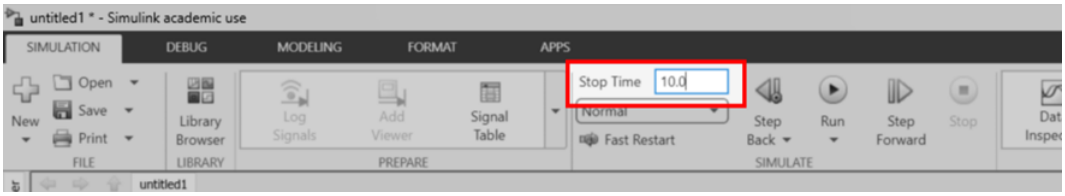


Berikutnya klik pada elemen blok yang dibutuhkan, maka blok tersebut akan muncul di kanvas SIMULINK. Apabila kita telah mengetahui nama *block* yang ingin diakses, kita juga dapat mengaksesnya dengan melakukan *double click* pada kanvas kosong SIMULINK dan ketikkan nama *block* nya.

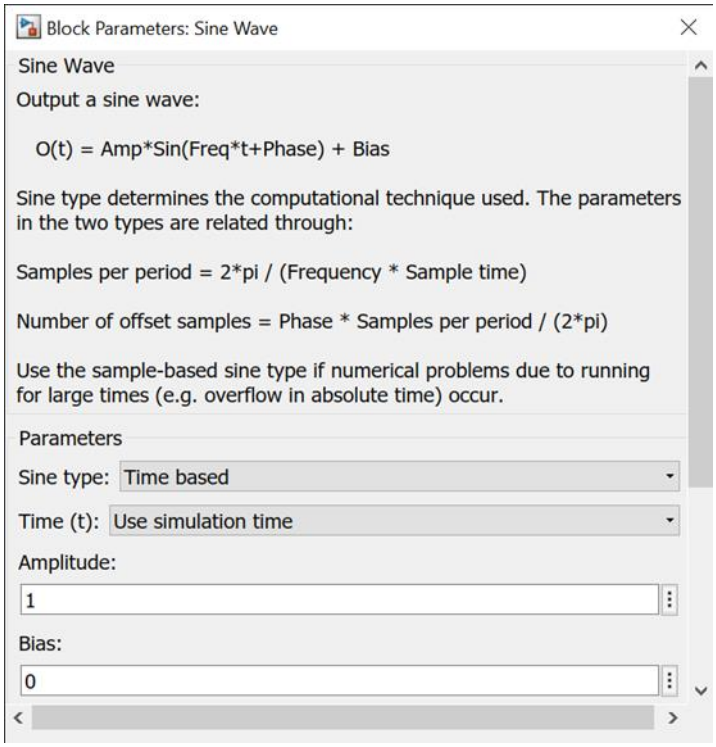


Blok-blok yang telah dimasukkan pada kanvas SIMULINK dapat dirubah namanya sesuai dengan yang diinginkan. Pertama lakukan klik pada blok yang inginkan, maka akan muncul tulisan berwarna biru pada blok. Kedua lakukan klik pada nama blok, maka akan membuka opsi untuk menulis ulang nama blok. Ketiga ubah nama sesuai dengan yang diinginkan kemudian klik '*enter*' atau klik pada area kosong pada kanvas. Cara ini juga berlaku untuk elemen lain, misalnya untuk kabel penghubung.

Sama seperti halnya MATLAB, SIMULINK juga menggunakan fungsi *run* untuk menjalankan simulasi, yang menjadi pembeda utama adalah pada SIMULINK terdapat waktu simulasi, pada *toolbar* diberi nama '*stop time*'. Seperti Namanya, *stop time* berfungsi untuk menentukan seberapa lama simulasi akan berlangsung. Nilai *default* dari stop time ini adalah 10 detik. Variasi nilai *stop time* dapat dilakukan langsung dengan mengganti nilainya sesuai dengan yang diinginkan.

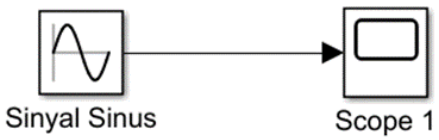


Mayoritas elemen-elemen blok pada SIMULINK memiliki nilai parameter didalamnya yang dapat divariasikan sesuai dengan kebutuhan. Untuk melihat nilai parameter ini dapat dilakukan dengan melakukan *double-click* pada blok yang dituju. Sementara untuk merubah nilainya dapat langsung dilakukan dengan mengetik ulang nilai yang diinginkan

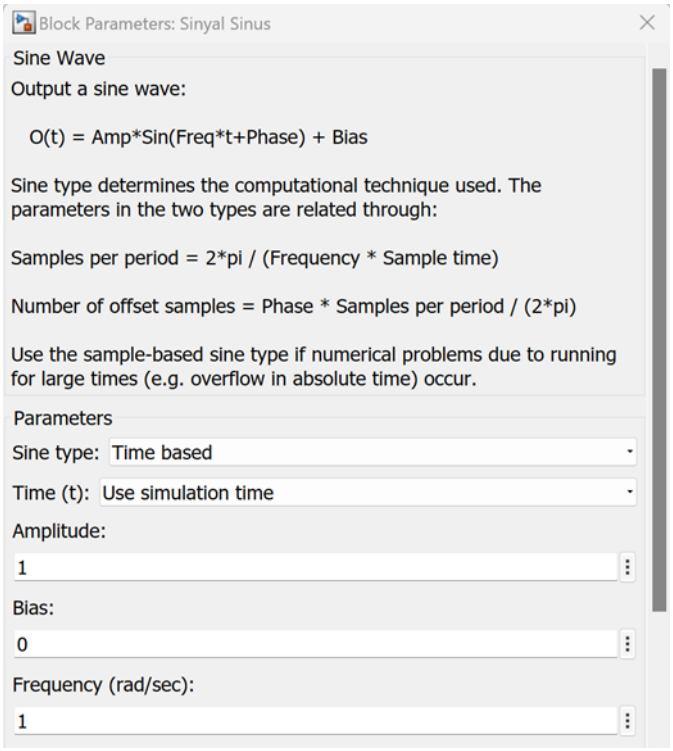


Nilai parameter tidak terbatas hanya angka saja, dapat juga memasukkan nilai parameter dalam bentuk variabel. Hal yang harus diingat ketika menggunakan nilai variabel sebagai parameter adalah harus memasukkan nilai variabel tersebut ke dalam *workspace* MATLAB terlebih dahulu sebelum melakukan *run* pada SIMULINK.

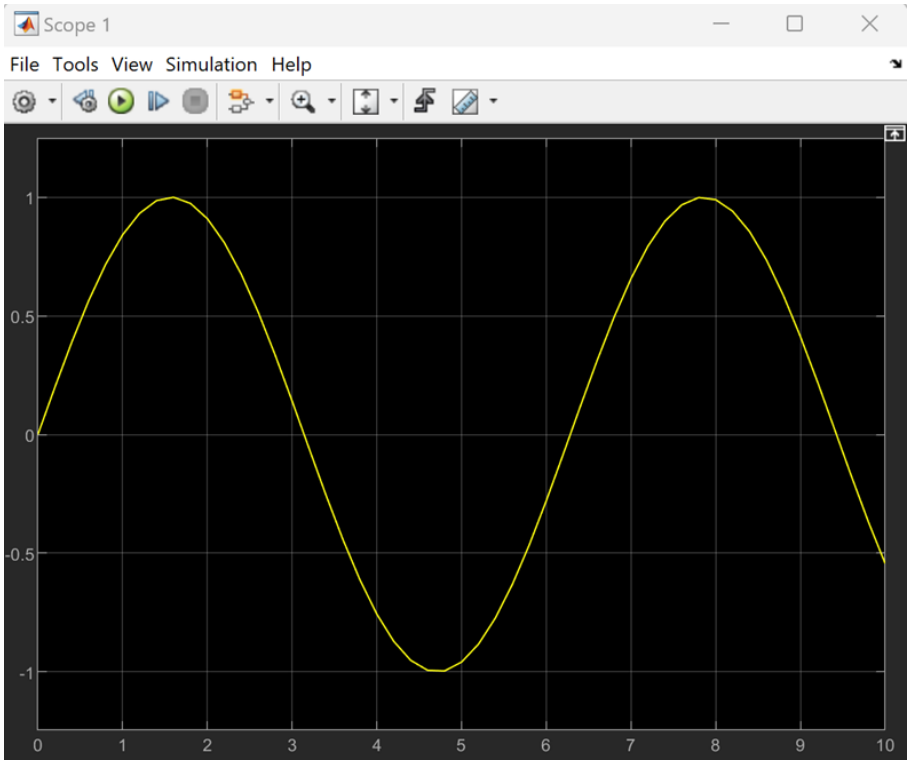
Setelah mengetahui cara untuk menambah blok dan memvariasikan parameter, berikutnya perlu diketahui cara melihat hasil keluaran dari simulasi. Blok yang paling sering digunakan untuk melihat output dari simulasi adalah blok '*scope*'. Blok ini memungkinkan untuk melihat sinyal keluaran sekaligus melakukan inspeksi tentang nilai-nilai keluaran per waktu. Diberi contoh sinyal input sinusoidal dengan frekuensi bernilai 1 rad/sec dan amplitudo sebesar 1. Maka block bisa disusun sebagai berikut:



Kemudian kita atur nilai parameter pada sinyal input seperti yang tertera diatas.



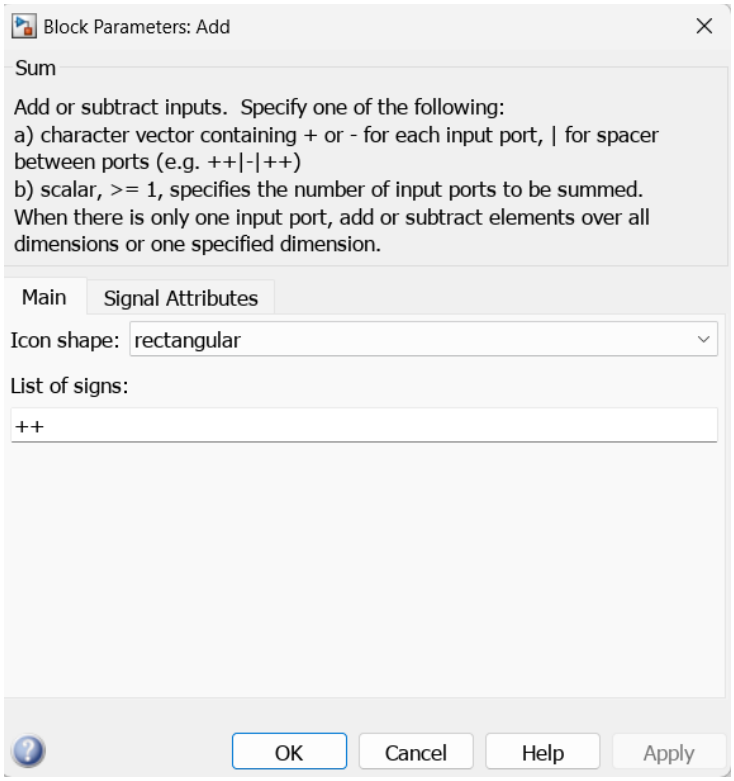
Setelah parameter diset, selanjutnya *run* SIMULINK kemudian untuk melihat hasilnya kita *double-click* pada scope sehingga diperoleh hasil sebagai berikut:



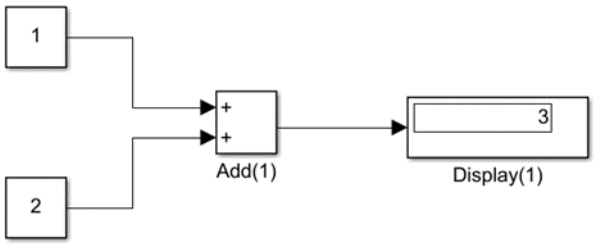
4.2 Algoritma Dasar

SIMULINK juga memungkinkan untuk melakukan simulasi operasi matematika, gerbang logika, maupun pernyataan bersyarat. Misalnya untuk **operasi**

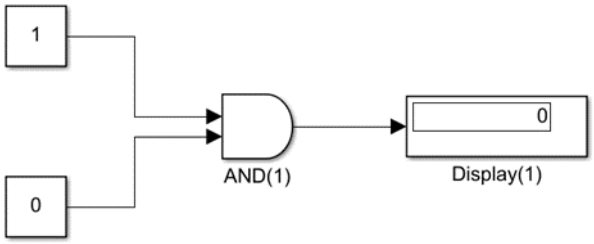
matematika, gunakan blok '*constant*' yang menghasilkan sinyal bernilai konstan, kemudian gunakan blok '*add*', blok ini berfungsi untuk menjumlahkan sinyal. Apabila melihat parameter pada blok, maka dapat juga difungsikan pengurangan dengan merubah tanda operasi dalam parameternya. Hal tersebut dapat dilakukan dengan mengubah nilai parameter pada '*List of signs*' menjadi + − atau − +. Kita juga dapat menambah jumlah input yang ingin dioperasikan dengan menambah tanda operasi seperti + + + atau + + −.



Terakhir gunakan blok '*display*' untuk menampilkan hasil perhitungan.



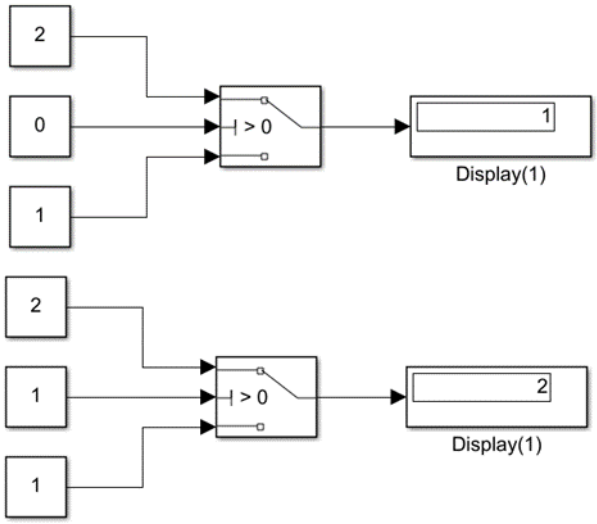
Jenis operasi matematika lain dapat ditemui pada *Library Browser > Math Operations*. Selanjutnya untuk membuat **gerbang logika** dapat menggunakan blok pada *Library Browser > Logic and Bit Operations*. Sebagai contoh kita gunakan logika AND seperti berikut



Terakhir untuk pernyataan bersyarat, pada SIMULINK dapat dibuat rangkaian yang memiliki arti yang sama dengan pseudocode tertentu. Misalnya terdapat algoritma dengan kode berikut

```
if (constant > 0)
     output 2
else
     output 1
end
```

Untuk mengimplementasikan kode diatas, digunakanlah block "*switch*".



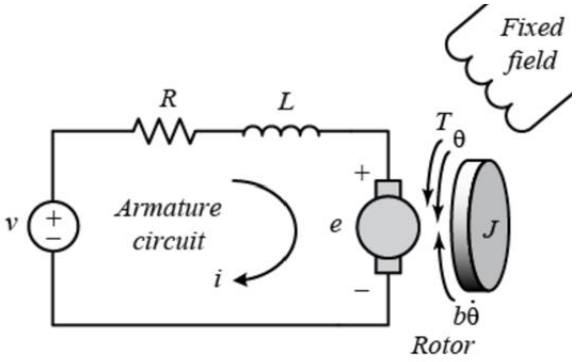
BAB 2 *Modelling*

Pada bagian ini akan dibahas mengenai cara melakukan *modelling* dinamika sistem melalui SIMULINK. Sistem yang akan digunakan yaitu berupa sistem kecepatan motor DC atau *DC Motor Speed*. Sistem tersebut akan dimodelkan melalui dua cara yaitu matematis dan *transfer function* sebagai berikut

2.1 *Modelling* Matematis

Motor DC merupakan jenis motor listrik yang menggunakan arus searah (DC) untuk menghasilkan gerakan mekanik. Prinsip kerja dasar dari DC motor adalah menggunakan interaksi antara medan magnet yang dihasilkan oleh arus searah yang

mengalir melalui gulungan atau kumparan (*coil*) yang disebut jangkar atau *armature*, dan medan magnet tetap yang dihasilkan oleh magnet permanen atau elektromagnet eksternal yang disebut medan magnet atau *fixed field*. Ketika arus mengalir melalui kumparan pada jangkar, medan magnet di sekitarnya berinteraksi dengan medan magnet tetap, sehingga menghasilkan gaya yang mendorong jangkar untuk berputar. Rotasi jangkar dapat digunakan untuk menggerakkan alat yang terhubung ke poros atau *shaft* motor, dan digunakan dalam berbagai cara sesuai dengan aplikasinya.



Pada contoh sistem diatas, diasumsikan bahwa input sistem adalah sumber tegangan (V), sedangkan outputnya merupakan kecepatan putar poros ($\dot{\theta}$). Kemudian diasumsikan torsi gesekan sebanding dengan kecepatan sudut poros.

Secara umum, torsi yang dihasilkan oleh motor DC sebanding dengan arus jangkar dan kekuatan medan magnet. Dikarenakan medan magnet konstan atau *fixed field*, maka torsi hanya sebanding dengan arus jangkar (i) dengan faktor konstan (Kt) sehingga diperoleh persamaan

$$T = K_t i$$

Kemudian, Gaya Gerak Listrik atau GGL balik (e), sebanding dengan kecepatan sudut poros dengan faktor konstan K_e

$$e = K_e \dot{\theta}$$

Dalam satuan SI, nilai faktor konstan torsi motor dan gaya gerak listrik balik sama, sehingga $K_t = K_e$. Oleh karena itu, akan digunakan variabel K untuk mewakili kedua konstanta sebelumnya. Berdasarkan rangkaian pada gambar di atas, kita dapat menurunkan persamaan menggunakan Hukum II Newton dan Hukum Kirchoff Tegangan sebagai berikut

$$J\ddot{\theta} + b\dot{\theta} = K_t i = T$$

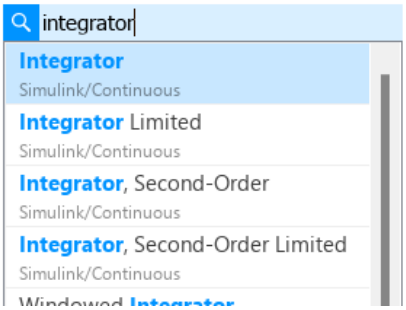
$$L\frac{di}{dt} + Ri = V - K_e \dot{\theta}$$

dimana J adalah momen inersia rotor, b adalah konstanta gesek *viscous* motor, R adalah resistansi rangkaian jangkar, dan L adalah induktansi rangkaian jangkar. Pertama, kita akan memodelkan integral dari percepatan rotasi ($\ddot{\theta}$ atau $d^2\theta/dt^2$) dan laju perubahan arus jangkar (di/dt).

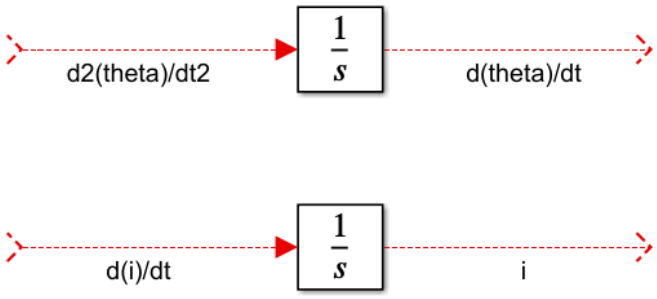
$$\int \frac{d^2\theta}{dt^2} dt = \frac{d\theta}{dt}$$
$$\int \frac{di}{dt} = i$$

Model SIMULINK dari kedua persamaan tersebut dapat dilakukan dengan cara sebagai berikut

1. Masukkan blok *Integrator* dari *Library Simulink*, dan tarik garis atau kabel dengan *drag* pada arah panah input serta output blok



2. Berikan label atau nama dengan *double click* pada kabel input yaitu $d^2\theta/dt^2$ dan output yaitu $d\theta/dt$ seperti pada gambar di bawah. Lakukan hal yang sama untuk persamaan arus



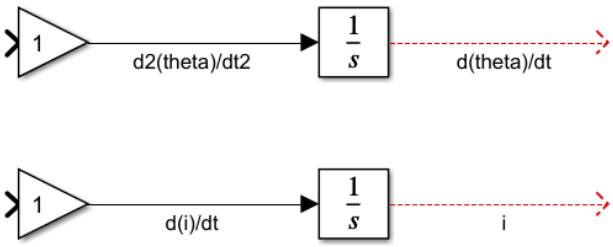
Selanjutnya, kita dapat menurunkan persamaan $d^2\theta/dt^2$ dan di/dt dari persamaan sebelumnya sebagai berikut

$$J\ddot{\theta} + b\dot{\theta} = K_t i$$
$$J\frac{d^2\theta}{dt^2} = K_t i - b\frac{d\theta}{dt}$$
$$\frac{d^2\theta}{dt^2} = \frac{1}{J}\left(K_t i - b\frac{d\theta}{dt}\right)$$

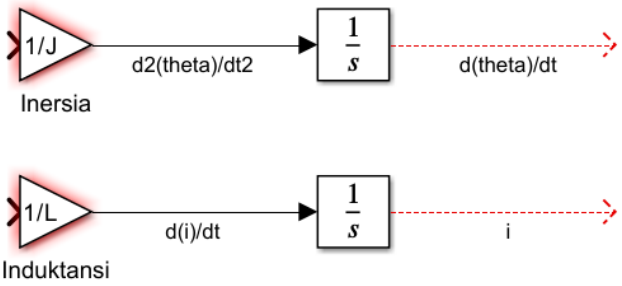
$$L\frac{di}{dt} + Ri = V - K_e\dot{\theta}$$
$$L\frac{di}{dt} = -Ri + V - K_e\frac{d\theta}{dt}$$
$$\frac{di}{dt} = \frac{1}{L}\left(-Ri + V - K_e\frac{d\theta}{dt}\right)$$

Kita dapat memodelkan persamaan di atas dalam SIMULINK sebagai berikut

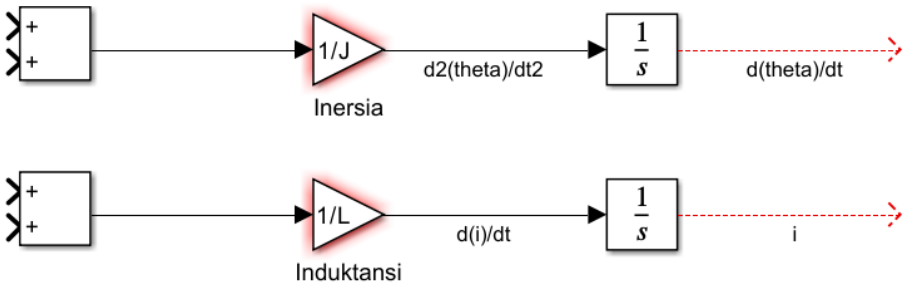
1. Masukkan dua blok *'gain'* dari *Library Simulink* dimana merepresentasikan nilai $1/J$ dan $1/L$. Hubungkan dengan masing-masing input integrator sebelumnya



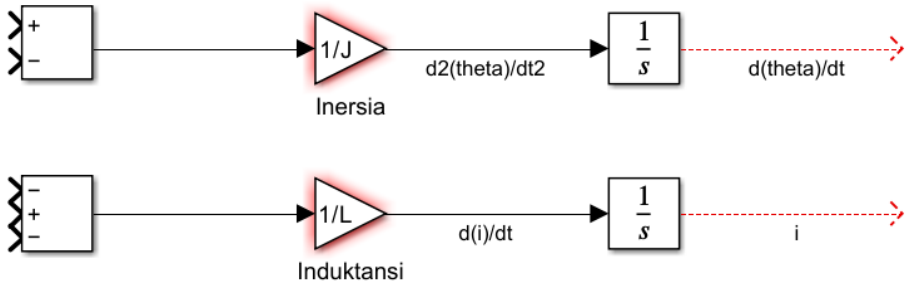
2. Ubah parameter blok *gain* dan nama blok sesuai dengan gambar di atas untuk inersia dan induktansi. Warna merah pada blok menandakan bahwa tidak adanya nilai variabel yang disebutkan dalam parameter blok pada *workspace* MATLAB



3. Masukkan dua blok '*add*' dari *Library Simulink*. Hubungkan blok tersebut dengan masing-masing blok *gain*

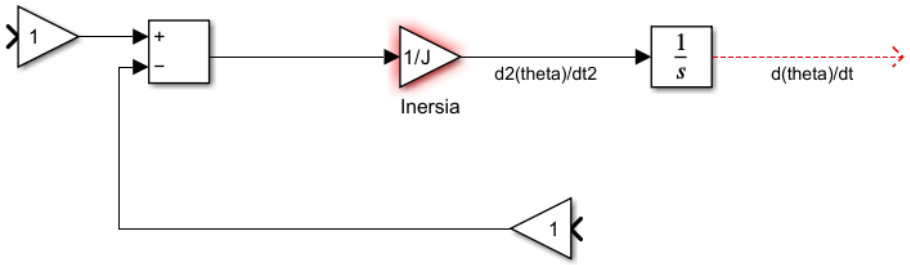


4. Ubah parameter blok '*add*' sesuai dengan banyak dan jenis operasi pada persamaan sebelumnya. Persamaan $d^2\theta/dt^2$ memiliki 2 operasi yaitu $+-$ dan persamaan di/dt memiliki 3 operasi yaitu $-+-$

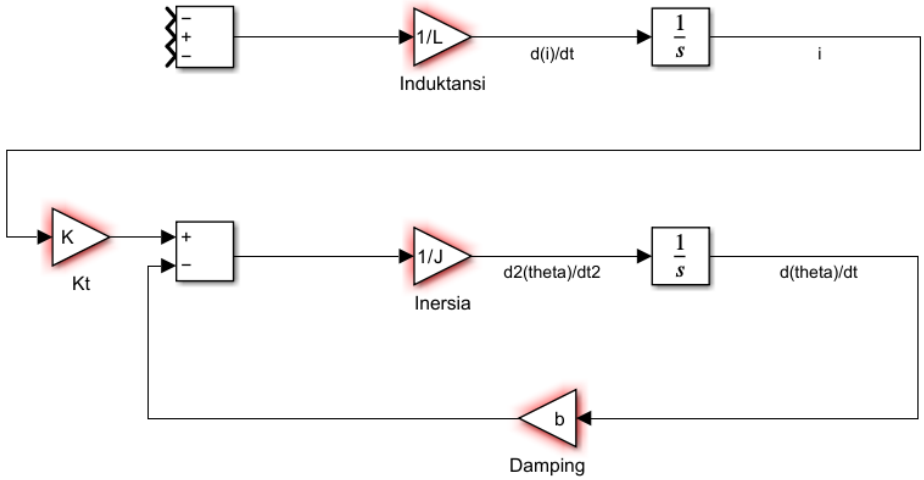


Selanjutnya kita akan menyelesaikan *modelling* dengan menambahkan elemen yang dijumlahkan pada masing-masing persamaan di atas. Pada persamaan $d^2\theta/dt$ dibutuhkan $K_t i$ (i didapat dari output persamaan arus) dan $b\frac{d\theta}{dt}$ ($\frac{d\theta}{dt}$ didapat dari output persamaan percepatan rotasi).

1. Masukkan dua blok '*gain*' pada masing-masing input blok '*add*' persamaan $d^2\theta/dt$. Untuk membalik arah blok seperti blok '*gain*' bawah dapat dilakukan dengan melakukan klik kanan pada blok dan pilih *Rotate & Flip* > *Flip Block*

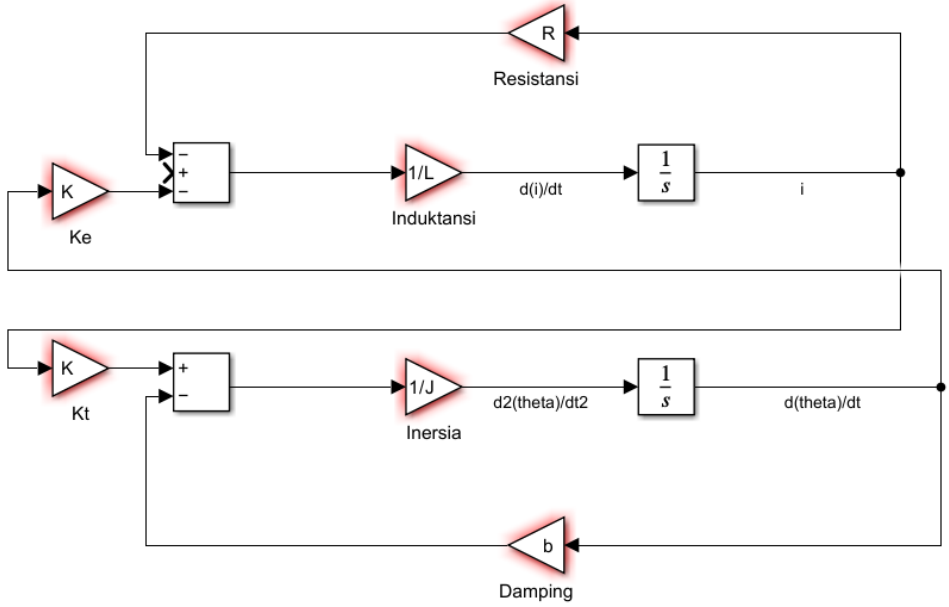


2. Ubah nilai parameter dan nama label masing-masing blok '*gain*'. Sambungkan blok atas dengan kabel i dan blok bawah dengan kabel $d\theta/dt$

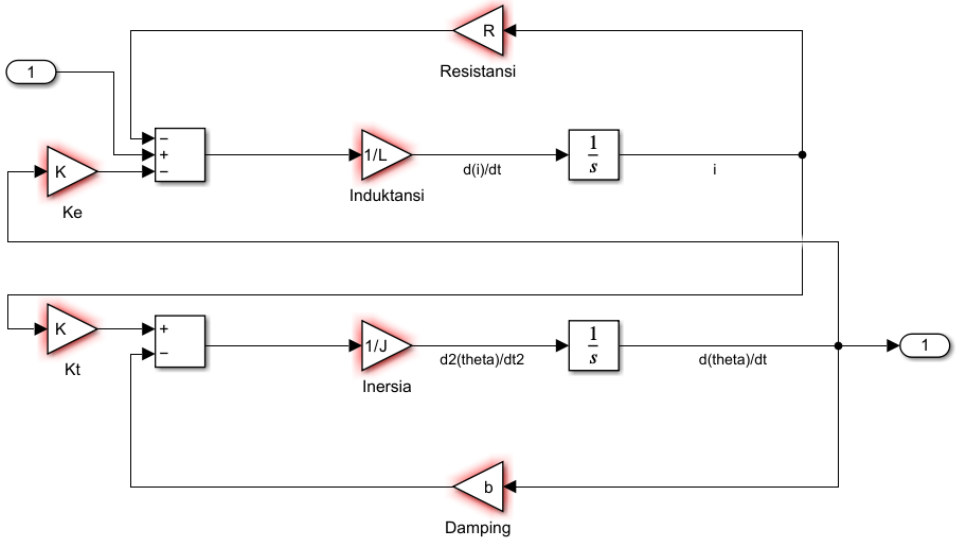


Pada persamaan di/dt dibutuhkan $-Ri$ (i didapat dari output persamaan arus), V (merupakan input dari sistem), dan $-K_e\frac{d\theta}{dt}$ ($d\theta/dt$ didapat dari output persamaan percepatan rotasi).

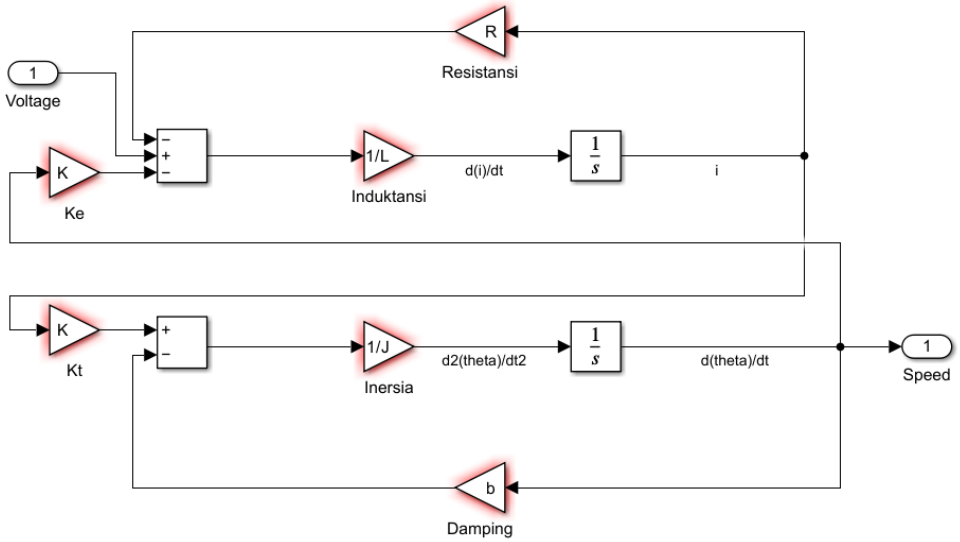
1. Masukkan dua blok '*gain*' untuk masing-masing input blok '*add*' untuk $-Ri$ dan $-K_e \frac{d\theta}{dt}$. Ubah nilai parameter dan nama masing-masing blok '*gain*' serta sambungkan dengan kabel yang sesuai seperti gambar



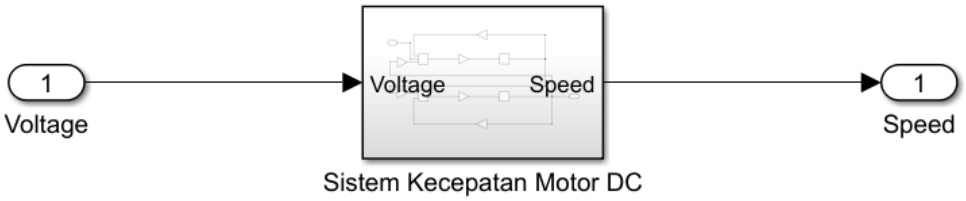
2. Tambahkan blok '*In1*' sebagai input sistem yaitu V dan blok '*Out1*' sebagai output sistem yaitu $\dot{\theta}$ atau $d\theta/dt$. Sambungkan input sistem dengan blok '*add*' atas dan output sistem degan kabel $d\theta/dt$



3. Tambahkan label pada kedua blok sebelumnya dan rapikan desain dari SIMULINK



Subsystem dapat digunakan untuk meringkas diagram blok di atas menjadi suatu blok tertentu. Cara untuk melakukannya yaitu dengan memblok diagram blok di atas atau dengan memencet $CTRL + A$ pada *keyboard*. Kemudian klik kanan pada salah satu elemen diagram blok dan tekan opsi ***Create Subsystem from Selection***.



2.2 *Transfer Function*

Terdapat cara lain dalam melakukan *modelling* suatu sistem pada SIMULINK yaitu menggunakan fungsi alih atau *transfer function* dari sistem secara langsung. Cara sebelumnya memerlukan blok yang banyak dan kompleks karena tiap elemen pada persamaan dijadikan blok tertentu. Masing-masing blok juga perlu disambungkan sesuai persamaan dimana hal ini memerlukan ketelitian agar tidak terjadi kekeliruan. Fungsi alih dari sistem dapat diperoleh menggunakan kedua persamaan sebelumnya yaitu

$$J\ddot{\theta} + b\dot{\theta} = Ki$$

$$L\frac{di}{dt} + Ri = V - K\dot{\theta}$$

Selanjutnya, dilakukan transformasi *Laplace* pada masing-masing persamaan sebagai berikut

$$Js^2\Theta + bs\Theta = KI$$

$$(Js^2 + bs)\Theta_{(s)} = KI_{(s)}$$

$$LsI + RI = V - Ks\Theta$$
$$(Ls + R)I_{(s)} = V_{(s)} - Ks\Theta_{(s)}$$

Seperti yang sudah diketahui, input dari sistem yaitu tegangan atau $V_{(s)}$ dan output dari sistem yaitu kecepatan rotasi atau $\dot{\Theta}_{(s)}$. Sehingga kita perlu mengeliminasi variabel lainnya yaitu $I_{(s)}$.

$$(Js^2 + bs)\Theta_{(s)} = KI_{(s)}$$
$$I_{(s)} = \frac{1}{K}s(Js + b)\Theta_{(s)}$$

$$(Ls + R)I_{(s)} = V_{(s)} - Ks\Theta_{(s)}$$
$$(Ls + R)\left(\frac{1}{K}s(Js + b)\Theta_{(s)}\right) + Ks\Theta_{(s)} = V_{(s)}$$
$$\frac{1}{K}\left(s(Js + b)(Ls + R)\right)\Theta_{(s)} + Ks\Theta_{(s)} = V_{(s)}$$

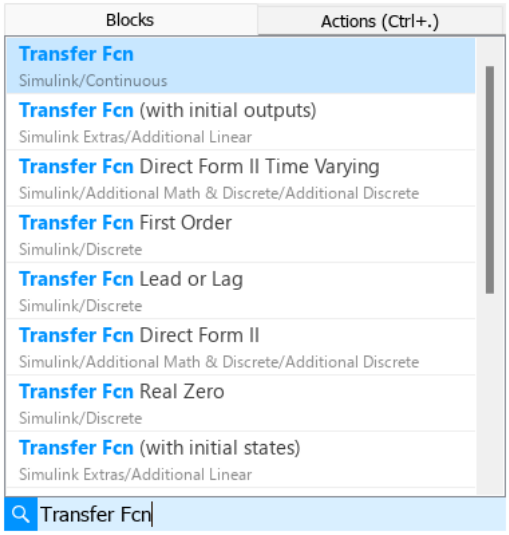
dimana kecepatan rotasi merupakan $\dot{\Theta}_{(s)} = s\Theta_{(s)}$

$$\frac{1}{K}(Js + b)(Ls + R)\dot{\Theta}_{(s)} + K\dot{\Theta}_{(s)} = V_{(s)}$$
$$\left(\frac{1}{K}(Js + b)(Ls + R) + K\right)\dot{\Theta}_{(s)} = V_{(s)}$$

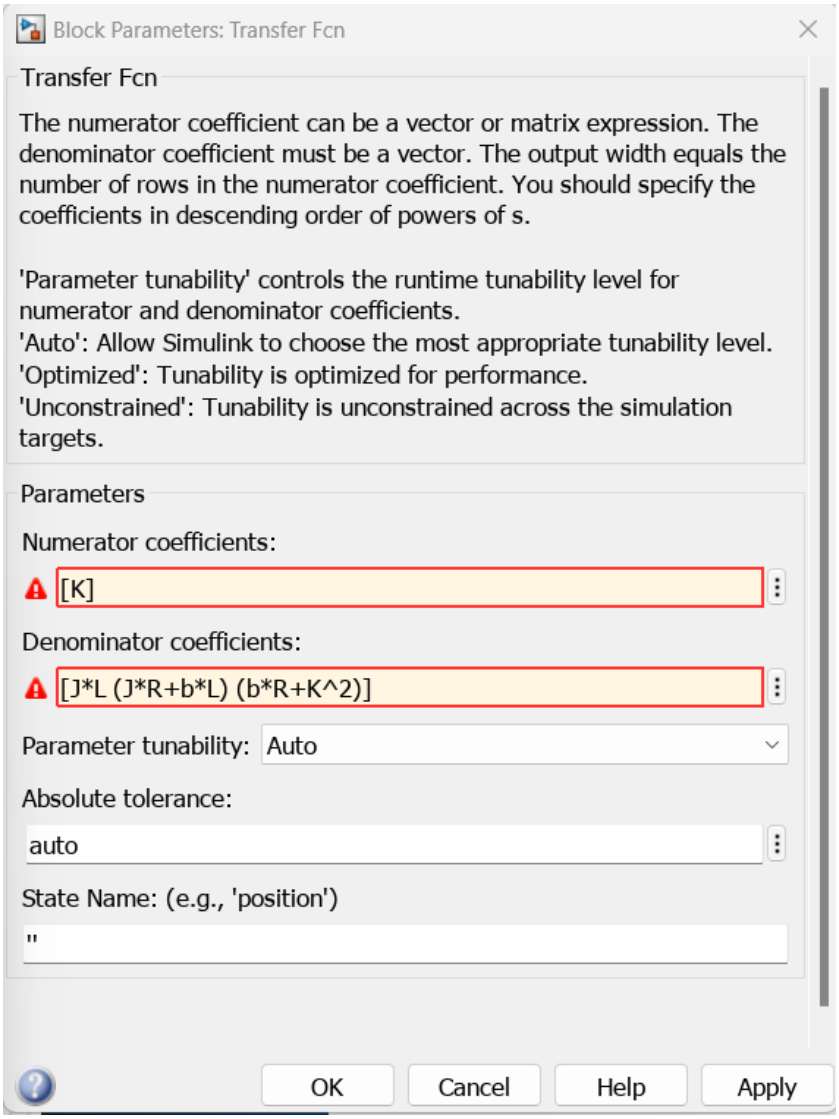
Fungsi alih atau *transfer function* merupakan perbandingan antara output terhadap input dari sistem. Dari persamaan terakhir didapat

$$G(s) = \frac{\dot{\Theta}(s)}{V(s)}$$
$$G(s) = \frac{1}{\left(\frac{1}{K}(Js + b)(Ls + R) + K\right)}$$
$$G(s) = \frac{K}{(Js + b)(Ls + R) + K^2}$$
$$G(s) = \frac{K}{JLs^2 + (JR + bL)s + bR + K^2}$$

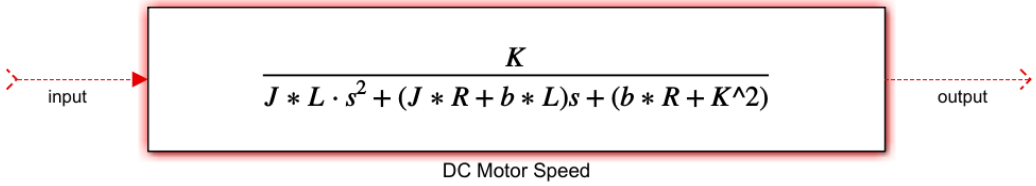
Kita dapat memasukkan model sistem dalam bentuk fungsi alih tersebut pada SIMULINK dengan menggunakan blok '*Transfer Fcn*'



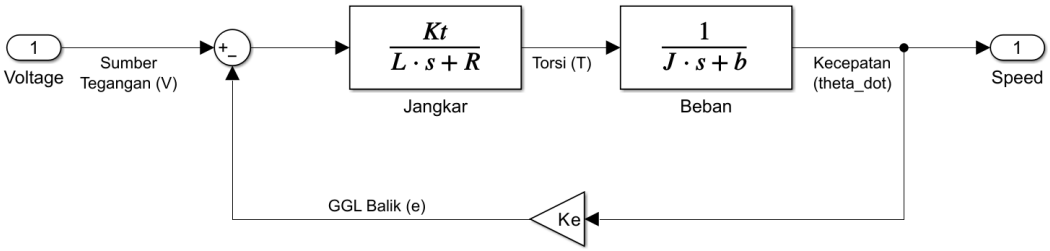
Sesuaikan nilai parameter dari blok yaitu koefisien pembilang dan penyebut dari persamaan fungsi alih di atas sebagai berikut



Sehingga didapat blok sebagai berikut dimana warna merah pada blok menandakan bahwa tidak adanya nilai variabel yang disebutkan dalam parameter blok pada *workspace* MATLAB seperti sebelumnya.



Secara umum, diagram blok dari sistem kecepatan motor DC seperti pada gambar di bawah dimana $K_t = K_e = K$. Terdapat 3 komponen penting dalam sistem ini yaitu jangkar, beban, dan GGL balik. Pada jangkar akan dihasilkan torsi yang kemudian akan masuk pada *load* atau beban sehinggga dihasilkan kecepatan sudut dari poros motor atau $\dot{\theta}$. Kecepatan ini juga menimbulkan GGL balik pada jangkar sehingga tegangan yang masuk pada jangkar merupakan selisih antara sumber tegangan dengan GGL tersebut.



Pembuktian secara matematis dari kedua diagram blok sebelumnya sama dapat dilakukan dengan menggunakan reduksi diagram blok yaitu

$$G(s) = \frac{\dot{\Theta}_{(s)}}{V_{(s)}} = \frac{G_{1(s)}}{G_{1(s)}G_{2(s)} + 1}$$

dimana $G_{1(s)}$ merupakan fungsi alih dari jangkar dan beban, serta $G_{2(s)}$ merupakan fungsi alih dari GGL balik.

$$G_{(s)} = \frac{\left(\frac{K}{Ls+R}\right)\left(\frac{1}{Js+b}\right)}{\left(\frac{K}{Ls+R}\right)\left(\frac{1}{Js+b}\right)K + 1}$$

$$G_{(s)} = \frac{\frac{K}{(Ls+R)(Js+b)}}{\frac{K^2}{(Ls+R)(Js+b)} + 1}$$

$$G_{(s)} = \frac{K}{(Ls+R)(Js+b) + K^2}$$

Dapat dilihat bahwa kedua diagram blok sebelumnya terbukti sama karena memiliki *transfer function* yang sama.

BAB 3 *Controlling*

Pada bagian ini akan dibahas mengenai desain kontrol melalui SIMULINK. Pada awalnya akan diuji atau analisis terlebih dahulu mengenai karakteristik sistem. Selanjutnya dilakukan *controlling* untuk mendapatkan karakteristik sistem yang diinginkan.

3.1 Uji Respon Sistem

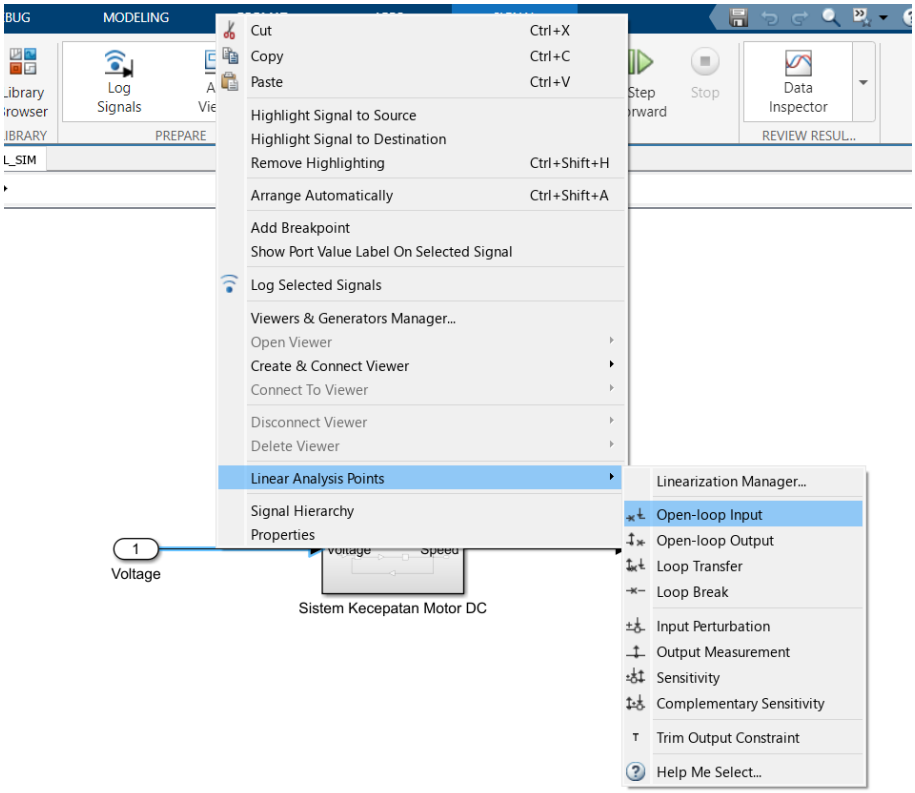
Respon dari sistem dapat diuji dengan dua cara yaitu melalui *tools* pada MATLAB maupun secara langsung menggunakan simulasi melalui SIMULINK. Tetapi, kita perlu memasukkan nilai parameter dari sistem yang belum diinputkan sebelumnya. Berikut adalah kode MATLAB yang dapat digunakan untuk menginputkan parameter sistem dimana nilai dapat disesuaikan keinginan

```
J = 0.01;
b = 0.1;
K = 0.01;
R = 1;
L = 0.5;
```

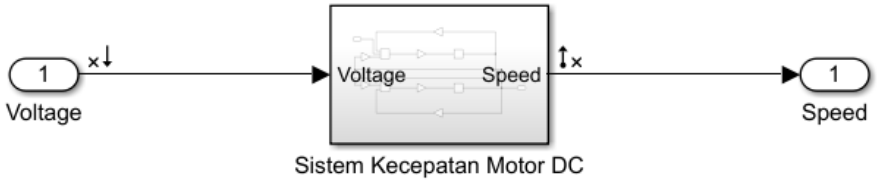
3.1.1 Linear Analysis Tool

Toolbox '*Simulink Control Design*' perlu di*install* terlebih dahulu sebelum menggunakan *tool* ini. Kita perlu mengidentifikasi input dan output dari model sistem yang akan digunakan terlebih dahulu. Hal ini dapat dilakukan dengan cara sebagai berikut

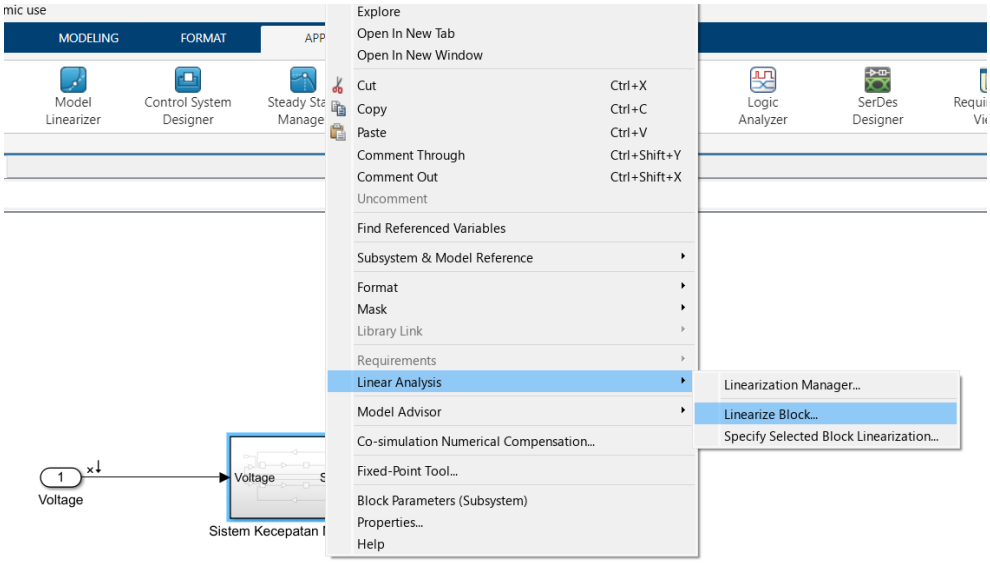
1. Klik kanan kabel atau sinyal yang merepresentasikan input *Voltage* pada sistem. Pilih ***Linear Analysis Points*** > ***Open-loop Input*** dari menu yang tampil



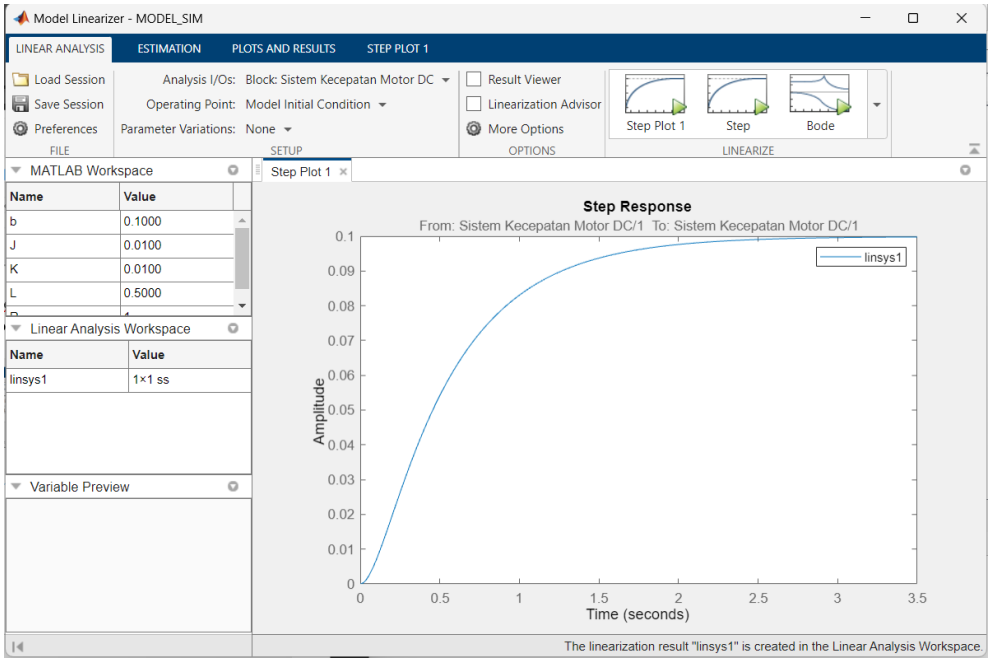
2. Klik kanan kabel atau sinyal yang merepresentasikan output *Speed* pada sistem. Pilih ***Linear Analysis Points*** > ***Open-loop Output*** dari menu yang tampil. Sehingga terdapat tanda pada input-output diagram sistem sebagai berikut



Selanjutnya kita dapat melakukan analisis dari karakteristik sistem melalui *Linear Analysis Tool*. Hal tersebut dapat dilakukan dengan klik kanan pada *subsystem* kemudian pilih ***Linear Analysis*** > ***Linearize Block..*** sebagai berikut



Selanjutnya kita dapat melakukan uji karakteristik sistem mulai dari respon *open loop* terhadap sinyal step, diagram Bode, dsb melalui *toolbar* bagian *Linearize*

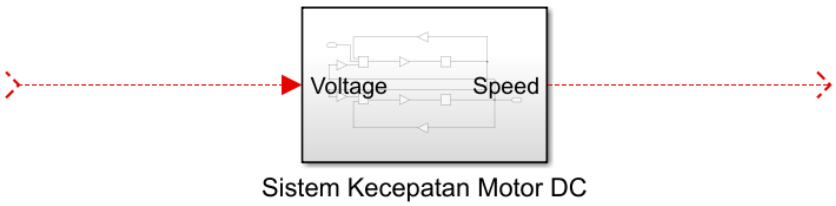


Gambar di atas merupakan respon *open loop* dari sistem terhadap sinyal *unit step*. Hasil analisis tersebut dapat diekspor pada *workspace* MATLAB melalui menu bagian '*Linear Analysis Workspace*'.

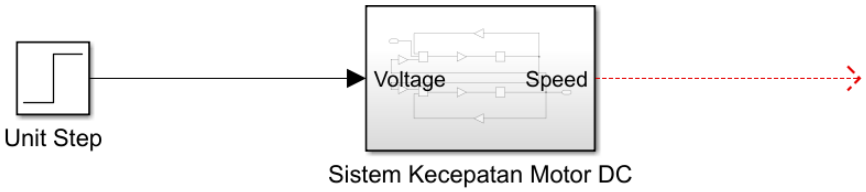
3.1.2 Uji Respon dengan Simulasi

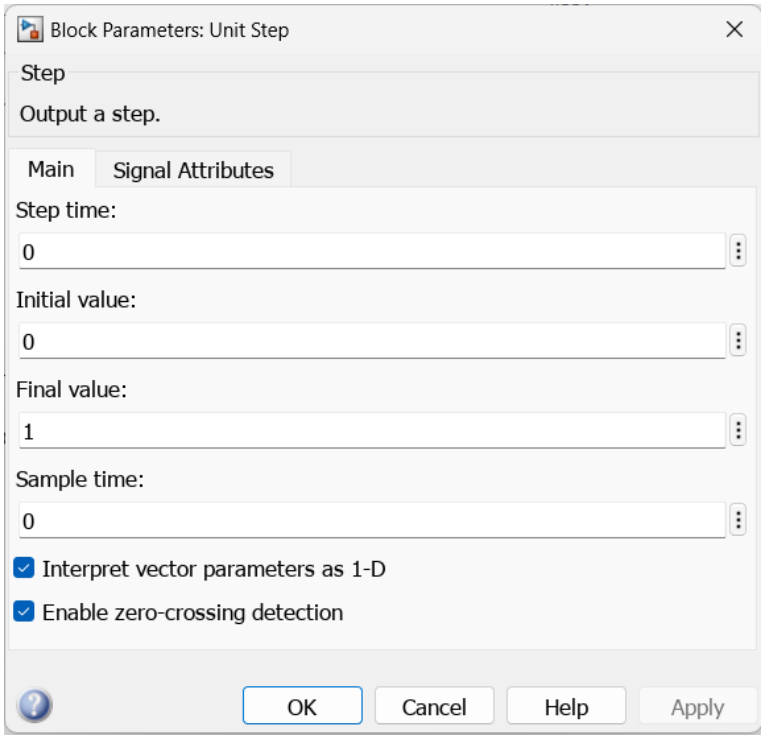
Uji respon *open loop* sebelumnya dapat dilakukan secara langsung melalui SIMULINK tanpa perlu melakukan ekstraksi model pada MATLAB. Cara yang dapat dilakukan yaitu

1. Hapus atau *delete* blok '*In1*' dan '*Out1*' dari diagram blok *subsystem* sebelumnya

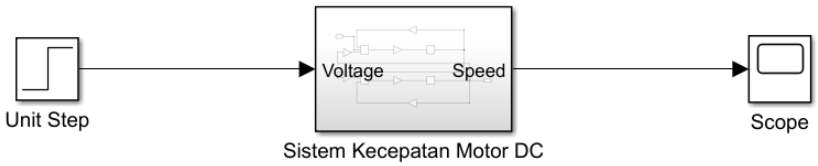


2. Tambahkan blok '*Step*' dari *Library Simulink* dan hubungkan terhadap input sistem. Pastikan nilai parameter dari blok tersebut yaitu memiliki *Step time* sebesar 0 dan *Final value* sebesar 1

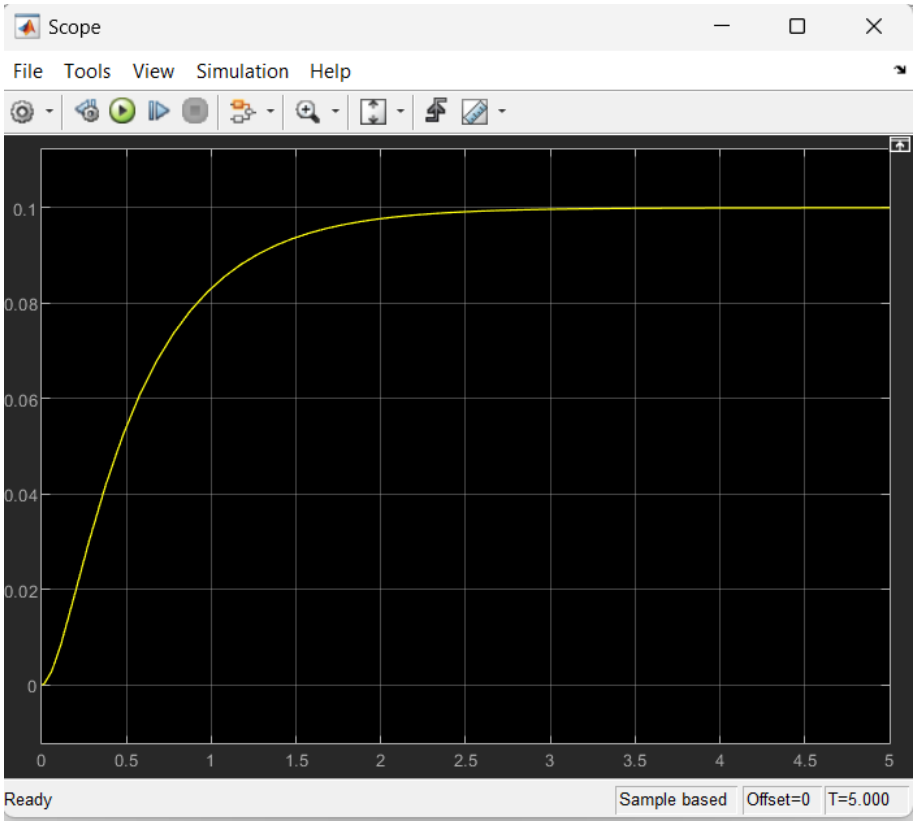




3. Tambahkan blok '*Scope*' dari *Library Simulink* dan hubungkan terhadap output sistem



4. Jalankan atau *run* simulasi pada SIMULINK. *Double click* pada blok '*Scope*' untuk melihat respon dari sistem, serta sesuaikan waktu simulasi atau *stop time*

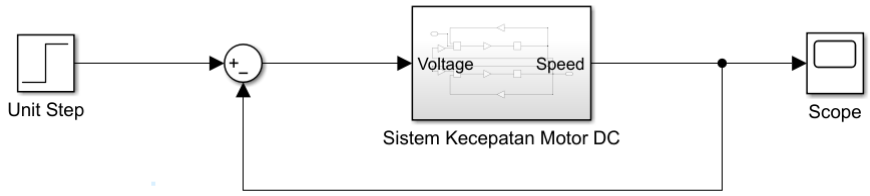


Dapat dilihat bahwa respon yang dihasilkan sama dengan cara sebelumnya atau melalui *tool* MATLAB.

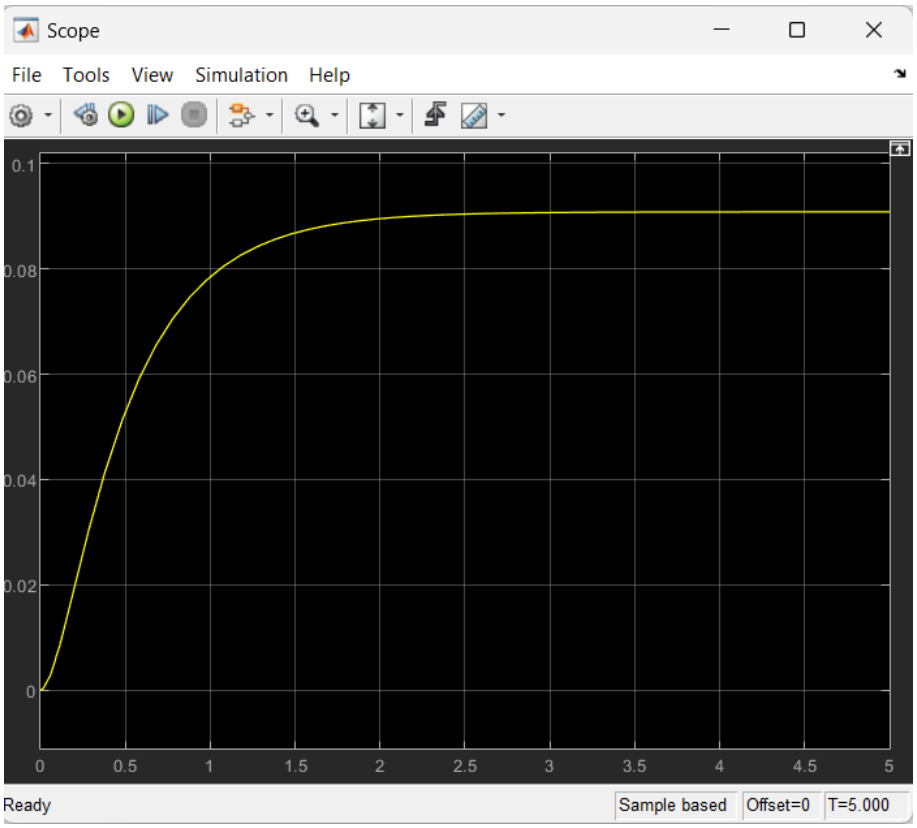
Selain respon *open loop*, kita juga dapat menguji respon *closed loop* dari sistem tanpa adanya kontroler. Hal ini dapat dilakukan dengan cara sebagai berikut

1. Tambahkan blok '*Sum*' pada diagram dan ubah parameter tanda menjadi + −

2. Sambungkan bagian + terhadap input *step* sebelumnya dan bagian − dengan output sistem



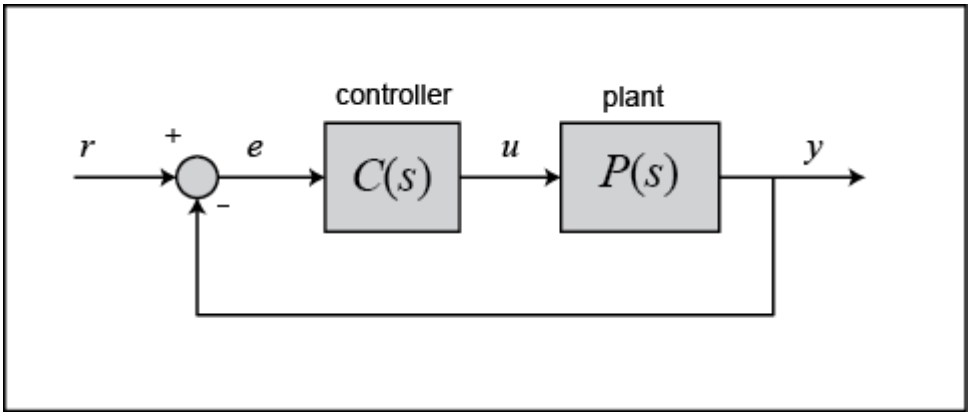
3. Jalankan atau *run* simulasi pada SIMULINK. *Double click* pada blok '*Scope*' untuk melihat respon dari sistem, serta sesuaikan waktu simulasi atau *stop time*



Kita dapat melakukan uji respon dari sistem terhadap sinyal lain seperti sinus, ramp, dsb melalui blok yang tersedia pada SIMULINK. Kita juga dapat melakukan uji respon sistem dalam bentuk *transfer function* dengan cara yang sama. Dari hasil analisis yang didapat selanjutnya dapat dilakukan desain kontroler yang sesuai.

3.2 Desain Kontroler

Pada bagian ini akan dibahas mengenai desain kontroler PID untuk sistem motor DC tersebut melalui SIMULINK. Berikut adalah diagram blok sistem *closed loop* dengan *plant* sistem adalah $P(s)$ dan kontroler $C(s)$.



Sinyal $\boldsymbol{u}$ merupakan output dari kontroler PID yang berperan sebagai input kontrol bagi *plant*. Persamaan input kontrol adalah sebagai berikut:

$$u(t) = K_p e(t) + K_i \int_{t_0}^{t} e(t) + K_d \frac{de(t)}{dt}$$

Variabel $\boldsymbol{e}$ adalah sinyal error pada sistem yang merupakan selisih antara input/referensi $\boldsymbol{r}$ dan output $\boldsymbol{y}$. Sinyal error $\boldsymbol{e}$ diinputkan ke kontroler PID. Di dalam kontroler PID, sinyal kontrol $\boldsymbol{u}$ didapat dengan menggunakan persamaan di atas dimana terdapat 3 gain yaitu K_p atau gain proporsional, K_i atau gain integral, dan K_d atau gain derivatif.

Sinyal kontrol $\boldsymbol{u}$ yang dihasilkan akan masuk ke *plant* dan didapatkan output $\boldsymbol{y}$. Output $\boldsymbol{y}$ di-umpan balik dan dibandingkan lagi dengan sinyal referensi $\boldsymbol{r}$ untuk mendapatkan sinyal error $\boldsymbol{e}$ yang baru. Jadi pada intinya, kontroler PID bekerja berdasarkan sinyal error yang dihasilkan pada *plant* atau sistem.

Fungsi alih atau *transfer function* dari kontroler PID dalam domain laplace yaitu

$$\frac{U(s)}{E(s)} = K_p + K_i \frac{1}{s} + K_d s$$

Karakteristik performa dari masing-masing jenis *gain* pada PID dapat dijelaskan melalui tabel berikut

Gain\Respon	***Rise Time***	***Overshoot***	***Settling Time***	**Error SS**
Kp	Menurun	Meningkat	Perubahan Sedikit	Menurun
Ki	Menurun	Meningkat	Meningkat	Mengeliminasi
Kd	Perubahan Sedikit	Menurun	Menurun	Tanpa Perubahan

Berdasarkan tabel tersebut, kita dapat menentukan *gain* mana yang perlu digunakan dalam desain kontroler PID. Karakteristik dari sistem atau *plant* dapat dianalisis menggunakan sinyal uji *step* seperti pada bagian sebelumnya. Sehingga nantinya kita dapat menentukan karakteristik mana yang perlu diperbaiki apakah karakteristik transien ataupun *steady-state* (SS).

Dari respon *open loop* dari sistem terhadap sinyal uji *unit step* dapat dilihat bahwa respon waktu atau transien sudah cukup cepat. Sehingga respon *steady state* nya saja yang perlu diperbaiki dimana terjadi *error* yang cukup besar yaitu hingga 90%.

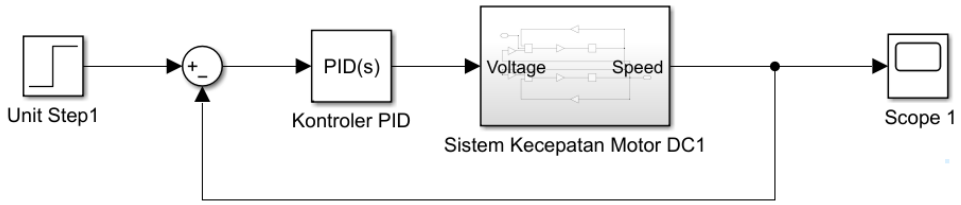
Berikut adalah karakteristik sistem yang diinginkan

- *Settling time* atau t_s kurang dari 2 detik
- *Overshoot* kurang dari 10%
- *Error* SS kurang dari 1%

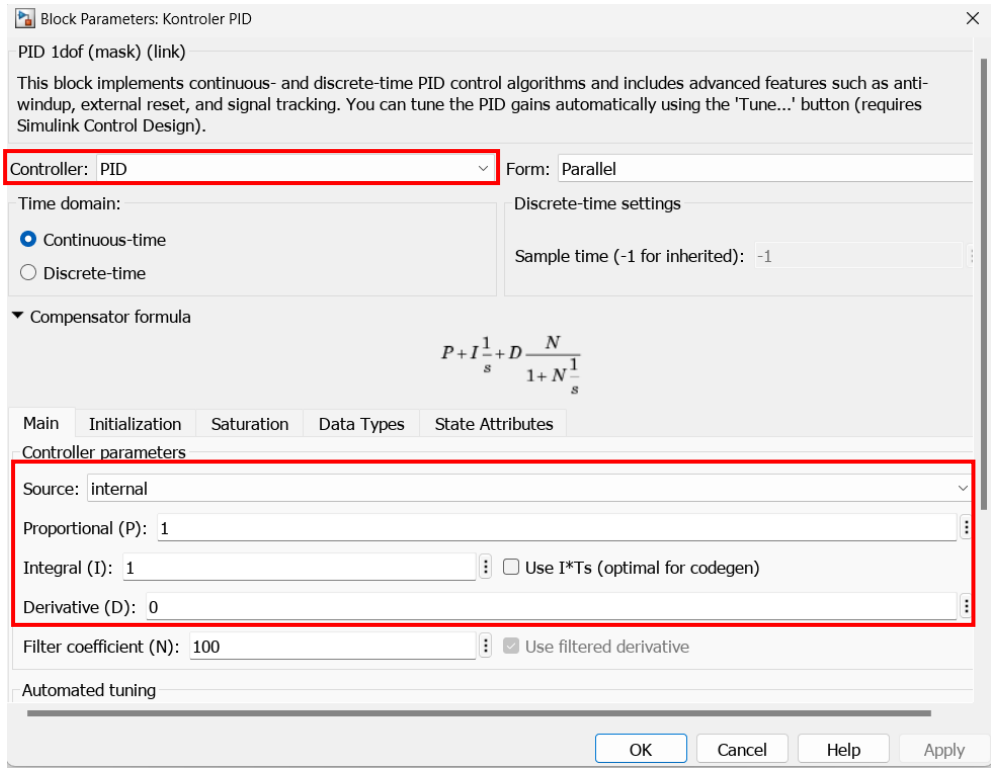
Terdapat beberapa cara untuk melakukan desain atau *tuning* kontroler PID. Berikut akan dibahas beberapa cara

3.2.1 *Model-Based Tuning*

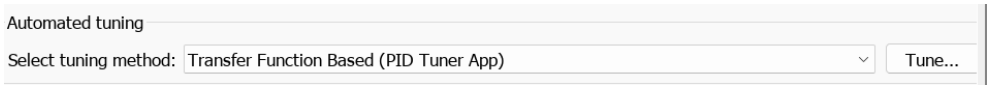
Pada metode pertama ini kita akan gunakan *tool* 'PID Tuner' yang tersedia pada blok SIMULINK. Langkah pertama yaitu merangkai diagram blok sistem *closed loop* dengan kontroler menggunakan blok '*PID Controller*'.



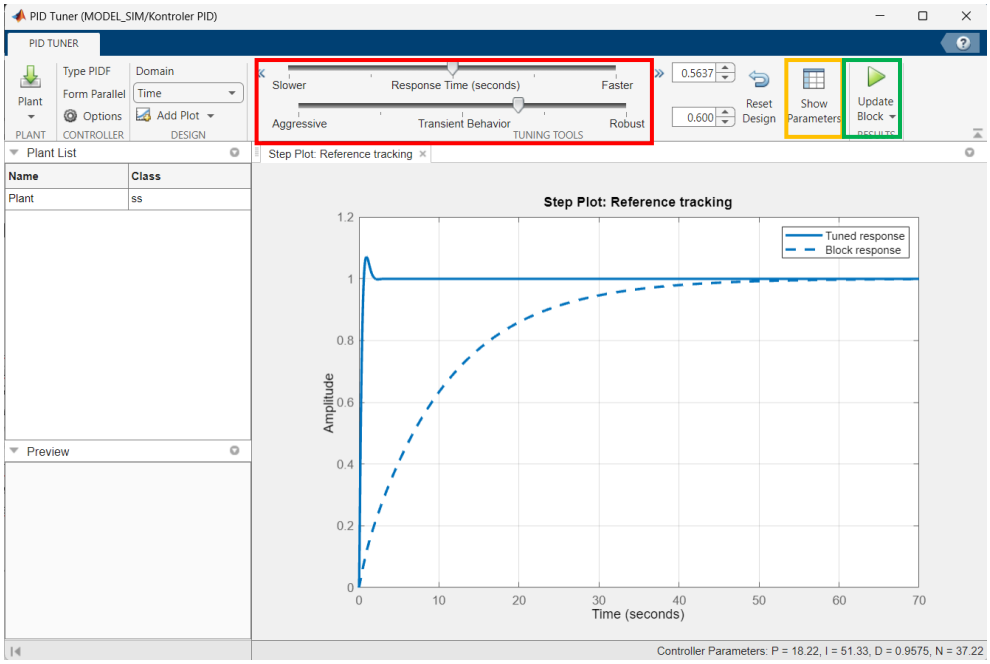
Selanjutnya *double click* pada blok kontroler PID sebelumnya sehingga tampil menu seperti gambar di bawah



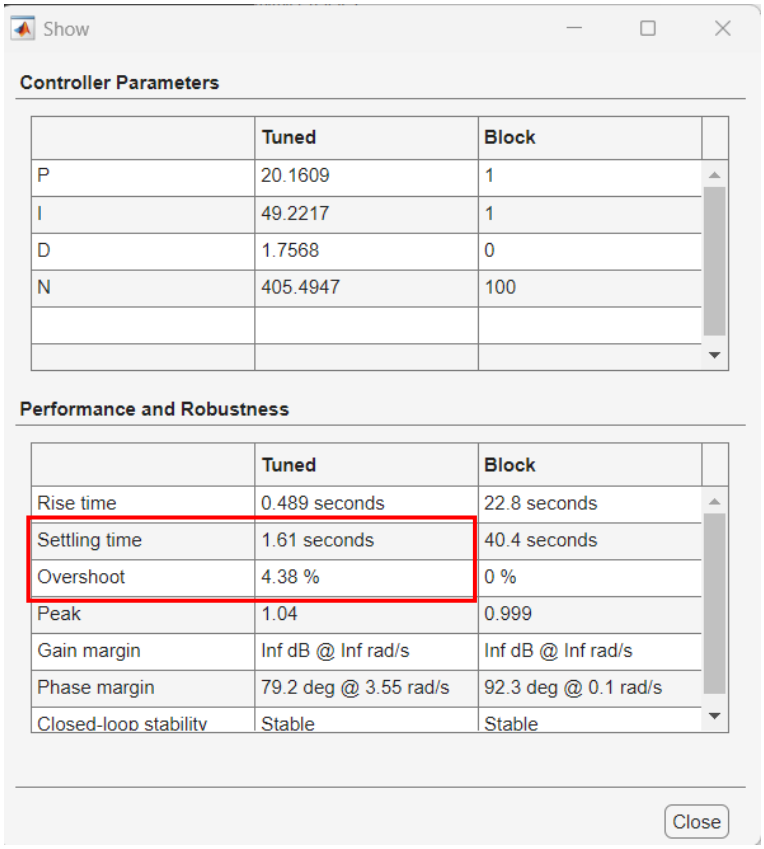
Apabila sebelumnya kita telah mendapatkan *gain* dari kontroler PID maka kita dapat menginputkan langsung pada bagian kotak merah. Pada bagian '*Source*', kita dapat mengatur apakah nilai *gain* PID langsung dituliskan pada menu tersebut (internal) atau melalui sinyal pada simulasi (eksternal).



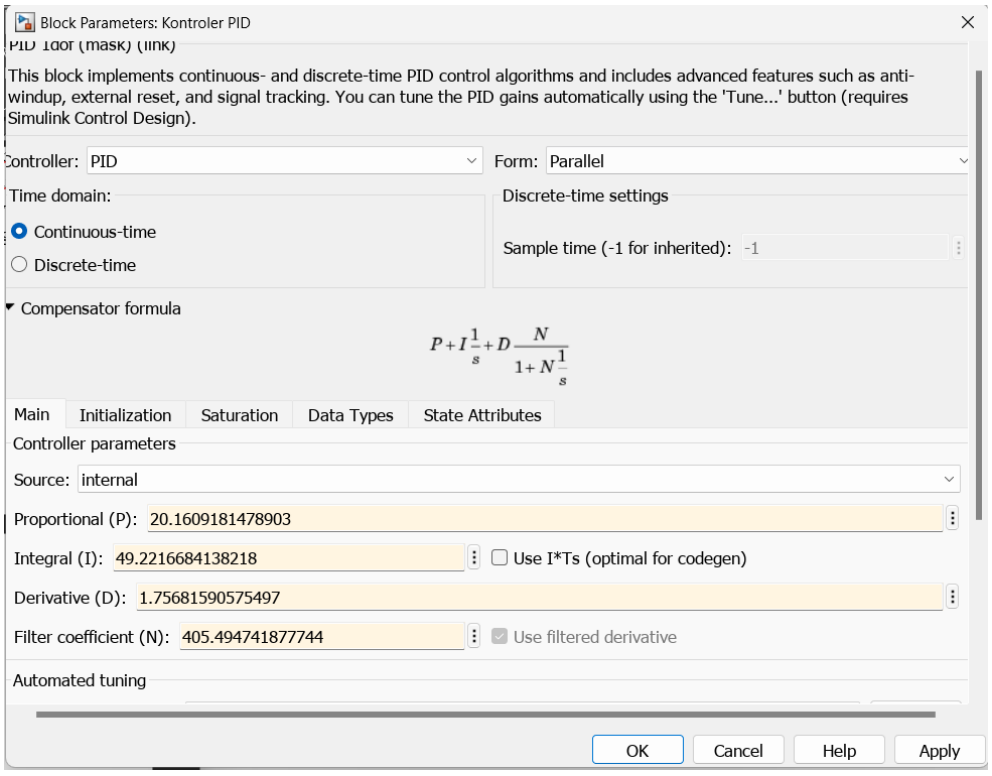
Kita harus menentukan terlebih dahulu jenis kontroler PID yang akan digunakan dimana pada contoh ini akan digunakan jenis PID. Untuk melakukan *tuning*, *scroll* menu hingga terdapat bagian '*Automated tuning*' dan klik tombol *Tune*. Menu *tool* 'PID Tuner' akan muncul seperti gambar di bawah



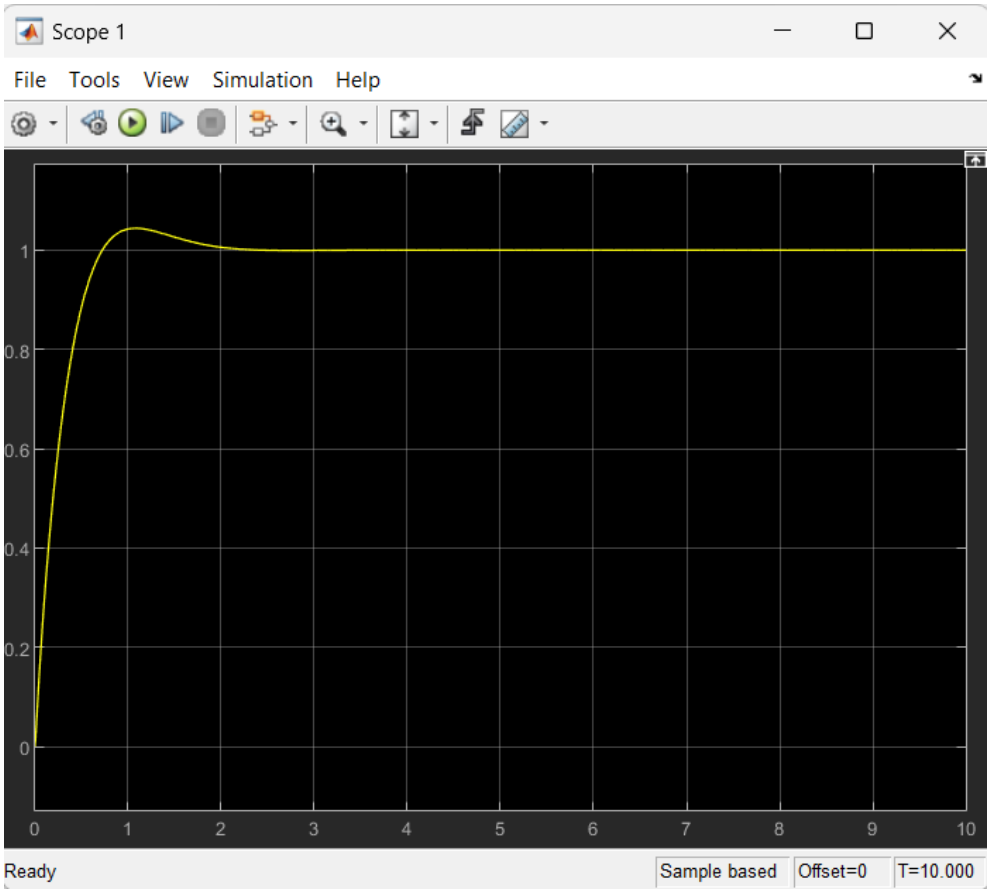
Terdapat beberapa bagian yang penting pada menu tersebut. Bagian yang diberikan kotak merah merupakan bagian untuk melakukan *tuning* berdasarkan *response time* dan *transient behavior*. *Response Time* akan mempengaruhi waktu dari respon dimana apabila semakin ke kanan (*faster*) maka respon semakin cepat dan diiringi *overshoot*. Sedangkan *Transient Behavior* mempengaruhi karakteristik transiennya dimana apabila semakin ke kanan (*robust*) maka sistem semakin *robust* dan *overshoot* semakin berkurang. Tetapi respon sistem juga menjadi lebih lambat. Kemudian bagian yang diberikan kotak berwarna kuning dapat digunakan untuk melihat nilai dari *gain* PID yang didapat serta performansi dari sistem setelah *tuning*. Berikut adalah salah satu contoh hasil *tuning* yang memenuhi spesifikasi yang diinginkan



Setelah didapat hasil yang sesuai maka kita dapat mengklik pada bagian kotak berwarna hijau atau '*Update Block*' untuk menerapkan hasil *tuning* pada blok '*PID Controller*' sebelumnya



Bagian yang berwarna kuning merupakan bagian yang telah di-*update* setelah dilakukan *tuning*. Kemudian klik tombol OK dan jalankan SIMULINK.



Kita juga dapat melakukan simulasi kontroler PID menggunakan blok-blok sederhana pada SIMULINK atau tanpa menggunakan blok '*PID Controller*'. Pada pembahasan sebelumnya diketahui bahwa fungsi alih dari kontroler PID yaitu

$$\frac{U(s)}{E(s)} = K_p + K_i\frac{1}{s} + K_d s$$

Dari persamaan tersebut, kita membutuhkan beberapa elemen yaitu 1 buah blok '*Sum*', 3 buah blok '*Gain*', 1 buah blok '*Integrator*', dan 1 buah blok '*Derivative*'. Tetapi pada pengaturan blok '*PID Controller*' dapat dilihat bahwa fungsi alih dari kontroler PID yaitu

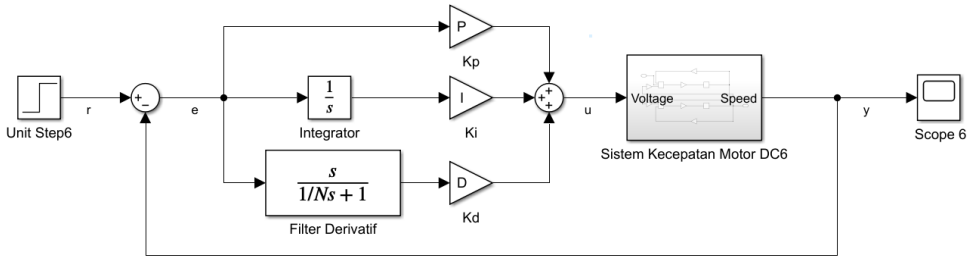
$$\frac{U(s)}{E(s)} = P + I\frac{1}{s} + D\frac{N}{1+N\frac{1}{s}} = P + I\frac{1}{s} + D\frac{s}{\frac{1}{N}s+1}$$

dimana $P = K_p, I = K_i, D = K_d$. Dapat dilihat bahwa fungsi derivatif diubah menjadi fungsi alih yaitu $\left(\frac{s}{\frac{1}{N}s+1}\right)$ dengan nilai N yang besar. Fungsi tersebut merupakan filter derivatif. Pada **sistem nyata**, kita tidak dapat mewujudkan derivatif murni karena *gain* menjadi *infinity* pada frekuensi *infinity* sehingga

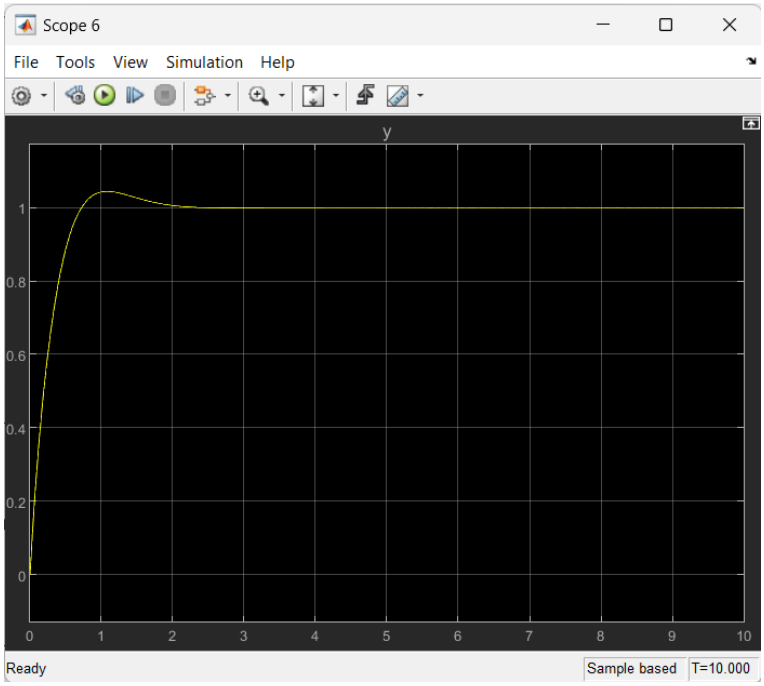
tidak dapat dihitung derivatif titik pada waktu tertentu dengan titik tunggal. Blok '*Derivative*' kita ubah menjadi blok '*Transfer Fcn*'. Berikut adalah kode untuk memasukkan *gain* yang didapat

```
P = 20.1609;
I = 49.2216;
D = 1.7568;
N = 405.4947;
```

Setelah itu, kita rangkai diagram blok sistem dengan kontroler sesuai dengan persamaan terakhir sebagai berikut



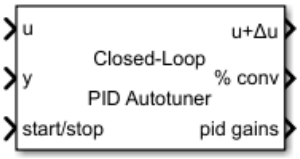
Nilai parameter dari masing-masing blok '*Gain*' disesuaikan dengan nama variabel *gain* yang diinputkan sebelumnya yaitu P, I, dan D. Parameter dari blok '*Sum*' diubah menjadi $+ + +$ karena dibutuhkan 3 penjumlahan sinyal. Kemudian pada blok '*Transfer Fcn*' disesuaikan dengan persamaan sebelumnya dimana pembilang yaitu $[1 \quad 0]$ dan penyebut yaitu $[1/N \quad 1]$. Setelah dijalankan simulasi dari diagram blok sistem tersebut maka dihasilkan respon sebagai berikut



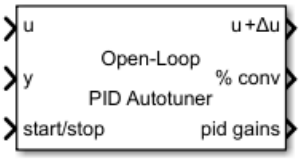
Dapat dilihat bahwa respon yang dihasilkan sama dengan respon dengan menggunakan blok '*PID Controller*'. Sehingga kita dapat membentuk berbagai fungsi dengan blok-blok sederhana pada SIMULINK.

3.2.2 *Real-Time Autotuning*

Pada SIMULINK terdapat 2 blok yang bernama '*Open-Loop PID Autotuner*' dan '*Closed-Loop PID Autotuner*'. Kedua blok tersebut dapat digunakan untuk melakukan *tuning* kontroler PID pada suatu *plant* atau sistem yang nyata ataupun virtual untuk mencapai *bandwidth* dan *phase margin* tertentu.

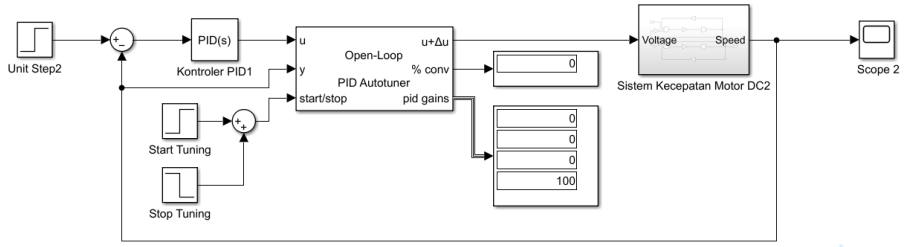


Perbedaannya yaitu seperti pada namanya, blok '*Closed-Loop PID Autotuner*' digunakan pada skema *closed loop* yang stabil dengan adanya kontroler PID awal.



Sedangkan blok '*Open-Loop PID Autotuner*' ini dapat digunakan tanpa adanya kontroler PID awal dan model parametrik dari *plant*. Kedua blok harus digunakan pada sistem yang stabil. Khusus untuk blok '*Open-Loop PID Autotuner*', sistem tidak boleh memiliki integrator lebih dari 1. Pada intinya, blok '*Open-Loop PID Autotuner*' digunakan untuk mendapatkan ***tuning* awal** dari kontroler PID dan '*Closed-Loop PID Autotuner*' digunakan untuk **penyempurnaan *tuning***. Berikut adalah langkah yang ditempuh untuk melakukan *tuning* dengan menggunakan blok '*Open-Loop PID Autotuner*'

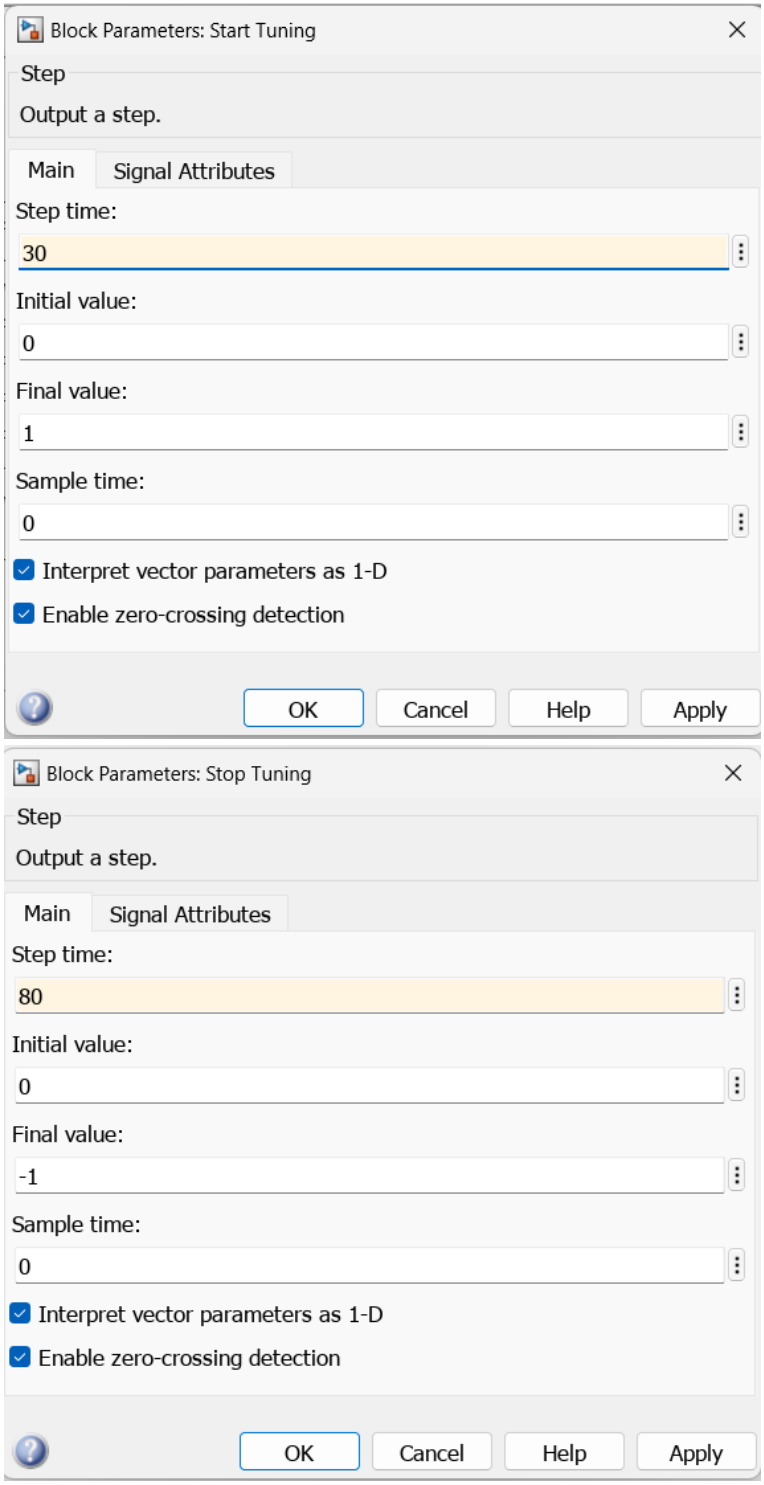
1. Susunlah diagram blok sistem seperti pada gambar di bawah dimana terdapat tambahan yaitu 1 blok '*PID Controller*', 2 blok '*Display*', 1 blok '*Sum*', 2 blok '*Step*', dan 1 blok '*Open-Loop PID Autotuner*'.



Fungsi blok '*Display*' yaitu untuk menampilkan nilai sinyal yaitu %konvergensi dan nilai *gain* PID hasil *tuning*. Blok '*Step*' dan '*Sum*'

digunakan untuk mengatur waktu pada simulasi dimana dilakukan atau tidaknya *tuning*.

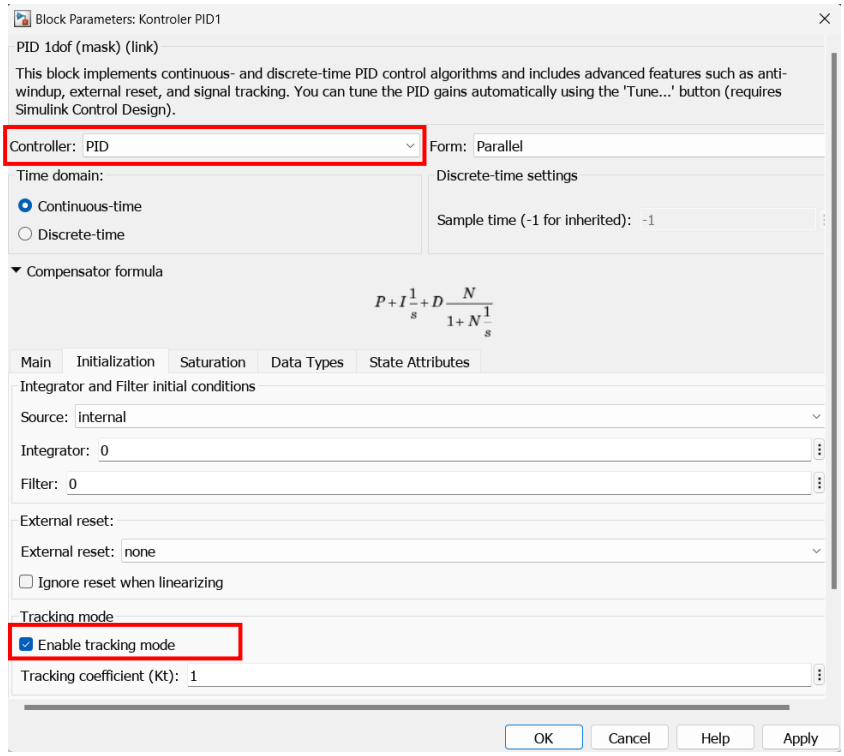
2. Sesuaikan parameter dari blok '*Step*' sebagai berikut



Gambar atas merupakan pengaturan untuk *start tuning* dimana terjadi pada waktu simulasi yaitu detik ke-30. Blok '*Open-Loop PID Autotuner*' akan melakukan *tuning* ketika diberi input bernilai 1 dan

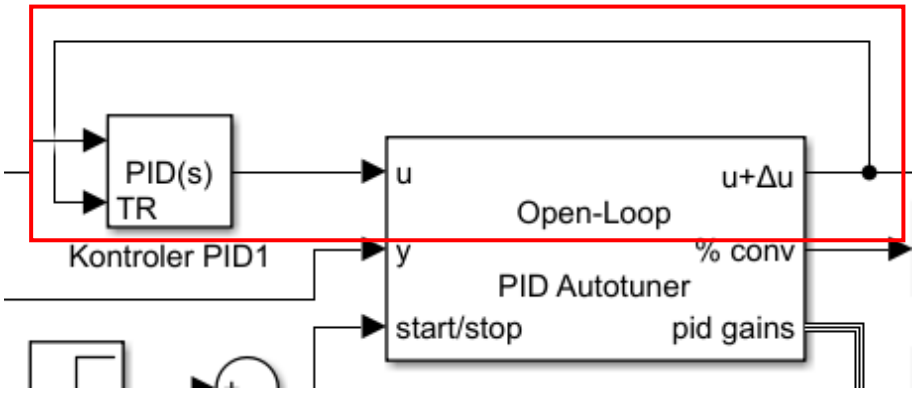
berhenti ketika bernilai 0. Sehingga gambar bawah diberikan *Final value* sebesar -1 (*summing* menjadi 0) pada detik ke-80.

3. Ubahlah parameter blok '*PID Controller*' mulai dari jenis kontroler yang akan digunakan serta *tracking mode* dihidupkan



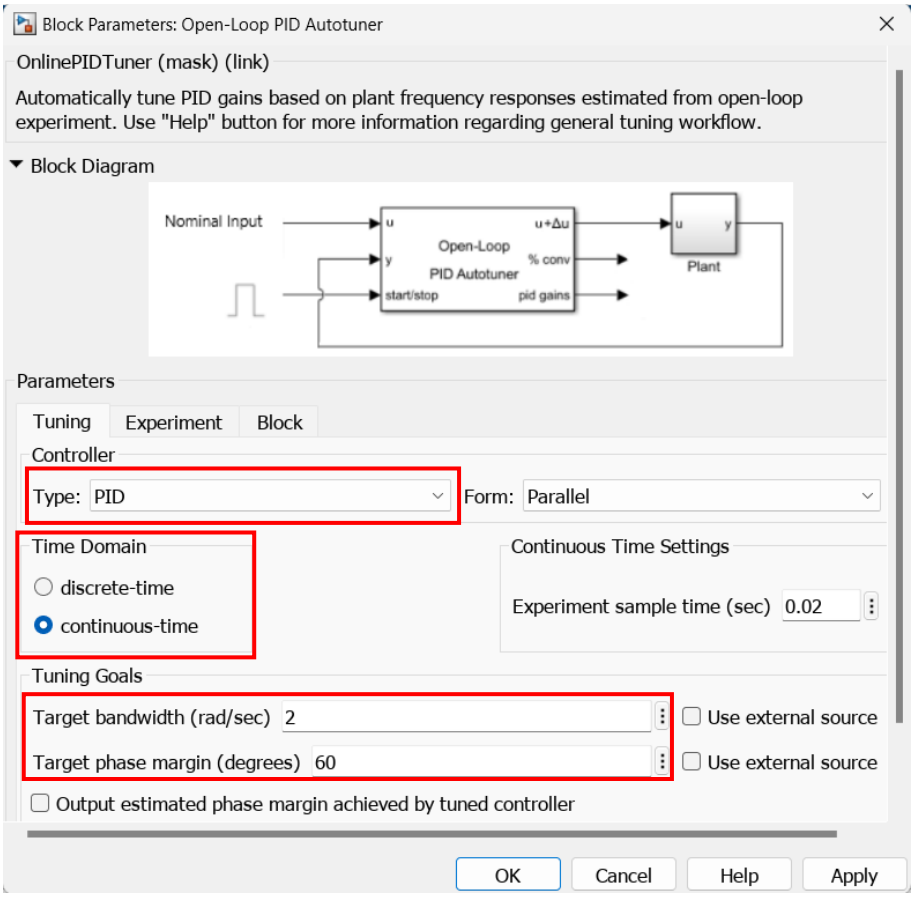
Tracking mode dihidupkan agar kontroler PID mengikuti sinyal referensi atau input yang digunakan mengingat skema *open loop* digunakan. Pada percobaan ini akan digunakan kontroler PI saja.

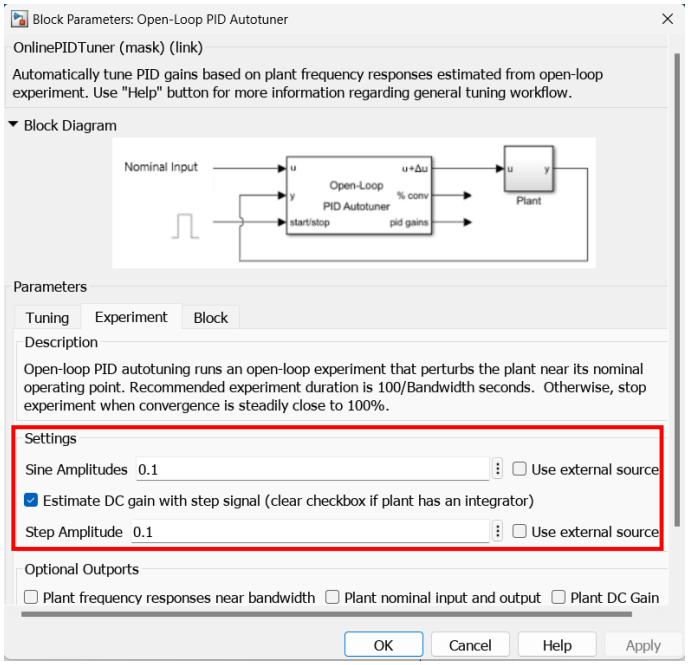
4. Sambungkan kabel $u + \Delta u$ terhadap input TR atau *tracking* pada kontroler PID



5. Ubahlah parameter dari blok '*Open-Loop PID Autotuner*' yaitu jenis dari kontroler PID yang digunakan, *time domain* yaitu kontinu, serta *tuning goals*. *Target bandwidth* menentukan seberapa cepat kontroler untuk merespon dimana digunakan 2 rad/s untuk *rise time* sebesar 1

detik. Sedangkan *Target phase margin* menentukan seberapa *robust* kontroler dimana digunakan 60 *degrees* untuk *overshoot* sekitar 5%.





Kemudian pada bagian *Experiment* diatur yaitu *Sine Amplitudes* dan *Step Amplitude* sebesar 0.1. *Sine Amplitudes* digunakan untuk menentukan amplitudo dari sinyal sinus *injected* yang berfungsi untuk

sinyal *tuning*. *Step Amplitude* menentukan besar amplitudo dari sinyal step *injected*. Opsi ini dimatikan dengan mematikan *Estimate DC gain* untuk sistem atau *plant* dengan **integrator tunggal**. Sistem kecepatan motor DC tidak memiliki integrator tunggal. Contoh sistem dengan integrator tunggal yaitu sistem **posisi** motor DC dimana posisi adalah $\Theta_{(s)} = \frac{1}{s}\dot{\Theta}_{(s)}$ sehingga *transfer function* nya yaitu

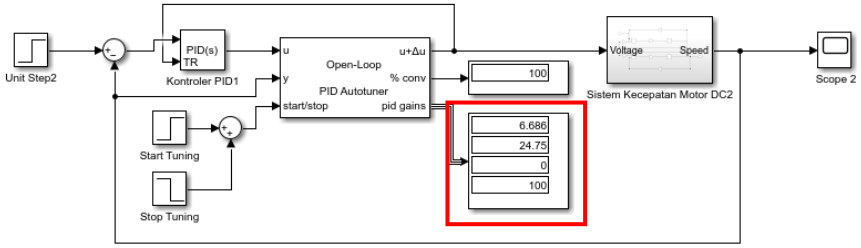
$$\frac{\dot{\Theta}_{(s)}}{V_{(s)}} = \frac{K}{JLs^2 + (JR + bL)s + bR + K^2}$$

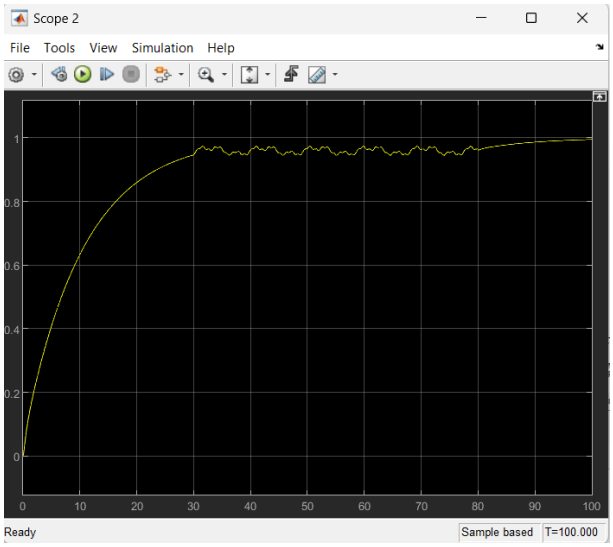
$$\frac{s\Theta_{(s)}}{V_{(s)}} = \frac{K}{JLs^2 + (JR + bL)s + bR + K^2}$$

$$\frac{\Theta_{(s)}}{V_{(s)}} = \frac{K}{s(JLs^2 + (JR + bL)s + bR + K^2)}$$

Dapat dilihat bahwa pada bagian penyebut terdapat faktor s atau integrator tunggal.

6. Jalankan simulasi dan amati respon yang terjadi.

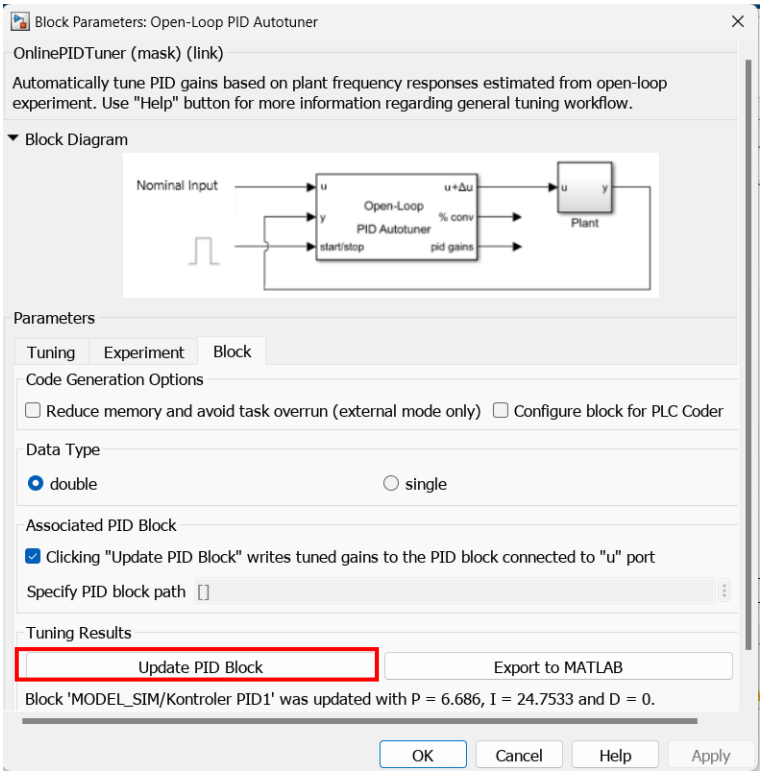




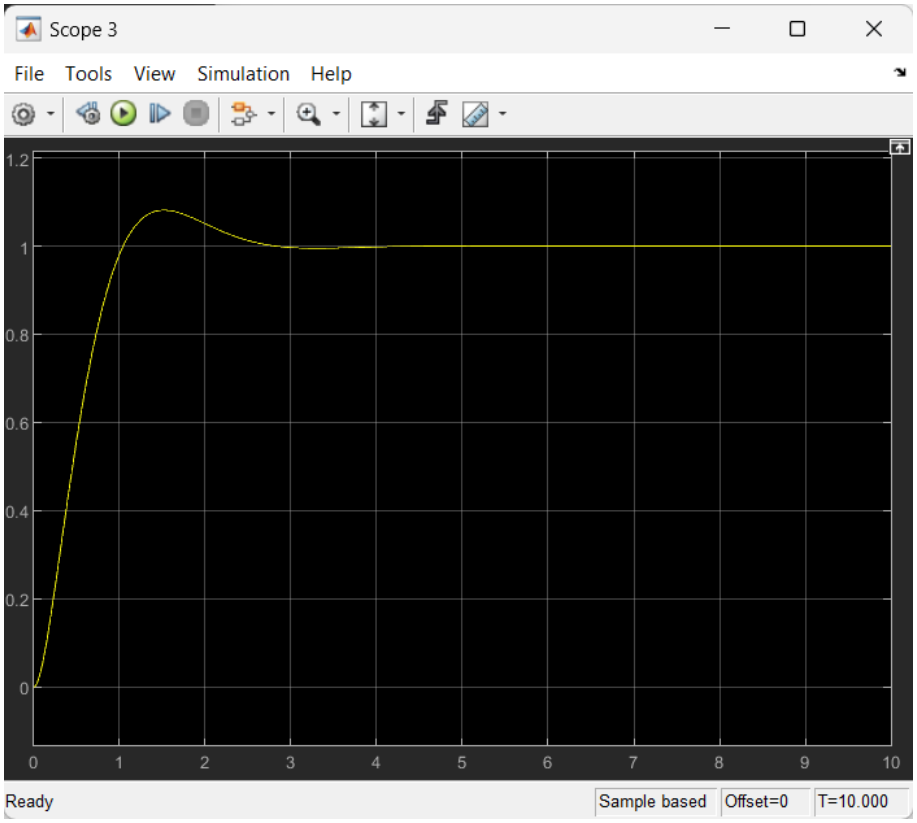
Pada blok '*Display*' yang ditandai kotak merah merupakan parameter *tuning* PID yang didapat yaitu K_p sebesar 6.686 dan K_i sebesar 24.75. Baris pertama untuk nilai K_p, baris kedua untuk nilai K_i, baris ketiga untuk nilai K_d, dan baris keempat untuk nilai N. Respon sistem yang didapat pada *scope* dapat dilihat bahwa *tuning* terjadi pada detik ke-30

(ditandai adanya amplitudo dari sinyal *injected*) dan detik ke-80 berhenti.

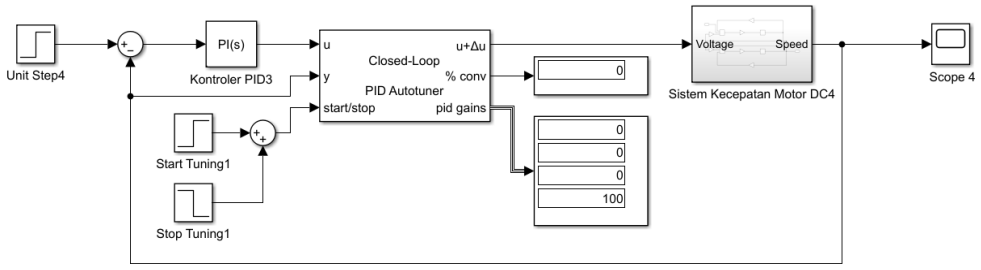
7. *Update* parameter dari kontroler PID dengan menggunakan opsi pada blok '*Open-Loop PID Autotuner*'



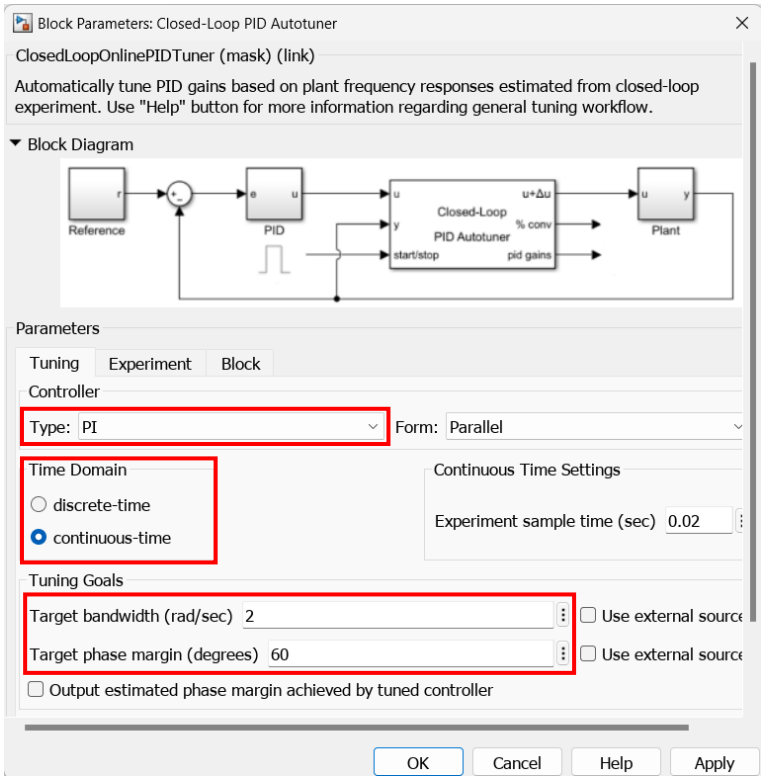
Notifikasi akan muncul pada bawah opsi tersebut. Dapat dilihat bahwa *gain* K_d bernilai 0 sehingga nantinya hanya digunakan kontroler PI saja tanpa adanya D (derivatif). Apabila diterapkan kontroler PID yang telah di*tuning* tersebut maka didapat respon sistem sebagai berikut



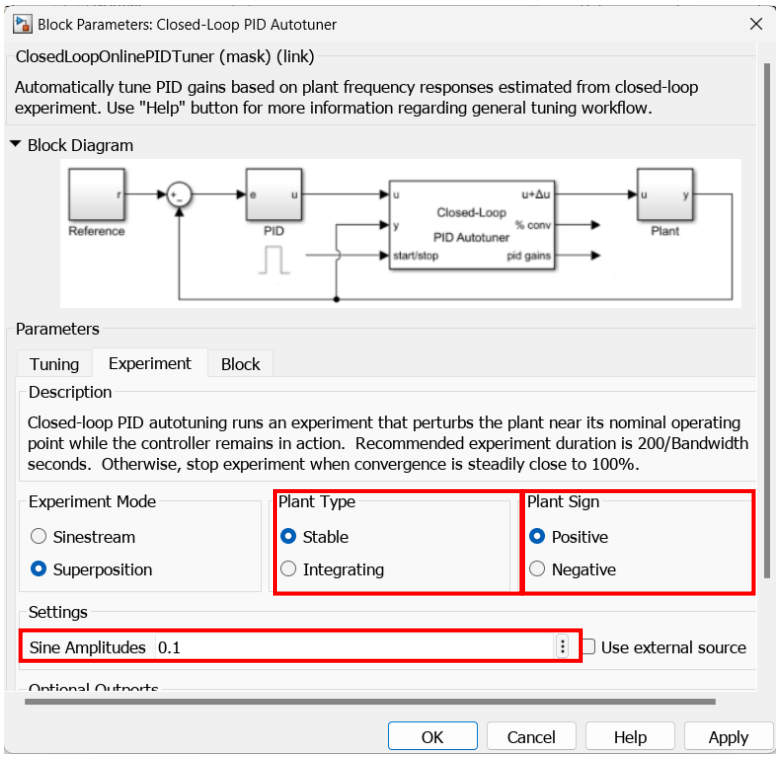
Respon sistem yang dihasilkan dengan menggunakan kontroler PID *tuning* awal cukup memuaskan dilihat dari respon waktu, *overshoot*, *error steady state* yang dihasilkan. T*uning* dengan blok '*Closed-Loop PID Autotuner*' akan dilakukan untuk melakukan penyempurnaan. Langkah yang ditempuh kurang lebih sama dengan sebelumnya. Perbedaannya yaitu pada jenis kontroler PID yang digunakan (hanya PI saja) serta *tracking mode* pada kontroler dimatikan. Diagram blok sistem yang digunakan yaitu



Berikut adalah pengaturan pada blok '*Closed-Loop PID Autotuner*' pada bagian *tuning* dimana yang diubah hanya bagian jenis kontroler, *time domain*, serta *tuning goals*. Efek dari *tuning goals* sama dengan sebelumnya sehingga nilainya dibuat sama.

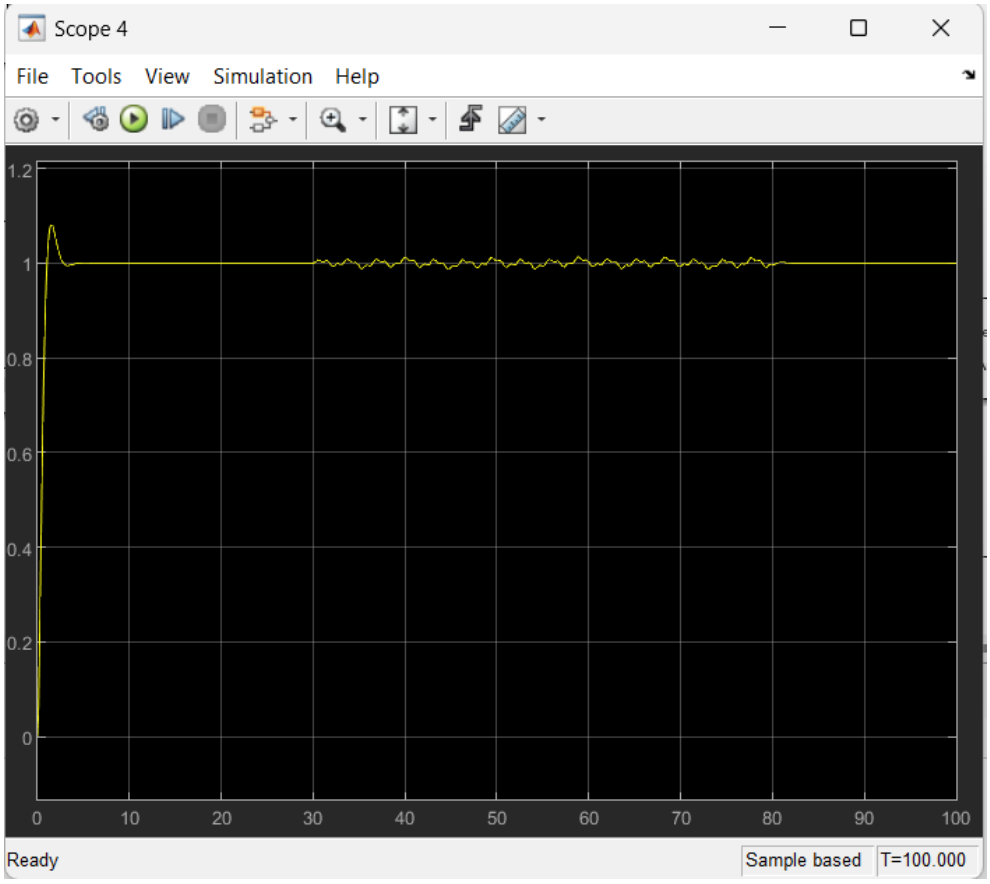


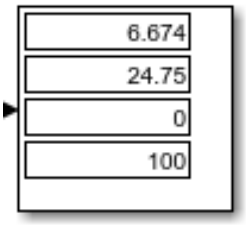
Kemudian pada bagian pengaturan *Experiment* yang diubah yaitu *plant type*, *plant sign*, serta *sine amplitudes*.



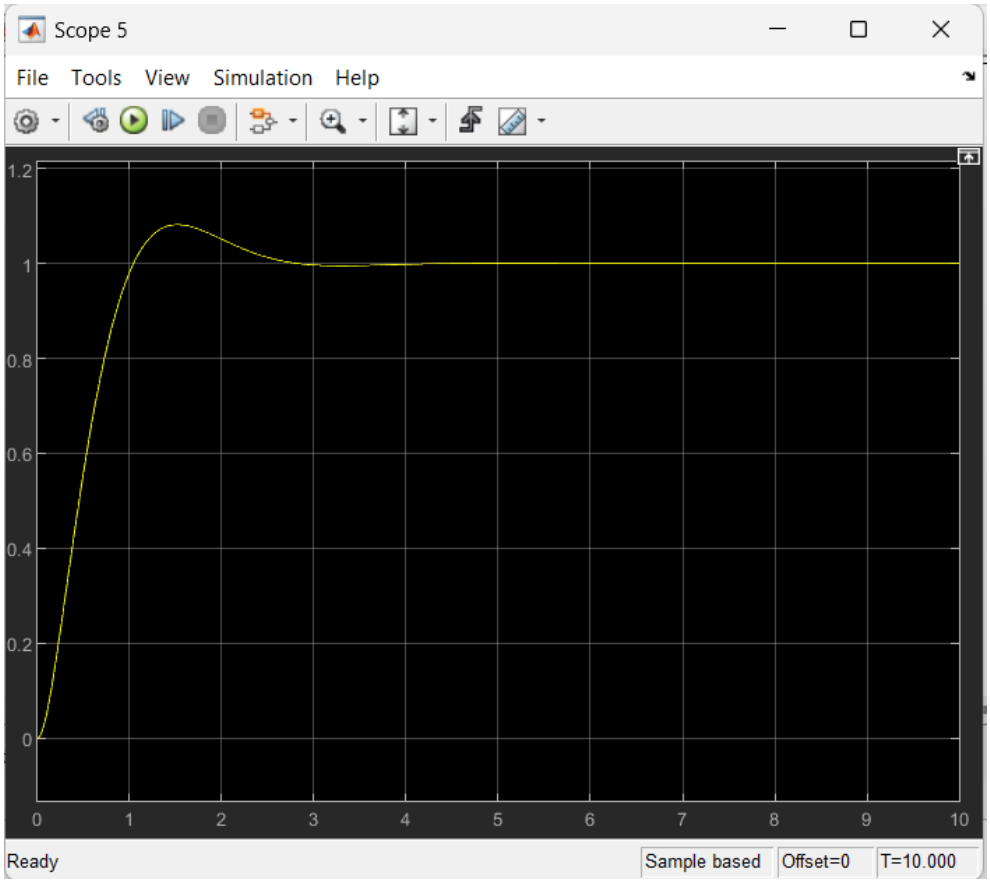
Pada ***plant type*** tetap *stable* karena sistem yang digunakan tidak memiliki integrator. ***Plant sign*** bernilai positif jika perubahan positif pada input sistem pada titik *nominal operating* menghasilkan perubahan positif pada output ketika sistem mencapai *steady state* yang baru serta begitu juga untuk negatif.

Sedangkan ***sine amplitudes*** digunakan untuk menentukan amplitudo sinyal sinus *injected* sehingga nilainya dibuat sama seperti sebelumnya. Setelah dijalankan maka didapatkan nilai *gain tuning* dan respon saat *tuning* sebagai berikut





Apabila diterapkan kontroler PID yang telah di*tuning* tersebut maka didapat respon sistem sebagai berikut



Dapat dilihat bahwa respon yang dihasilkan dengan kontroler PID *tuning* terbaru kurang lebih sama dengan *tuning* dengan '*Open-Loop PID Autotuner*'. Hal tersebut dikarenakan hasil *tuning* pertama telah menunjukkan hasil yang memuaskan. Tetapi apabila dibandingkan dengan hasil *tuning* dengan metode *model-based* maka *gain* pada *tuning* metode *real-time* lebih kecil dengan respon pada *model-based* yang **sedikit** lebih baik.