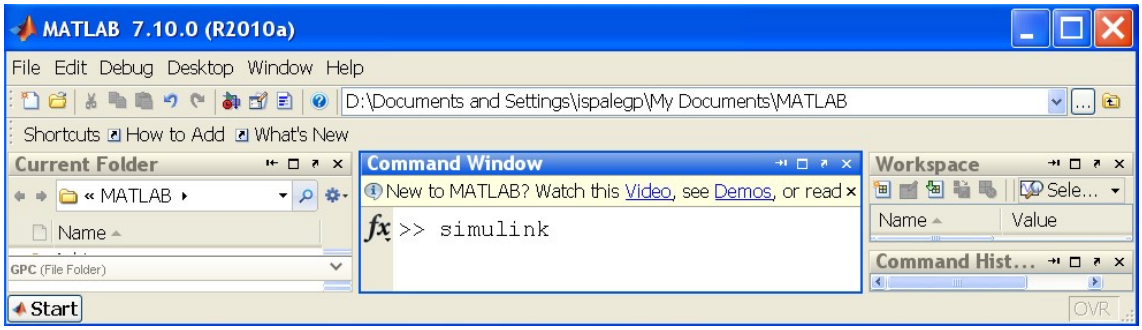
LEHEN URRATSAK Simulink-ekin (MatLab)

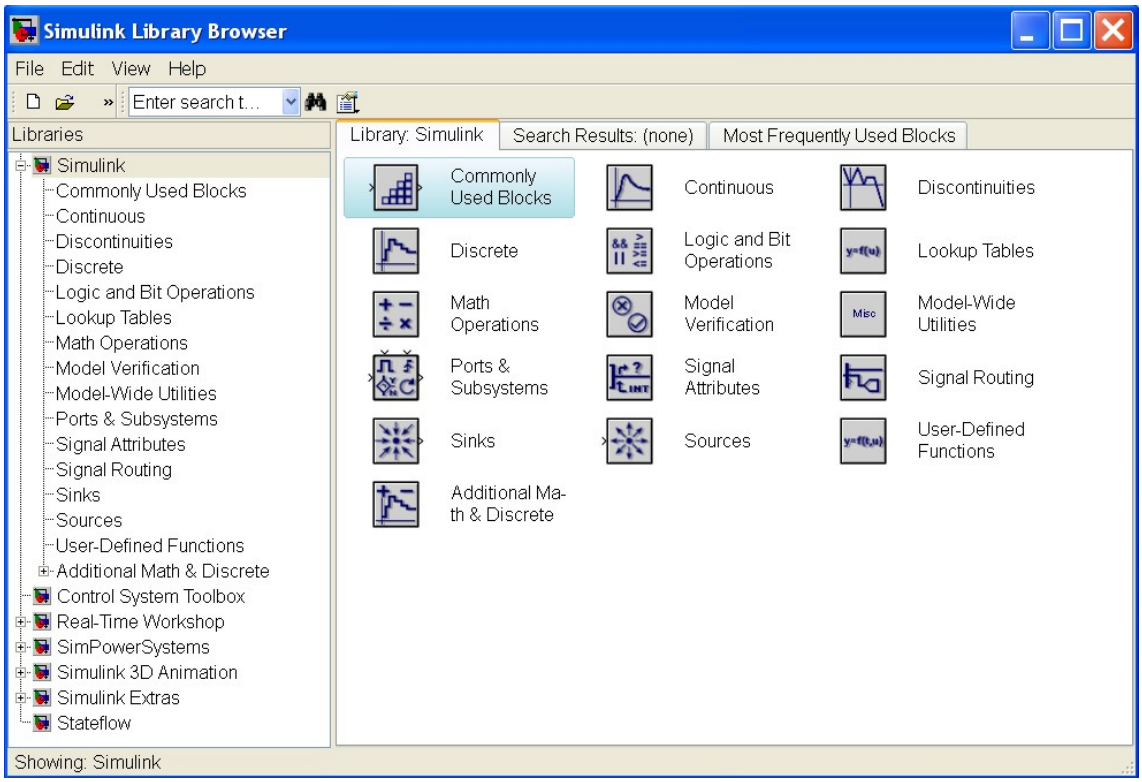
Patxi Alkorta Egiguren & Esteban Jacob Taquet (2016)

Simulink MatLab-en ingurune grafikoa da, sistema lineal nahiz ez linealen portaera oso modu eroso eta eraginkorrean simulatzeko software tresna. Tresna grafikoa izateak, bere erabilera erraza eta intuitiboa izatea dakartza.

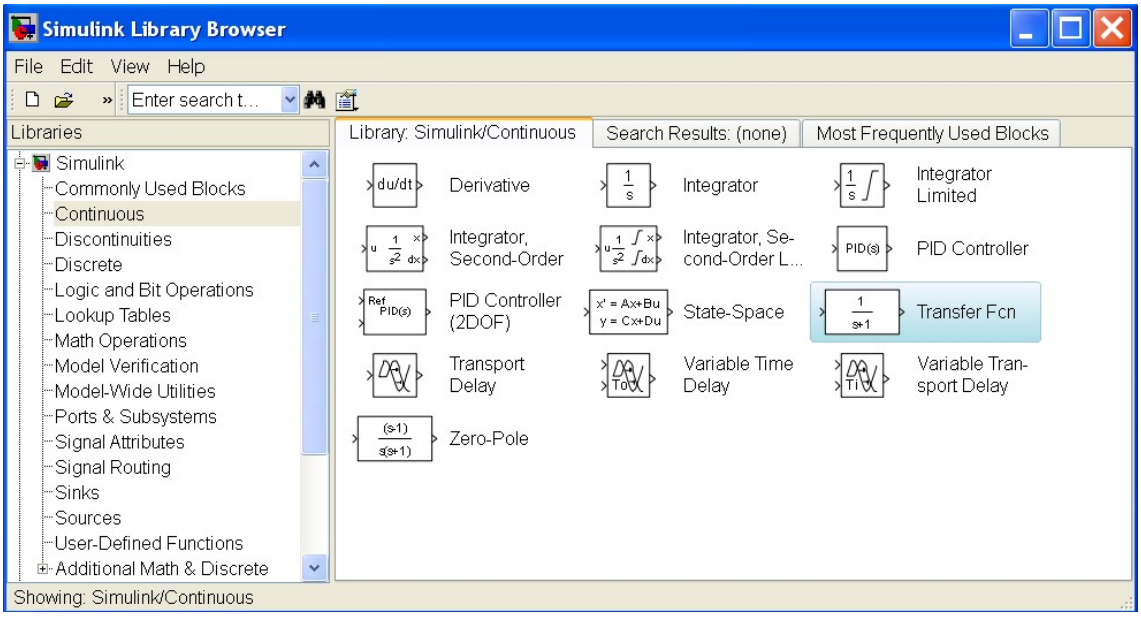
Simulink egikaritu edo deitu ahal izateko *MatLab* ingurunea irekita izan behar da, eta botoia sakatu edo bestela komandoen lerroan (*Command Window*) `>> simulink` aginduaren bidez



Behin *Simulink* zabalduz, hau izango da leihoaren itxura (*Simulink Library Browser*): ezkerraldean instalaturik eta ingurune grafiko honetan erabili litezkeen liburutegiak (*Libraries*) edo *Toolbox*-ak agertzen dira. Eskuinaldean, hainbat erlaitz (pestaña) agertzen dira, non ezkerraldean sailkatuiko liburutegian aukeran dauden objektuak/funtzioak edo objektuen taldeak erakusten dituen: kasu honetan *Simulink* liburutegia sailkatu da, eta eskuinaldean bertan aukeran dauden objektu taldeak agertzen dira, *Library: Simulink*.

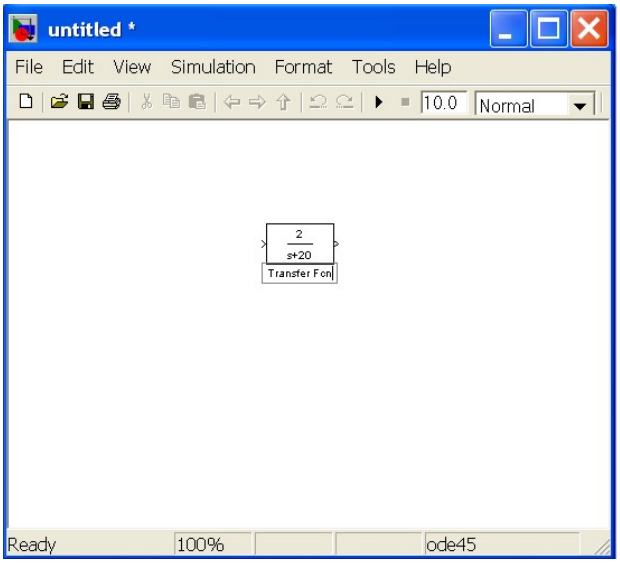


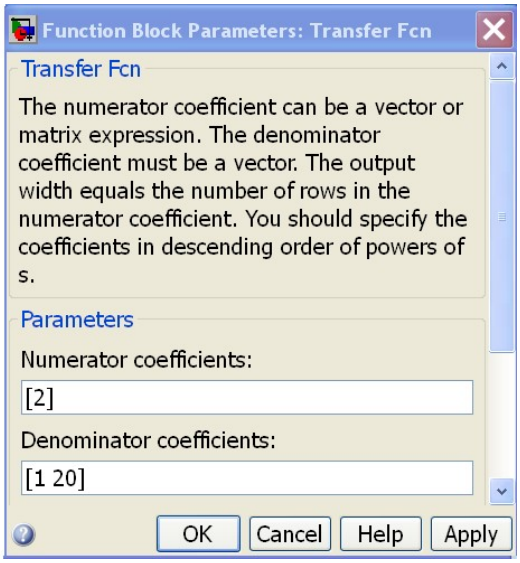
Kontrol-ingeniaritzan gehien erabiltzen den objektuen talde edo familia *Continuous* izenez ezagutzen dena da, denbora jarraituko sistemena.



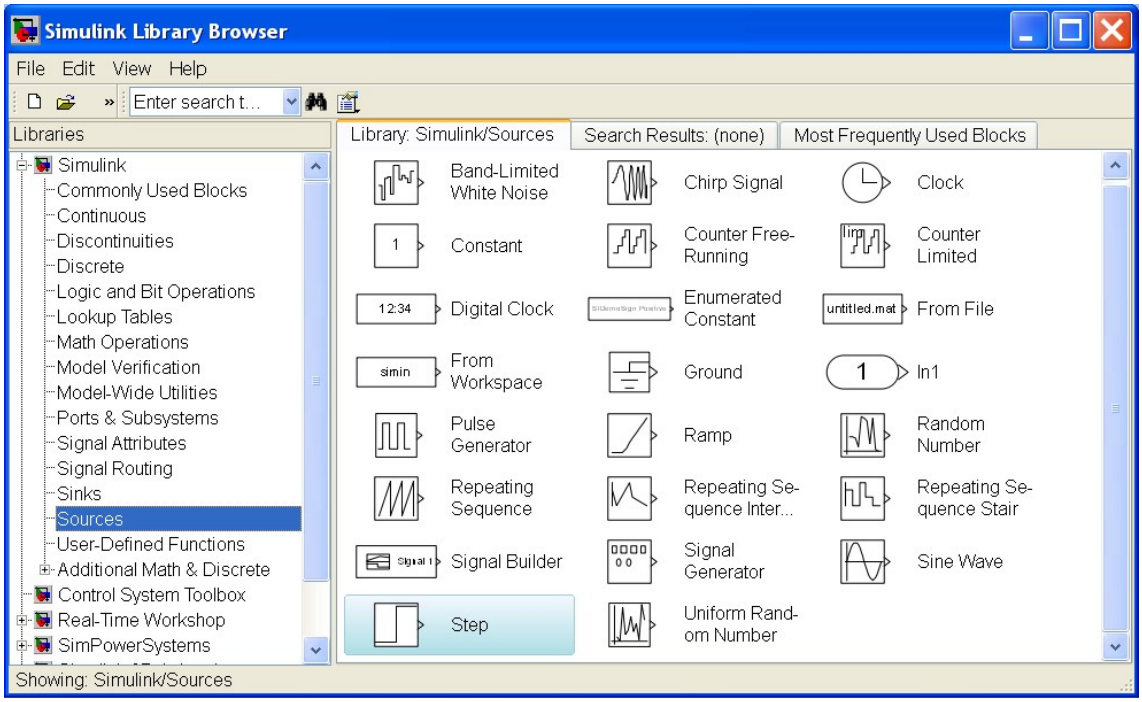
Sistema jakin batekin lan egin ahal izateko, lehenengo *Simulink*-eko fitxategi berri bat zabaldu behar da, non sistemaren blokeen diagrama sortuko den, eta horretarako  botoia sakatu edo bestela *File* menutik *File/New/Model* egikaritu behar da. Jarraian, ingurune honetan sistema bat sortzeko hiru modu desberdin deskribatuko dira: transferentzia funtzioa (G(s)), egoera ekuazioak (EE), eta ekuazio diferentzialak (ED), erabiliz.

Behin fitxategi berri bat zabalduz, eta bistan edukiz leiho hau nahiz *Simulink Library Browser*, *Transfer Function* blokea klikatu eta fitxategi berrira eramango da arrastaka. Orain, suposa dezagun *G(s)=2/(s+20)* transferentzia funtzioa definitu nahi dela, orduan click bikoitza egiten da blokearen gainean eta *G(s)* berriaren parametroak editatzen dira, *Apply* sakatzen da horiek gaurkotzeko, eta parametroen leihoa ixten da *OK* sakatuz. Blokearen beheko aldean clickatuz, *Transfer Function*-en, bere izena editatu eta *G(s)* jartzen da.

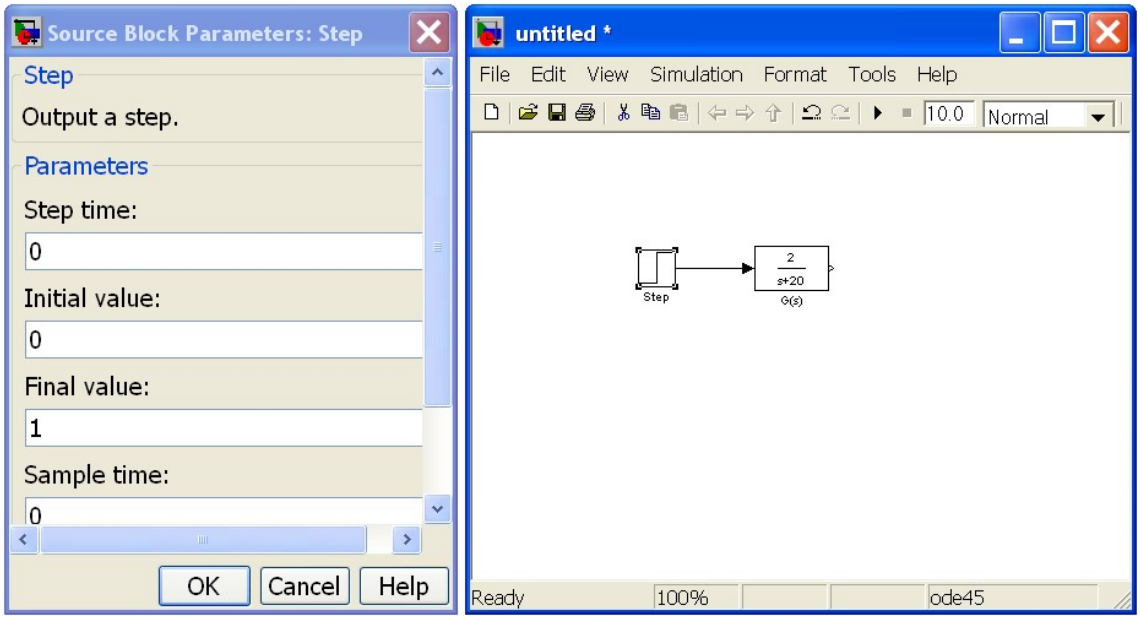




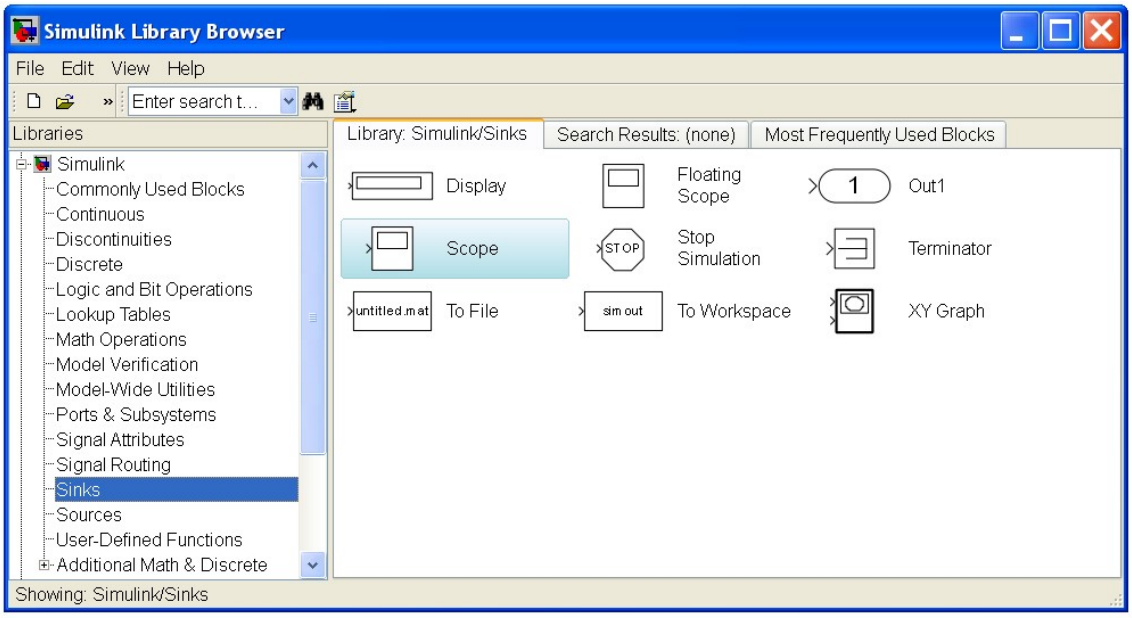
Jarraian, maila unitarioa sarrera sailkatzen da *G(s)*-ren sarreran konektatzeko. Horretarako *Sources* liburutegiko *Step* ojektua sailkatu eta lehengo diagramara arrastatzen da.



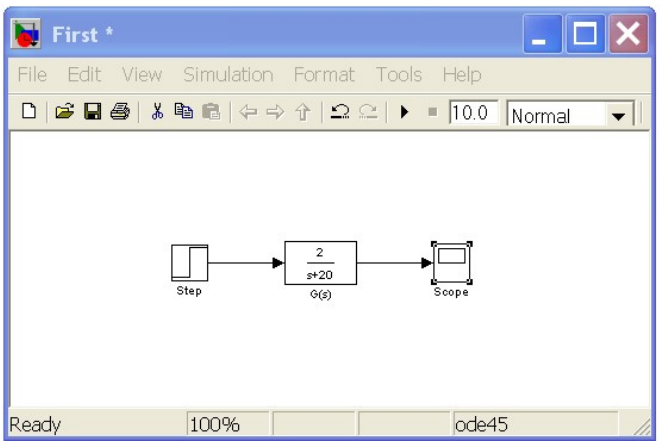
Ondoren, *Step* iturriaren irteeran klikatuz eta *G(s)* blokearen sarrerara arrastatuz eta bertan utziz, sarrera sistemarekin konektatzea lortzen da. Gainera, *Step* iturria editatu eta bere *Step time* parametroa 0 balioarekin jartzen da, non horrela bere *Final value* (1) balioa 0 s-tik aurrera ematen hasten den.

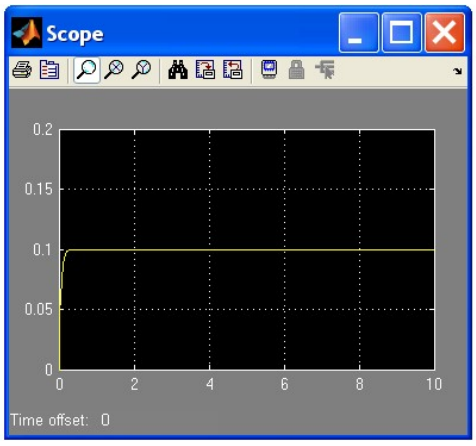


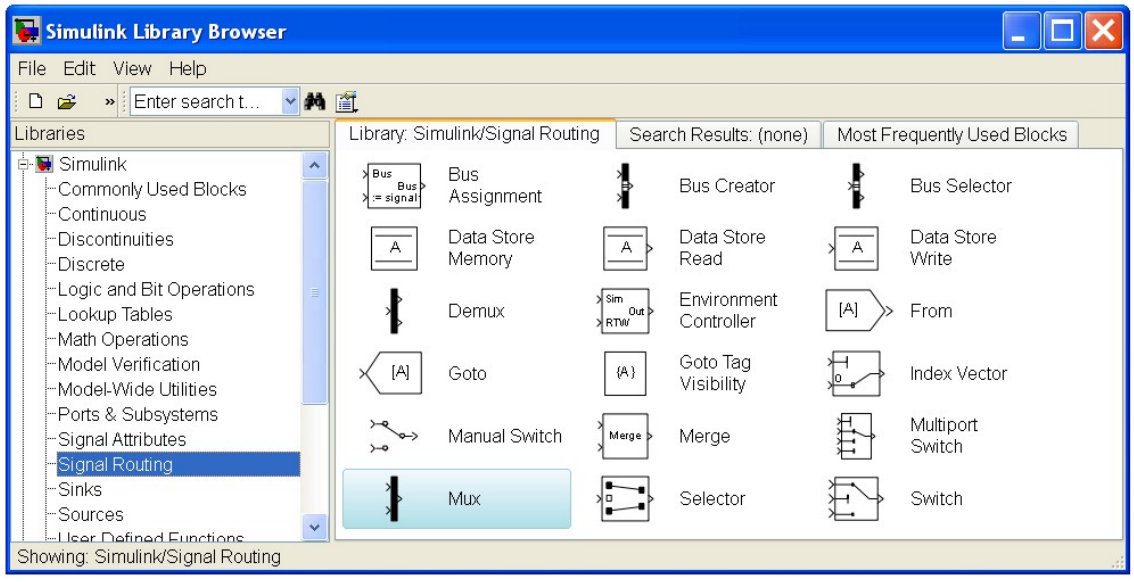
Irteera edo beste seinaleren bat bistaratu ahal izateko, *Scope* bistaratzailea erabiltzen da, non *Sinks* liburutegian aurkitzen den, eta diagramara arrastatzen den.



Jarraian, *G(s)* sistemaren irteeraren eta *Scope* bistaratzailearen artean konexio bat sortzen da, eta diagrama First (*First.mdl*) izenarekin gordetzen da. Ondoren, simulazioa egikaritzen da diagramako ▶ botoia sakatuz. Emaitza, sarrera maila unitarioaren aurreko *G(s)* sistemaren denbora-erantzuna da, eta berari konektaturiko bistaratzailean ikus daiteke.

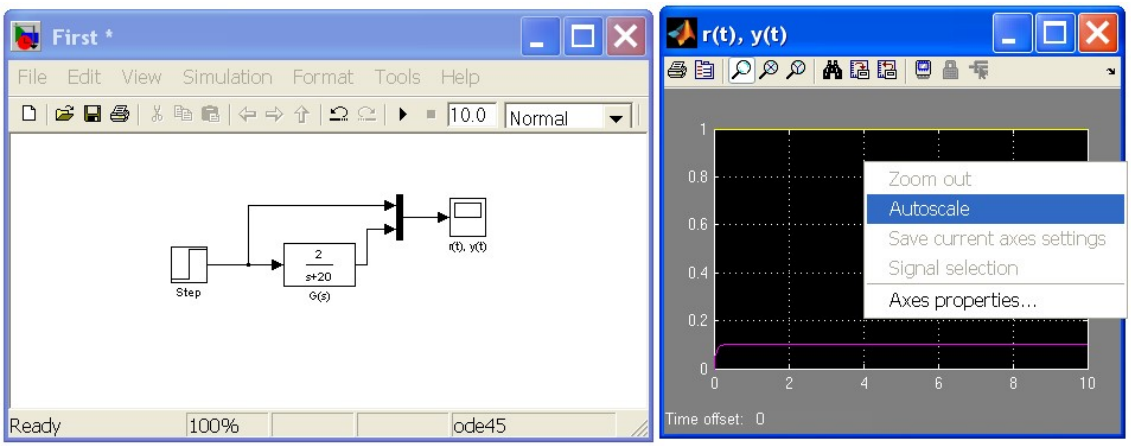




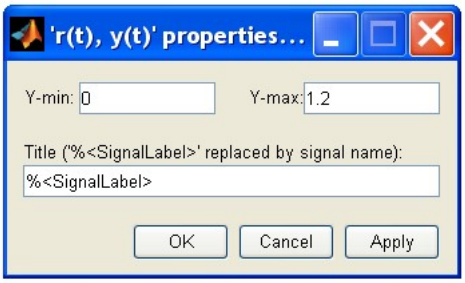


Sarrerako seinalea bistaratzeko, beste *Scope* bistaratzaile bat konektatu daiteke berari, baina askotan hobe izaten da *Scope* bistaratzaile kopuru txikiagoa erabiltzea, non horietako bakoitzean seinale bat baino gehiago bistaratzen den. Horrela, elementu gutxiago dituen diagrama bat edukiko da, eta gainera, seinaleak beraien artean konpara daitezke. *Scope* berean hainbat seinale bistaratu ahal izateko, *Mux* multiplexorea erabiltzen da, non *Signal Routing* liburutegian aurkitzen den.

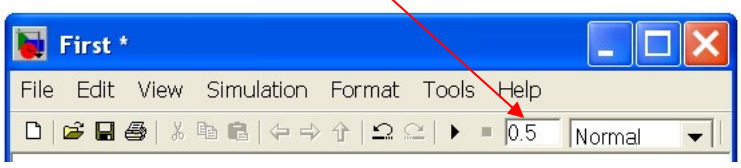
G(s) sistemaren sarrera (*r(t)*) nahiz irteera (*y(t)*) multiplexorearen bi sarreretan konektatuz, eta bere irteera *Scope* bistaratzaileari konektatuz, bi seinaleak bistaratzaile berdinean bistaratu daitezke. Horrela, bistaratzailearen berezko izena *r(t), y(t)* izenarengatik ordezkatzen da. Ikusi daiteke lehen seinalea (sarrera) oria dela, eta bigarrena (irteera), morea. *Mux* gailuaren sarrera kopurua handitu daiteke bertan klik bikoitza eginez eta dagokion balioa jarriz (aurrerago).



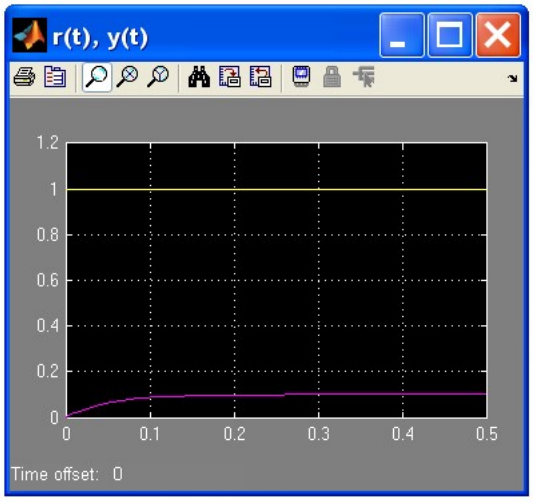
Seinaleren bat ez bada bistaratzen, eskala muga baino handiagoa delako, orduan arratoriaren eskuineko botoiarekin klikatu dezakegu bistaratzailearen gaiean eta *Autoscale* egikaritu. Eskuzko doiketa bat egin nahi bada, orduan *Axes properties…* egikaritu beharko da, eta ondoren *y* ardatzaren balio minimo eta maximoa aukeratu daitezke, adib. *0* eta *1.2*, hurrenez hurren.



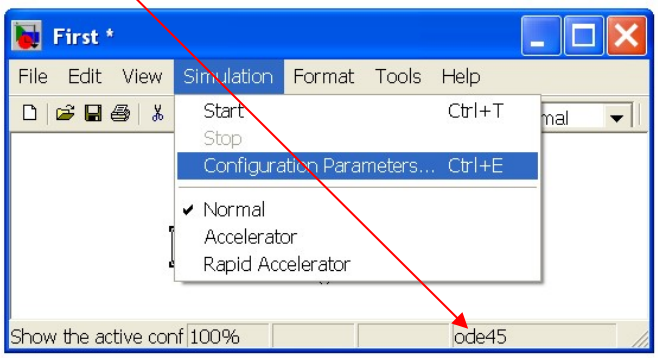
Besterik adierazi ezean (berez), simulazioaren iraupena 10 s-koa da, baina gehienetan komeni da aldatzea, egin behar den entsegu eta sistemaren dinamikaren arabera. Gure kasuan, kontuan izanik aurreko emaitza, argi dago 1 s edo 0.5 s-ko simulazio denbora nahikoa dela. Parametro hau aldatzeko, nahi den balioa dagokion laukian jarri behar da.

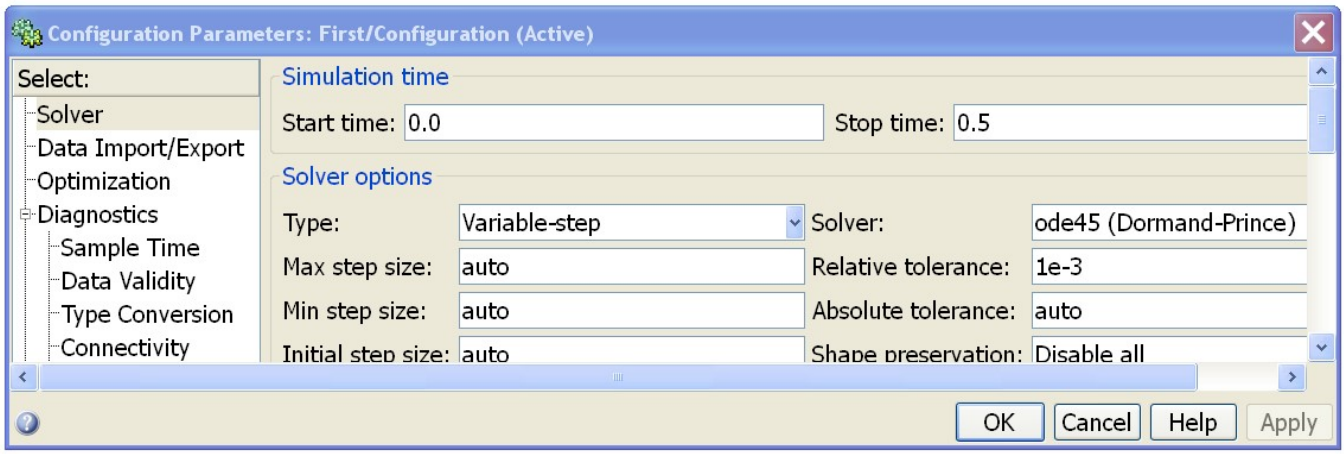


Emaitza bezala, irteeraren grafika handiago bat lortzen da, lehengo bistaratzailean eta informaziorik galdu gabe.



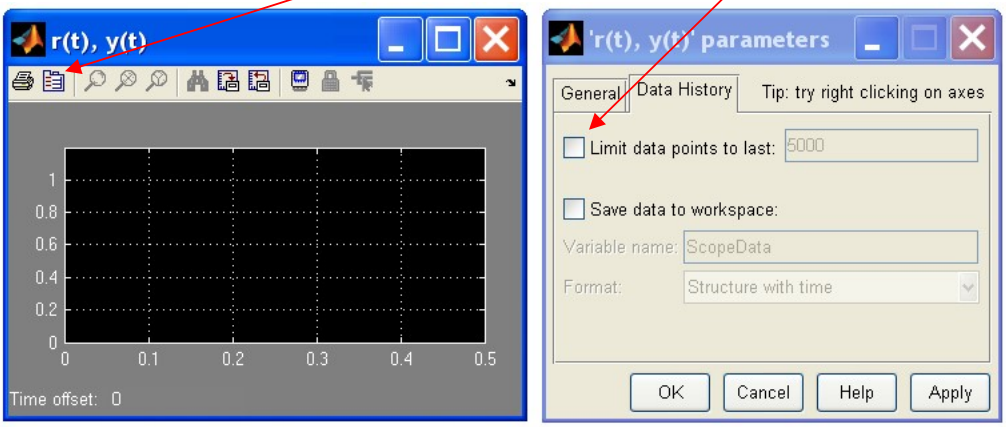
Simulazio denbora *Configuration Parameters* leihotik ere alda daiteke, *Solver* (ezkerrean) sailkatuz, *Simulation* menuko *Configuration Parameters* egikarituz lortzen delarik. *Stop time* parametroak simulazioaren azken unea definitzen du. Leiho honek, *Simulink*-ek berari diagramak adierazten dion ekuazioa ebazteko erabiliko duen zenbakizko/diskretizazio metodoa sailkatzeko ere balio du, non besterik adierazi ezean (berez) *ode45 (Dormand-Prince)* den, beste funtzioen artean.



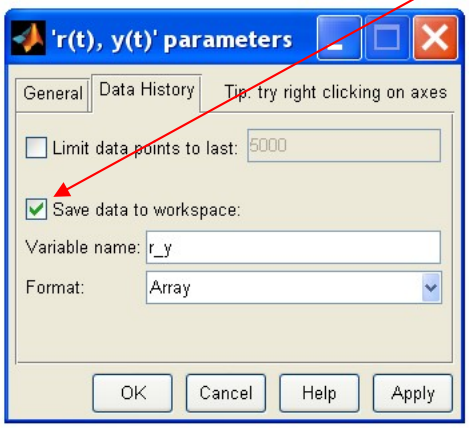


Simulazio denbora handiak erabiltzean eduki dezakegun arazoetariko bat, seinaleak denbora guztian ez direla adierazten da. Honen arrazoia da, *Scope* bistaratzaileak duten bistaratzeko lagin kopurua 5000-ra mugatua dagoela, memoria aurrezteko hartzen den

neurri bezala. Aukera hau ezgaitzeko, bistaratzailearen *Parameters* botoian sakatu behar da, eta agertzen den leihoan, *Data Histoy* erlaitzean (pestaña), *Limit data points to last* laukia ezgaitu.



Simulink-ek, *Scope* bistaratzailearen bidez, simulazio baten ondorioz sorturiko datuak *Workspace*-n gordetzeko aukera ematen du. Horretarako, *Save data to workspace* laukitxoa sailkatzen/gaitzen da, aldagaiari izena eman (kasu honetan bi direnez, *r(t)* eta *y(t)*), kasu honetan *r_y*, eta *Array* en *Format* sailkatzen da. Eragiketa hau *Sinks* liburutegiko *To Workspace* funtzioarekin ere egin ahal da.



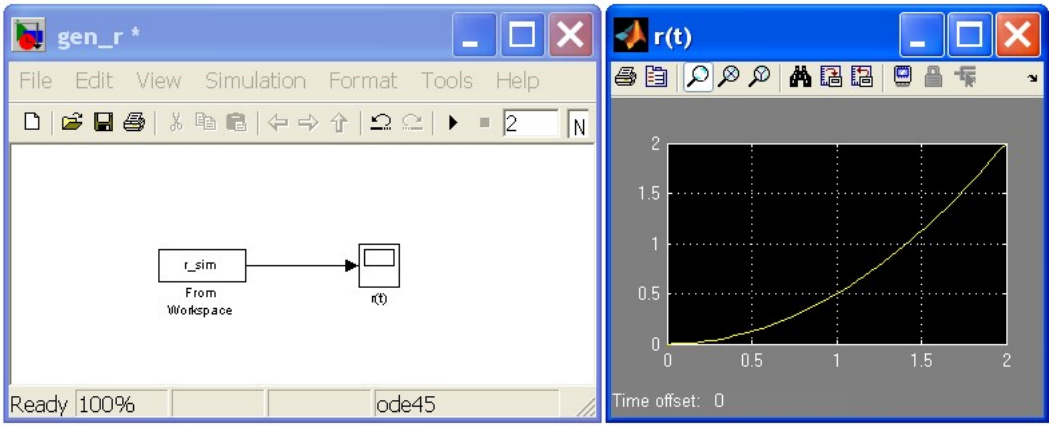
Behin simulazioa eginez gero, array edo bektore hori erabilia izan daiteke, adibidez, *MatLab*-eko *Figure* grafiko batean. Horretarako *plot()* funtzioa erabiltzen da:

```
>> plot(r_y(:,1),r_y(:,2)) % r(t) bistaratzen du Figure-an
>> plot(r_y(:,1),r_y(:,3)) % y(t) bistaratzen du Figure-an
```

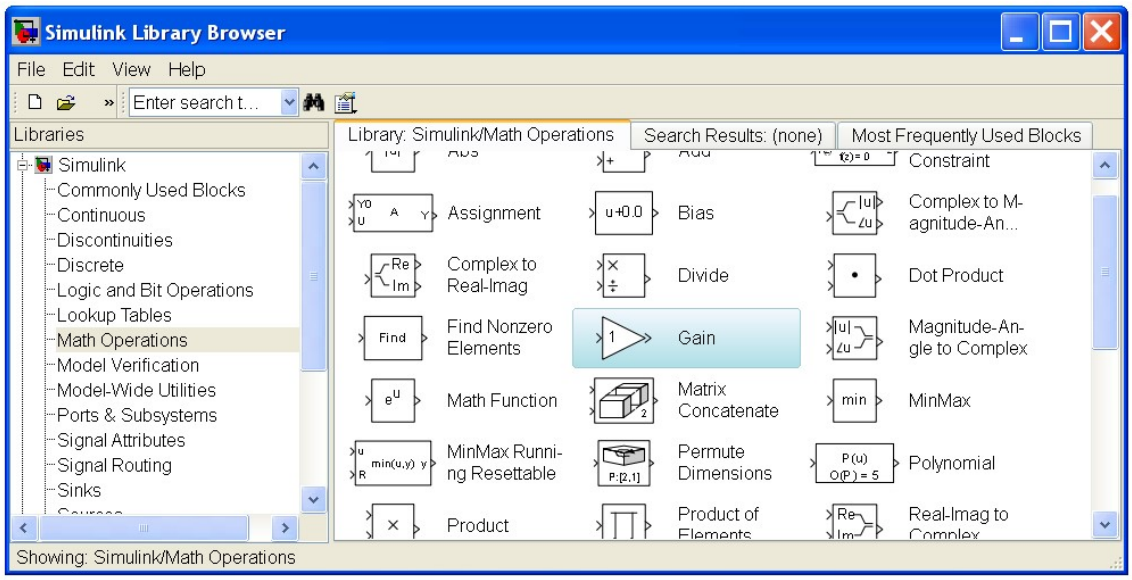
Badago *Simulink*-eko elementu bat non *MatLab*-en sorturiko bektore bat aplikatzen duen, *Sources* liburutegiko *From Workspace* funtzioa da hain zuzen ere. Horretarako, *MatLab*-en sortzen da nahi de bektorea, adibidez parabola unitarioa, $r(t)=1/2 \cdot t^2$. Lehenengo *t* denbora bektorea sortzen da, adib. 2 s 50 laginduna, ondoren *r(t)* seinalea, eta azkenik *r_sim* egituraren eremu desberdinei 3 sententzien bidez esleitzen zaizkie beraiei dagozkien balioak.

```
>> t=linspace(0,2,50);
>> r=0.5*t.*t;
>> r_sim.signals(1).values=r';
>> r_sim.time=t';
>> r_sim.signals(1).dimensions=1;
```

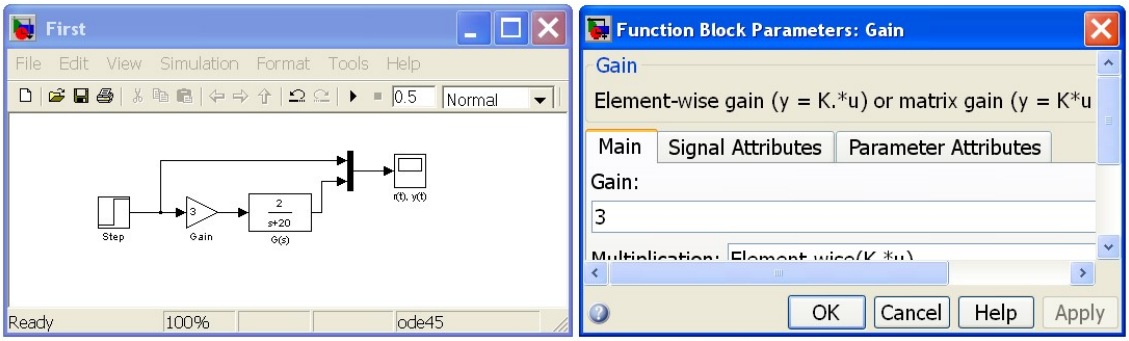
Simulink-eko diagraman, *From Workspace* motako iturriak (sarrera) *r_sim* izena du eta *t*-ri esleituriko denbora berdina simulatzen da.



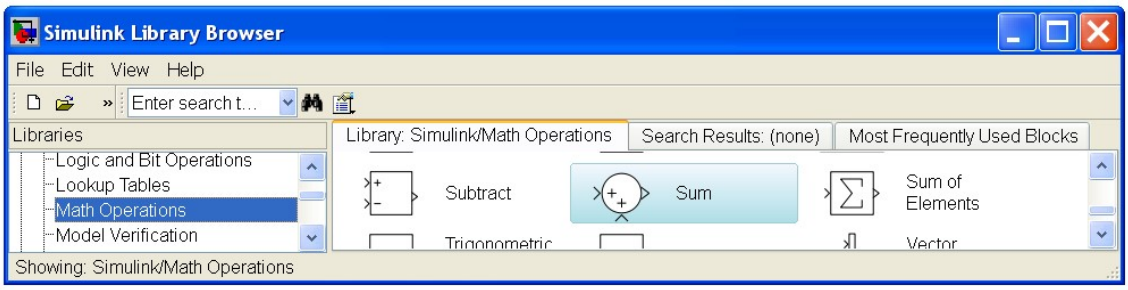
Simulink-en diagrametan oso erabiliak diren elementuak *Math Operations* liburutegikoak dira. Horrela, *Gain* blokeak bere barnean duen konstantearengatik (bere gainean klik bikoitza eginez) bidertzen du bere sarreran aplikatzen den seinalea, eta emaitza bere irteeran ematen du.



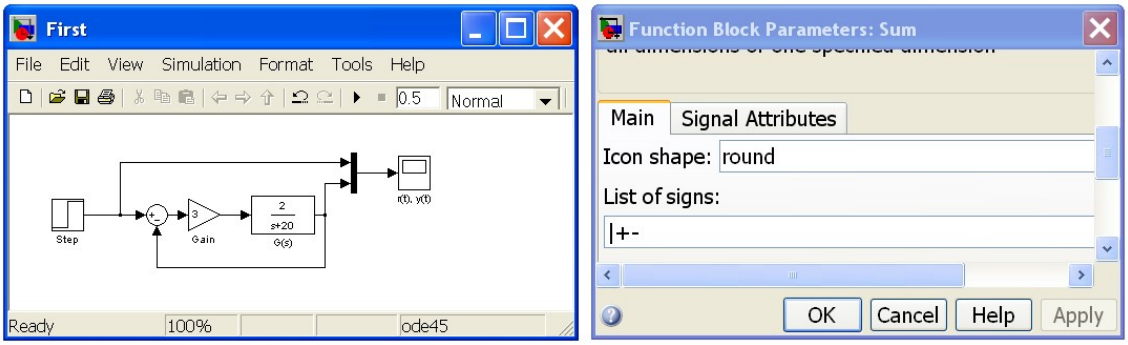
Irudian ikus daiteke nola 3 balioa duen *Gain* edo biderkatzailea sistemaren sarreraren aurretik konektatu dela. Honek *Step* maila unitarioa den sarrera 3-rengatik biderkatuko du, sistemaren sarrerara aplikatu aurretik.



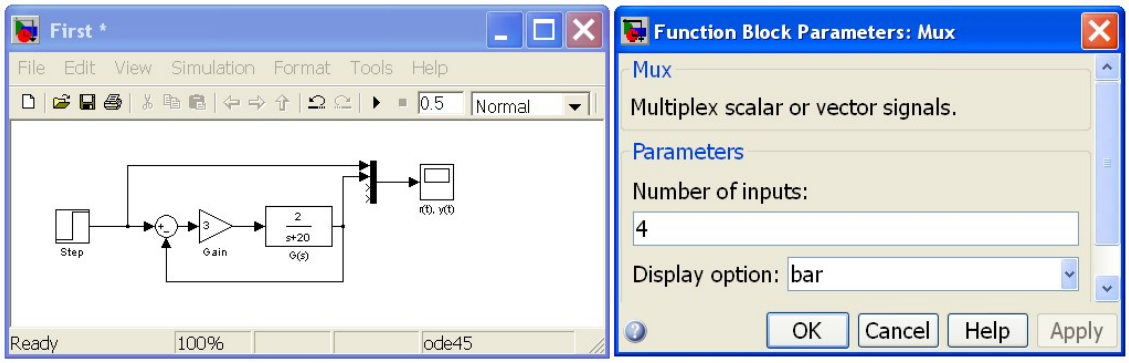
Sum ere sarritan erbiliko dugun beste elemento bat da, batez ere begizta itxia erabiltzen den sistemetan.



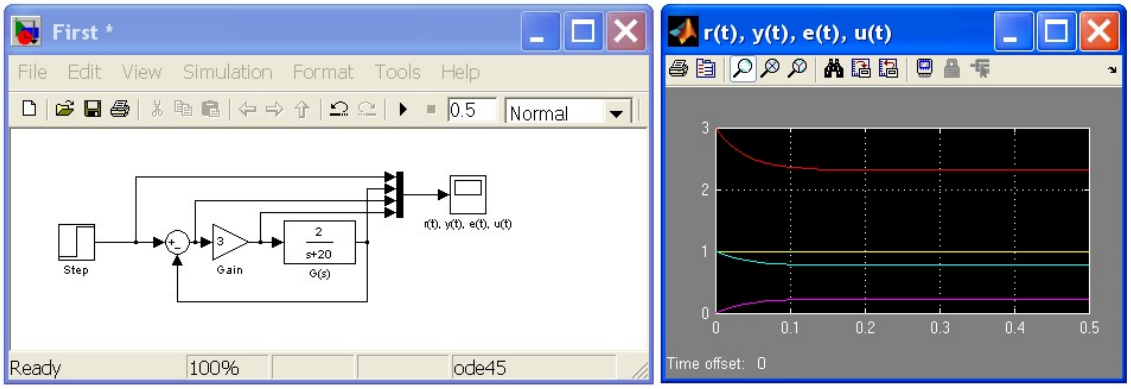
Aurreko diagrama hartuz, berrelikadura negatibo unitarioa gehitu nahi bazaio, *Sum* bloke bat sartu eta dagokion bezala konektatu ondoren, editatu egiten da (klik bikoitza) eta bigarren zeinua – (eskubikoa) jartzen da.



r(t) sarrera eta *y(t)* irteerataz gain, *e(t)* errore seinalea eta *u(t)* kontrol seinalea ere bistaratzeko *Scope* berdinean, multiplexorea editatu eta 4 sarrera behar direla adierzi beharko litzateke.



Simulazioaren emaitza ondorengo grafikoan (eskubia) ikus daiteke, non ikus daitekeen multiplexorearen 3. seinalea (*e(t)*) urdina dela, eta 4.a, gorria (*u(t)*).



Erosotasunagaitik, edo sistema hobeto dokumentatua edukitzeko, edo baita tamaina jakin bateko (ertain edo handi) diagrametan, non parametro batzuen balioak aldatu eta simulazioak askotan errepikatu dire, komeni da sistema osatzen duten bloke desberdinak parametrizatzea. Aurreko kasua hartuz adibide bezala, suposa genezake *G(s)* sistemak jarraiko *RL* zirkuitua adierazten duela, non sarrera zirkuituari aplikatzen zaion $v_i(t)$ tensioa den, eta irteera horren ondorioz bultzatzen de *i(t)* korrontea.

Horrela,

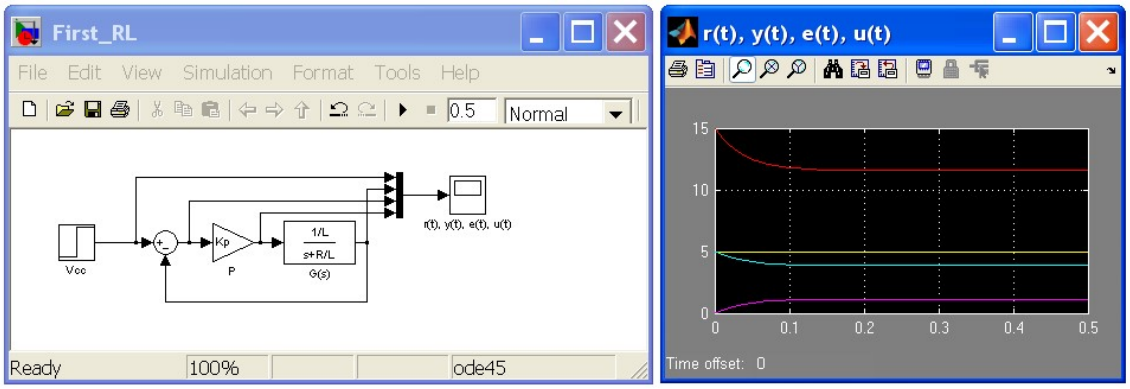
$$G(s) = \frac{I(s)}{Vi(s)} = \frac{1/L}{s + R/L}$$

orduan suposatuz $v_i(t)=Vcc$, $R=10\ \Omega$, $L=0.5\ H$ direla

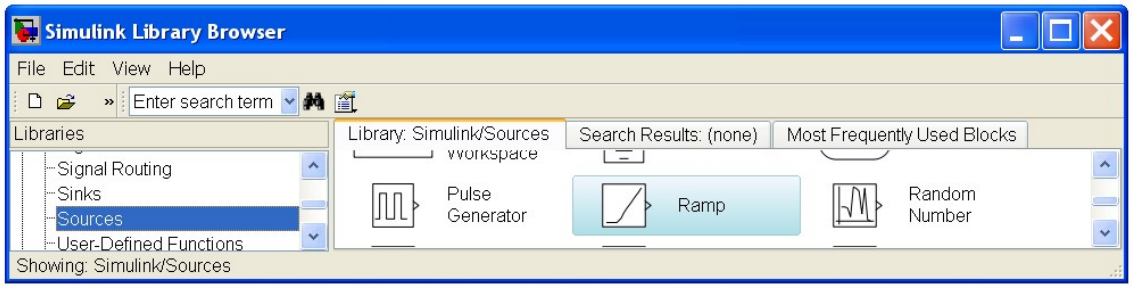
non $Vcc=5\ V$ hartzen den, eta 3 irabazpena kontroladore proportzionalarengatik ordezkatzen du, eta *MatLab*-en komandoen lerroan parametro hauek sortzen dira, beraien balioekin,

```
>> Vcc=5;R=10;L=0.5;Kp=3;
```

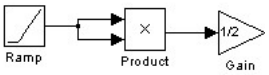
eta jarraian ondorengo blokeen diagrama sortzen da, *First_RL.mdl* (adib. Lehengo *First.mdl* hartuz oinarri gisa, eta dagozkion aldaketak eginez). Simulazioa egin ondoren, ikusi daiteke seinaleen formak aurreko irudikoak bezalakoak direla, beraien balioak izan ezik (oraingo sarrera 5 delako lehengo 1en ordez).



Erregulazio automatikoan asko erabiltzen diren sarreren artean arrapala unitarioa aurkitzen da, $r(t) = t$



eta parabola unitarioa, hau da $r(t) = 1/2 \cdot t^2$. Baina azken hau ez denez agertzen definitua *Sources* liburutegian, sortu egin behar da. Hau egiteko modu bat beheko azpidiagrama eraikiz da,



non *Product* funtzioa *Math Operations* liburutegitik lortua izan den.

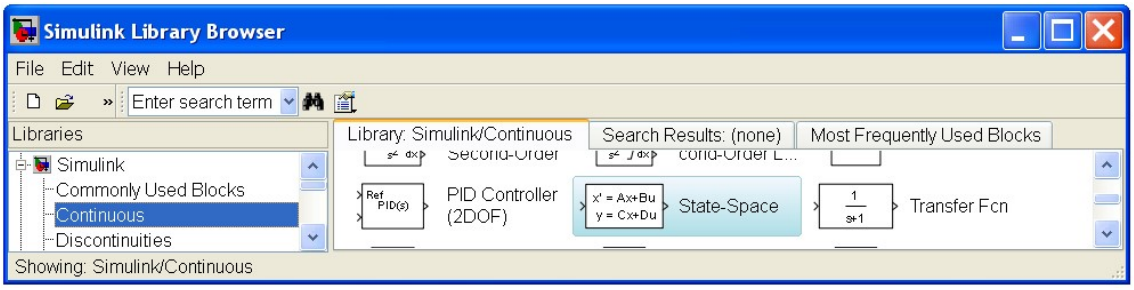
Simulink-ek sistema baten dinamika definitzeko beste modu bat ere eskeintzen du, egoera ekuazioen (*EE*) bidez: egoera ekuazioa (1) eta irteera ekuazioa (2). Transferentzia funtzioarekin konparatuz (*G(s)*), non Laplace-en *s* aldagaian definitzen den, egoera ekuazioak *t* denbora aldagaian definitzen dira.

$$\begin{cases} \dfrac{dx(t)}{dt} = A \cdot x(t) + B \cdot u(t) & (1) \\ y(t) = C \cdot x(t) + D \cdot u(t) & (2) \end{cases}$$

Sistemaren ekuazio diferentzialetan oinarrituz, (1) eta (2) ekuazio matrizialen erara ordenatzea pretenditzen da, nahiz eta lehen ordenako sistemetan ekuazio eskalarrak izaten diren. Adierazteko, aurreko sistemaren ekuazio diferentziala erabiliko da, jada parametrizatua, eta (1) eta (2) egoera ekuazioen arabera ordenatua.

$$\begin{cases} \dfrac{di(t)}{dt} = -R/L \cdot i(t) + 1/L \cdot u(t) & (1) \\ y(t) = 1 \cdot x(t) + 0 \cdot u(t) & (2) \end{cases}$$

Fitxategi berri bat zabaltzen da eta *State-Space* objetua sailkatu, *Continuous* liburutegikoa

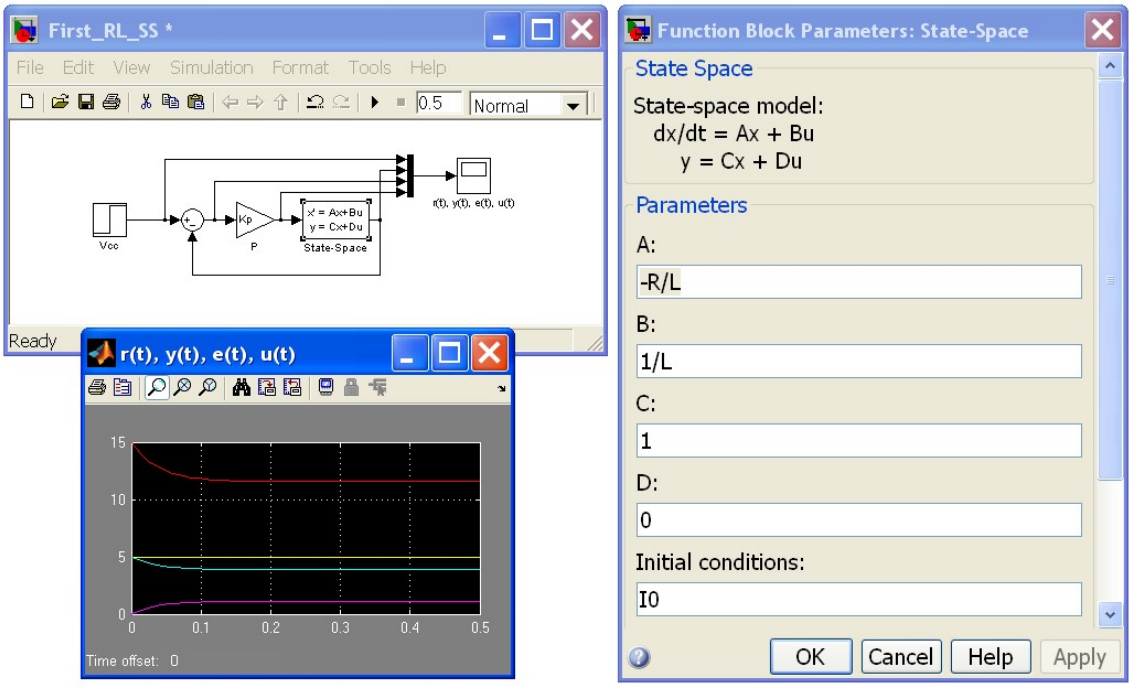


Eta diagramara arrastatu, non blokean klik bikoitza eginez, dagozkion balioak hasieratzen diren. Ondoren, sistema berdina eraikitzen da.

Ikus daitekeen bezala, sistemaren erantzuna *Vcc* sarreraren aurrean, lehengo berdina da, sistema bera delako, nahiz eta beste modu batetara errepresentatua egon. Ohartu beharra dago, funtzio honetan *I0* parametro berria agertzen dela sistemaren hasierako baldintzak definitu ahal izateko, *Initial conditions*, non *Transfer Function*-ek ezin dituen definitu. Kasu honetan eman zaion balioa hutsa da, nulua, hau da

```
>> Vcc=5;R=10;L=0.5;Kp=3;I0=0;
```

Kontrol-ingeniaritzako teoriaren arabera, egoera ekuazioak sistema ez linealen dinamika adierazten uzten dute, baina *Continuous* liburutegiko *State-Space* funtzio honek sistema linealak adierazteko balio du soilik. Sistema ez linealen egoera ekuazioen notazioa erabili ahal izateko *Simulink*-en, *User-Defined Functions* liburutegiko *S-Function* edo bestela *S-Function Builder* funtzioa erabili beharko litzateke.

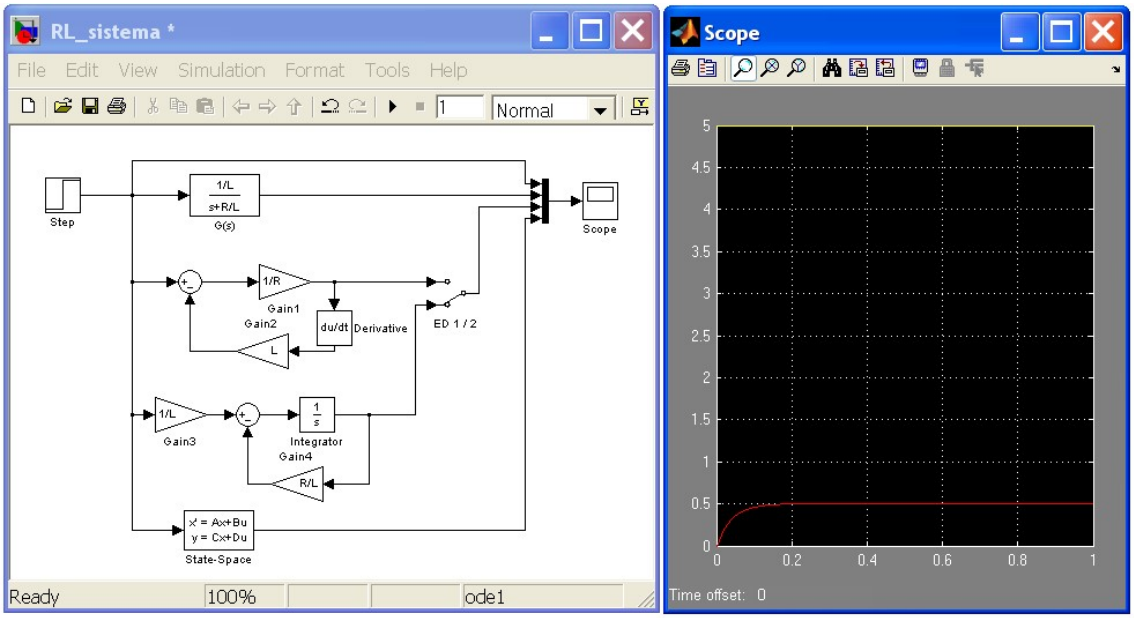


Sistema baten dinamika *Simulink*-en definitzeko hirugarren eta azken modua, ekuazio diferentzialak (*ED*) erabiltzea da, zuzenean. Sistemaren dinamika osatzen duten ekuazio guztiak eraiki eta elkarrekin konektatzea, hainbat badira, eginez lortzen da. Posible den guztietan, termino deribatuen ordez termino integralak erabili behar dira, bestela ereduak huts egin bait dezake (ED 1). Adibidez, ondorengo ekuazio diferentziala eraikitzen da,

$$\frac{di(t)}{dt} = -R/L \cdot i(t) + 1/L \cdot u(t)$$

eta irteera integratzen da (ED 2),

$$\int \frac{di(t)}{dt} dt = i(t)$$



Aurreko irudian ikus daitekeen bezala, ED 2 ekuzioaren irteerak transferentzia funtzioarekin nahiz egoera ekuazioarekin bat egiten du, non hasierako baldintzak baliogabeak ziren (I0=0). Halere, ekuazio diferentziala hasierako baldintzak barneratzeko gai da: ED 2-ren integrala (*Integrator*) editatuz eta balioa (parametroa) *I0* jarriz *Initial condition*-en. Simulazioa errepikatuz, baina I0 = -0.5 jarriz, ikusi daiteke egoera ekuazioaren erantzuna eta ED 2 ekuazio diferentzialarena berdinak direla, non transferentzia funtzioarena desberdina den, ezina baitu hasierako baldintzak kontuan hartu (morea).