ETF Sarajevo
Postdiplomski studij na A i E

Metode modeliranja procesa

Sarajevo ,April 2004

Općenito o SIMULINK-u

Simulink je jedan od paketa u sastavu popularnog softwareskog alata MATLAB. i trenutno je uključen u posljednju verziju MATLAB-a 6.5 (Release 13) , kao Verzija 5.
Sa Simulink-om mi možemo da krenemo od modeliranja i simulacije idealizovanih linearnih modela ka realističnijim nelineranim modelima, koji uključuju trenje, otpor zraka, proklizavanje u reduktoru, ograničivači pomaka, i drugi tipovi nelinearnosti koji opisuje fenomene iz realnog svijeta.

Simulink je softwareski paket za modeliranje, simulaciju i analizu dinamičkih sistema. Podržava linearne i nelinearne sisteme, modele u kontinualnom vremenu, diskretnom vremenu uzorkovanja, ili bilo koji hibrid od ova dva. Sistemi mogu takodjer biti višebrzinski, tj. imati djelove koji se sampluju ili ažuriraju sa različitim brzinama.

Za modeliranje, Simulink obezbjedjuje grafički korisnički interfejs (GUI), za gradnju modela kao blok dijagrama, koristeći operacije miša tipa klik i vuci (click and drag). Sa ovim interfejsom mi možemo crtati modele kao što bi to radili sa olovkom i papirom. Simulink uključuje bogatu biblioteku potrošača (sink) , izvora (source) , linearnih i nelinearnih komponenti, i konektora. Korisnik može takodjer kastomizirati i kreirati njegove vlastite blokove. (vidjeti kasnije u opisu o S-funkcijama).

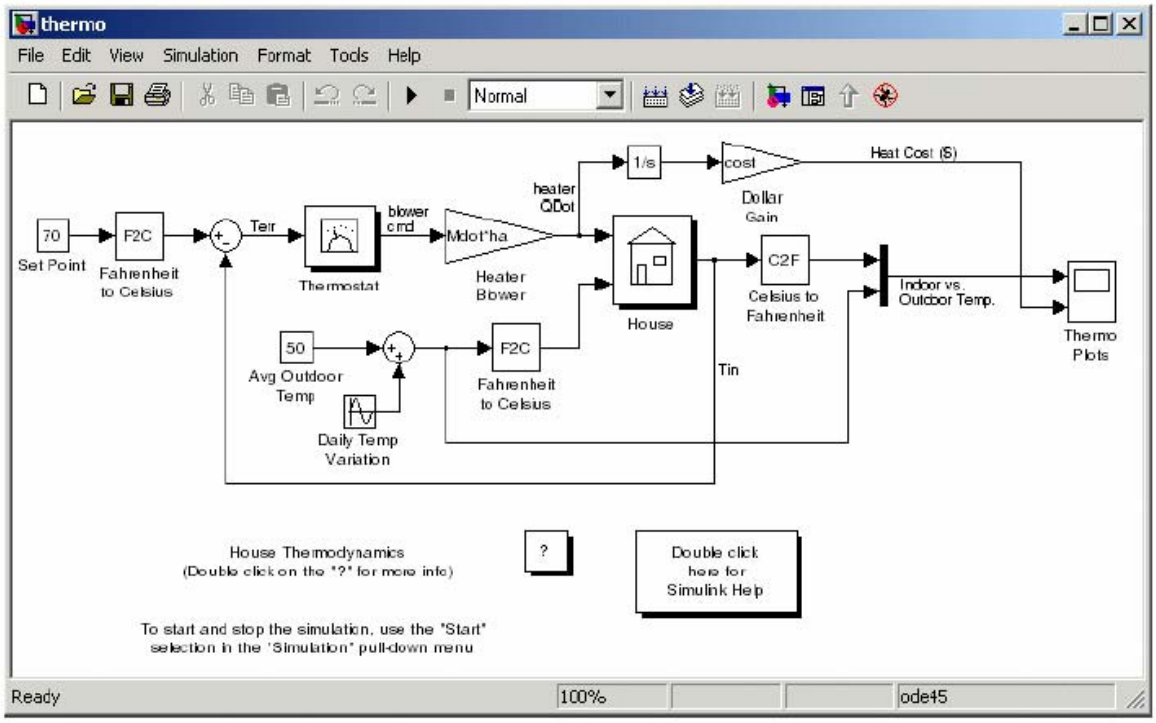
Modeli su hijerarhijski , tako da korisnik može graditi modele koristeći top-down ili bottom-up (odozdo na gore) pristup. Korisnik može da posmatra sistem na visokom nivou, zatim da dvaput klikne na blok da bi silazio unutra kroz nivoe i vidio rastući nivo detalja modela. Ovaj pristup omogućuje takodjer da korisnik može da sagleda strukturu modela i vidi kako je organiziran, i kako njegovi pojedini dijelovi interaktiraju.

Nakon što je model definiran, on se može simulirati, koristeći niz integracionih metoda , bilo sa menija Simulinka, ili unošenjem komandi u komandni prozor MATLAB-a. Meniji su naročito pogodni za interaktivni rad , dok pristup preko komandne linije je vrlo pogodan za slučajeve kada se izvršava više simulacija, (napr. ako radimo Monte Carlo simulacije , ili želimo da predjemo kroz ukupan opseg vrijednosti parametara). Koristeći osciloskope (scopes) i druge blokove za prikazivanje podataka, korisnik može vidjeti rezultate simulacije dok se simulacija još izvršava. Nadalje, korisnik može promjeniti parameter i trenutačno vidjeti šta se dešava, za istraživanja tipa "šta ako" (what if). Rezultati simulacije se mogu staviti u radni prostor MATLAB-a za daljnji postprocesiranje i vizualizaciju.

Alati za analizu modela uključuju linearizaciju i alate za pojednostavljenje modela, kojima se može pristupiti sa komandne linije MATLAB-a.

Opis jednostavnog Simulink modela.

Interesantan demo program u Simulinku je termodinamski model kuće. Program se starta ukucavanjem **thermo** u komandni prozor MATLABA-a . Ova komanda će startati Simulink i otvoriti prozor kao na slici:



Da bi se startala simulacija, treba otvoriti meni Simulation i kliknuti na **Start** ili kliknuti na mali trokut usmjeren udesno u toolbaru (traci alata).

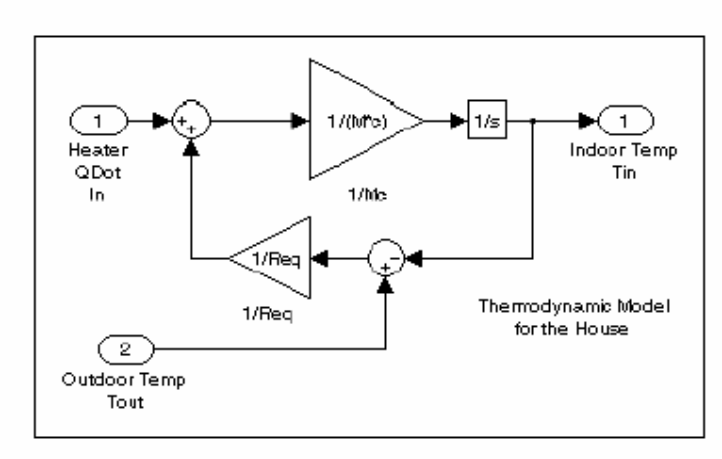
Za zaustavljanje simulacije dovoljno je klinuti na **Simulation>>Stop** ako se već nije ranije završila sama prema vremenu trajanja simulacije konfiguriranom u polju **Simulation>>Simulation parameters>>Solver>>Stop time.**

Opis modela

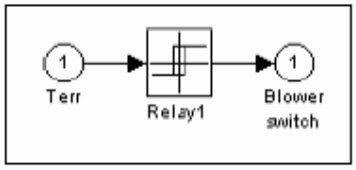
Model koristi podsisteme (subsystems) da pojednostavi dijagram modela i kreira module koji se mogu više puta koristiti. Podsistem je grupa blokova koji su predstavljeni blokom podsistema. Ovaj model sadrži pet podsistema :

- thermostat
- house
- temp convert F/C (2)
- temp convert C/F (1)

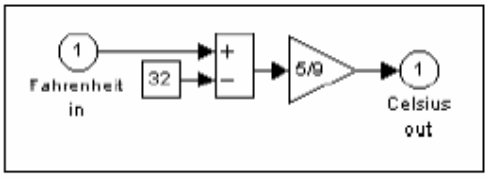
Da bi vidjeli unutarnju strukturu podsistema , treba kliknuti dva puta na njegov blok. Tako naprimjer ako kliknemo dvaput na blok "House", otvoriće se blok struktura tog subsistema kao na slici:



Podsistem termostata modelira rad termostata , odredjujući kada će se grijanje uključiti ili isključiti. Ako dvaput kliknemo na taj blok , otvoriće sa njegov subsistem kao na slici:



Vanjska i unutarnja temperatura su pretvarane iz Farenhajtovih stepeni u Celzijusove i obratno od strane identičnih podsistema kao na slici:



F2C konvertor

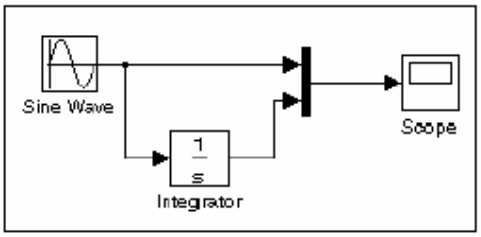
Kada je grijanje uključeno, troškovi grijanja se računaju i iscrtavaju u plotu "Heat cost ($)" . Unutarnja temperatura se prikazuje na osciloskopu Indor Temp.

Ovaj demo ilustrira nekoliko taskova koji su zajednički svim modelima:

- Izvršavanje simulacije se starta sa komandom Start , i zaustavlja sa komandom Stop, koje su dostupne iz menija Simulation ili direktno na toolbaru.
- Pri gradnji modela možemo enkapsulirati grupu povezanih blokova u jedan blok, koji se naziva podsistem.
- Možemo kreirati kastomiziranu ikonu i dizajnirati dijalog boks za svaki blok , koristeći osobinu maske (masking feature),
- Scope blokovi prikazuju grafičke izlaze vrlo slićno kako to čine i osciloskopi.

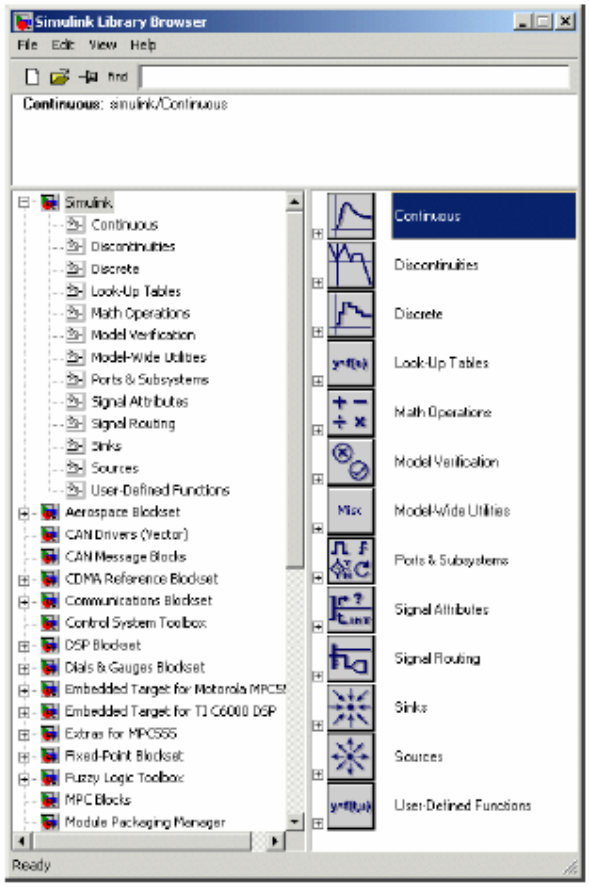
Gradnja jednostavnog modela

Na slijedećem primjeru ćemo pokazati kako se gradi jednostavni model koristeći mnoge komande i akcije koje se koriste kod gradnje modela.
Model uključuje blok generisanja sinusnog valnog oblika i prikazuje taj oblik. Blok dijagram modela izgleda kao na slijedećoj slici:



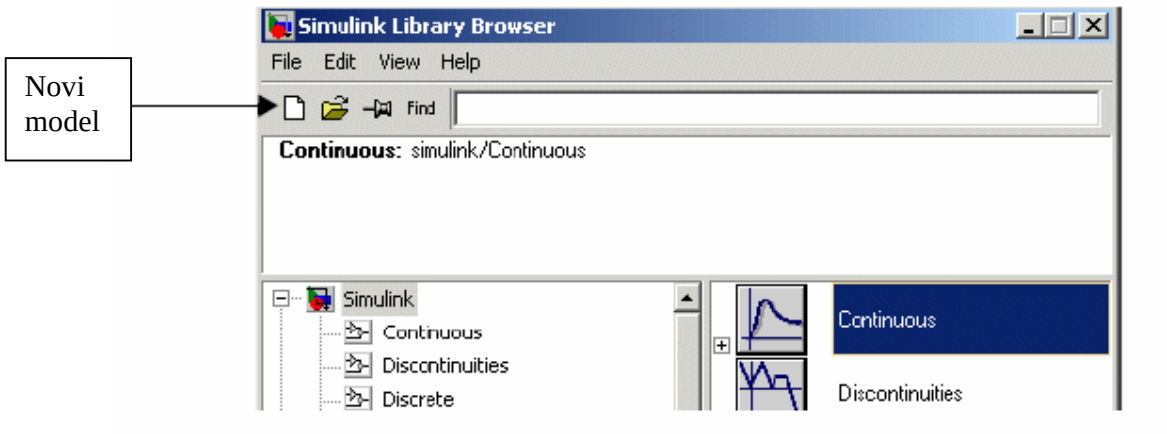
Da bi se kreirao model, moramo prvo ući u Simulink GUI , bilo kucajući na komandnoj liniji **simulink**, ili klikajući direktno na ikonu na toolbaru.

Kad se otvori prozor, on će pokazati browser Simulink biblioteke kao na slici:

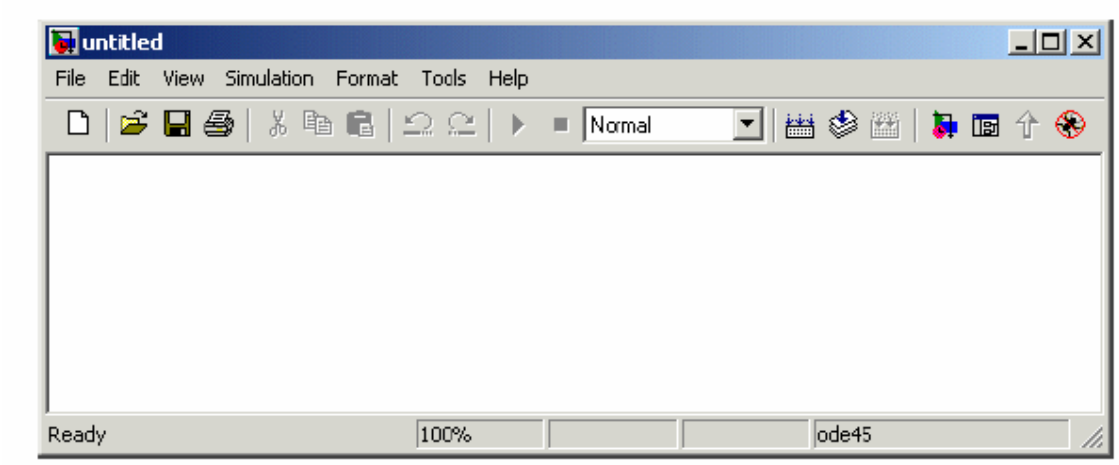


Browser Simulink biblioteke

Da se kreira novi model treba izabrati ili kliknuti na **New Model** na tollbaru , kao na slici :



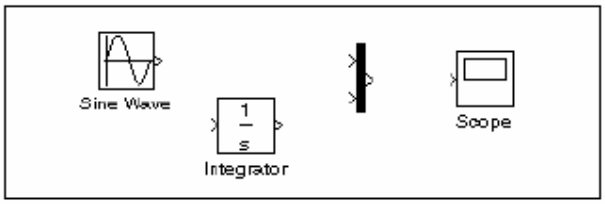
Simulink će otvoriti novi prozor:



Da bi kreirali zadati model, treba da kopiramo blokove iz biblioteka Simulinka:

- iz source biblioteke (izvora) – blok Sine wave
- iz sink biblioteke (potrošača signala) – Scope blok
- iz bibliteke Continious – blok Integratora
- iz biblioteke Signal Routing – mux blok (multipleksera)

Kada su svi blokovi kopirani u prozor modela, izgledaće otprilike kao :



Ako pogledamo pažljivije ikone blokova, vidjećemo male strelice na izlazu iz sinusnog bloka i dvije na lijevoj strani mux bloka. Simbol **>** se naziva izlaznim portom (**output port**), ako simbol je usmjeren prema bloku , onda je to ulazni port (**input port**). Signal putuje iz izlaznog porta u ulazni port slijedećeg bloka kroz liniju koja ih spaja.

Primjetimo da kod povezivanja blokova ima jedna linija koja spaja koja se naziva linija ogranak (**branch line**), koja spaja Sinusni blok sa integratorskim blokom i nosi isti signal i ka Mux bloku.

Izvršićemo simulaciju modela za interval vremena od 10 sec. Da bi se postavilo to vrijeme treba otvoriti **Simulation parameters** iz **Simulation** menija. Vidimo da je to i default vrijeme u polju **Stop Time**.

Unošenje Simulink komandi

U Simulinku se komande unose kroz:

- Selektiranje komandi iz menija Simulinka
- Selektiranje komandi kroz kontekst senzitivni Simulink meni
- Klikanjem ikona na Simulinkovom toolbar-u
- Unošenjem komandi u MATLAB komandni prozor

Prozori Simulinka

Simulink koristi posebne prozore da prikaže browser blokova biblioteke, biblioteku blokova i grafički simulacioni izlaz (scope).

Kreiranje podsistema

Kako model raste u veličini i kompleksnosti, korisnik ga može pojednostaviti grupišući blokove u podsisteme. Korištenje podsistema ima slijedeće prednosti:

- Pomaže reducirati broj blokova koji su prikazani u jednom prozoru modela.
- Dozvoljava da zajednički držimo funkcionalno povezane blokove
- Omogućava nam da uspostavimo hijerarhijski blok dijagram , gdje podsistemski blok je jedan sloj a blokovi koji čine podsistem su drugi blok.

Možemo kreirati podsistem na dva načina:

- Dodati podsistemski blok u model, a onda otvoriti taj blok i dodati blokove koje sadrži u prozor podsistema
- Dodati blokove koji čine podsistem , a onda grupisati ove blokove u podsistem.

Kreiranje podsistema dodajući podsistemski blok

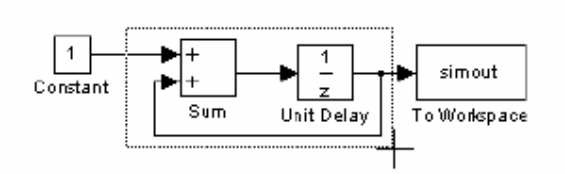
Da se kreira podsistem prije dodavanja blokova koje će sadržavati, treba dodati podsistemski blok u model, a onda dodati blokove koji ga čine:

1. Kopirati podsistemski blok iz **Ports&Subsystems** biblioteke u model
2. Otvoriti podsistem kliknuvši dvaput na njega.

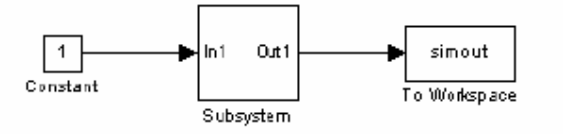
Kreiranje podsistema grupiranjem postojećih blokova

Ako model već sadrži blokove koje želimo konvertovati u podsistem, možemo kreirati podsistem grupiranjem ovih blokova:

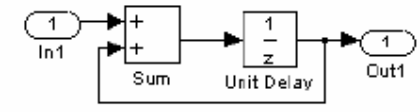
1. Okružiti blokove koje želimo pretvoriti u podsistem sa blokom vezanja kao na slici



2. Izabrati **Create Subsystem** iz **Edit** menija ili iz pop-up menija kada se klikne desnim tasterom na selekciju. Simulink će zamjeniti selektirane blokove sa blokom podsistema.



Ako otvorimo podsistemski blok , Simulink će ga pokazati u novom prozoru (ako smo izabrali u **Preferences windows reuse**), kao na narednoj slici:



Promjetimo da Simulink dodaje Inport i Outport blokove da bi prikazao ulaze i izlaze u podsistem.

Ponovno korištenje prozora (window reuse)

Korisnik može specificirati da li Simulink navigacione komande koriste tekući prozor ili novi prozor da prikažu podsistem i njegovog roditelja (gornji sistem). Meni **Preferences** u **File** meniju , omogućavaju konfigurisanje slijedećih kombinacija:

Tip reusa	Akcija kod otvaranja prozora	Go to Parent (Esc) akcija
none	Podsistem se pojavljuje u novom prozoru	Prozor roditelj se prebacuje ispred
reuse	Podsistem zamjenjuje roditelja u tekućem prozoru	Prozor roditelja zamjenjuje podsistem u tekućem prozoru
replace	Podsistem se pojavljuje u novom prozoru. Prozor roditelja iščezava	Ponovo se pojavljuje prozor roditelja a iščezava prozor podsistema
mixed	Podsistem se pojavljuje u svom vlastitom prozoru	Prozor roditelja iskače naprijed. Prozor podsistema iščezava

Kreiranje uslovno izvršivih podsistema

Uslovno izvršivi podsistem je onaj čije izvršenje zavisi od vrijednosti ulaznog signala. Signal koji kontroliše da li se podsistem izvršava se zove kontrolni signal (**control signal**). Signal ulazi u blok podsistema na kontrolnom ulazu (**control input**).
Uslovno izvršivi podsistemi mogu biti vrlo korisni kada gradimo kompleksne modele koji sadrže komponente čije izvršenje zavisi od drugih komponeneta.

Simulink podržava tri tipa uslovno izvršavanih podsistema:

- **Omogućeni podsistem** (enabled subsystem) se izvršava dok je kontrolni signal pozitivan. On počinje izvršenje u trenutku kada vremenski step kontrolnog signala prolazi kroz nulu (iz negativnog u pozitivni pravac) i nastavlja izvršenje sve dok kontrolni signal ostaje pozitivan.

- **Trigerovani podsistem** (triggered subsystem) se izvršava svaki put kada se pojavi trigerski dogadjaj. Trigerski dogadjaj se može pojaviti na rastučoj ili opadajučoj ivici trigerskog signala, koji može biti kontinualni ili diskretni.

- **Trigerovani i omogućeni podsistem (** triggered and enabled subsystem) , se izvršava jedanput na vremenski step kada se pojavi trigerski dogadjaj , ako kontrolnii signal omogućenja (enable control signal) ima pozitivnu vrijednost za taj step.

- **Iskaz u kontroli toka programa (** control flow sattement) , izvršava C – logiku toka programa pod nadzorom bloka za kontrolu toka programa.
  U svim slučajevima , blokovi koji se izvršavaju su locirani unutar kontrolisanog podsistema. U slučaju **if-then-else** i **switch** kontrole izvršenja programa, kontrolni blok je lociran izvan kontrolisanog podsistema i izdaje

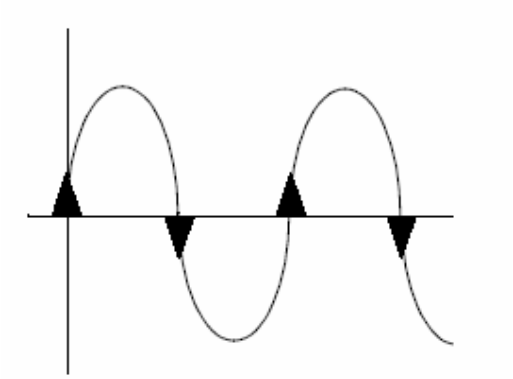
kontrolni signal ka bloku **Action Port**, koji se nalazi unutar kontrolisanog podsistema. U slučaju **while** , **do-while** i **for** komandi za kontrolu toka programa , blok sa iterativnom kontrolom je lociran unutar podsistema, kojeg kontroliše bez vidljivog kontrolnog signala.

Omogućeni podsistemi

Omogućeni podsistemi su podsistemi koji se izvršavaju u svakom koraku simulacije kada kontrolni signal ima pozitivnu vrijednost.
Omogućeni podsistem ima jednostruki kontrolni ulaz, koji može biti skalar ili vektor:

- ako je ulaz skalar, podsistem se izvršava ako je ulazna vrijednost veća od nule
- ako je ulaz vektor, podsistem se izvršava ako bilo koji od vektorskih elemenata je veći od nule

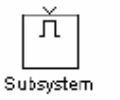
Naprimjer, ako je kontrolni ulazni signal sinusoida, podsistem je alternativno omogućen i onemogućen, kako je prikazano na slijedećoj slici. Strelica na gore označava omogućen a strelica na dole onemogućen:



Simulink koristi metod nagiba prolaska kroz nulu (zero crossing slope), da odredi da li će se omogućenje pojaviti. Ako signal prolazi kroz nulu i nagib je pozitivan , podsistem je omogućen. Ako je nagib negativan kod prolaska kroz nulu, podsitem je onemogućen.

Kreiranje omogućenog sistema

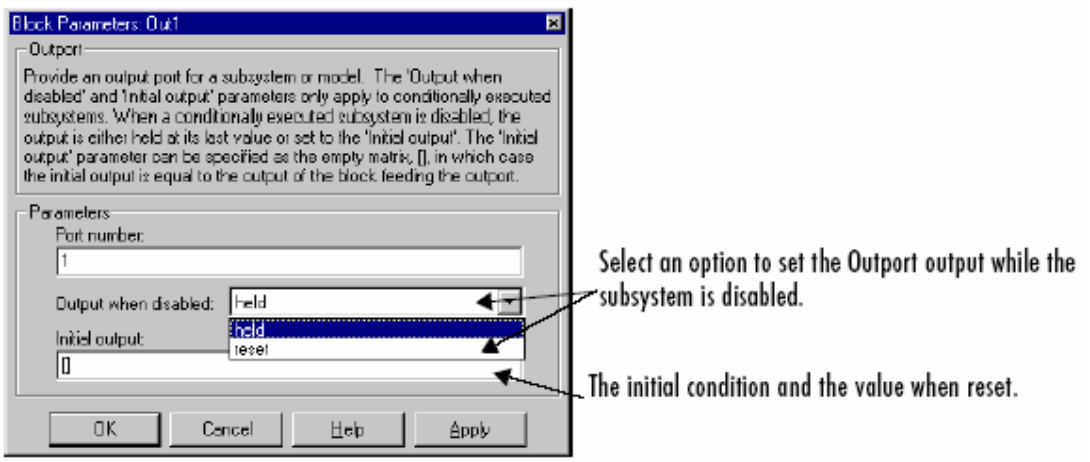
Korisnik kreira omogućeni podsistem kopiranjem Enable bloka iz biblioteke **Ports&Subsystems** u podsistem. Simulink dodaje enable simbol kao i enable port za kontrolni signal na ikoni bloka podsistema.



Setovanje izlaznih vrijednosti dok je podsistem onemogućen

Mada se omogućeni podsistem ne izvršava dok je onemogućen, izlazni signal je još uvjek raspoloživ za druge blokove. Dok je podsistem onemogućen, korisnik može izabrati da drži izlaze podsistema na njihovim prethodnim vrijednostima ili da ih resetuje na njihove početne vrijednosti.
Otvoriti za svaki Outport blok dijalog box i izabrati parametar u polju **Output when disabled**, kao što je pokazano na narednoj slici:

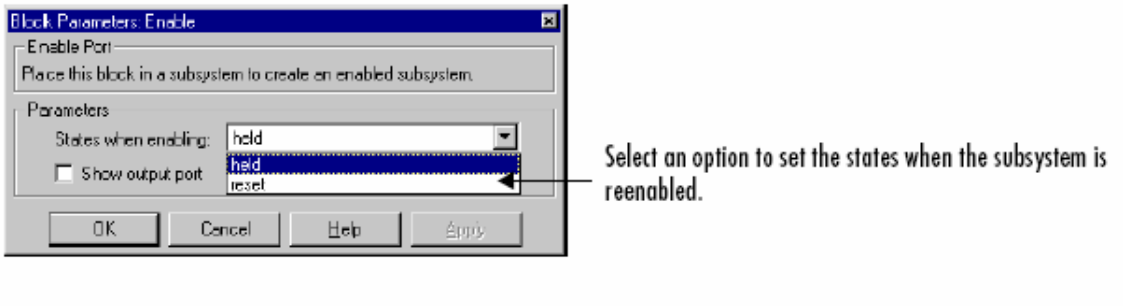
- izabirući **held** , prouzrokuje izlaz da održava svoju najsvježiju vrijednost
- izabirući **reset,** prouzrokuje da se izlaz vraća na svoju početnu vrijednost. Treba postaviti i polje **Initial output** na željenu početnu vrijednost.



Postavljanje stanja kada je podsistem ponovno omogućen

Kada se omogućeni podsistem izvršava, korisnik može izabrati da li da drži stanja podsistema na njihovim prethodnim vrijednostima ili resetuje ih na početne uslove.
Da bi ovo uradili , treba otvoriti Enable blok diajlog boks i izabrati jedan od izbora za polje **States when enabling ,** kao što je pokazano na narednoj slici:

- Izabrati held da se stanja zadrže na njihovim posljednjim vrijednostima
- Izabrati reset da se stanja vrate na njihove početne vrijednosti



Blokovi koji mogu biti u omogućenom podsistemu

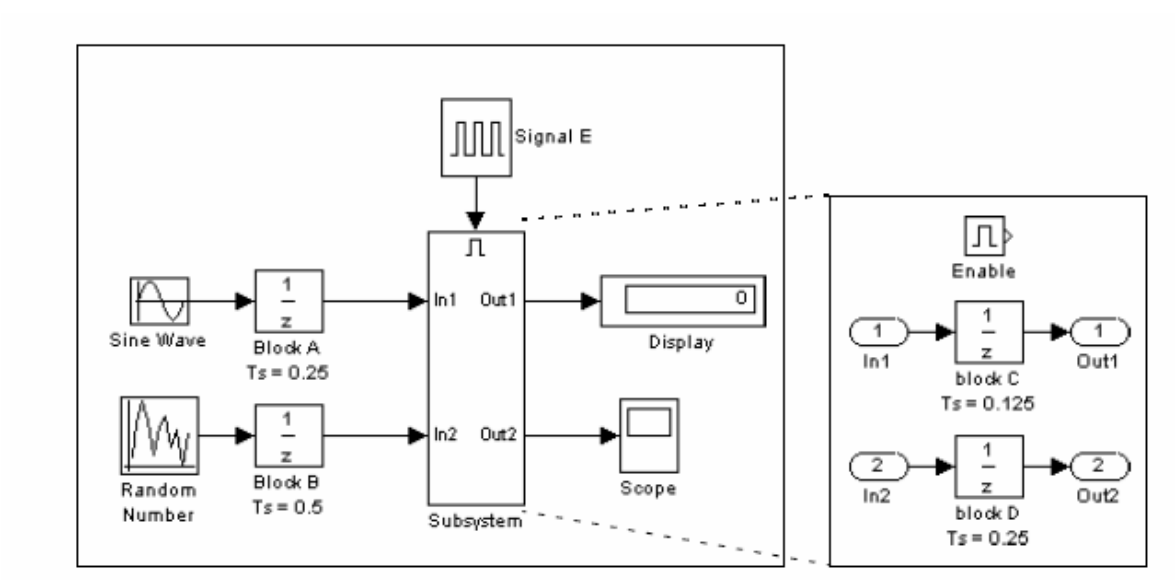
Omogućeni podsistem može sadržavati bilo koje blokove, bilo kontinualne ili diskretne. Diskretni blokovi u omogućenom podsistemu se izvršavaju samo kada se podsistem izvršava , i samo onda kada su njihova vremena sampliranja sinhronizovana sa vremenima sampliranja simulacije. Omogućeni podsistemi i osnovni model koriste zajednički sat.

Opaska
Omogućeni podsistemi mogu sadržavati **Goto** blokove. Medjutim, samo portovi stanja se mogu spojiti na Goto blokove u omogućenom sistemu. Pogledati Simulink demo model **clutch** , kao primjer kako koristiti Goto blokove u omogućenom podsistemu.

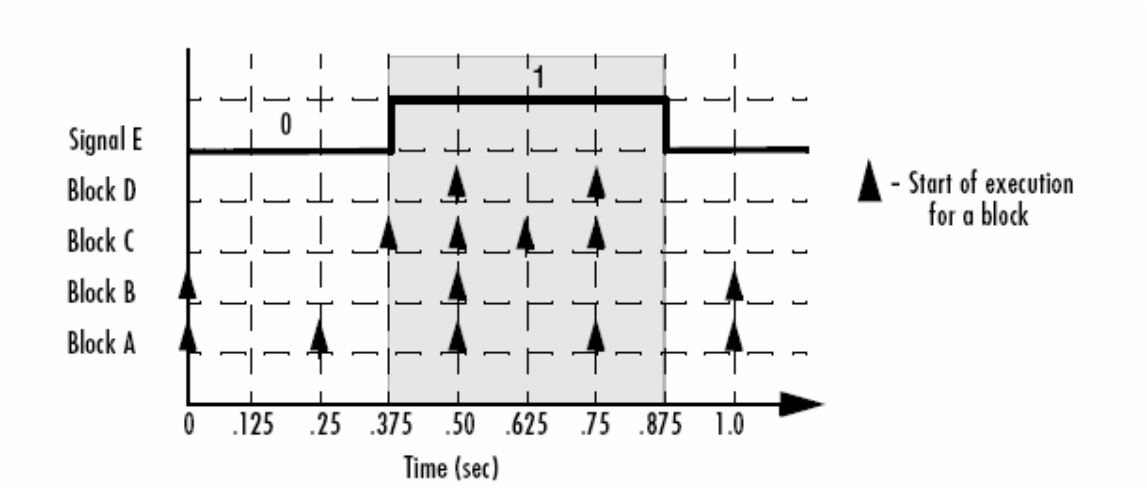
Kao drugi primjer uzmimo sistem koji sadrži četiri diskretna bloka i kontrolni signal. Diskretni blokovi su:

- Blok A, koji ima vrijeme sampliranja od 0.25 sec
- Blok B, koji ima vrijeme sampliranja od 0.5 sec
- Blok C , unutar omogućenog podsistema, koji ima vrijeme sampliranja od 0.125 sec
- Blok D, takodjer unutar omogućenog podsistema, koji ima vrijeme sampliranja od 0.25 sec

Enable kontrolni signal se generiše od strane bloka generatora impulsa (pulse generator), koji je labeliran Signal E, koji se mjenja od 0 do 1 kod 0.375 sec i vraća na 0 kod 0.875 sec.



Na chartu koji slijedi vidi se kada se diskretni blokovi izvršavaju:



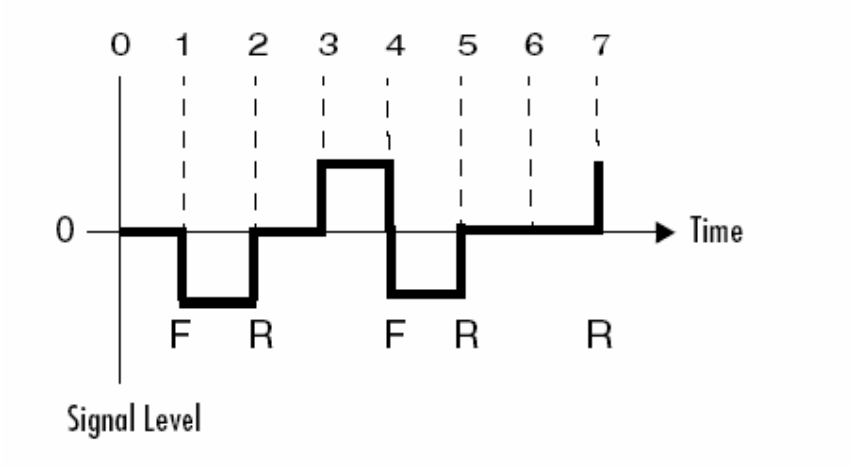
Blokovi A i B se izvršavaju nezavisno od enable kontrolnog signala pošto oni nisu dio omogućenog podsistema. Kada enable kontrolni signal postaje pozitivan, blokovi C i D se izvršavaju kod njihovih doznačenih brzina sampliranja, sve dok enable kontrolni signal ne postane ponovno nula. Primjetimo da blok C se ne izvršava kod 0.875 sec , kada enable kontrolni signal se mjenja na nulu.

Trigerovani podsistemi

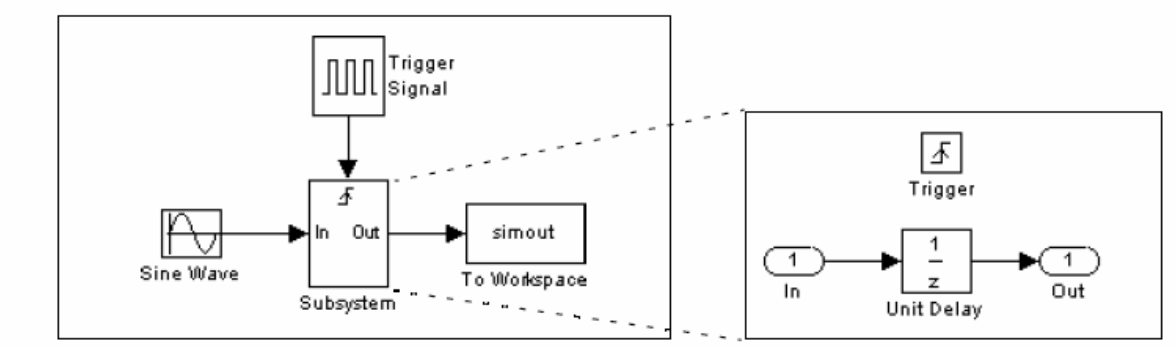
Trigerovani podsistemi su podsistemi koji se izvršavaju svaki put kada se pojavi trigerski dogadjaj.
Trigerski podsistem ima jedan kontrolni ulaz, koj se zove **trigger input** (trigerski ulaz), koji odredjuje da li će se podsistem izvršiti. Možemo izabrati izmedju tri tipa trigerskih dogadjaja da prisilimo trigerski podsistem da počne izvršenje:

- **rastući** (rising) triger – izvršenje podsistema kada kontrolni signal poraste sa negativne ili nulte vrijednosti na pozitivnu vtijednost (ili nulu ako je početna vrijednost negativna)
- **opadajući** (falling) triger – izvršenje podsistema kada kontrolni signal padne sa pozitivne ili nulte vrijednosti na negativnu vrijednost (ili nulu ako početna vrijednost je pozitivna).
- **bilo koji (either)-** trigeruje izvršenje podsistema kada signal je bilo rastući ili opadajući.

Opaska :
U slučaju diskretnih sistema, signal koji raste ili pada od nule se smatra trigerskim dogadjajem samo ako se signal zadržao na nuli više od jednog vremenskog stepa nakon pada ili rasta. Ovo eleminira lažne trigere prouzrokovane sa smapliranjem kontrolnog signala.
Naprimjer , u vremenskom dijagramu na slijedećoj slici, za diskretni sistem, rastući triger (R) se neće pojaviti u vremenskom koraku 3, pošto je signal ostao na nuli samo jedan vremenski step, kada se pojavio njegov porast:



Jednostavan primjer trigerskog podsistema je dat na narednoj slici:

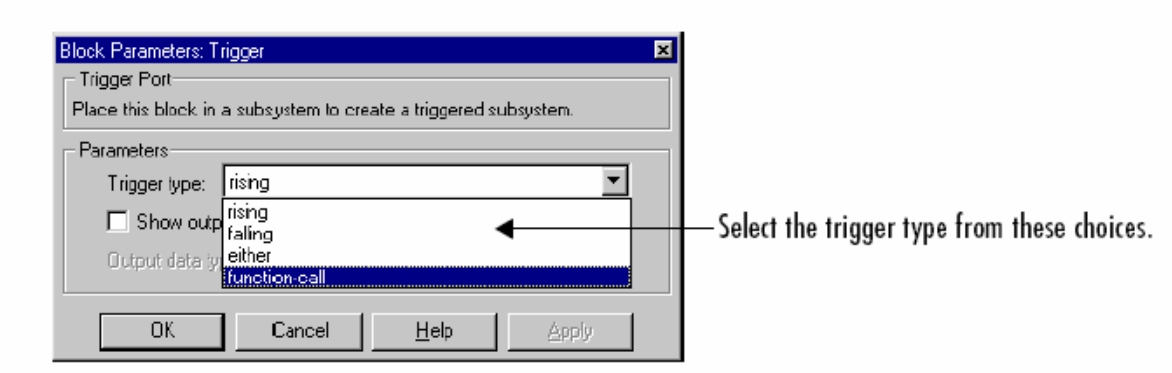


U ovom primjeru, podsistem se trigeruje na rastuću ivicu kvadratičnog signala koji je trigerski kontrolni signal.

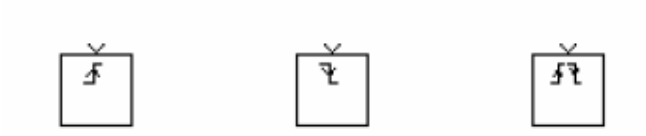
Kreiranje trigerskog podsistema

Korisnik kreira trigerski podsistem kopiranjem trigerskog bloka iz **Ports&Subsystem** biblioteke u podsistem. Simulink dodaje trigerski simbol i trigerski kontrolni ulazni port na ikonu podsistemskog bloka.

Da se izabere tip trigera, treba otvoriti trigerski blok dijalog boks i izabrati jedan od izbora za **Triger type** parametar, kao što je pokazano na slici:



Simulink koristi različite simbole na trigerskom bloku i bloku podsistema da indicira rasteču ili opadajuću ivicu (ili obadvije). Naredna slika pokazuje trigerske simbole na bloku podsistema:



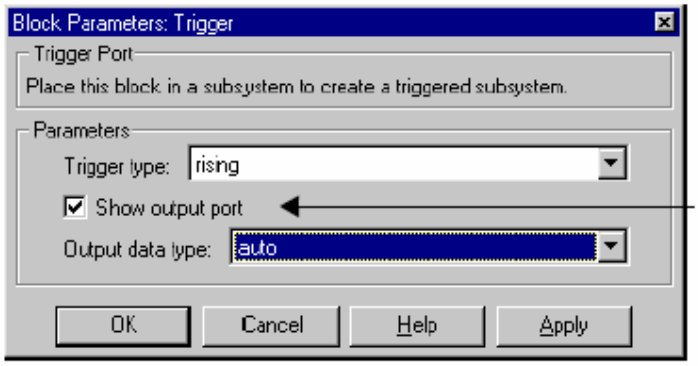
rastuča opadajuća obadvije

Izlazi i stanja izmedju trigerskih dogadjaja

Za razliku od omogućenih podsistema, trigerski podsistemi ovjek drže svoje izlaze na posljednjoj vrijednosti izmedju trigerskih dogadjaja. Takodjer, trigerski podsistemi ne mogu resetovati njihova stanja kada se trigeruju, stanja bilo kojih diskretnih blokova se drže zamrznuta izmedju trigerskih dogadjaja.

Izbacivanje na izlaz trigerskog kontrolnog signala

Opcija na dijalog boksu za trigerski blok omogućava korisniku da izbaci van bloka trigerski kontrolni signal. Da bi se izbacio, treba izabrati **Show output port** check boks.



Polje (**Output data type**) Izlazni tip podataka , omogućava korisniku da specificira tip podatka izlaznog signala kao **auto , int8**, ili **double.** Opcija auto prouzrokuje da tip

podatka izlaznog podatka bude setovan na tip podatka (bilo **int8** ili **double**) na portu na koji je signal spojen.

Podsistemi sa pozivom funkcije (function call subsystem)

Korisnik može kreirati trigerski podsistem čije izvršenje je odredjeno sa logikom koja je interno unutar S-funkcije , umjesto sa vrijednošću signala. Ovi podsistemi se zovu **podistem sa pozivom funkcije.**

Blokovi koji mogu biti sadržani u trigerovanom podsistemu

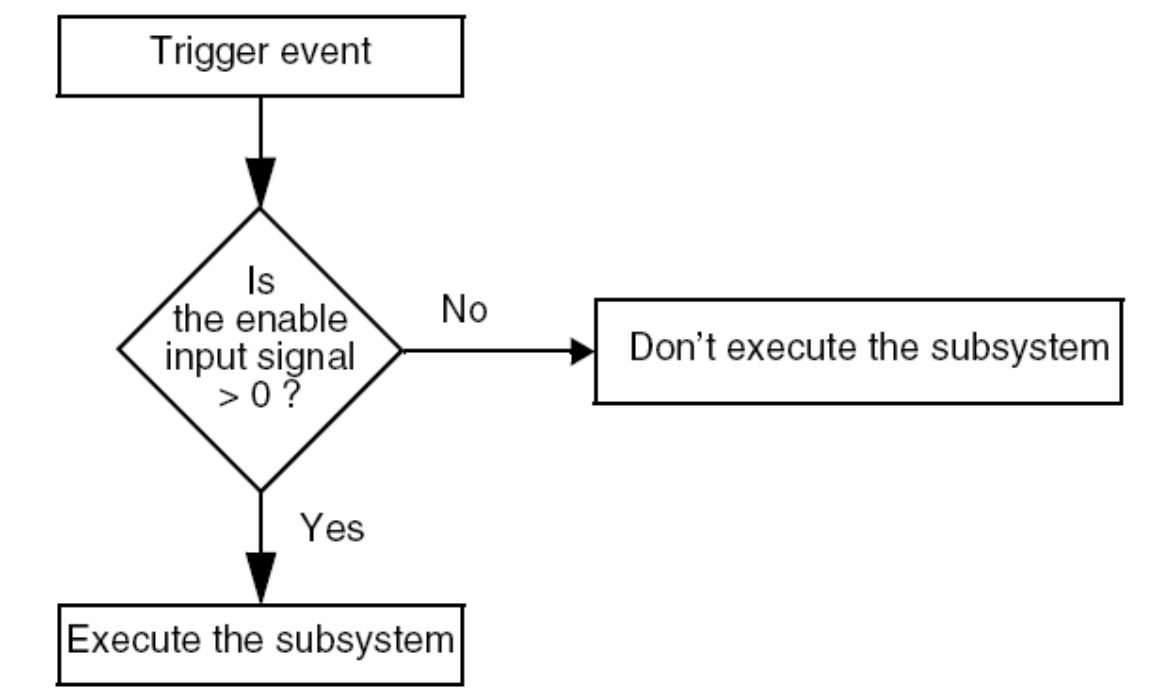
Trigerski sistemi se izvršavaju samo u specifičnim trenutcima u toku simulacije. Kao rezultat , jedini blokovi koji su pogodni za korištenje unutar trigerovanog podsistema su:

- blokovi koji naslijedjuju vrijeme samplovanja, kao što je blok logičkog operatora ili blok pojačanja.

- diskretni blokovi koji imaju vremena samplovanja setovana na -1, što indicira da je vrijeme sampliranja naslijedjeno od vodećeg bloka (bloka koji je ispred).

Trigerovani i omogućeni podsistemi

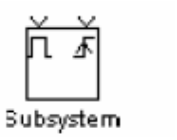
Treča vrsta uslovno izvršavanih podsistema kombinira obadva tipa uslovnih izvršenja. Ponašanje ovog tipa podsistema, nazvan trigerski i omogućeni podistem (triggered and enabled), je kombinacija omogućenog podsistema i trigerskog podsistema, kao što je pokazano na slijedećem dijagramu toka:



Trigerski i omogućeni podsistem sadrži i ulaz omogućenja (enable input port) i trigerski input port. Kada se desi trigerski dogadjaj, Simulink provjerava enable ulazni port da evaluira enable kontrolni signal. Ako je vrijednost veća od nule, Simulink izvršava podsistem. Ako su obadva ulaza vektori, podsistem se izvršava ako barem jedan element svakog vektora je nenulti.

Kreiranje trigerskog i omogućenog podsistema

Korisnik kreira trigerski i omogućeni podsistem vučenjem i Enable i Trigger bloka iz **Ports&Subsystem** biblioteke u podsistem. Simulink dodaje enable i trigger simbole kao i trigger i enable ulaze na ikonu bloka podsistema.

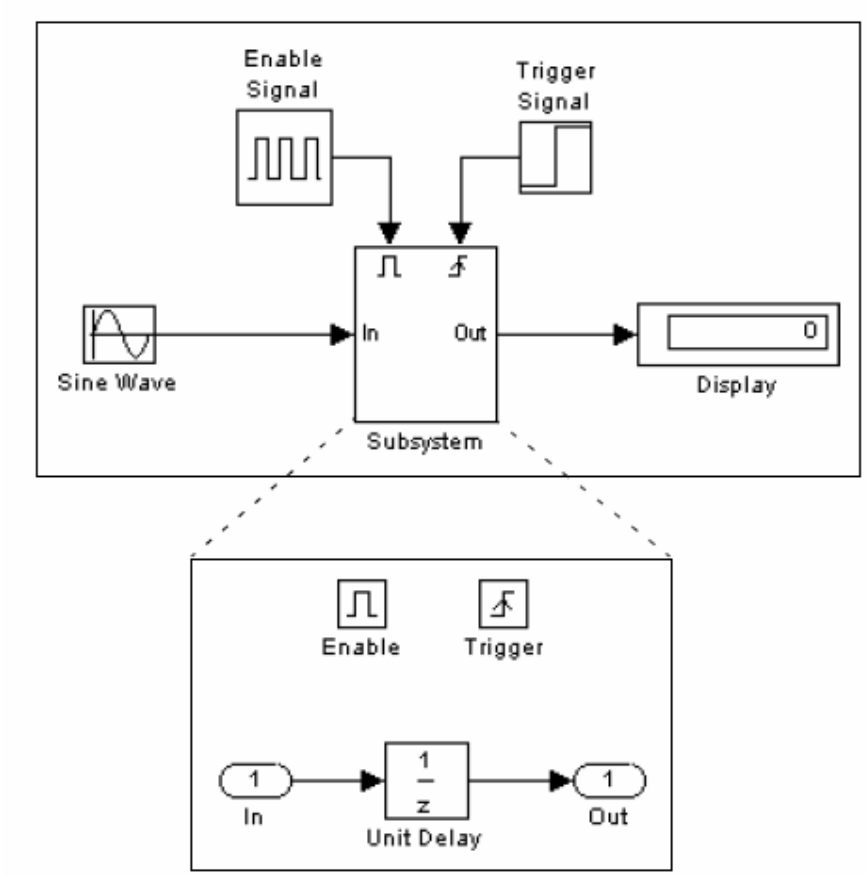


Mi možemo postaviti izlazne vrijednosti kada su trigerski i omogućeni podistem onemogućen, kao što to radimo i za omogućen podsistem. Takodjer možemo specificirati koje su vrijednosti stanja kada se podsistem ponovo omogućije.

Postavljanje parametara za Enable i trigger blokove je odvojeno. Procedure su iste kao one već opisane za individualne blokove.

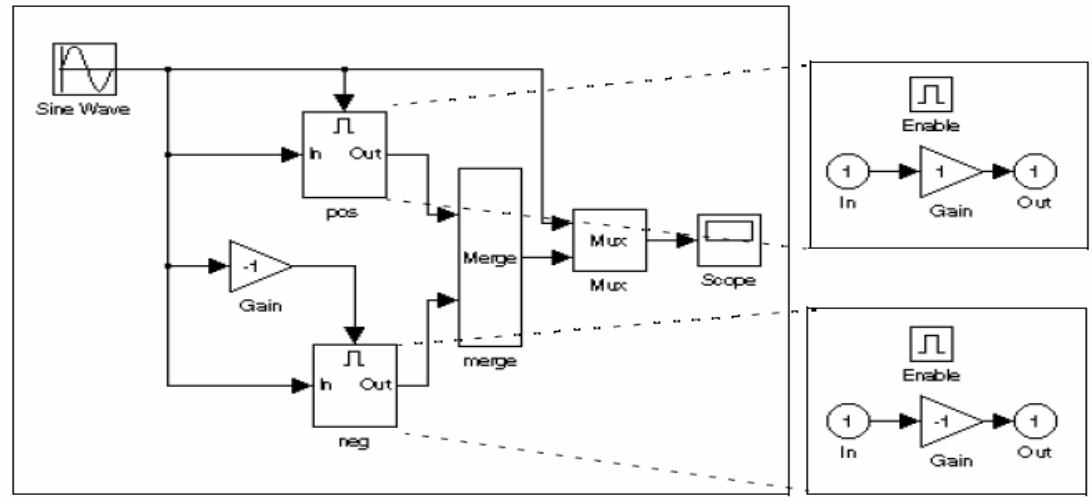
Primjer trigerovanog i omogućenog podsistema

Jednostavan primjer trigerovanog i omogućenog podsistema je ilustriran na modelu koji je prikazan na slici:



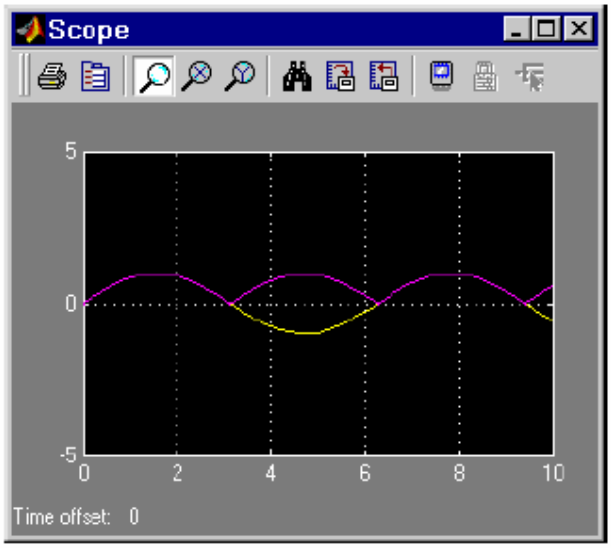
Kreiranje alternativno izvršavanih podsistema

Korisnik može koristiti uslovno izvršavane podsisteme u kombinaciji sa blokovima spajanja (merge blocks) , da kreira setove podsistema koji se alternativno izvršavaju, zavisno od tekućeg stanja modela. Na primjer, slijedeća slika pokazuje model koji koristi dva omogućena bloka i Merge blok da se modelira invertor , tj. AC/DC pretvarač.



U ovom primjeru, blok koji je označen "pos" je omogućen kada AC valni oblik je pozitivan: on prenosi valni oblik neizmjenjen na njegov izlaz. Blok označen "neg" je omogućen kada je valni oblik negativan: on invertira valni oblik. Blok spajanja (merge) prenosi izlaz tekućeg omogućenog bloka na Mux blok, koji ga prenosi na izlaz, zajedno sa orginalnim valnim oblikom, na blok osciloskopa.

Na osciloskopu će se pojaviti displej kao na slici:



Blokovi kontrole toka programa

Blokovi kontrole toka programa se koriste da implementiraju logiku slijedećih C-tip iskaza programske kontrole u Simulinku:

- for
- if-else
- switch
- while (uključuje while i do-while iskaze kontrole toka)

Mada su svi pobrojani iskazi implementabilni u Stateflow, ovi blokovi su namjenjeni da obezbjede korisnicima Simulinka alate koji ispunjavaju njihovu potrebu za jednostavnim logičkim iskazima.

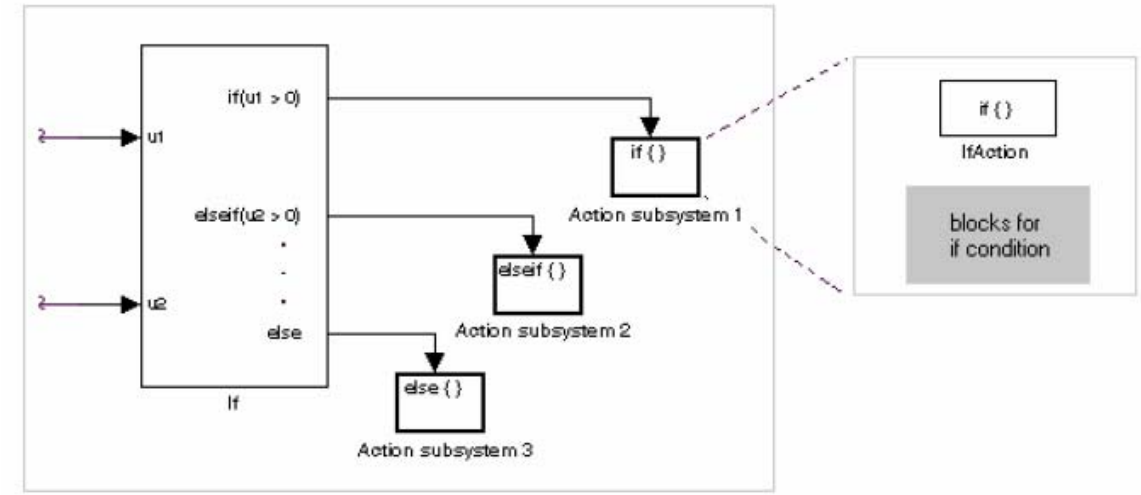
Kreiranje uslovnih iskaza kontrole toka programa

Korisnik može kreirati C-tip uslovne iskaze kontrole toka programa koristeći obične podsisteme i slijedeće blokove iz biblioteke **Ports&subsystems** .

C iskaz	Blokovi koji se koriste
If-else	If, Action port
Switch	Switch case, Action port

If-else iskazi kontrole toka

Slijedeći dijagram prikazuje generalizirani **if-else** iskaz kontrole programskog toka , implementiran u Simulinku.



Konstruisaćemo Simulink if-else iskaz kontrole toka na slijedeći način:

- Obezbjediti ulazni podatak u **If** blok za konstruciju **if-else** uslova. Ulazi u If blok su setovani u dijalog boksu If blok osobina. Interno, oni su označeni kao u1, u2,..., un i koriste se da konstruišu izlazne uslove.

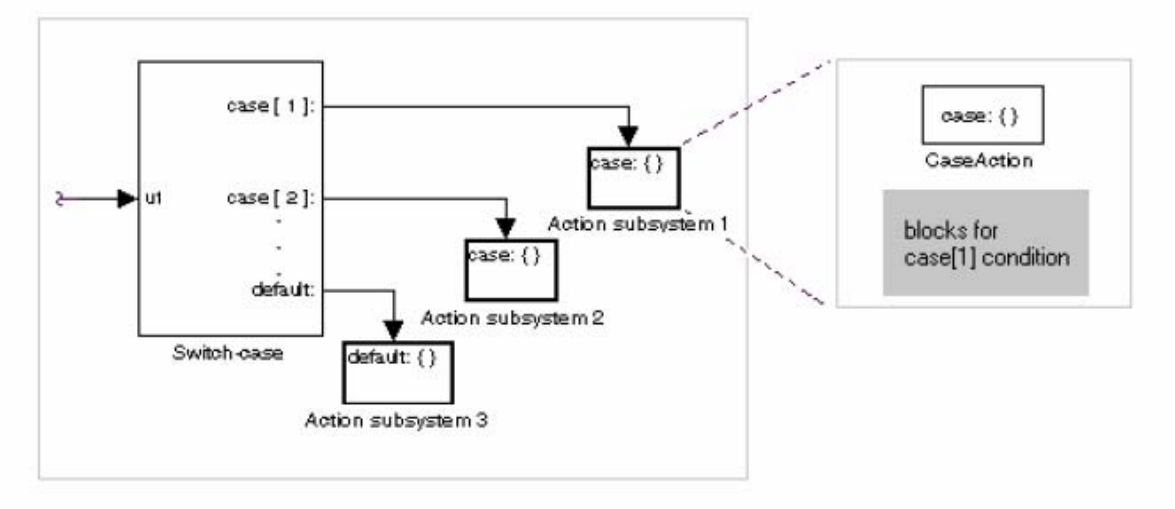
- Setovati izlazni port **if-else** uslova za If blok.
  Izlazni portovi za If blok se takodjer postavljaju u svom dijalog boksu osobina. Koristimo ulazne vrijednosti u1,u2,...un da izrazimo uslove za if, elsif, i else uslove polja u dijalogu. od ovih, samo se zahtjeva if polje. Mi možemo unjeti višestruke elseif uslove i izabrati check boks da omogućimo else uslov.

- Spojiti svaki uslovni izlazni port sa Action podsistemom.
  Svaki if, elseif, i else uslov izlazni port na If bloku je spojen na podsistem koji će se izvršiti ako je slučaj port istinit . Korisnik kreira ove podisteme stavljajući Action port blok u podistem. Ovo kreira **atomic Action** podistem sa portom koji se zove **Action**, koji se može onda povezati na uslov na If bloku. Jedanput kada je spojen, podsistem uzima identitet uslova sa kojm je spojen i ponaša se kao omogućeni podsistem.

Opaska :
Svi blokovi u Action podsistemu koji su vodjeni sa If ili Switch caseom mora se izvršavati sa istom brzinom kao i vodeći blok.

Switch iskazi kontrole programskog toka

Slijedeći dijagram prikazuje poopšteni switch iskaz kontrole toka implementiran u Simulinku.



Konstruisati Simulink iskaz za switch kontrolu toka na slijedeći način:

- obezbjediti ulaz podataka u argument ulaz bloka Switch case. Ulaz u Switch case blok je argument u switch control flow iskazu. Ova vrijednost odredjuje odgovarajući slučaj za izvršenje. Noninteger ulazi u ovaj port se skračuju.
- Dodati slučajeve na switch case blok baziran na numeričkoj vrijednosti ulaza argumenta,
  Korisnik dodaje cases na Switch case bloku putem dijalog boksa osobina Switch case bloka. Case mogu biti jednostruki ili viševrijednost (multivalued). Korisnik može takodjer dodati opcioni default case, koji je istinit ako ostali cases nisu istiniti. Jedanput kada se doda , ovi slučajevi se pojavljuju kao izlazni portovi na Switch Case bloku.

- Spojiti case izlazni port svakog Switch case bloka na Action podsistem
  Svaki case izlaz Switch case bloka je spojen na podsistem koji će se izvršiti ako je case porta istinit. Korisnik kreira ove podsisteme postavljajući Action Port blok u podistem. Ovo kreira atomic podistem sa portom koji se naziva Action, koji onda povezujete na uslov na Switch case bloku. Jedanput kada je spojen, podsitem uzima na sebe identitet uslova i ponaša se kao omogućeni podsitem. Postaviti sve programske blokove izvršene za taj case u ovaj podistem.

Opaska :

Nakon što je podsistem za specifičan slučaj izvršen, jedan implicirani prekid (break) se izvršava koji izlazi iz iskaza switch kontrole toka upotpunosti. Simulink switch kontrolni iskazi ne ispoljavaju "propadajuče" ponašanje kao switch iskazi u C.

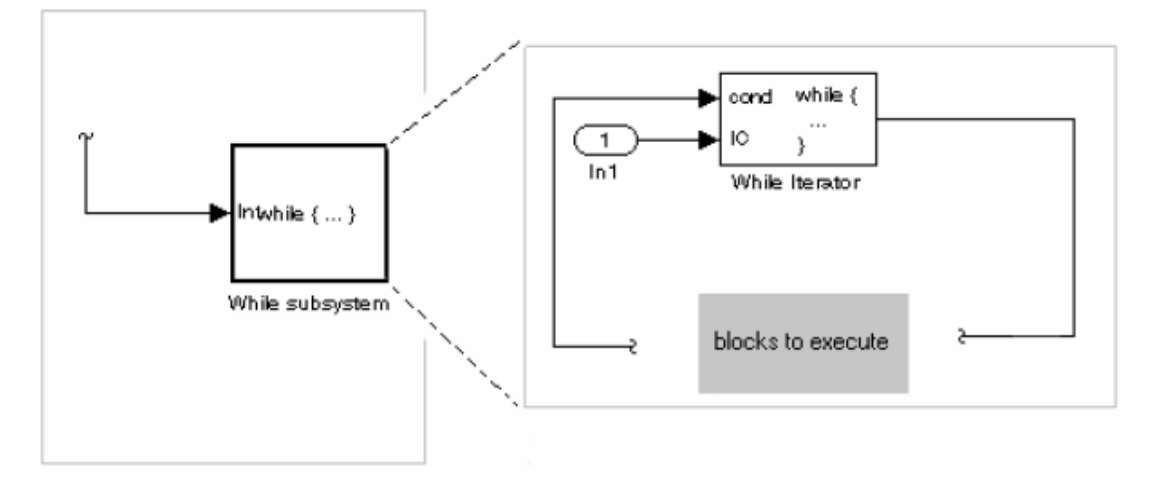
Kreiranje iterativnih iskaza kontrole programa

Korisnik može kreirati C-tip iterator kontrolne iskaze koristeći podsisteme i slijedeće blokove iz biblioteke **Ports&Subsystems:**

C iskaz	Korišteni blokovi
Do-while	While iterator
for	For iterator
while	While iterator

While kontrolni iskazi

Slijedeći dijagram pokazuje poopšteni C-tip while kontrolnog iskaza implementiranog u Simulinku.



U Simulinku while kontrolni iskaz , While iteratorski blok iterira sadržaj While podsistema, jedan atomic podsistem. Za svaku iteraciju While Iterator bloka, programiranje bloka While podsistema izvršava jedan prolaz kroz svoje blokove.
Konstruisati Simulink while kontrolni iskaz kako slijedi:

- Postaviti While Iterator blok u podsistem
  Host podsistem postaje while kontrolni iskaz kao što je indicirano sa njegovom novom labelom , while $\{.....\}$. Ovi podsistemi se ponašaju kao trigerovani podsistemi. Ovaj podsistem je host za blok programiranje koje želimo da iteriramo sa While Iterator blokom.

- Obezbjediti ulaz podataka za početne uslove na ulaznom portu While Iterator bloka
  While Iterator blok zahtjeva početni uslov na ulazu (označen sa IC) za njegovu prvu iteraciju. Ovo mora imati porijeklo van While podsitema. Ako je ova vrijednost različita od nule, desiće se prva iteracija.

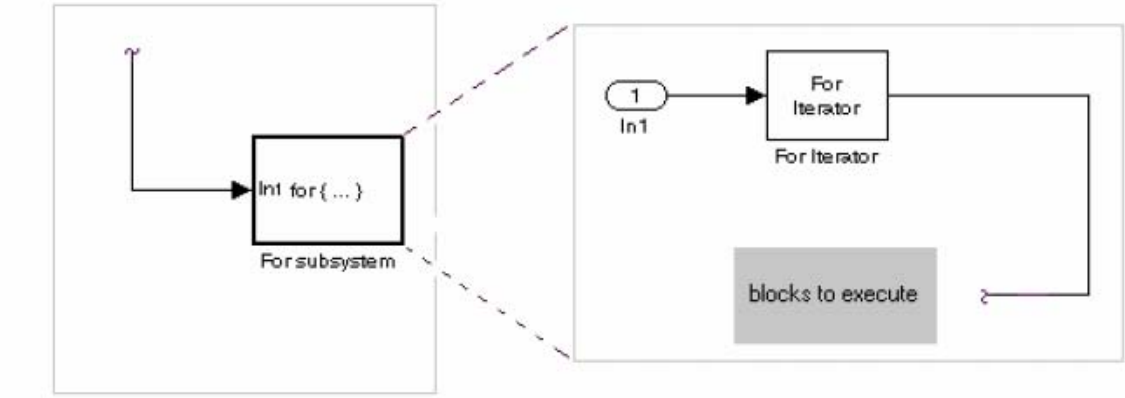
- Obezbjediti ulazne podatke za uslovne portove While Iterator bloka.
  Uslovi za preostale iteracije se prenose na port ulaznih podataka označen kao **cond**. Ulaz u ovaj port mora dolaziti unutar While podsistema.

- Možemo postaviti While Iterator blok da izbacije svoju Iterator vrijednost kroz dijalog boks njegovih osobina.
  Vrijednost iteratora je 1 za prvu iteraciju i inkrementirana za q za svaku narednu iteraciju.

- Korisnik može promjeniti iteraciju While Iterator bloka na do-while putem njegovog dijalog boksa za osobine.
  Ovo će promjeniti i labelu host podsistema to do $\{.....\}$ while. Sa do-while iteracijom, While iteracioni blok nema više port za početni uslov IC, pošto svi blokovi u podsistemu se izvršavaju jedanput prije nego uslovni port (označen cond) je čekiran.

For kontrolni iskaz

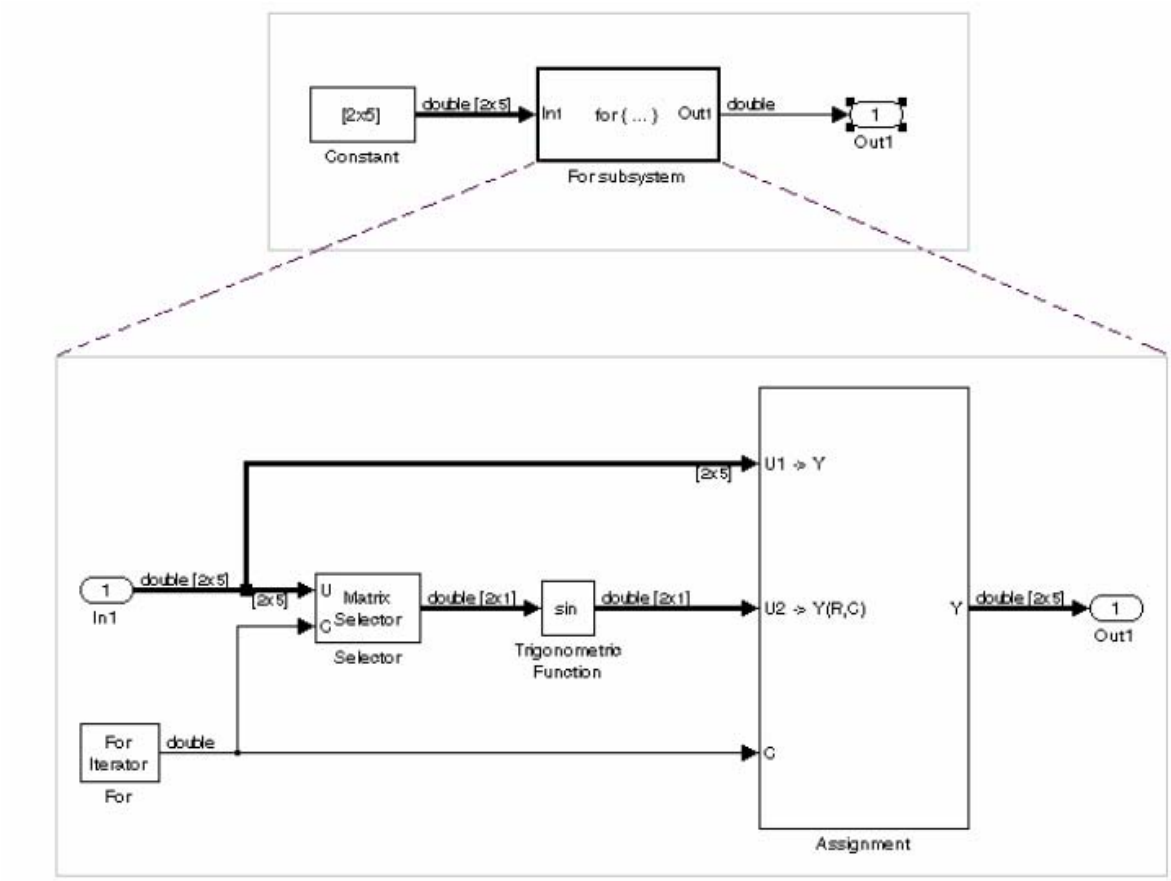
Slijedeći dijagram pokazuje opšti for kontrolni iskaz kako je implementiran u Simulinku:



U Simulinku for kontrolnom iskazu , For Iterator blok iterira sadržaj For Iterator podsistema , atomic podsistema. Za svaku iteraciju For Iterator bloka, programiranje bloka For Iterator podsistema izvršava jedan kompletan ciklus kroz njegove blokove.

Konstruišemo Simulink za **for** kontrolni iskaz na slijedeći način:

- Povući For Iterator podsistem blok iz biblioteke u prozor modela
- Može se postaviti za For Iterator blok da uzme vanjski ili interni ulaz za broj iteracija koje izvršava.
  Putem dijalog boksa osobina For Iterator bloka možemo postaviti da uzima ulaz za broj iteracija putem porta označenog sa N. Ovaj ulaz mora dolaziti izvana za For Iterator podsistem. Može se takodjer postaviti broj iteracija direktno u diajlog boksu osobina.
- Možemo takodjer postaviti For iterator blok da izbacuje svoju iteracionu vrijednost radi korištenja u programiranju unutar bloka For Iterator podsistema.
  Iteratorska vrijednost je za prvu iteraciju1 i inkrementira se za 1 za svaku narednu iteraciju. For iteracioni blok radi dobro sa blokom doznačavanja (assignment blokom) da ponovno doznači vrijednosti u vektoru ili matrici. Ovo je demonstrirano u slijedećem primjeru. Primjetimo dimenzije matrica u podatcima koji se prenose.



Gornji primjer izbacuje sinusni signal iz jedne ulazne 2 x 5 matrice (2 reda , 5 kolona), koristeći For podsistem koji sadrži Assignment blok. Proces je kako slijedi:

1. 2 x 5 matrica je ulaz u Selector blok i Assignment blok
2. Selector blok izvlači 2 x 1 matricu iz ulazne matrice kod vrijednosti kolone indicirane sa tekučom iteracionom vrijednošću od For Iterator bloka.

3. Uzima se sinus od 2 x 1 matrice
4. Vrijednost sinusa 2 x 1 matrice se prenosi na Assignment blok.
5. Assignment blok koji uzima originalnu 2 x 5 matricu kao jedan od svojih ulaza, doznačuje 2 x1 matricu natrag u originalnu matricu u koloni koja je indicirana sa iteracionom vrijednošću.
   Redovi specificirani za ponovno doznačavanje u dijalog boksu osobina za Assignment blok u gornjem primjeru su [1,2]. Pošto postoje samo dva reda u originalnoj matrici, mogli bi takodjer specificirati -1 za redove , tj. sve redove.

Poredjenje stateflow i iskaza kontrole toka (stateflow and control flow)

Stateflow već posjeduje logičke mogućnosti Simulink kontrolnih iskaza. On može zvati podsisteme poziva funkcije (function-call subsystem). Medjutim, pošto Stateflow obezbjedjuje mnogo više u logičkoj sofistikaciji, ako su naši zahtjevi jednostavniji, možemo naći da su mogućnosti Simulink blokova kontrole zadovoljavajuće. Nadalje, iskazi kontrole programa nude nekoliko pogodnosti koje će biti opisane u nastavku.

Vremena sampliranja

Function-call podistemi koje Stateflow može pozivati su trigerski podsistemi. Trigerski podsistemi naslijedjuju njihova jednostavna vremena iz pozivajučeg bloka. Ipak, Action podsistemi korišteni u if-else i switch kontrolnim iskazima i While i For podsistemi koji čine while i for kontrolne iskaze su omogućeni podsistemi. Omogućeni podsistemi mogu imati njihova vlastita vremena samplovanja, nezavisno od pozivajućeg bloka. Ovo takodjer dozvoljava da se koristi više kategorija blokova u itriranom podsistemu nego u Function-call podsistemu.

Resetovanje stanja kada je ponovno omogućen

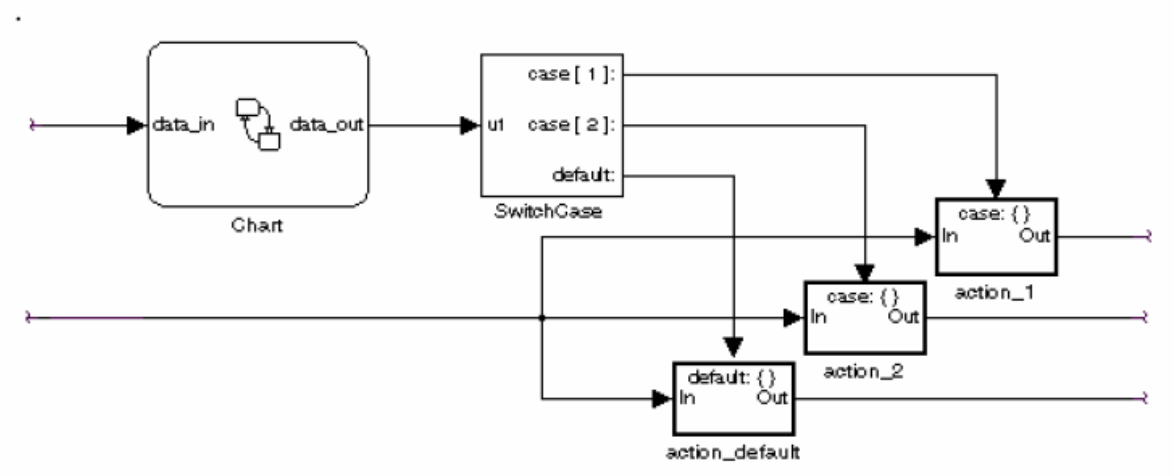
Simulink blokovi iskaza kontrole toka dozvoljavaju da zadržimo ili resetujemo (na njihove inicijalne vrijednosti) vrijednosti stanja za Action, For i While podsisteme kada su ponovno omogućeni.

Koristeći Statefow sa blokovima kontrole toka

Korisnik može željeti da razmatra mogućnost korištenja Stateflow i Simulink blokova zajedno. Naredna sekcija sadrži neke primjere koji daju sugestije kako se ovo dvoje može kombinirati.

Korištenje stateflow sa if-else ili switch podsistemima

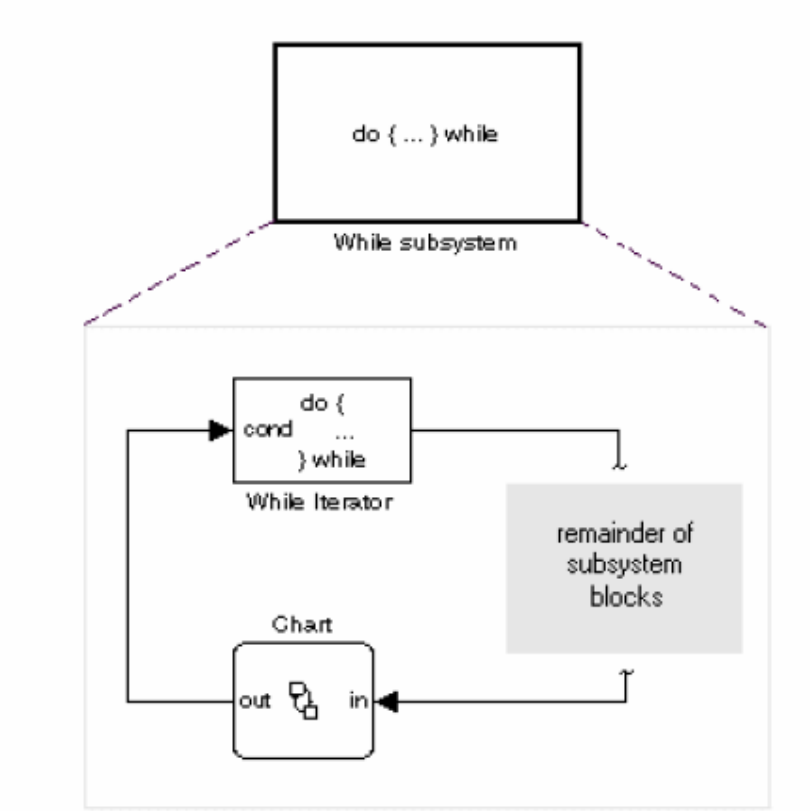
U slijedećem modelu , Stateflow postavlja jednu od niza vrijednosti u Stateflow data objekat. Nakon završetka charta , Simulink iskaz if kontrole toka koristi ovaj podatak da napravi uslovnu odlluku:



U ovom slučaju kontrola je data Switch Case bloku, koji koristi vrijednost da izabere jedan od nekoliko case podsistema da izvrši.

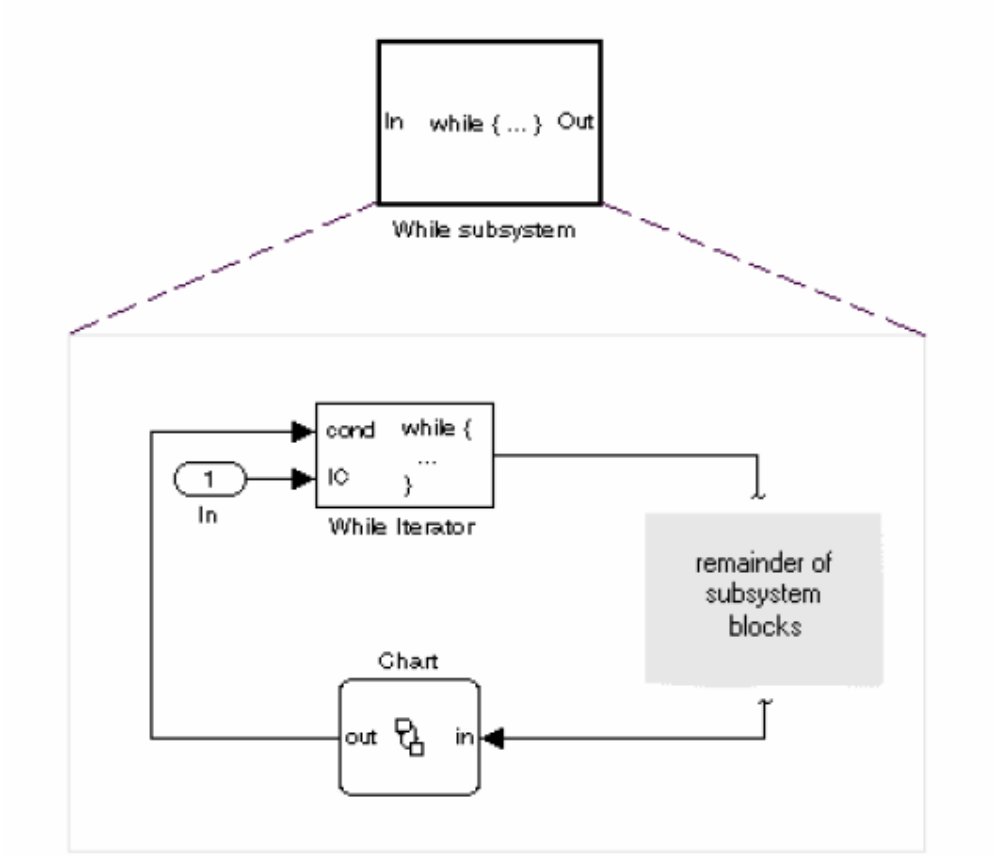
Korištenje Stateflow sa While podsistemima.

U slijedećem dijagramu , Stateflow izračunava vrijednost objekat podataka koji je na raspolaganju uslovnom ulazu na While Iterator bloku u do-while modu.



While Iterator blok ima iterativnu kontrolu nad host podsistemom, koji uključuje Stateflow chart blok. U do-while modu, While blok je garantiran da radi za prvu vrijednost iteracije (= 1). Za vrijeme tog vremena, Stateflow chart je probudjen i postavlja vrijednost podataka koje koristi While Iterator blok , koji je evaluiran kao uslov za slijedeći next while iteraciju.

U slijedećem dijagramu , While blok je sada postavljen u while mode. U ovom modu , While Iterator blok mora imati ulaz u njegov port početnih uslova da bi se izvršio sa svojom prvom iteracionom vrijednošću. Ova vrijednost mora dolaziti izvan While podsistema .



Ako je početni uslov istinit, While Iterator blok probudjuje Stateflow chart i izvršava se do svog završetka. Za vrijeme tog vremena Stateflow chart postavlja podatke, koje While iterator uslovni port koristi kao uslov za slijedeću iteraciju.

Diskretizator modela

Diskretizator modela selektivno zamjenjuje kontinualne Simulink blokove sa diskretnim ekvivalentima. Diskretizacija je kritični korak u dizajnu digitalnog regulatora kao i za hardware kod simulacija u konturi. Korisnik može koristiti ovaj alat da pripremi kontinualne modele za korištenje sa Real Time radionicom (workshop) sa imbediranim

koderom (Real time Workshop with embedded Coder), koji podržava samo diskretne blokove.

Diskretizator modela omogućava korisniku da :

- identificira kontinualne blokove modela
- Promjeni parametre bloka od kontinualnih na diskretne
- Primjeni setinge diskretizacije na sve kontinualne blokove u modelu ili na izabrane blokove
- Kreira konfigurabilne podsisteme koji sadrže višestruke diskretizacione kandidate zajedno sa originalnim kontinualnim blokovima.
- Preključuje izmedju različitih diskretizacionih kandidata i podiže rezultirajuće simulacije modela.

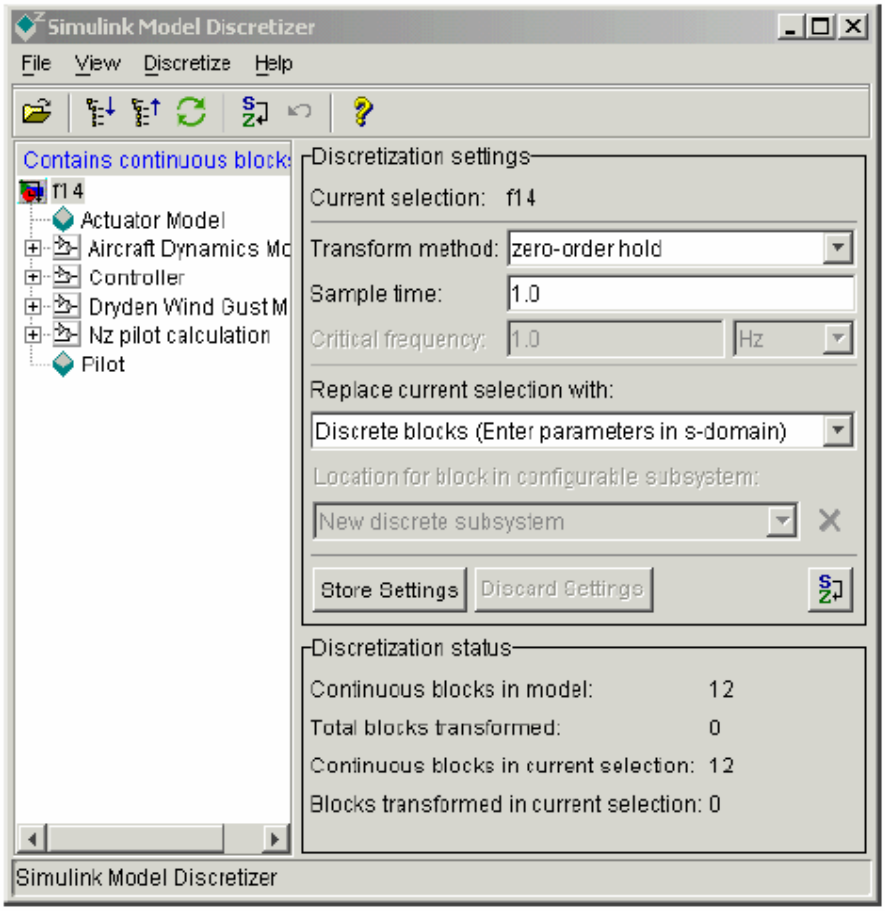
Zahtjevi

Da bi se koristio diskretizator modela, potrebno je imati instaliran Control system Toolbox , verzija 5.2 ili viša.

Diskretizacija modela sa njegovog GUI interfejsa

Start diskretizera modela

Da bi se otvorio alat , treba izabrati **Model Discretizer** iz **Tools** menija u Simulink modelu. Ovo će prikazati prozor Simulink Model Discretizer-a kao na slici:



Alternativno, može se otvoriti diskretizator modela iz MATLAB komandnog prozora koristeći **slmdldiscui** funkciju.

Slijedeća komanda će otvoriti **Simulink model discretizer** prozor sa f14 modelom :

slmdldiscui('f14')

Specifikacija metode transformacije

Metod transformacije specificira tip algoritma koji se koristi u diskretizaciji.

Transform method lista sadrži slijedeće opcije:

- zero –order hold (hold kolo nultog reda na ulazima)
- first-order hold (hold kola prvog reda , sa linearnom interpolacijom medju ulazima)
- tustin Bilinearna (Tustin) interpolacija
- tustin sa prewarpingom
  Tustin aproksimacija sa odmotavanjem (prewarping) frekvencije
- upareni polovi i nule (matched pole-zero)
  Upareni polovi i nule metod koji vrijedi samo za SISO sisteme.

Specifikacija vremena sampliranja

Unjeti vrijeme sampliranja u polje **Sample time**. Korisnik može specificirati vrijeme ofseta unoseći dvoelementni vektor za diskretne blokove ili kofigurabilne podsisteme. Prvi element je vrijeme sampliranja a drugi element je vrijeme ofseta. unos [1.0 0.1] će specificirati vrijeme sampliranja od 1 sekunde sa 0.1 sekunda ofsetom. Ako ofset nije specificiran default vrijednost je nula.

Možemo takodjer unjeti varijable iz radnog prostora kada diskretiziramo blokove u s-domen.

Specificiranje metoda diskretizacije

Metod diskretizacije se specificira u polju **Replace current selection width.** Opcije na raspolaganju su:

- **Discrete blocks (Enter parameters in s-domain)**
  Kreira diskretne blokove čiji parametri su zadržani od odogovarajućeg kontinualnog bloka

- **Discrete blocks (Enter parameters in z – domain)**
  Kreira diskretni blok čiji parametri su "hard-coded" vrijednosti stavljene direktno u dijalog bloka.

- **Configurable subsystem (Enter parameters in s-domain)**
  Kreira višestruke diksretizacione kandidate koristeći s-domen vrijednosti za tekuču selekciju. Konfigurabilni podsistem se može sastojati od jednog ili više blokova.

- **Configurable subsystem (Enter paramters in z – domain)**
  Kreira višestruke diskretizacione kandidate u z – domenu za tekuču selekciju. Konfigurabilni podsistem se može sastojati od jednog ili više blokova

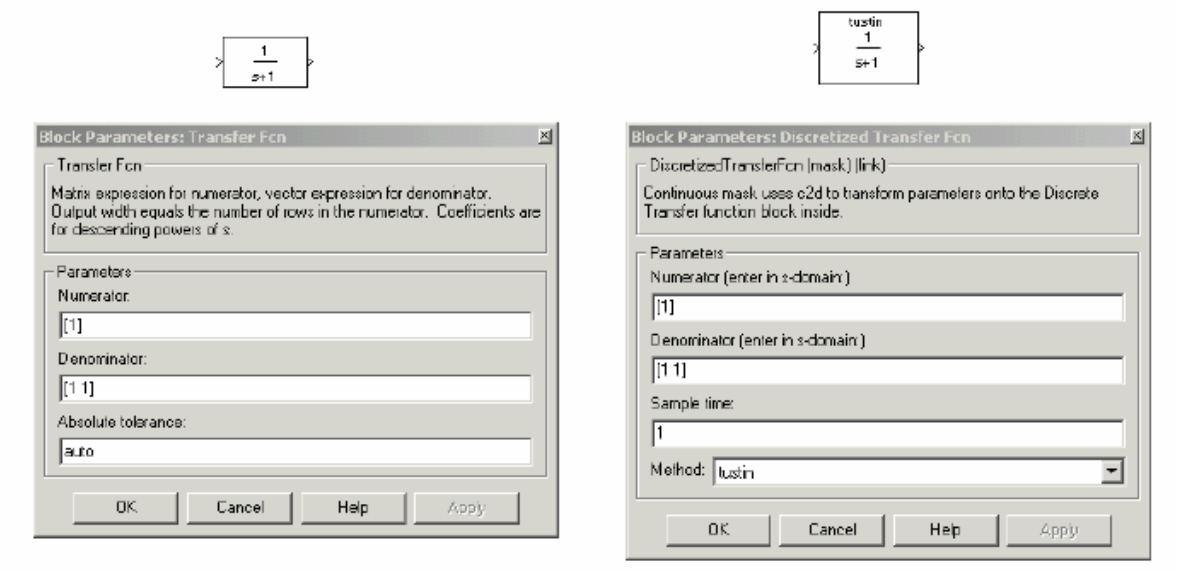
Diskretni blokovi (unošenje parametara u s-domen)

Kreira diskretni blok čiji parametri su zadržani iz odgovarajućeg kontinualnog bloka. Vrijeme sampliranja i diskretizacioni parametri su takodjer prisutni u dijalog boksu bloka. Ovaj blok je implementiran kao maskirani diskretni blok koji koristi c2d da transformira kontinualne parametre u diskretne parametre u maskiranom inicijalizacionom kodu.

Ovi blokovi imaju jedinstvenu mogućnost ponovnog vraćanja na kontinualno ponašanje ako vrijeme sampliranja se promjeni na nulu. Unoseći vrijeme sampliranja kao varijablu radnog prostora (Ts, naprimjer), dozvoljava laganu promjenu od kontinualnog na diskretni model i natrag na kontinualni.

Opaska :
Parametri nisu podesivi kada Inline parameters su izabrani u dijalog boksu modelovih **Simulation Parameters.**

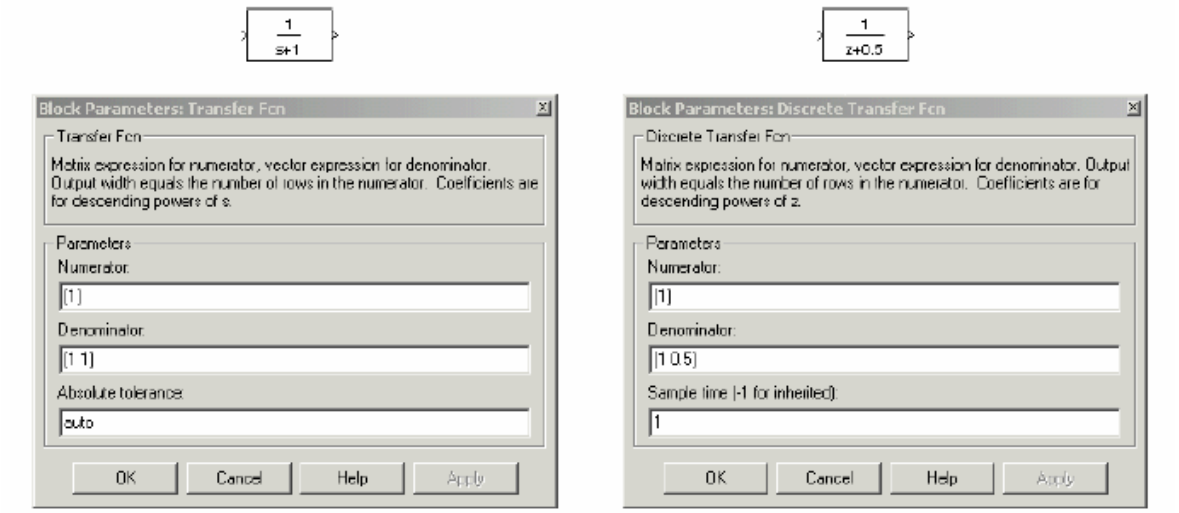
Naredna slika pokazuje blok kontinualne prenosne funkcije i desno od njega je taj blok nakon dikretizacije u s –domenu. Dijalog boks **Block Parameters** za svaki blok je pokazan ispod bloka:



Diskretni blokovi (unesi parametre u z-domen)
Discrete blocks (Enter parameters in z-domain)

Kreira diskretni blok čiji parametri su "hard-coded" vrijednosti smještene direktno u dijalog bloka. Diskretizator modela koristi **c2d** funkciju da dobije diskretizacione parametre ako su potrebni.

Naredna slika pokazuje blok kontinualne prenosne funkcije a do njega je blok prenosne funkcije koji je bio diskretiziran u z – domenu. Dijalog boks **Block Parameters** za svaki blok je pokazan ispod bloka.



Konfigurabilni podsistem (unošenje parametara u s-domen)
Configurable subsystem (Enter parameters in s-domain)

Kreirati višestruke diskretizacione kandidate koristeći s-domen vrijednosti za tekuću selekciju. Konfigurabilni podsistem se može sastojati od jednog ili više blokova.

Polje **Location for block in configurable subsystem** postaje aktivno kada se ova opcija izabere. Ova opcija dozvoljava korisniku da kreira novi konfigurabilni sistem ili da prepiše preko postojećeg.

Konfigurabilan podsistem (unjeti parametre u z-domen)
Configurable subsystem (Enter parameters in z-domain)

Kreirati višestruke diskretizacione kandidate u z-domenu za tekuću selekciju. Konfigurabilan podsistem se može sastojati od jednog ili više blokova.
Polje **Location for block in configurable subsystem** postaje aktivno kada se izabere ova opcija. Ova opcija dozvoljava da ili kreiramo novi konfigurabilni podsistem ili prepišemo preko postojećeg.

Konfigurabilni podsistemi se pohranjuju u biblioteku koja sadrži diskretizacione kandidate kao i originalni kontinualni blok. Biblioteka se naziva **<model name>_disc_lib** i biće pohranjena u tekući direktorij. Naprimjer biblioteka koja sadrži konfigrabilni podsistem kreiran iz f14 modela će biti nazvana **f14_disc_lib**.

Ako se kreiraju višestruke biblioteke iz istog modela, tada imena fajlova će se inkrementirati. Naprimjer, biblioteka drugog konfigurabilnog podsistema bit će nazvana **f14_disc_lib2.**

Korisnik može otvoriti konfigurabilnu biblioteku podsistema sa desnim klikom na Simulink model i izabirući **Link options >> Goto library block** , iz pop-up menija.

Diskretizacija blokova

Da diskretiziramo blokove koji su linkovani sa bibliotekom, moramo ili diskretizirati blokove u samoj biblioteci ili onemogućiti linkove sa bibliotekom u prozoru modela.
Postoje dva metoda za diskretizaciju blokova

Selekcija blokova i diskretizacija

1. Izabrati blok ili više njih u lijevom dijelu prozora Model Discretizer-a gdje je pokazana drvo struktura modela . Da bi se izabrali višestruki blokovi, treba pritisnuti na **Ctrl** taster dok se vrši izbor blokova.

2. Izabrati komandu **Discretize current block** iz **Discretize** menija ako je jedan blok izabran ili izabrati **Discretize selected blocks** iz **Discretize** menija ako je više blokova izabrano.

Možemo takodjer diskretizirati tekući izabrani blok klikanjem na taster Discretize koji izgleda kao:

Pohranjivanje diskretizacionih podešenja i primjenjivanje na izabrane blokove u modelu.

1. Unjeti diskretizaciona podešenja za tekući blok
2. Kliknuti na **Store Settings**

Ovo će dodati tekući blok sa diskretizacionim podešenjima u grupu prethodno setovanih blokova.

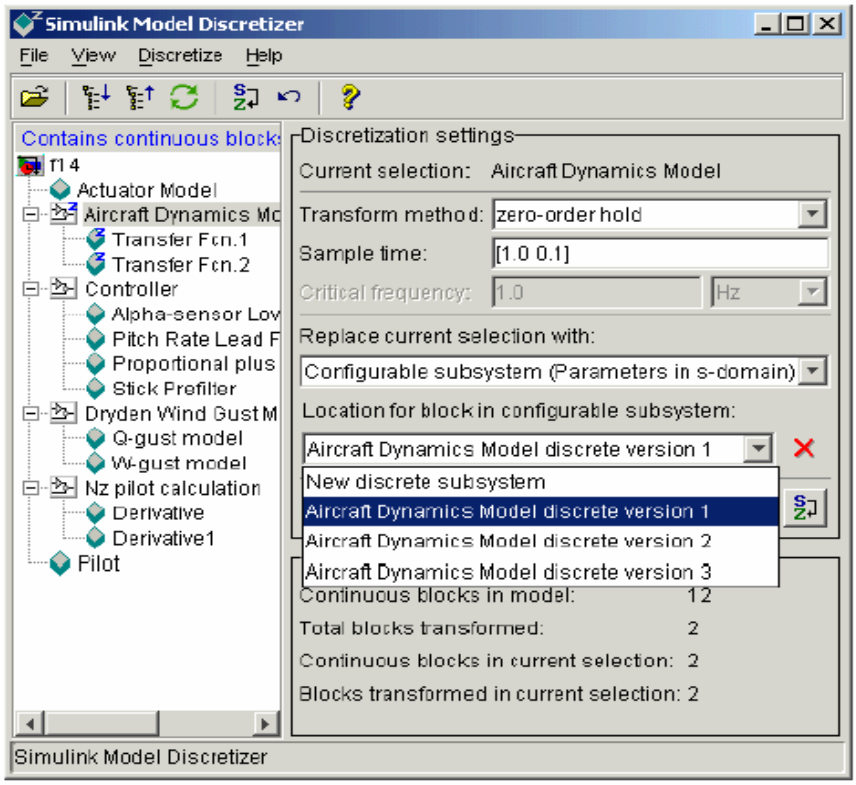
3. Ponoviti korake 1 i 2 ako je potrebno
4. Izabrati **Discretize preset blocks** iz **Discretize** menija

Analiza diskretizovanog modela

Model Discretizer prikazuje model u hijerarhijskom stablu .

Gledanje diskretizovanih blokova

Ikona bloka u prikazu stabla će postati naglašena sa "**z**" kada je blok diskretiziran. Naredna slika pokazuje da je podsistem Aircraft Dynamics Model diskretizovan u konfigurabilni podsistem sa tri diskretizaciona kandidata. ostali blokovi u ovom f14 modelu nisu bili diskretizirani.

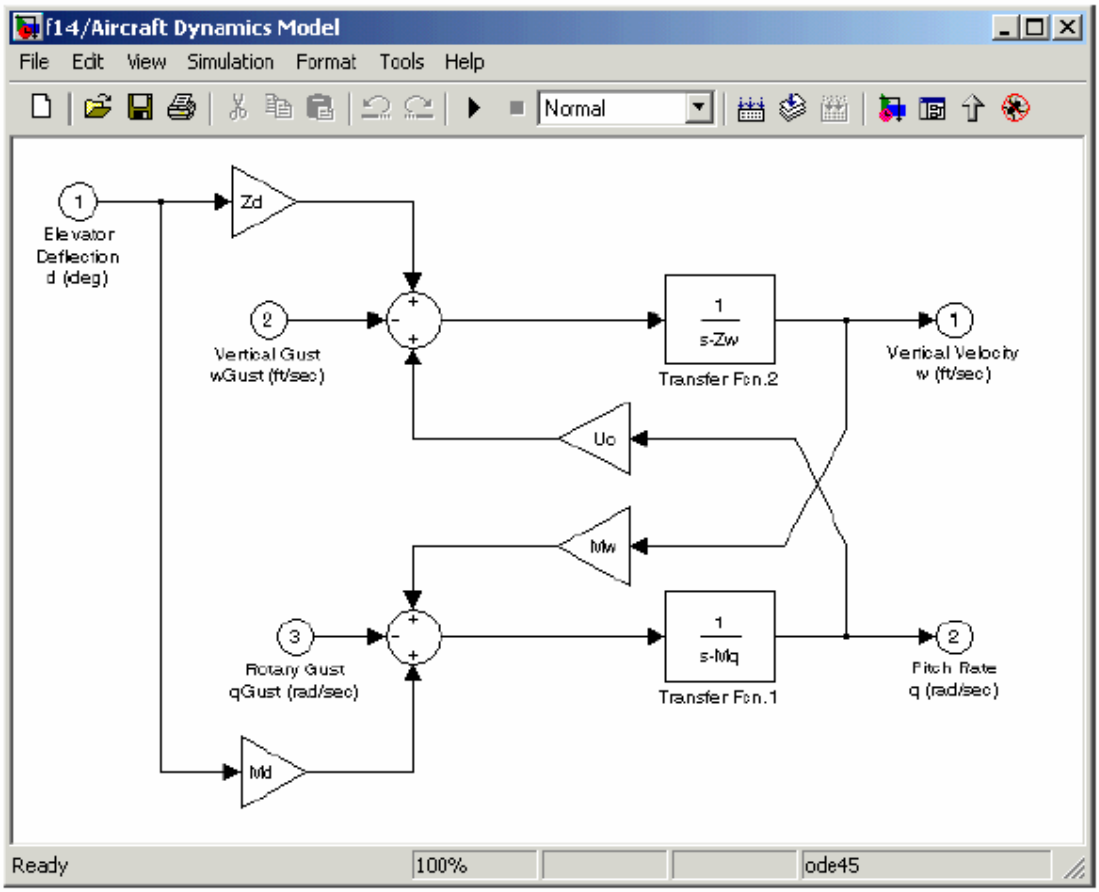


Diskretizacija blokova iz Simulink modela

Možemo zamjeniti kontinualne blokove u Simulink modelu sa ekvivalentnim blokovima diskretiziranim u s-domenu koristeći diskretizacionu biblioteku.
Procedura u nastavku pokazuje kako zamjeniti kontinualni blok prenosne funkcije u podsistemu Aircraft Dynamics Model za model f14 sa diskretiziranim blokom prenosne funkcije iz biblioteke diskretizacije. Blok je diskretiziran u s-domenu sa metodom transformacije sa hold kolom nultog reda i sa 2 sekunde vremenom sampliranja.

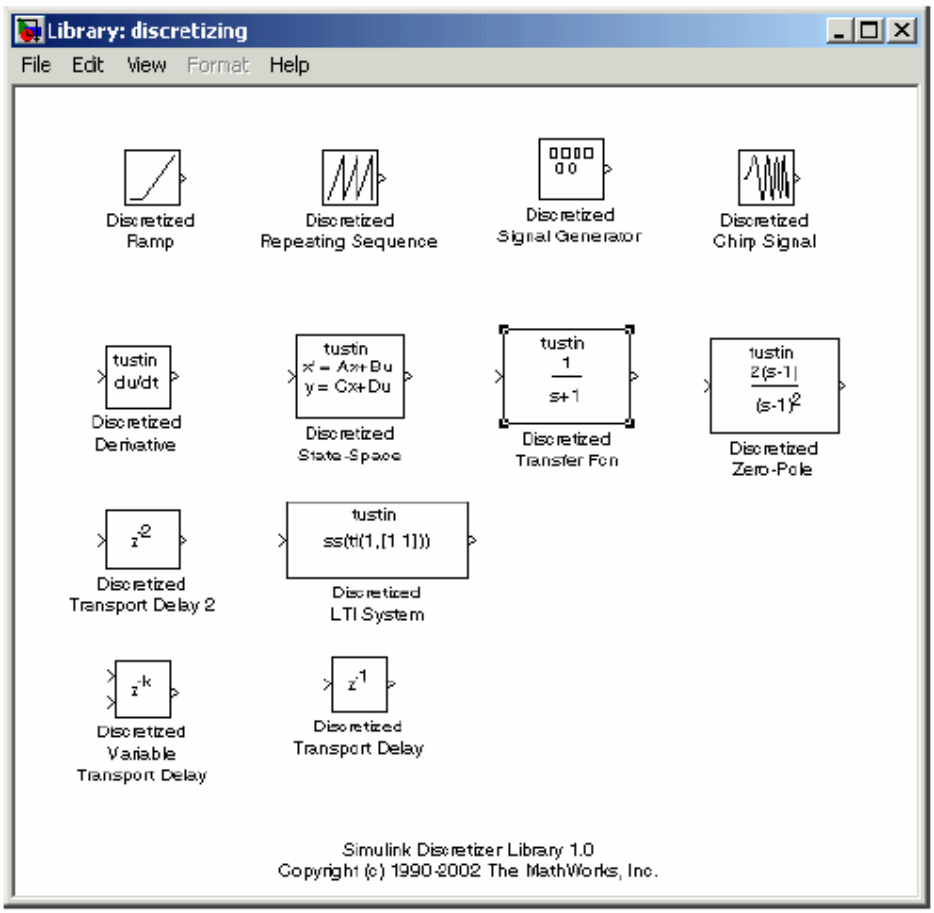
Postupak je slijedeći :

1. Otvoriti f14 model
2. Otvoriti podsistem Aircraft Dynamics Model u f14 modelu.

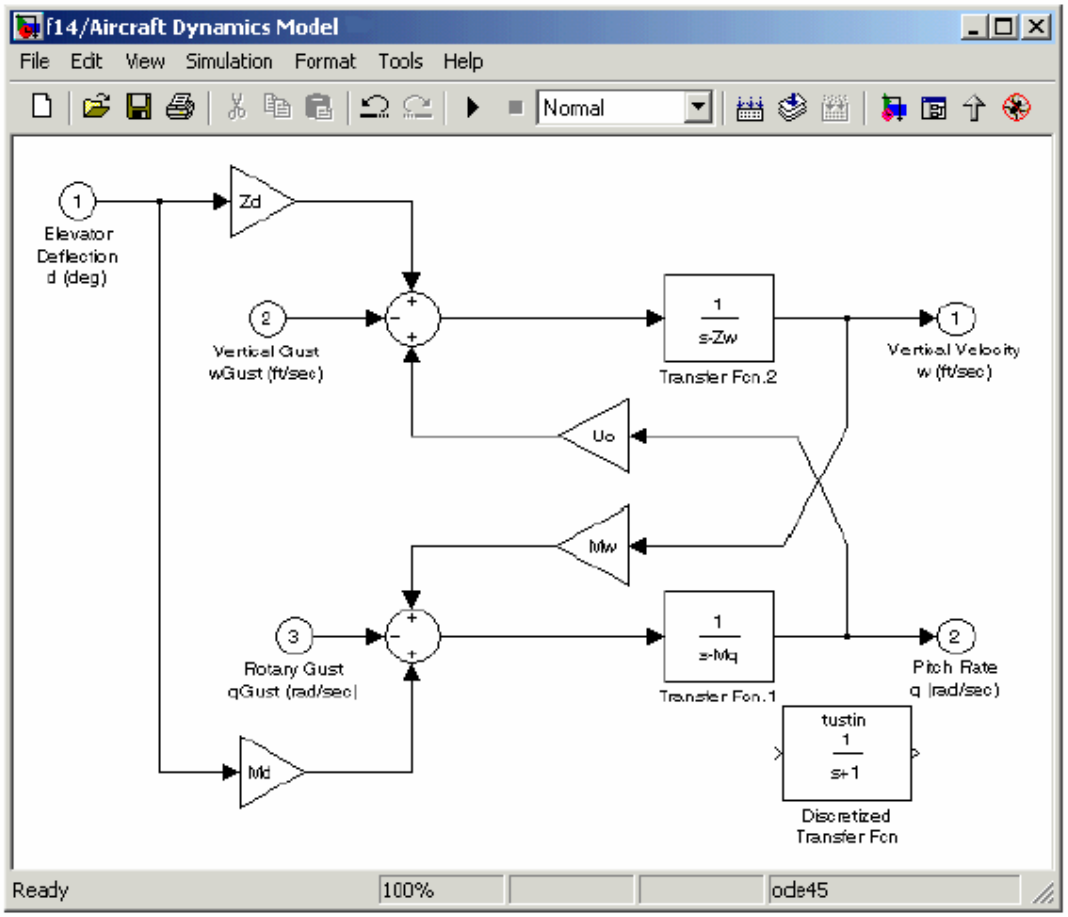


3. Otvoriti prozor biblioteka za diskretizaciju

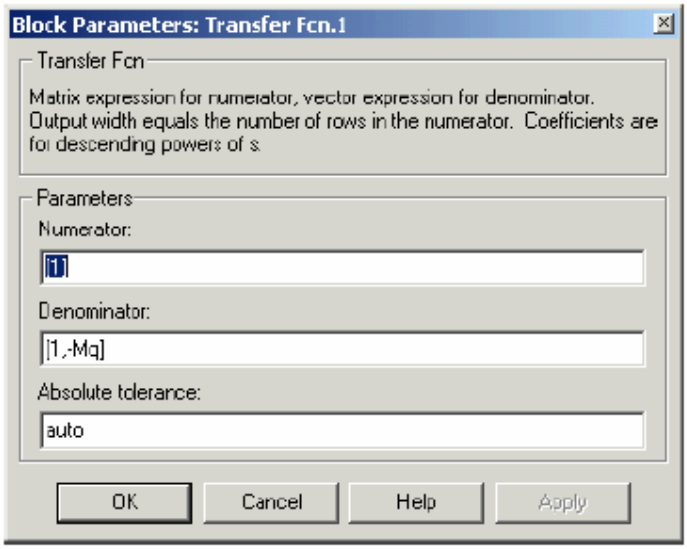
Unjeti discretizing komandu na komandnu liniju MATLAB-a. Otvoriće se biblioteka Library : discretizing . Ova biblioteka sadrži s-domen diskretizacione blokove.



4. Dodati Discretized Transfer Fcn blok u prozor f14/aircraft Dynamics model klikajući na njega u biblioteci i prevlačeći ga u prozor modela kao na slici :

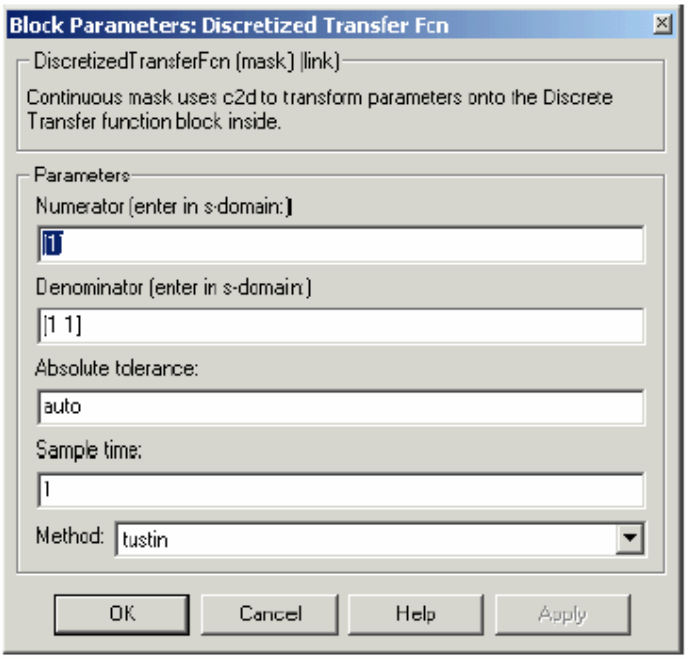


5. Otvoriti parametarski dijalog boks za Transfer Fcn.1 blok.
Dvaput kliknuti na taj blok i otvoriće se dijalog boks kao na slici :

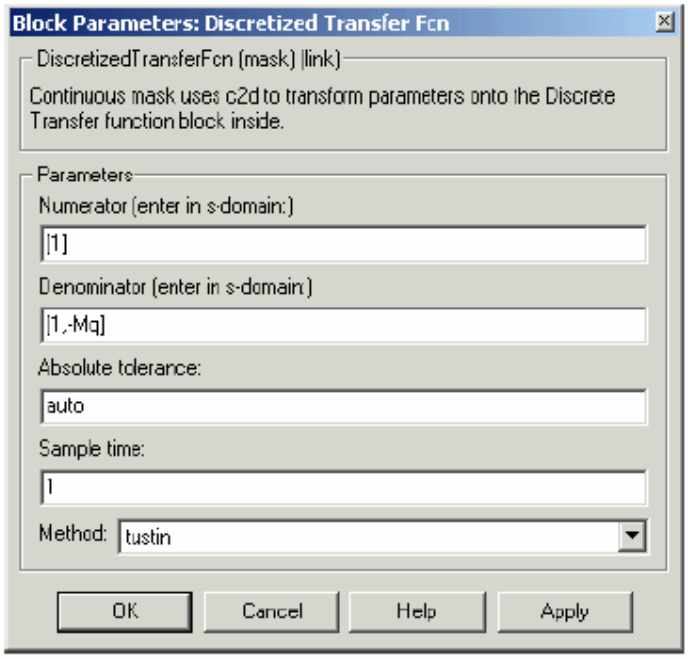


6. Otvoriti diajlog boks parametara za Discretized transfer Fcn blok.

Dvaput kliknuvši na blok otvoriće se dijalog boks parametara kao na slici:

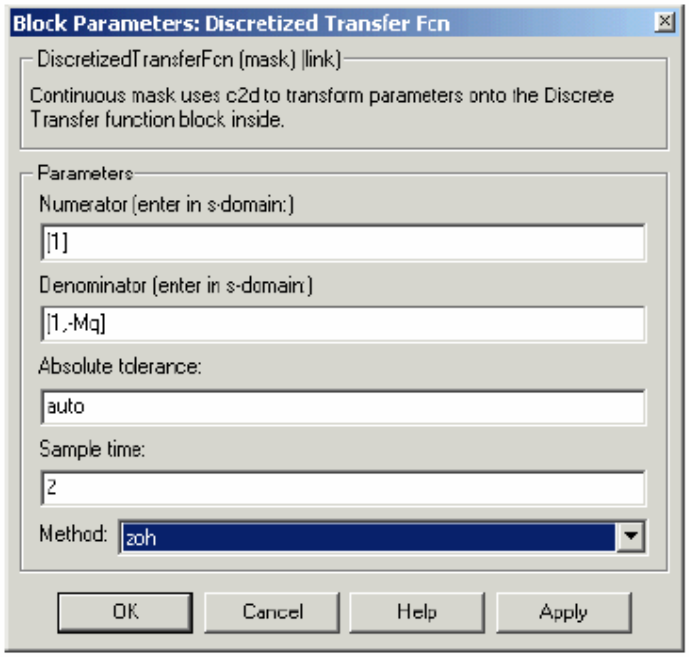


Kopirati informacije o parametrima iz Transfer Fcn.1 bloka u Discretized Transfer Fcn. blok dijalog boks.



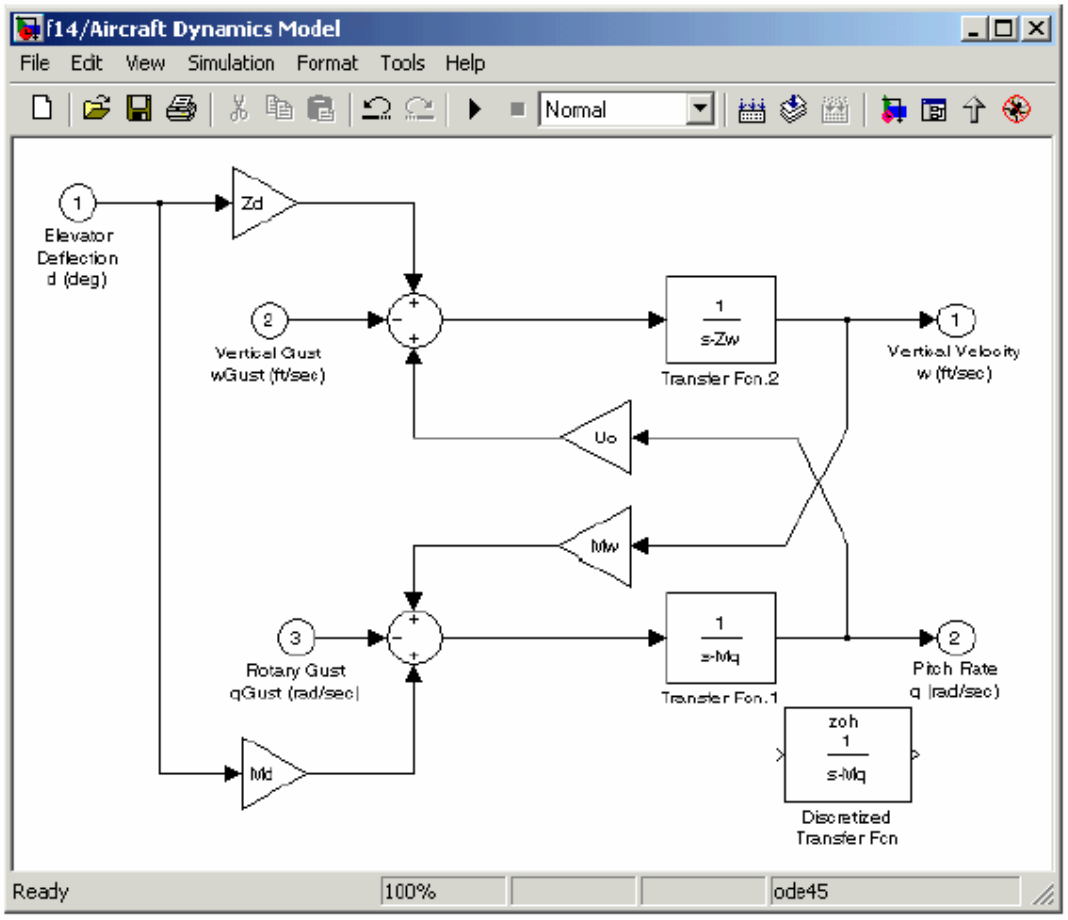
7. Unjeti 2 u **Sample time** polje.
8. Izabrati **zoh** u **Method** padajučoj listi.

Dijalog boks za parametre za Discretized Transfer Fcn. sada će izgledati kao na slici :

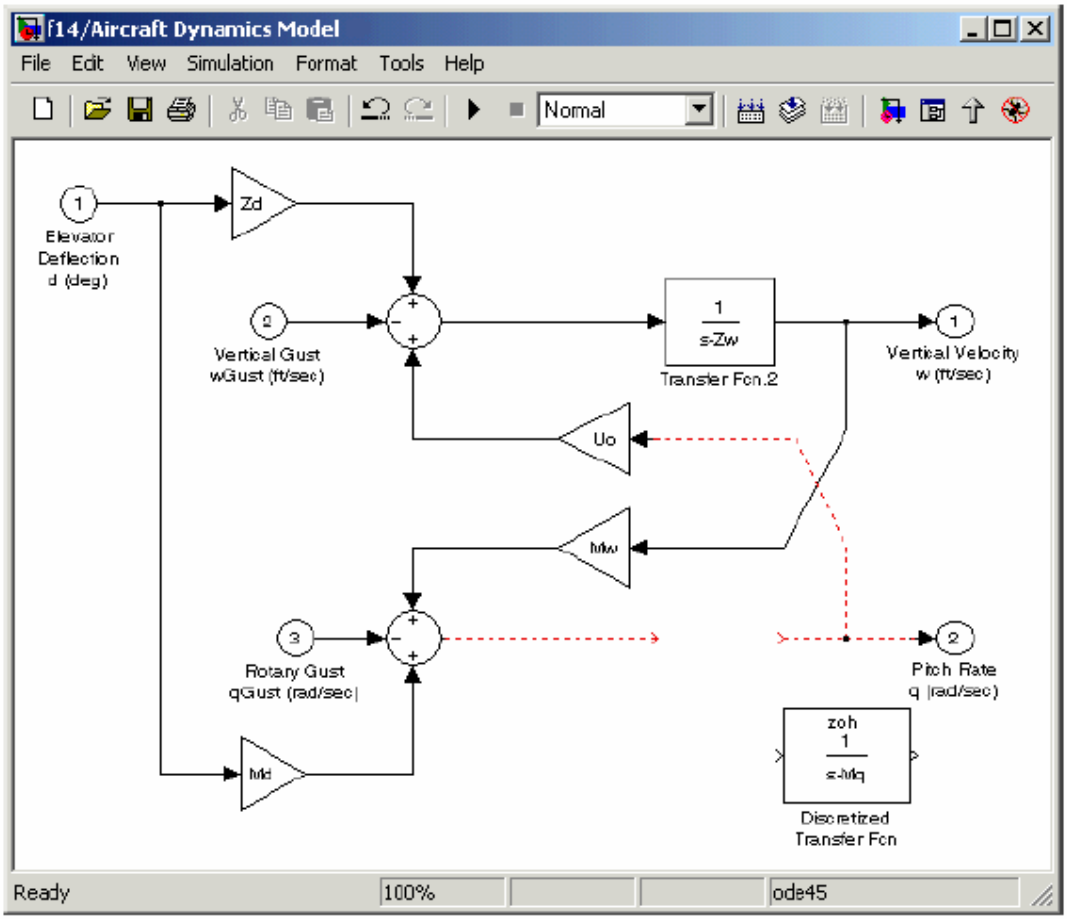


9. Kliknuti na OK
10.

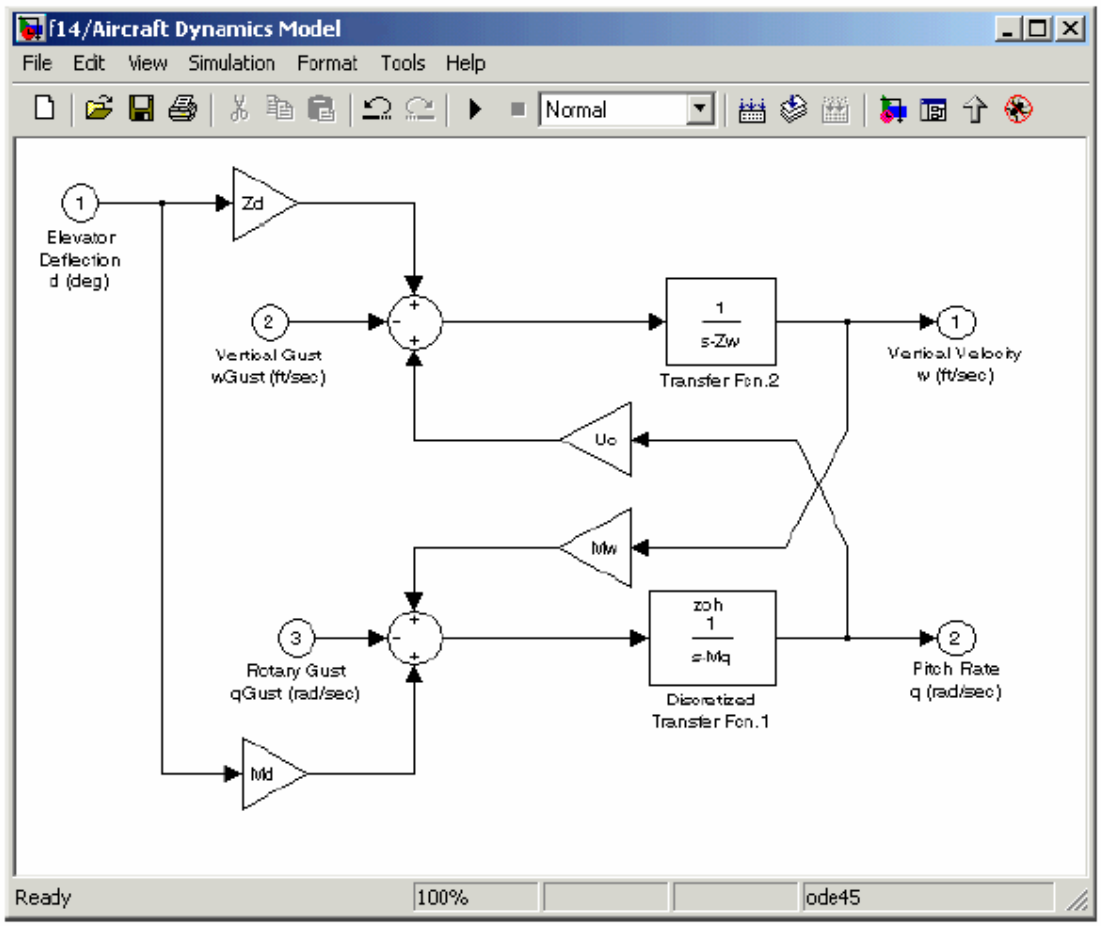
Prozor f14/Aircarft Dynamics Model će sada izgledati kao na narednoj slici:



11. Pobrisati originalni Transfer Fcn.1 blok. Sada će prozor modela izgledati kao:



12. Dodati i uključiti Transfer Fcn. blok u model , koji će sada izgledati kao na slici:



Diskretizacija modela iz MATLAB komandnog prozora

Koristiti **sldiscmdl** funkciju da diskretiziramo Simulink modele iz MATLAB komandnog prozora. Možemo specificirati metod transformacije, vrijeme sampliranja, i metod diskretizacije sa **sldimscmdl** funkcijom.

Naprimjer, slijedeća komanda diskretizira f14 model u s-domenu sa 1 sekundnim vremenom sampliranja koristeći zoh metod transformacije.

```
sldiscmdl('f14',1.0,'zoh')
```

Korištenje callback rutina

Možemo definirati MATLAB izraze koji se izvršavaju kada se na blok dijagram ili blok djeluje na neki poseban način. Ovi izrazi , koji se nazivaju **callback routines** , su pridružene sa blokom, portom ili parametrima modela. Na primjer, callback pridružen sa OpenFcn parametrom bloka se izvršava kada model koristi dvostruki klik na ime bloka ili na promjenu staze (path).

Praćenje callbackova (tracing callbacks)

Praćenje callbackova dozvoljava da odredimo callbackove koje Simulink kreira i u kojem redu ih Simulink poziva kada otvorimo ili simuliramo model. Da omogućimo praćenje callbacka, treba izabrati **Callback tracing** opciju na Simulink **Preferences** dijalog boksu, ili

execute set_param (0, 'CallbackTracing', 'on')

Ova opcija prouzrokuje da Simulink izlista callbackove u komandnom prozoru MATLAB-a , u redoslijedu kako su pozivani.

Kreiranje callback funkcija modela

Korisnik može kreirati callback funkcije modela interaktivno ili programski. Treba koristiti Callback površinu u dijalog boksu **Model Propreties** da se interaktivno kreiraju callbackovi modela. Da se kreiraju callbackovi programski, treba koristiti komandu **set_param** da se doznači MATLAB izraz koji implementira funkciju parametru modela koji korespondira sa callbackom.

Naprimjer, naredna komanda evaluira varijablu testvar kada korisnik dvaput klikne Test blok u **mymodel.**

```
set_param('mymodel/Test', 'OpenFcn', testvar)
```

Možemo analizirati clutch sistem (kočnica) u primjeru clutch.mdl , za rutine pridružene sa mnogim callbackovima modela.

Parametri callbacka modela

Slijedeća tabela daje listu parametara modela koji se koriste da specificiraju callback rutine modela i indicira kada odgovarajuće callback rutine su izvršene.

Parametar	Kada se izvršava
CloseFcn	Prije nego što je blok dijagram zatvoren
PostLoadFcn	Nakon što je model loadovan u memoriju. Definiranje callback rutine za ovaj parametar može biti korisno za generisanje interfejsa koji zahtjeva da je model već bio loadovan
InitFcn	Pozivana na startu simulacije modela
PostSaveFcn	Nakon što je model pohranjen
PreLoadFcn	Prije nego što je model loadovan. Definišući callback rutinu za ovaj parametar može biti korisno za loadovanje varijabli koje koristi model.
PreSave Fcn	Prije nego što je model pohranjen
StartFcn	Prije nego što simulacija starta
StopFcn	Nakon što se simulacija zaustavi. Izlaz se upisuje u varijable u radnom prostoru i fajlove prije nego se StopFcn izvrši.

Kreiranje blok callback funkcija

Korisnik može kreirati callback funkcije modela interaktivno ili programski. Koristiti Callback panel u diajlog boksu **Block Propreties** modela , da se kreira interaktivno callback modela. Da se kreiraju callbackovi programski, treba koristiti set_param komandu da doznači MATLAB izraz koji implementira funkciju na parametar bloka koji korespondira callbacku.

Opaska :
Callback za maskirani podsistem ne može direktno referencirati parametre maskiranog podsistema. Razlog je u tome što Simulink evaluira callbackove bloka u osnovnom radnom prostoru modela . Blok callback , pak, može koristiti get_param da dobije vrijednost mask parametra , tj. **get_param (gcb, 'gain')**, gdje gain je ime mask parametra u tekučem bloku.

Callback parametri bloka

Naredna tabela izlistava parametre za koje možemo definisati callback rutine bloka, i indicira kada su ove callback rutine izvršene. Rutine koje se izvršavaju prije ili nakon što se desi akcija , javljaju se trenutno prije ili nakon akcije.

Parametar	Kada se izvršava
CloseFcn	Kada je blok zatvoren koristeći **close_system** komandu
CopyFcn	Nakon što je blok kopiran. Callback je rekurzivan za blokove podsistema (tj. ako kopiramo podistemski blok koji sadrži blok za koji je CopyFcn parametar definiran, rutina se takodjer izvršava. Rutina se takodjer izvršava ako komanda **add_block** je korištena da kopira blok.
DeleteFcn	Prije nego se blok briše. Ovaj callback je rekurzivan za blokove podsistema
DestroyFcn	Kada je blok uništen
InitFcn	Prije nego što je blok dijagram kompiliran i prije nego su parametri bloka evaluirani.
LoadFcn	Nakon što je blok dijagram loadovan. Ovaj callback je rekurzivan za blokove podsistema.
ModelCloseFcn	Prije nego što je blok dijagram zatvoren. Ovaj callback je rekurzivan za blokove podsistema
MoveFcn	Kada je blok pomaknut ili mu je promjenjena velićina
NameChangeFcn	Nakon što se ime bloka i/ili staza promjeni. Kada se promjeni staza bloka podsistema, on rekurzivno zove ovu funkciju za sve blokove koje sadrži nakon pozivanja svoje vlastite **NameChangeFcn** rutine
OpenFcn	Kada se blok otvori . Ovaj se parametar općenito koristi sa podsistemskim blokovima. Ova rutina se izvršava kada dvaput kliknemo na blok ili kada komanda **open_system** se poziva sa blokom kao argumentom. OpenFcn parametar nadjahuje normalno ponašanje pridruženo otvaranju bloka, koje je da pokaže dijalog boks bloka ili da otvori podsistem.
ParentCloseFcn	Prije zatvaranja podsistema koji sadrži blok ili kada je blok

	napravljen kao dio novog podsistema , koristeći **new_system** command .
PreSaveFcn	Prije nego što se blok dijagram pohrani. Ovaj callback je rekurzivan za podsistemske blokove.
PostSaveFcn	Nakon što se pohrani blok dijagram. Ovaj callback je rekurzivan za podsistemske blokove.
StartFcn	Nakon što je blok dijagram kompiliran i prije nego simulacija starta. U slučaju bloka S-funkcije, StartFcn se izvršava odmah prije prvog izvršenja funkcije bloka **mdlProcessParameters**
StopFcn	Kod svakog završetka simulacije. U slučaju bloka S-funkcije, StopFcn izvršava se nakon što se izvrši **mdlTreminate** funkcija bloka.
UndoDeleteFcn	Kada je povraćena komanda brisanja bloka

Port callback parametri

Ulazni i izlazni portovi bloka imaju jedan callback parametar, **ConnectionCallback**. Ovaj parametar dozvoljava da se setuju callbackovi na portove koji su trigerovani svaki put kada se promjeni konektivnost ovih portova. Primjeri promjene konektivnosti uključuju brisanje blokova spojenih na port i brisanje , diskonekcija, ili konekcija grana ili linija na port.

Koristiti get_param da se dobije handle porta i set_param da se postavi callback na port. Naprimjer, predpostavimo da tekući izabrani blok ima jednostruki ulazni port. Slijedeći code fragment setuje "**foo**" kao konekcioni callback na ulazni port.

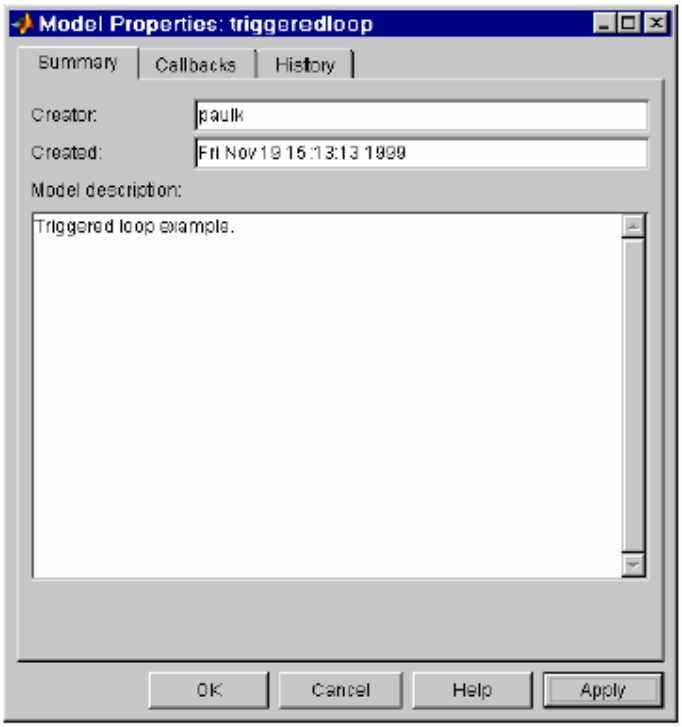
```
phs = get_param(gcb, 'PortHandles');
set_param(phs.Inport, 'ConnectionCallback', 'foo');
```

Prvi argument callback funkcije mora biti handle porta. Callback funkcija može imati i druge argumente (i vraćenu vrijednost) takodjer. Na primjer, slijedeće je validna signatura callback funkcije.

```
function foo(port, otherArg1, otherArg2)
```

Dijalog boks osobina modela

Model Propreties dijalog boks dozvoljava da se setuju razni parametri kontrole verzije i funkcije callback modela. Da se prikaže dijalog boks, izabrati **Model Propreties** iz iz Simulink **File** menija.



Dijalog boks uključuje slijedeće panele:

Summary pane
Callback pane
History pane

Općenito o blokovima

Blokovi su elementi od kojih su izgradjeni Simulink modeli. Možemo modelirati bilo koji dinamički sistem kreirajući i povezujući blokove na odgovarajuće načine. Naredna sekcija će pokazati kako da koristimo blokove da gradimo modele dinamičkih sistema.

Virtualni blokovi

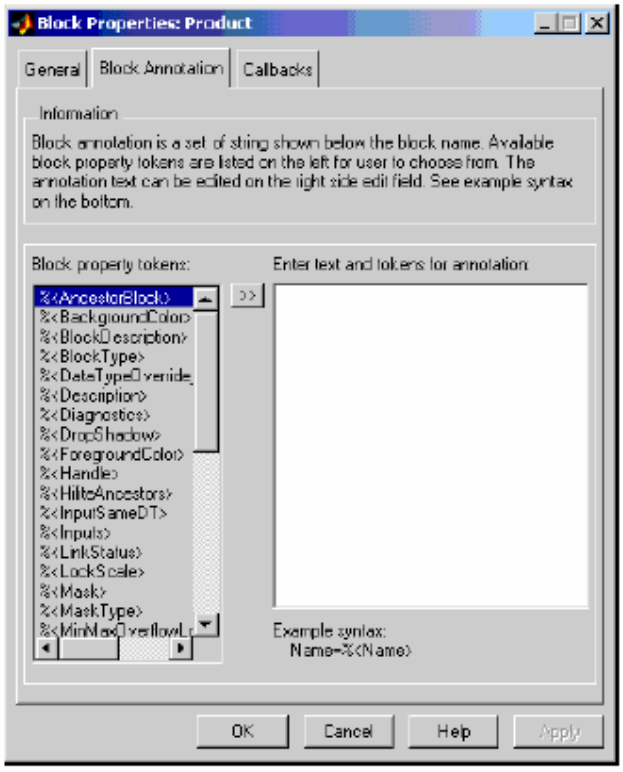
Kada kreiramo modele, moramo biti svjesni da Simulink blokovi ulaze u dvije bazne kategorije: nevirtualni i virtualni blokovi. Nevirtualni blokovi igraju aktivu ulogu u simulaciji sistema. Ako dodamo ili otklonimo nevirtualni blok, mi mjenjamo ponašanje modela. Virtualni blokovi , nasuprot ovome, ne igraju aktivnu ulogu u simulaciji, oni pomažu da se grafički organizuje model. Neki Simulinkovi blokovi su virtualni u odredjenim okolnostima a nevirtualni udrugim. Takvi blokovi se zovu uslovno virtualnim blokovima. Slijedeća tabela ilustrira Simulink virtualne i uslovno virtualne blokove.

Virtualni i uslovno virtualni blokovi

Ime bloka	Uslov pod kojim je blok virtualan
Bus Selector	Uvjek virtualan
Demux	Uvjek virtualan
Enable Port	Uvjek virtualan
From	Uvjek virtualan
Goto	Uvjek virtualan
Goto Tag Visibility	Uvjek virtualan
Ground	Uvjek virtualan
Inport	Virtualan izuzev ako je blok rezidentan u uslovno izvršivom podsistemu i ima direktnu konekciju na outport blok
Mux	Uvjek virtualan
Outport	Virtualan kada blok je rezidentan unutar bilo kojeg podsistemskog bloka (uslovnog ili ne) , i nije rezidentan u osnovnom (najvišeg nivoa) prozoru Simulinka.
Selector	Virtuelan izuzev u matričnom modu
Subsystem	Virtualan izuzev ako je blok uslovno izvršen i/ili blok opcija **Treat as Atomic unit** je izabrana
Terminator	Uvjek virtualan
Trigger port	Virtualan kada izlazni port nije prisutan

Panel anotacije bloka

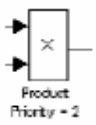
Panel anotacija (opis) bloka nam omogućava da prikažemo vrijednosti izabranih parametara bloka u anotaciji koja će se pojavliti ispod ikone bloka.



Unjeti tekst anotacije u polje teksta koje se pojavljuje na desnoj strani panela. Ovaj tekst uključuje tokene osobina bloka, naprimjer:

```
%<Name>
Priority = %<priority>
```

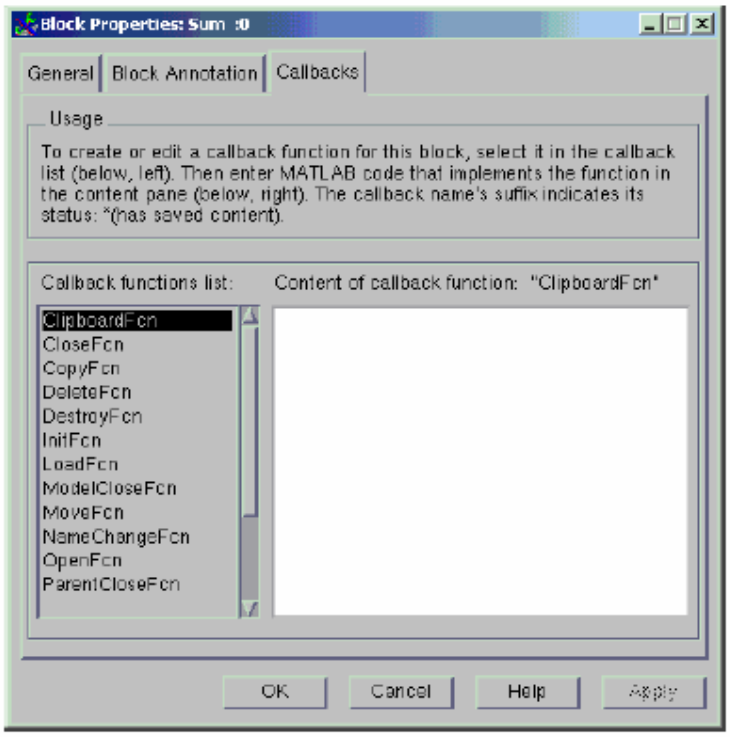
u formi %<param> gdje param je ime parametra bloka. Kada se pokaže anotacija , Simulink zamjenjuje tokene sa vrijednostima odgovarajućih parametara, kao:



Tag osobina bloka izlistava na lijevoj strani panela sve tagove koji su validni za selektirani blok. Da se uključi jedan od izlistanih tagova u anotaciju, izabrati tag i onda kliknuti na dugme izmedju tag liste i anotacionog polja.
Moguće je kreirati anotacije bloka i programski.

Callback panel

Callaback pane omogućava nam da specificiramo implementacije za callbackove bloka



:

Da specificiramo implementaciju za callback, izabrati callback u callback listi na lijevoj strani panela. Nakon toga unjeti MATLAB komande koje implementiraju callback polje sa desne strane. Klik **Ok** ili **Append** da se pohrani promjena. Simulink dodaje zvijezdu na ime pohranjenog callbacka da indicira da je bio implementiran.

Programsko kreiranje anotacija bloka

Možemo koristiti parametar bloka **AttributesFormatString** da pokažemo izabrane parametre bloka ispod bloka kao "atributi string formata" , tj. string koji specificira vrijednosti atributa bloka (parametara). Treba koristiti komandu Simulinka **set_param** da se postave ovi parametri na željeni string formata atributa.

Format string atributa može biti bilo koji tekst string koji ima embedded imena parametara. Uronjeno (embedded) ime parametra je ime parametra ispred koga je % i nakon koga je tekst parametra , naprimjer %<priority>. Možemo koristiti karakter nove linije (\n) da se pokaže svaki parametar na posebnoj liniji. Naprimjer, specifirajući atribure format stringa kao:

```
pri=%<priority>\ngain=%<Gain>
```

za blok pojačanja, imaćemo u prozoru modela:

Ako vrijednost parametra nije string ili cijeli broj, Simulink prikazuje N/S (not supported – nije podržano), za vrijednost parametra. Ako je ime parametra nevalidno, Simulink će pokazati ???? kao vrijednost parametra.

Kontrola i pokazivanje redoslijeda izvršenja blokova

Editor Simulinka nam omogućava da kontrolišemo i prikažemo redoslijed u kojem Simulink izvršava blokove.

Doznačavanje prioriteta blokova

Možemo doznačiti prioritete izvršenja nevirtualnim blokovima ili virtualnim blokovima podsistema u modelu. Blokovi višeg prioriteta se izvršavaju prije blokova nižeg prioriteta , mada ne uvjek i prije blokova koji nemaju doznačene prioritete.

Možemo doznačiti interaktivno blokove prioriteta ili programski. Da se setuju prioroteti programski, koristiti komandu:

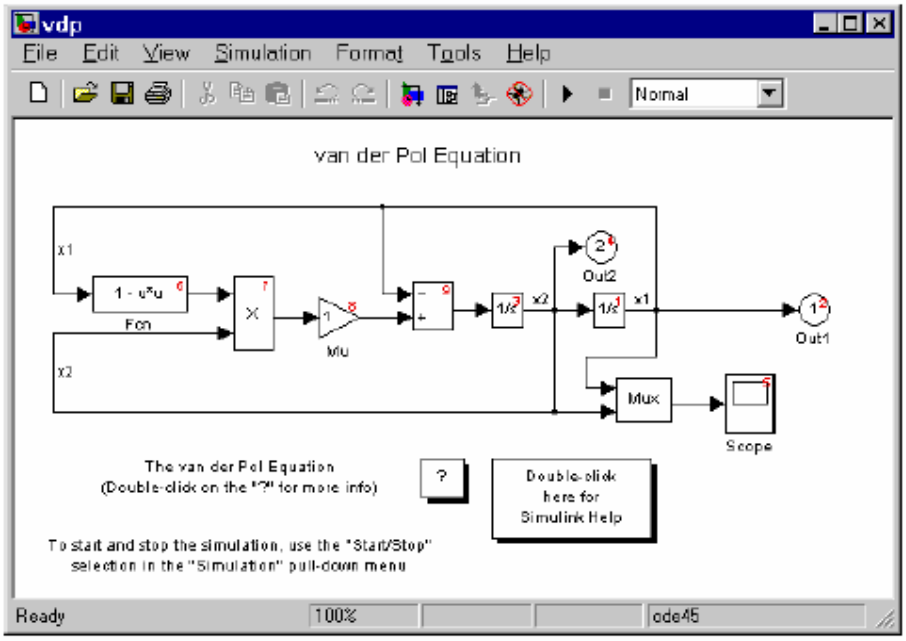
```
set_param(b,'Priority','n')
```

gdje je b staza bloka a n je cijeli broj. (negativni brojevi i 0 su takodjer validne vrijednosti prioriteta). Ukoliko je ovaj broj niži, utoliko je veći prioritet , tj. 2 je veći prioritet od 3. Da bi se postavio prioritet bloka interaktivno, , treba unjeti prioritet u polje **Priority** u dijagram boksu **Block Propreties .**

Simulink prihvata specifikaciju prioriteta koju je unjeo korisnik , samo onda kada je konzistentna sa Simulinkovim algoritmom sortiranja. Ako je specificiran prioritet nekonzistentan, Simulink ignoriše specificirani prioritet i smješta blok na odgovarajuću lokaciju u redoslijedu izvršenja blokova. Ako Simulink nije u stanju da prihvati priorotet koji specificira korisnik , on će pokazati dijagnostičku poruku "**Block Priority Violation**".

Prikazivanje redoslijeda izvršenja blokova

Da bi se pokazao redoslijed izvršenja blokova za vrijeme simulacije, izabrati **Execution order** komandu iz Simulink **Format** menija. Izabirući ovu opciju prouzrokuje da Simulink pokaže broj u gornjem desnom uglu svakog bloka u blok dijagramu.



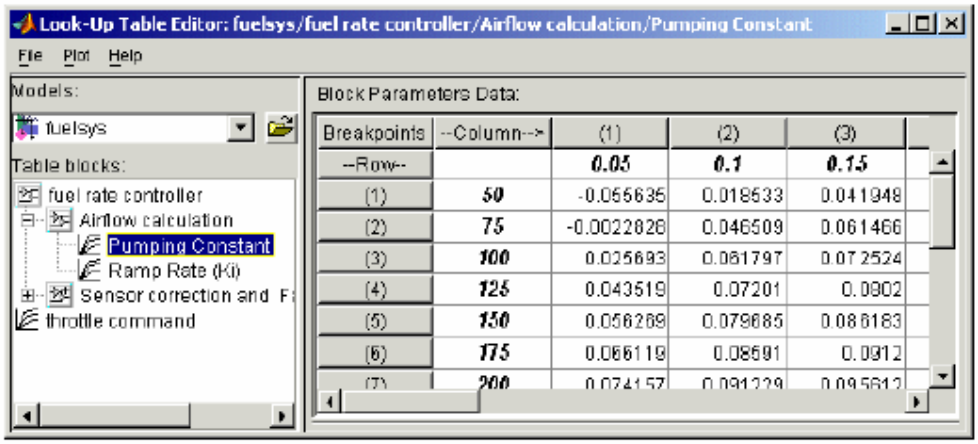
Broj označava redoslijed izvršenja bloka relativno u odnosu na druge blokove u dijagramu. Na primjer, 1 indicira da je blok prvi blok koji će se izvršiti u svakom vremenskom koraku, 2 indicira da je blok drugi , itd.

Editor look-up tabele

Editor look-up tabele nam omogućava da pregledamo i promjenimo elemente tabele bilo koje look-up table (LUT) bloka u modelu, uključujući kastomizirane LUT blokove koje je kreirao korisnik, koristeći Simulinkov Mask Editor. Možemo takodjer koristiti dijalog boks parametara bloka da editiramo njegovu tabelu.
Ipak, to zahtjeva da prvo otvorimo podsistem koji sadrži blok, a onda dijalog boks njegovih parametara.
Nasuprot, LUT editor nam omogućava da izbjegnemo ove korake.

Primjetimo da ne možemo koristiti LUT Editor da promjenimo dimenzije look-up tabele. Za ovu namjenu moramo koristiti dijalog boks za parametre bloka.

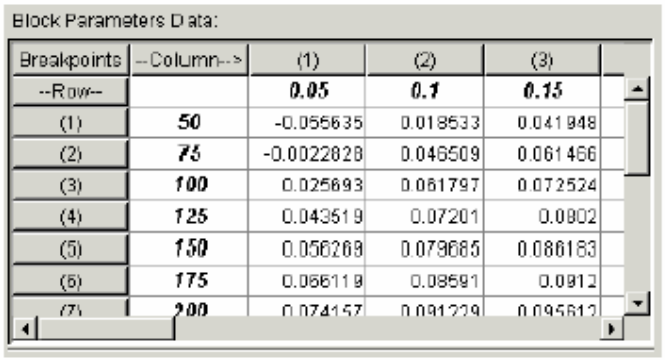
Da otvorimo editor , treba izabrati **Look-up table editor** iz Simulink **Tools** menija. Pojaviće se editor kao na slici:



Editor sadrži dva panela. Panel na lijevoj strani je browser LUT bloka. On omogućava browsovanje i izabiranje LUT blokova u otvorenom modelu. Panel na desnoj strani omogućava da editiramo izabranu look-up tabelu bloka.

Editiranje vrijednosti tabele

Tabela **Block parameters data** LUT Editora omogućava editiranje look-up tabele LUT bloka koji je izabran u drvetu selekcije na lijevoj strani.

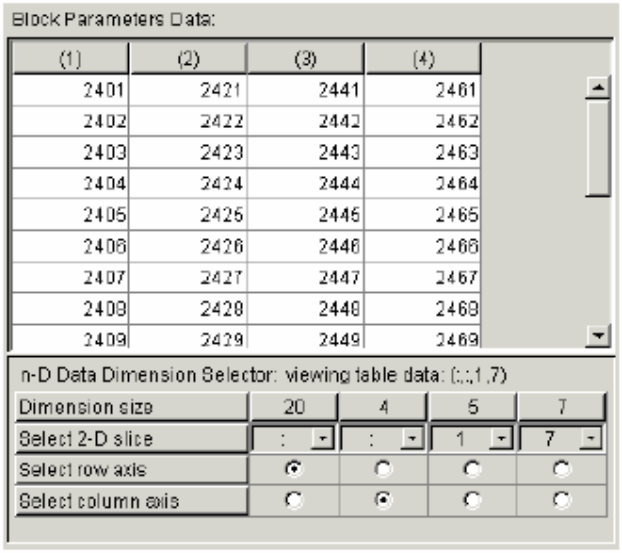


U prozoru će biti pokazana cijela tabela ukoliko je ona jedno ili dvodimenzionalna ili dio dvodimenzionalne ako tabela ima više od dvije dimenzije. Da bi se promjenila bilo koja vrijednost , treba dvaput kliknuti na vrijednost. LUT editor će zamjeniti vrijednost sa poljem editiranja koji sadrži tekuću vrijednost. Sada se može editirati ta vrijednost , zatim pritisnuti **Return** ili kliknuti van polja da se potvrdi promjena.

Da bi se konačno ažurirala tabela , treba izabrati **Update block data** iz **File** menija.

Prikazivanje N-D tabela

Ako look-up tabela od LUT bloka ima više od dvije dimenzije, vidjeće se dvo dimenzionalni segment te tabele kao na slici:

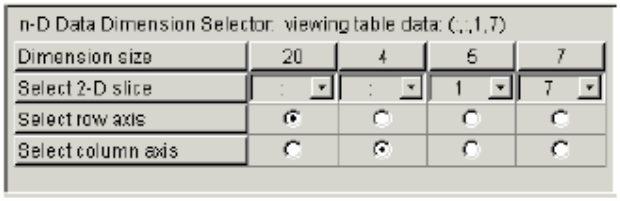


n-Data Dimension Selector koji se vidi ispod tabele specificira koji segment će se pojaviti na ekranu i omogućava selekciju nekog drugog segmenta. Selektor se sastoji od 4 x N polja kontrolnih elemenata, gdje N je broj dimenzija u look-up tabeli. Prva kolona korespondira prvoj dimenziji tabele, druga kolona drugoj dimenziji tabele itd.

Najgornji red selektorskog polja prikazuje veličinu svake dimenzije. Preostali redovi specificiraju koje dimenzije tabele korespondiraju osovinama redova i kolona segmenta i indekse koji izabiru segmenti iz preostalih dimenzija.

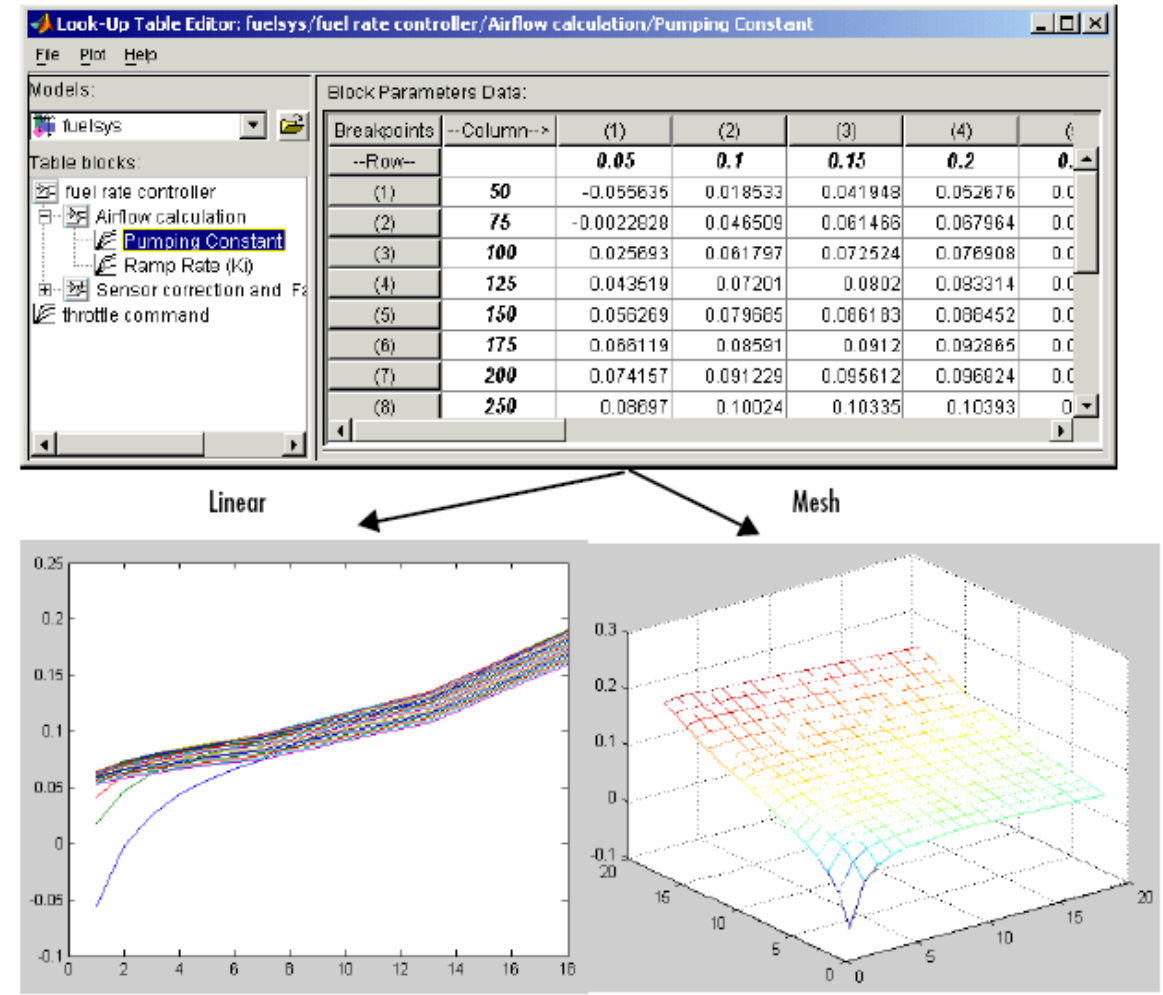
Da se izabere drugi segment tabele, kliknuti **Select row axis** i **Select column axis** radio dugmad, u kolonama koji korespondiraju dimenzijama koje želimo da gledamo. Nakon toga izabiremo indekse segmenta iz popup indeksne liste u preostalim kolonama.

Naprimjer, slijedeći selektor prikazuje segment (:,:, 1.7) iz 4-D tabele.



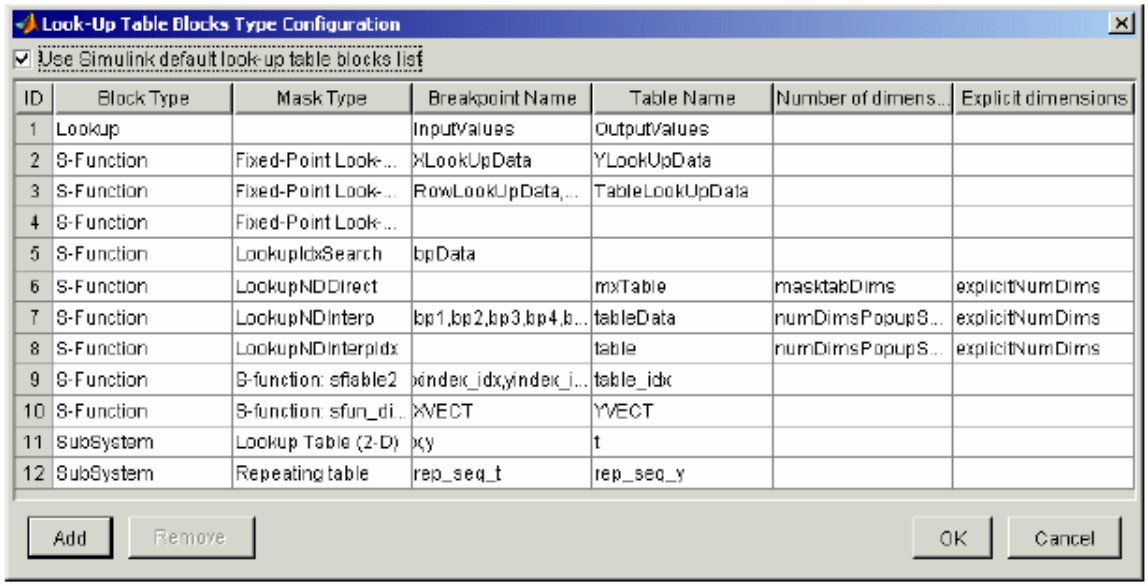
Crtanje LUT tabela

Izabrati **Linear** ili **Mesh** iz **Plot** menija LUT editora da se pokaže linearni ili mesh plot tabele ili segmenta tabele, koja je trenutno prikazana u tabeli na ekranu:



Editiranje kastomiziranih LUT blokova

Možemo koristiti LUT editor da editiramo korisnički krojene (custom) blokove look-up tabela, koje je korisnik kreirao. Da bi to mogli uraditi, moramo prvo konfigurisati LUT editor da prepozna kastomizirane LUT blokove, na taj način što ćemo izabrati komandu **Configure** iz **File** menija. Pojaviće se **Look-up Table Blocks Type Configuration** dijalog boks kao na slici:



Po defaultu , dijalog boks će prikazati tabelu tipova LUT blokova koje je LUT editor prepoznao. Svaki red tabele prikazuje ključne atribute tipa LUT bloka.

Rad sa bibliotekama blokova

Biblioteke omogućuju korisnicima da kopiraju blokove u njihove modele iz vanjskih biblioteka i automatski ažuriraju kopirane blokove kada se primjene izvorni blokovi. Korištenje biblioteka omogućava korisnicima da razviju svoje vlastite bibioteke blokova, ili da oni koji koriste biblioteke od drugih (tkzv. blocksets), da omogući da njihovi modeli automatski uključe najnovije verzije ovih blokova.

Terminologija

Library (biblioteka) – kolekcija bibliotečkih blokova. Biblioteka mora biti eksplicitno kreirana koristeći komandu **New Library** iz **File** menija.

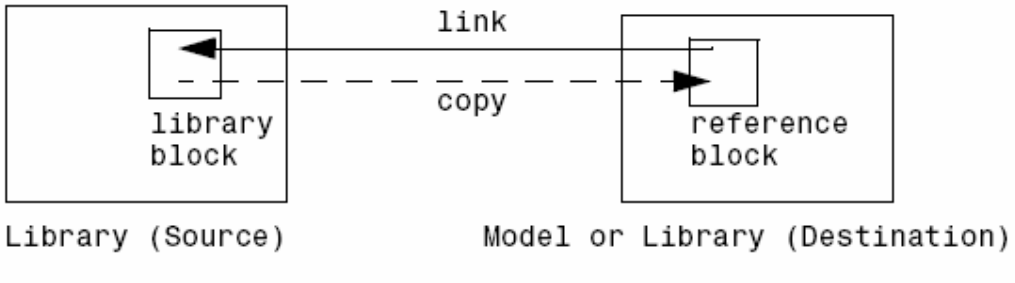
Library block – blok u biblioteci

Reference block – kopija bloka iz biblioteke.

Link – Konekcija izmedju referentnog bloka i njegovog bibliotečkog bloka koja omogućava Simulinku da ažurira referentni blok kada se promjeni bibliotečki blok.

Copy – operacija koja kreira referentni blok bilo iz bibliotečkog bloka ili drugog refrentnog bloka.

Naredna slika ilistrira ovu terminologiju



Onemogućavanje linkova sa bibliotekom

Simulink dozvoljava korisniku da onemogući linkovane blokove u modelu. Simulink će ignorisati onemogućene linkove kada se simulira model. Da se onemogući link, treba izabrati link, zatim izabrati **Link options** iz menija **Edit** , i zatim izabrati komandu **Disable link**. Da se restaurira onemogućeni link, treba izabrati **Restore link** iz **Link Options** menija.

Modificiranje linkovanog podsistema

Simulink omogućava korisniku da modificira podsisteme koji imaju linkove sa bibliotekom. Ako modifikacija mjenja strukturu podsistema, mora se onemogućiti link iz referentnog bloka do bibliotečkog bloka. Ako korisnik pokuša da modificira strukturu linka podsistema, Simulink ga promptira da onemogući link. Primjeri strukturnih modifikacija uključuju dodavanje ili brisanje bloka ili linije ili promjenu broja portova na bloku. Primjeri nestrukturnih promjena su promjene u vrijednosti parametara koje ne utiču na strukturu podsistema.

Propagacija modifikacija linkova

Simulink dozvoljava modelu da ima aktivne linkove sa nestrukturnim ali ne sa strukturnim promjenama. Ako restauriramo link koji ima strukturne promjene, Simulink će promptirati korisnika da ili propagira ili odbaci ove promjene. Ako on odluči da propagira promjene, Simulink ažurira bibliotečki blok sa promjenama koje su unesene u referentnom bloku. Ako pak odluči da odbaci promjene, Simulink zamjenjuje modificirani referentni blok sa originalnim bibliotečkim blokom. U svakom slučaju, krajnji rezultat je da referentni blok će opet biti egzaktna kopija bibliotečkog bloka.

Ako restauriramo link sa nestrakturalnim promjenama, Simulink omogućava link , bez promptiranja korisnika da propagira ili odbaci promjene. Ako želimo da propagiramo promjene ili odbacimo ih u nekom kasnijem vremenu, treba izabrati referentni blok , zatim izabrati **Link options** iz **Edit** menija , i zatim **Propagate/Discard** changes.
Ako želimo da vidimo razlike kod nestrukturalnih promjena izmedju referentnog i njegovog bibliotečkog bloka, izabraćemo **View changes** iz **Link options** menija.

Ažuriranje linkovanog bloka

Simulink ažurira zastarjele referentne blokove u modelu ili biblioteci u slijedećim vremenima:

- kada se loaduje model ili biblioteka
- kada izaberemo **Update Diagram** komandu iz **Edit** menija ili izvršimo simulaciju.
- Kada ispitujemo (query) **LinkStatus** parametara bloka, koristeći **get_param** komandu
- Kada koristimo **find_system** komandu

Prekid linka sa bibliotečkim blokom

Korisnik može prekinuti link izmedju referentnog bloka i njegovog bibliotečkog bloka da prouzrokuje da referentni blok postane samo jednostavna kopija bibliotečkog bloka, nelinkovana (nevezana) sa bibliotečkim blokom. Promjene u bibliotečkom bloku neće više uticati na referentni blok. Prekid linkova sa bibliotečkim blokovima nam može omogučiti da transportujemo model kao samostalan (stand-alone), bez bibloteka.

Da se prekine link izmedju referentnog bloka i njegovog bibliotečkog bloka, treba onemogučiti blok. Nakon toga izabrati **Break Library Link** iz menija **Link options.**

Može se pohraniti sistem i prekinuti svi linkovi izmedju referentnih blokova i bibliotečkih blokova sa komandom:

```
save_system('sys', 'newname', 'BreakLinks')
```

Opaska
Prekidanje bibliotečkih linkova u modelu ne garantira da korisnik može izvršavati model samostalno (stand-alone), naročito ako model uključuje blokove iz biblioteka drugih izvora , ili opcione Simulink blocksetove. Može biti moguće da bibliotečki blok poziva funkcije koje su uključene u biblioteku, i može se izvršavati samo ako je biblioteka instalirana na sistemu na kojem se izvršava model. Nadalje, prekidanje linka može prouzrokovati da model otkaže kada se instalira nova verzija biblioteka na sistemu.

Naprimjer, predpostavimo blok poziva funkciju koja je isporućena sa bibliotekom. Sada predpostavimo da je nova verzija biblioteke eliminirala funkciju. Izvršavanje modela sa nelinkovanom kopijom bloka rezultiraće u pozivanju sada nepostojeće funkcije, prouzrokujući simulaciju da otkaže. Da bi se izbjegli ovakvi problemi, treba generalno izbjegavati da se prekidaju linkovi sa bibliotekama drugih proizvodjaća i sa opcionimSimulink blocksetovima.

Nalaženje biblotečkog bloka za referentni blok

Da se nadje izvorna biblioteka i blok koji je linkovan na referentni blok, treba izabrati referentni blok , a zatim **Go To Library Link** iz podmenija **Link options** na meniju **Edit**.

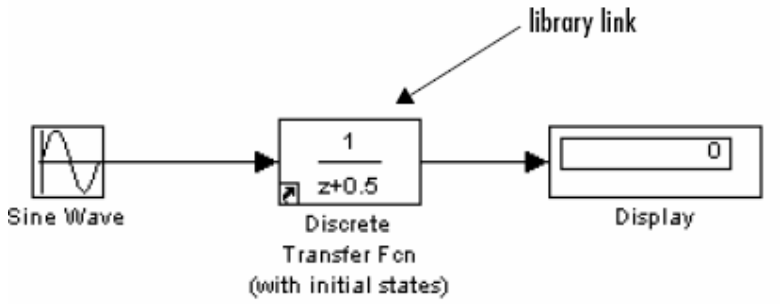
Status bibliotečkog linka

Svi blokovi imaju **LinkStatus** parametar koji indicira da li je blok referentni blok. Parametar može imati slijedeće vrijednosti:

Status	Opis
none	Blok nije referentni blok
resolved	Link je razriješen (nadjen)
unresolved	Link nije razriješen
implicit	Blok je unutar linkovanog bloka
Inactive	Link je onemogućen

Prikazivanje bibliotečkih linkova

Simulink opciono prikazuje strelicu na dnu lijevog ugla svake ikone koja predstavlja bibliotečki link u modelu:



Ova strelica dozvoljava da odmah uočimo da li ikona predstavlja link na bibliotečki blok ili je lokalna instanca bloka. Da se omoguće prikazi bibliotečkih linkova, izabrati **Library Link Display** u meniju **Format** i onda izabrati bilo **User** (prikazuje samo linkove sa korisničkim bibliotekama) ili **All** (prikazuje sve linkove).

Boja strelice linka indicira status linka

Boja	Status
crna	Aktivan link
siva	Neaktivan link
crvena	Aktivan i modifikovan

Rad sa signalima u Simulinku

Signali su tokovi vrijednosti koje se pojavljuju na izlazima Simulink blokova kada se model simulira. Primjetimo da linije u modelu Simulinka koje spajaju blokove predstavljaju logičke a ne fizičke konekcije medju blokovima.

Dimenzije signala

Blokovi Simulinka mogu izbacivati jedno ili dvodimenzionalne signale. Jednodimenzionalni (1-D) signal se sastoji od toka jedno dimenzionalnih polja (arrays) izlaza sa frekvencijom od jednog polja (vektora) po simulacionom koraku. Dvodimenzionalni (2-D) signal se sastoji od toka dvodiemenzionalnih polja emitiranih sa frekvencijom od 2-D polje (matrica) po bloku i vremenu samplovanja.
U Simulinku se općenito 1-D signali nazivaju vektorima a 2-D signali matricama. Polje sa samo jednim elementom se naziva skalarom. Vektor red je 2-D polje (array) koje ima samo jedan red. Vektor kolona je 2-D polje koje ima samo jednu kolonu.

Blokovi Simulinka variraju u dimenzionalnosti signala koje oni mogu prihvatiti i izbaciti za vrijeme simulacije. Neki blokovi mogu prihvatiti i izbaciti signale bilo koje dimenzije. Neki mogu pihvatiti ili izbaciti samo skalar ili vektorske signale. Da bi se odredila dimenzionalnost signala nekog specifičnog bloka, treba vidjeti njihov opis u "help" fileu.

Tipovi podataka signala.

Tip podataka se odnosi na format koji se koristi da interno prikaže vrijednost signala. Po default tip signala u Simulinku je **double** (dvostruka preciznost). Medjutim, mogu se kreirati i signali drugih tipova podataka. Simulink podržava isti opseg tipova podataka kao i MATLAB.

Kompleksni signali

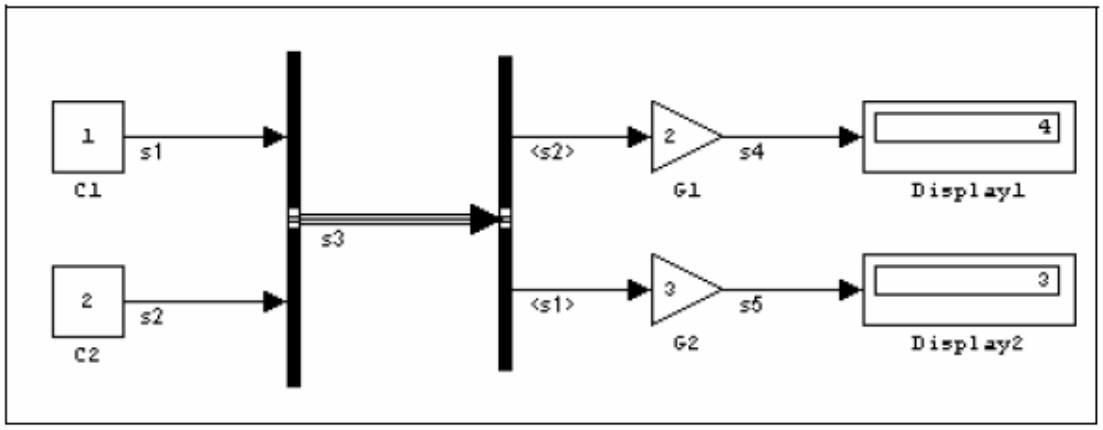
Vrijednosti Simulink signala mogu biti kompleksni brojevi. Signal čije su vrijednosti kompleksni brojevi se zove kompleksni signal.

Virtualni signali.

Virtualni signal je signal koji predstavlja drugi signal grafički. Virtualni blokovi, kao što je **Bus Creator** ili podsistemski blok , generiraju virtualne signale. Kao i virtualni blokovi, virtualni signali omogućavaju korisniku da grafički pojednostavi svoj model. Naprimjer, koristeći blok **Bus Creator**, možemo reducirati veliki broj nevirtualnih signala (tj. signala koji izlaze iz nevirtualnih blokova) , na jedan virtualni signal, i na taj način čineći model lakšim za razumjevanje. Možemo zamisliti virtualni signal kao omotnicu koja uvezuje veći broj signala zajedno.

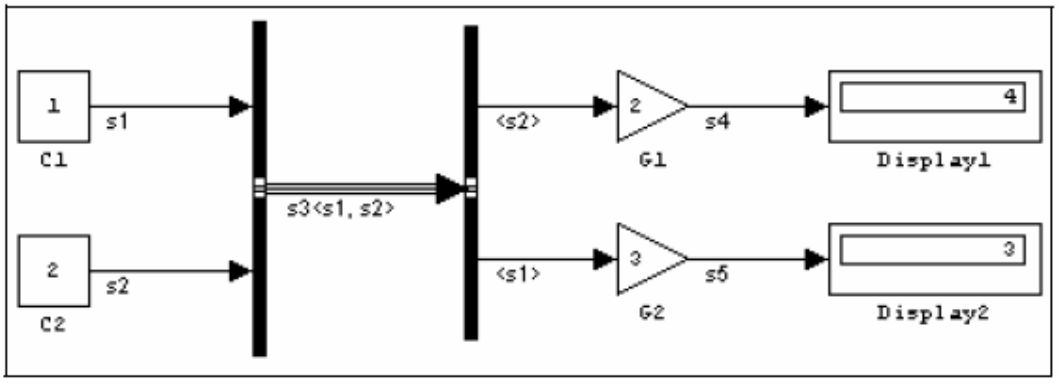
Virtualni signali su čisto grafički entiteti. Oni nemaju matematsko niti fizičko značenje. Simulink ih ignoriše kada simulira model.

Kada god izvršavamo ili ažuriramo model, Simulink odredjuje nevirtualni signal (ili signale) , koristeći proceduru poznatu kao propagacija signala (signal propagation). Kada se izvršava model, Simulink koristi odgovarajuče nevirtualne signale, odredjene preko propagacije signala, da vodi blokove na koje su spojeni virtualni signali. . Posmatrajmo naprimjer slijedeći model:



Signali koji vode blokove pojačanje G1 i G2 su virtualni signali koji odgovaraju signalima s2 i s1. Simulink odredjuje ovo automatski kada god ažuriramo ili simuliramo model.

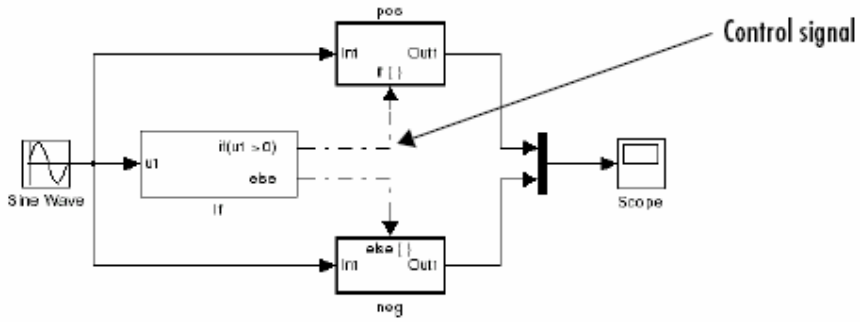
Opcija **Show Propagated Signals** pokazuje nevirtualne signale predstavljene virtualnim signalima u labelama virtualnih signala:



Opaska
Virtualni signali mogu predstavljati virtualne kao i nevirtualne signale. Naprimjer, možemo koristiti blok Bus Creator da kombinujemo višestruke virtualne i nevirtualne signale u jedan virtualni signal. Ako za vrijeme propagacije signala Simulink ustanovi da je komponenta virtualnog signala sama po sebi virtualna, Simulink odredjuje njene nevirtualne komponente koristeći propagaciju signala. Ovaj se proces nastavlja sve dok Simulink ne odredi sve nevirtualne komponente virtualnog signala.

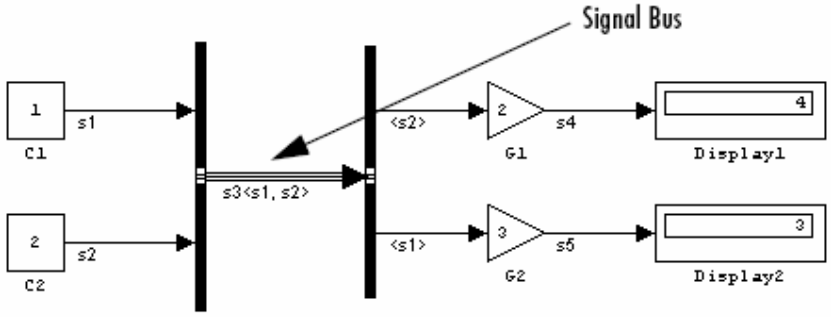
Kontrolni signali

Kontrolni signal je signal koji se koristi od jednog bloka da inicira izvršenje drugog bloka, napr. poziv funkcije (fuction call) ili akcionog podsistema (action subsystem). Kada mi ažuriramo ili startamo simulaciju blok dijagrama, Simulink koristi crta-tačka liniju da ponovo iscrta linije koje predstavljaju kontrolne signale dijagrama , kako je to prikazano na slijedećem primjeru na slici:



Signalni basevi

Možemo koristiti blokove Bus Creator i Bus Selector da kreiramo signalne baseve:



Signalni bas je virtualni signal koji predstavlja set signala. On je analogan skupu žica koje su omotane zajedno kao višežilni kabel. Simulink koristi specijalni stil linije da prikaže baseve signala. Ako izaberemo **Signal Dimensions** iz **Format** menija , Simulink će pokazati broj komponenti signala koji se prenose kroz bas.

Odredjivanje dimenzije izlaznih signala

Ako blok može izdavati neskalarne signale, dimenzije signala koje blok izbacuje zavise od parametara bloka, ako je riječ o bloku izvoru, u protivnom , izlazne dimenzije zavise od ulaza u blok i parametara.

Odredjivanje izlaznih dimenzija izvornog bloka

Izvorni blok je blok koji nema ulaza. Primjeri blokova izvora su : blok konstante, i sinusni blok. Izlazna dimenzija izvornog bloka je ista kao i od parametara njegovog izlaza ako je isključen ček boks **Interpret Vector Parameters as 1-D** (to jest, nije izabran u dijalog boksu za parametre). Ako **Interpret Vector Parameters as 1-D** je izabran, izlazna dimenzija je jednaka dimenziji parametara izlaza, ukoliko te dimenzije parametara nisu N x 1 ili 1x N. U ovom posljednjem slučaju , izlaz iz bloka je vektorski signal sa širinom N.

Kao primjer kako parametri izlaza izvornog bloka kao i **Interpret Vector Parameters as 1-D** parametar odredjuju dimenzionalnost njegovog izlaza, posmatrajmo blok konstante. Ovaj blok izbacuje konstantan signal jednak njegovom parametru **Constant value.**
Slijedeća tabela ilustrira kako dimenzionalnost parametra **Constant value** kao i setovanje **Interpret Vector Parameters as 1-D** parametra odredjuje dimenzionalnost izlaza iz bloka:

Konstantna vrijednost	Interpret Vector Parameters as 1-D	Izlaz
2-D skalar	off	2-D skalar
2-D skalar	on	1-D skalar
1 x N matrica	off	1 x N matrica
1 x N matrica	on	N elementni vektor
N x 1 matrica	off	N x 1 matrica
N x 1 matrica	on	N elementni vektor
M x N matrica	off	M x N matrica
M x N matrica	On	M x N matrica

Simulink izvorni blokovi dozvoljavaju korisniku da specificira dimenzije signala koje će on izbaciti na izlaz. Zbog toga mogu biti korišteni da uvedu signale različitih dimenzija u model.

Odredjivanje dimenzija izlaza blokova koji nisu izvori

Ako blok ima ulaze, dimenzije njegovih izlaza su, nakon skalarne ekspanzije, iste kao i ulaza. (Svi ulazi moraju imati iste diemenzije , kao što će to biti kasnije diskutirano).

Pravila dimenzija signala i parametara.

Kada se kreira Simulink model, moramo poštovati slijedeča pravila koja se odnose na dimenzije signala i parametara.

Pravilo dimenzije ulaznog signala.

Neskalarni ulazi u blok moraju imati iste dimenzije. Blok može imati mješane skalarne i neskalarne ulaze, sve dok svi neskalarni ulazi imaju iste dimenzije. Simulink ekspandira skalarne ulaze da imaju iste dimenzije kao i neskalarni ulazi, i na taj način obezbjedjuje da je pravilo poštovano.

Pravilo dimenzije parametara bloka

Općenito parametri bloka moraju imati iste dimenzije kao i odgovarajući ulazi. Postoje dva izuzetka iz ovog opšteg pravila:

- blok može imati skalarne parametre koji odgovaraju neskalarnim ulazima. U ovom slučaju, Simulink ekspandira skalarni parametar da ima iste dimenzije kao i odgovarajući ulaz, i na taj način obezbjedjuje da se sačuva opšte pravilo.

- Ako je ulaz vektor, odgovarajući parametar može biti bilo N x1 ili 1 x N matrica. U ovom slučaju Simulink primjenjuje N matričnih elemenata na odgovarajuće elemente ulaznog vektora. Ovo izuzeće dozvoljava da se koristi MATLAB vektor reda ili kolone, koji je ustvari 1 x N ili N x 1 matrica , da specificira parametre koji se primjenjuju na vektorske ulaze.

Pravilo konverzije vektorskog ili matričnog ulaza

Simulink konvertira vektore u red ili kolonu matricu kao i red i kolonu matricu u vektore pod slijedećim okolnostima:

- Ako je vektorski signal spojen na ulaz koji zahtjeva matricu, Simulink konvertuje vektor u jedan red ili kolona matricu

- Ako jednokolonska ili jedan red matrica je spojena na ulaz koji zahtjeva vektor, Simulink konvertuje matricu u vektor.

- Ako se ulazi u blok sastoje od mješavine vektora i matrica i matrični ulazi imaju jedan red ili kolonu, Simulink konvertuje vektore u matrice koje imaju jednu kolonu ili red.

Opaska

Možemo konfigurisati Simulink da prikaže poruku upozorenja ili greške ako konverzija vektora ili matrice se javlja za vrijeme simulacije.

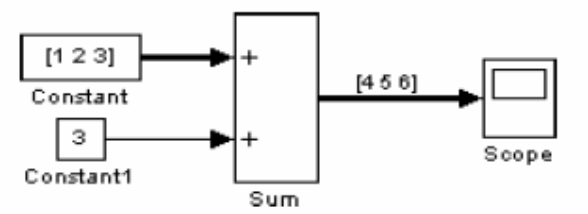
Skalarna ekspanzija ulaza i parametara

Skalarna ekspanzija (*scalar expansion*) je konverzija skalarne vrijednosti u neskalarno polje istih dimenzija. Mnogi Simulink blokovi podržavaju skalarnu ekspanziju ulaza i parametara.

Skalarna ekspanzija ulaza

Skalarna ekspanzija ulaza se odnosi na ekspanziju skalarnih ulaza da se upare sa dimenzijama drugih neskalarnih ulaza ili neskalarnih parametara. Kada je ulaz u blok mješavina skalarnih i neskalarnih signala, Simulink ekspandira skalarne ulaze u neskalarne signale koji imaju iste dimenzije kao i drugi neskalarni ulazi. Elementi ekspandiranog signala su jednaki vrijednosti skalara iz kojeg je signal ekspandiran.

Slijedeći model ilustrira skalarnu ekspanziju ulaza. Ovaj model dodaje skalarne i vektorske ulaze. Ulaz iz bloka Constant1 je skalar ekspandiran da se upari (match) dimenziju vektorskog ulaza iz Constant bloka. Ulaz je ekspandiran u vektor [3 3 3].

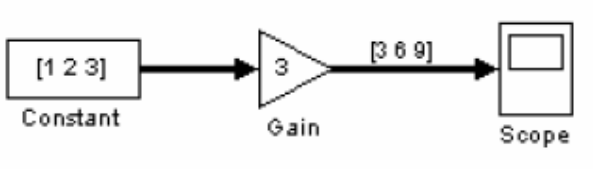


Kada je izlaz bloka funkcija parametra i parametar je neskalar, Simulink ekspandira skalarni ulaz da se upari sa dimenzijama parametra. Na primjer, Simulink ekspandira skalarni ulaz za blok pojačanja (Gain) da se upari sa dimenzijama neskalarnog parametra gain bloka.

Skalarna ekspanzija parametara

Ako blok ima neskalarni ulaz i odgovarajući parametar je skalar, Simulink ekspandira skalarni parametar da ima isti broj elemenata kao ulaz. Svaki element ekspandiranog parametra jednak je vrijednosti originalnog skalara. Simulink nakon toga primjenjuje svaki element ekspandiranog parametra na odgovarajući ulazni element.

Ovaj primjer pokazuje da skalarni parametar (Gain –pojačanje) je ekspandiran na vektor sa identičnim vrijednostima elemenata da se upari sa velićinom ulaza bloka, tj. sa troelementnim vektorom.

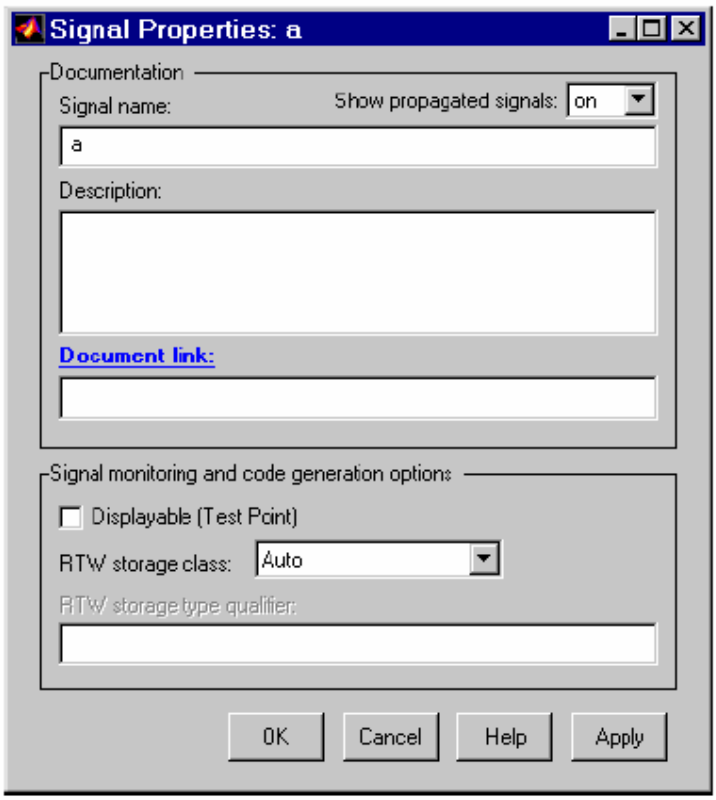


Setovanje osobina signala

Signali imaju osobine. Koristiti dijalog boks **Signal Propreties** da se pogledaju ili setuju osobine signala. Da se prikaže dijalog boks, izabrati liniju koja nosi signal i izabrati **Signal Propreties** iz Simulink menija **Edit**.

Dijalog boks osobina signala

Dijalog boks **Signal Propreties** omogućava da pregledamo i editiramo osobine signala.



Dijalog boks uključuje izmedju ostalih i slijedeća kontrolna polja:

Show propagated signals

Ova opcija će se pojaviti samo za signale koji imaju porijeklo iz bilo kojeg virtualnog bloka osim Bus Selector block.

Možemo izabrati jednu od slijedećih opcija

Opcija	Opis
off	Ne prikazati signale predstavljene sa virtualnim signalom u labeli signala
on	Prikazati virutalne i nevirtualne signale predstavljene sa virtualnim signalom u labeli signala. Na primjer, predpostavimo da virtualni signal s1 predstavlja nevirtualni signal s2 i virtualni signal s3. Ako je izabrana ova opcija , labela s1 će biti s1< s2,s3>
all	Prikazuje sve nevirtualne signale koje predstavljaju virtualni signal bilo direktno ili indirektno. Na primjer, prepostavimo da virtualni signal s1 predstavlja nevirtualni signal s2 i virtualni signal s3 predstavlja nivirtualne signale s4 i s5. Ako ova opcija je izabrana labela s1 je s1<s2,s4,s5>

Document link

U ovo polje treba unjeti MATLAB izraz koji pokazuje dokumentaciju za signal. Da bi se pokazala dokumentacija, kliknuti na "Document Link". Na primjer, unoseći izraz :

```
web(['file:///' which('foo_signal.html')])
```

u ovo polje prouzrokovaće da MATLAB default Web server prikaže foo_signal.html dokument kada kliknemo na labelu polja.

Displayable (test Point)

Treba izabrati ovu opciju da se indicira da signal može biti pokazan za vrijeme simulacije.

Slijedeća dva kontrolna polja postavljaju osobine koje će koristiti Real-Time Workshop da generira kod iz modela.

RTW storage class

Treba izabrati klasu pohranjivanja za signal iz liste. Detalji o ovome izboru se mogu naći u manualu za Real-Time Workshop.
Ako želimo da pokažemo **Storage class** signala na blok dijagramu, treba izabrati ovu opciju iz menija **Format**.

RTW storage type qualifier

Izabrati tip pohranjivanja za ovaj signal iz liste. Detalji o ovome izboru se mogu naći u manualu za Real-Time Workshop.

Rad sa kompleksnim signalima

Po default, vrijednosti signala Simulinka su realni brojevi. Medjutim, modeli mogu kreirati i manipulirati signalima koji imaju kompleksne brojeve kao vrijednosti.

Možemo unjeti kompleksnu vrijednost signala u MATLAB radni prostor na slijedeće načine:

- Loadovati kompleksnu vrijednost podatka iz MATLAB radnog prostora u model putem inporta iz korijenskog bloka

- Kreirati konstantni blok u modelu i postaviti njegovu vrijednost na kompleksni broj.

- Kreirati realne signale koji odgovaraju realnom i imaginarnom dijelu kompleksnog signala, i zatim kombinirati dijelove u kompleksni signal, koristeći konverzioni blok Real-Imag to Complex.

Možemo manipulisati kompleksnim signalima preko blokova koji ih prihvataju. Ako nismo sigurni da li blok prihvata kompleksne signale, možemo provjeriti u online dokumentaciji za njih.

Provjeravanje spajanja signala

Mnogi Simulink blokovi imaju ograničenja na tipove signala koje prihvataju. Prije simulacije modela, Simulink čekira sve blokove da obezbjedi da oni mogu prihvatiti tipove izlaznih signala na portovima na koje su spojeni.

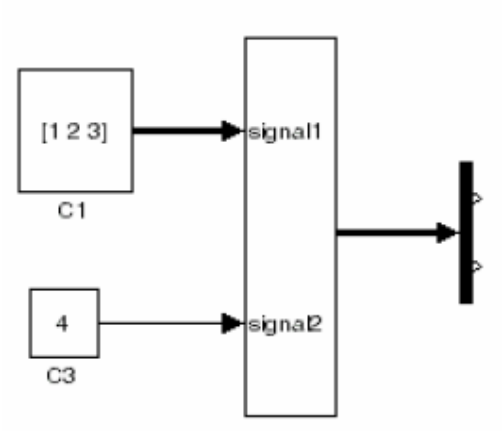
Ako postoji bilo kakva inkompatibilnost, Simulink izvještava o grešci i završava simulaciju. Da bi se otkrile takve greške prije neko što se pokrene simulacija, izabrati **Update Diagram** iz Simulink **Edit** menija. Simulink izvještava bilo koju nevalidnu konekciju koju nadje u procesu ažuriranja dijagrama.

Prikazivanje signala

Format meni nudi slijedeće opcije za prikazivanje atributa signala.

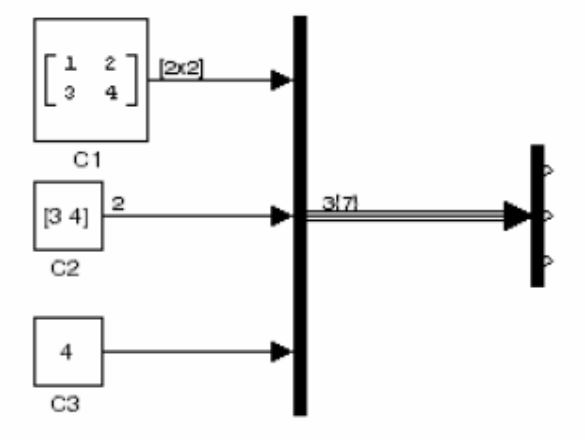
Široke linije neskalara

Crta linije koje nose vektorske ili matrične signale šire nego što su linije koje nose skalarne signale.



Dimenzije signala

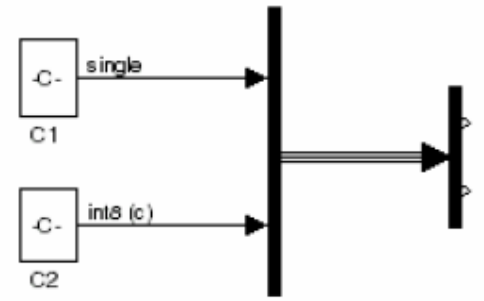
Prikazuje dimenzije neskalarnih signala pored linije koja nosi signal:



Format prikaza zavisi od toga da li linija predstavlja jedan signal ili bas. Ako linija predstavlja jednostruki vektorski signal, Simulink pokazuje širinu signala. Ako linija predstavlja matrični signal, Simulink će pokazati njenu dimenziju kao [N1 x N2] gdje N_i je velićina i-te dimenzije signala. Ako linija predstavlja bas koji prenosi signale istog tipa podataka, Simulink prikazuje N $\{M\}$ gdje N je broj signala koji bas prenosi a M je totalni broj signalnih elemenata koje bas prenosi. Ako bas nosi signale različitih tipova podataka, Simulink prikazuje samo totalni broj signalnih elemenata $\{M\}$.

Tipovi podataka porta

Prikazuje tip podatka signala pored izlaznog porta koji emituje signal



Oznaka **c** koja dolazi poslije tipa podatka indicira da je signal kompleksan.

Imena signala

Možemo doznačiti imena signalima sa:

- Editiranjem labele signala
- Editiranjem polja Name u dijalog boksu osobina signala (**Signal Propreties**)
- Setovanjem imena parametra porta ili linije koja predstavlja signal, naprimjer.

```
p= get_param ( gcb, 'PortHandles')
l = get_param ( p.Inport, 'Line')
 set:param ( 1,'Name', 's9' )
```

Rad sa grupama signala

Blok graditelja signala (signal builder block) omogućava nam da kreiramo medjusobno izmjenjljive grupe izvora signala i brzo preklapamo grupe u i iz modela. Signalne grupe mogu značajno olakšati testiranje modela, naročito kada se koristi u sprezi sa Simulink blokovima i opciono alatom za pokrivanje modela (model coverage tool).

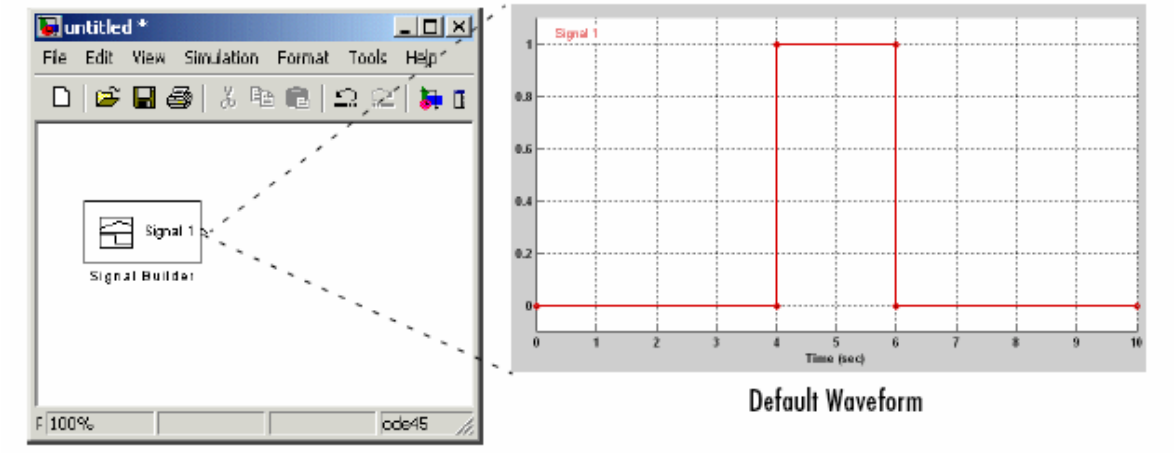
Rad sa signalnim grupama

Blok graditelja signala (Signal Builder block), omogućava korisniku da kreira izmjenjljive grupe izvora signala i brzo preklučuje grupe u i iz modela. Signalne grupe mogu značajno olakšati testiranje modela, naročito kada se koriste u sprezi sa Simulink blokovima potvrde (assertion blocks), i opcionim alatom za pokrivanje modela (Model Coverage tool).

Kreiranje seta signalne grupe

Da se kreira izmjenjljivi set signalnih grupa , uraditi:

1. Vuči instancu Signal Builder bloka iz Simulink biblioteke izvora i spustiti ga u model.



Po default blok predstavlja jednu signalnu grupu koja sadrži jedan signalni izvor koji izbacije četvrtku.

2. Koristiti editor bloka da se kreiraju dodatne signalne grupe, dodati signale u signalne grupe, modificirati postojeće signale i signalne grupe, i izabrati signalnu grupu koju blok izbacuje.

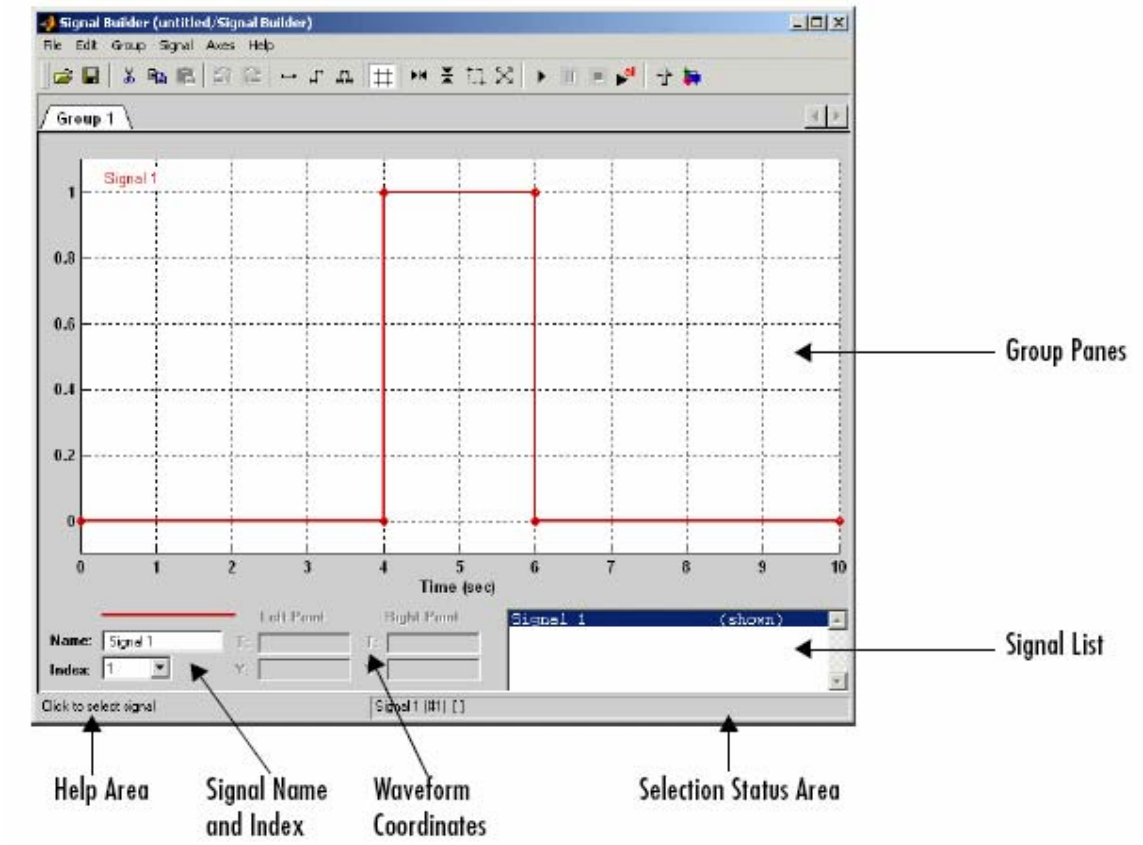
3. Povezati izlaz bloka na dijagram.

Možemo kreirati koliko god želimo Signal Builder blokova u modelu, od kojih svaki predstavlja poseban set izmjenjljivih grupa izvora signala.

Dijalog boks buildera signala

Dijalog boks Buildera signala dozvoljava korisniku da definira valne oblike signala koje blok izbacuje. Možemo specificirati bilo koji valni oblik koji je linearan po segmentima.

Da se otvori dijalog boks , dvaput kliknuti na ikonu bloka. Pojaviće se dijalog boks Signal **Buidera.**



Signal Builder dijalog boks omogućava korisniku da kreira i modificira signalne grupe predstavljene sa Signal Builder blokom. **Signal Builder** dijalog boks uključuje slijedeće kontrolne elemente:

Group panes

Prikazuje se set izmjenjljivih grupa izvora signala predstavljenih sa blokom.

Signal axes

Signali se pojavljuju na posebnim osovinama koje dijele zajednički opseg vremena. Ovo omogućava korisniku da lagano poredi relativni tajming promjena u svakom signalu.

Signal Builder automatski skalira opseg svake ose da prilagodi signalu kojeg pokazuje. Koristiti Axes meni Signal Buildera da se promjeni vrijeme (T) i amplituda (Y) opsega selektiranih osa.

Signalna lista

Prikazuje imena i vidljivost signala koji pripadaju tekuće izabranoj grupi signala. Klikanjem na ulaz u liste selektiramo signal. Dvostrukim klikanjem na ulaz signala u listi krijemo ili prikazujemo valne oblike signala na panelu grupe.

Selekcija zone statusa

Prikazuje ime tekučeg izabranog signala i indeks tekučeg izabranog segmenta ili tačke valnog oblika.

index

Indeks tekućeg izabranog signala. Indeks indicira izlazni port kod kojeg se signal pojavljuje. Indeks 1 indicira najgornji izlazni port, 2 indicira drugi port sa vrha, itd.

Editiranje signalnih grupa

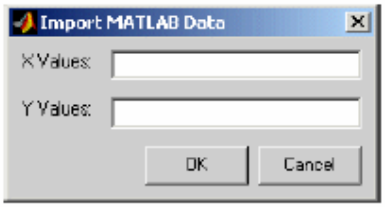
Dijalog boks Signal Buildera nam omogućava da kreiramo, promjenimo ime, pomjeramo i brišemo signalne grupe iz seta grupa predstavljenih sa blokom Signal Buildera.

Editiranje signala

Dijalog boks Signal Buildera nam omogućava da kreiramo, izrezujemo i lijepimo , krijemo i brišemo signale iz signalnih grupa.

Kreiranje signala

Da bi se kreirao signal u tekuće selektiranoj signalnoj grupi, izabrati **New** iz **Signal** menija. Pojavljuje se meni valnih oblika. Meni uključuje set standardnih valnih oblika (**Constant, Step** , itd) i **Custom** valne oblike. Izabrati jedan od valnih oblika. Ako izaberemo standardni valni oblik, Signal Builder dodaje signal sa tim valnim oblikom na tekuće selektiranu grupu. Ako izaberemo **Custom**, pojaviće se dijalog boks kastom valnog oblika.



Dijalog boks nam dozvoljava da specificiramo kastom valni oblik segmentno linearan, koji se dodaje grupama definiranim sa blokom Signal Buildera. Unjeti kastom vremensku koordinatu valnog oblika u **T Values** polje a odgovarajući signal amplituda u

Y Values polje. Ulazi i obadva polja može biti bilo koji MATLAB izraz koji se evaluira u vektor. Rezultirajući vektori moraju biti iste dužine. Izabrati **OK**.

Promjena indeksa signala

Da bi se promjenio indeks signala , izabrati signal i zatim izabrati komandu **Change Index** iz **Signal** menija.

Editiranje valnih oblika

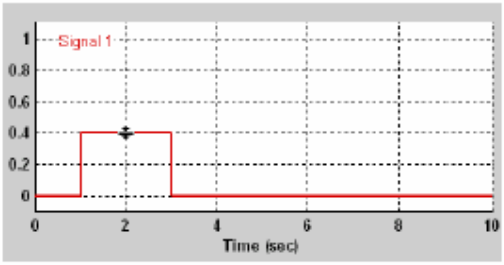
Dijalog boks **Signal Builder** dozvoljava da se promjeni oblik, boja, i stil linije i debljina izlaza valnih oblika signala signalne grupe.

Promjena valnog oblika

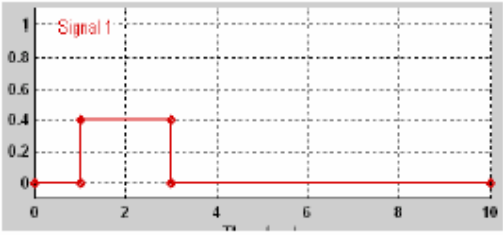
Dijalog boks Signal Buildera nam omogućava da promjenimo oblik valnog oblika izabirući i vukući njegove segmente i tačke ili editirajući koordinate segmenata i tačaka.

Selekcija valnog oblika

Da bi se izabrao valni oblik, kliknuti lijevim tasterom na bilo koju tačku ili valni oblik:

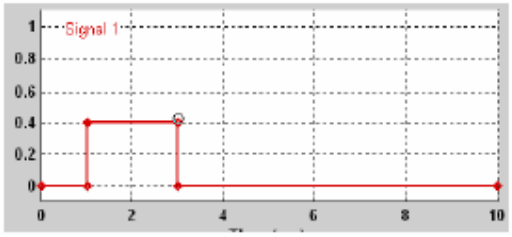


Signal Builder prikazuje tačke valnog oblika da indicira da je valni oblik izabran:

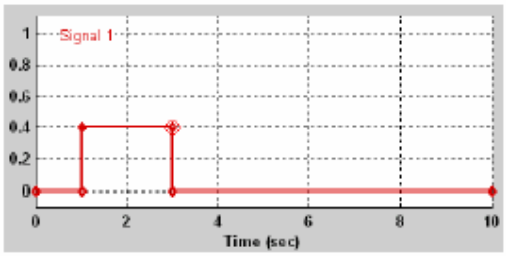


Selektiranje tačaka

Da se selektira tačka valnog oblika , prvo treba izabrati valni oblik. Nakon toga pozicionirati kurzor miša iznad tačke. Kurzor promjeni oblik da indicira da je iznad tačke:

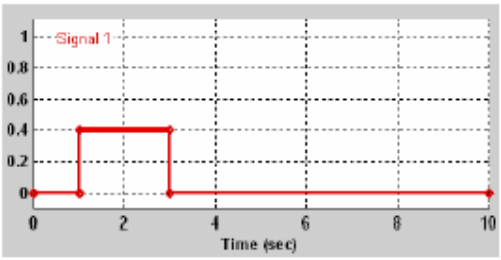


Kliknuti lijevim tasterom miša. Signal Builder će nacrtati krug oko tačke da indicira da je izabran.



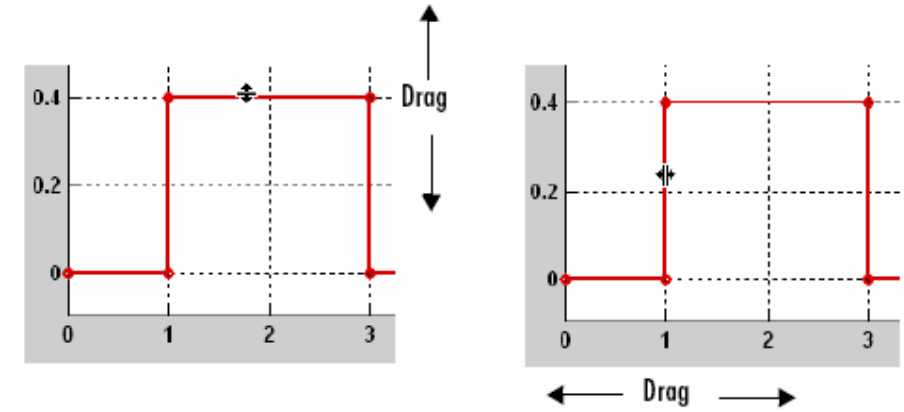
Selekcija segmenata

Da bi se selektirao segment linije, prvo treba selektirati valni oblik. Signall Builder podebljava segment da indicira da je izabran.



Vučenje segmenta

Da bi se prevukla linija segmenta na novu lokaciju, pozicionirati kurzor miša iznad linije segmenta. Kurzor miša mjenja oblik da pokaže smjer u kojem možemo vući segment.

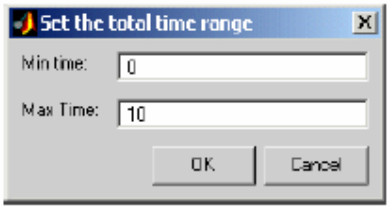


Vučenje tačaka

Da bi se vukla tačka duž amplitude signala (vertikalne ose), treba vući kurzor miša nad tom tačkom. Kurzor mjenja oblik u kružič da indicira da se može vući tačka. Vuči tačku paralelno x osi do željene lokacije. Da bi se vukla tačka duž vremenske (horizontalne) ose, treba pritisnuti **Shift** taster dok se vuče tačka.

Vremenski opseg signal Buildera

Vremenski opseg Signal Buildera odredjuje opseg vremena na kojem je izlaz eksplicitno definisan. Po defaultu, opseg vremena je izmedju 0 i 10 sek. Možemo promjeniti i početno i krajnje vrijeme opsega. Da bi to uradili treba izabrati komandu **Change time range** sa **Axes** menija. Pojaviće se slijedeći dijalog boks:



Editirati **Min. time** i **Max. time** polja da se unesu početno i završno vrijeme novog vremenskog opsega.

Izvoz podataka iz signalne grupe

Da bi se izvezli podatci iz signalne grupe u MATLAB radni prostor , izabrati **Export to workspace** iz **File** menija.

Pojaviće se slijedeći dijalog boks:

Signal Builder izvozi podatke po defaultu u varijable radnog prostora koje se zovu **channels**. Ako se želi promjeniti ime unjeti novo ime u polje **Variable name**. Builder izvozi podatke u radni prostor kao vrijednost specificirane varijable. Izvezeni podatci su u obliku polja struktura.

Izvršenje različitih signalnih grupa u nizu

Alatna traka (toolbar) Signal Buildera uključuje standardnu dugmad Simulinka za izvršenje simulacije. Ovo omogućava da se izvršava nekoliko različitih signalnih grupa u nizu. Na primjer, možemo otvoriti dijalog boks, izabrati grupu, izvršiti simulaciju , izabrati drugu grupu, izvršiti simulaciju itd. iz dijalog boksa Signal Buildera.

Izvršenje svih signalnih grupa

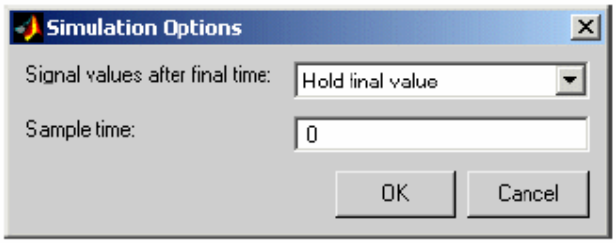
Da bi se izvršavale sve signalne grupe koje su definirane u bloku Signal Buildera, treba otvoroti dijalog boks i izabrati **Run all**  taster u traci alata .

Komanda **Run all** će izvršiti seriju simulacija, jednu za svaku signalnu grupu definisanu blokom. Ako smo instalirali opcioni alat Model Coverage Tool, komanda **Run all** će konfigurisati alat da prikuplja i pohranjuje podatke za svaku simulaciju u radnom prostoru MATLAB-a i prikaže izvještaj kombinovanih okvirnih (coverage) rezultata na kraju simulacije. Ovo nam omogućava da brzo odredimo kako dobro set signalnih grupa testira naš model.

Da bi zaustavili seriju simulacija koje su pokrenute sa **Run all** komandom, treba unjeti **Control-c** na komandnoj liniji MATLAB-a.

Dijalog boks simulacionih opcija

Dijalog boks **Simulation Options** omogućava korisniku da specificira simulacione opcije koje se odnose na Signal Builder. Da bi se prikazao diajlog boks treba izabrati komandu **Simulation Options** iz **File** menija. Pojaviće se slijedeći dijalog boks:

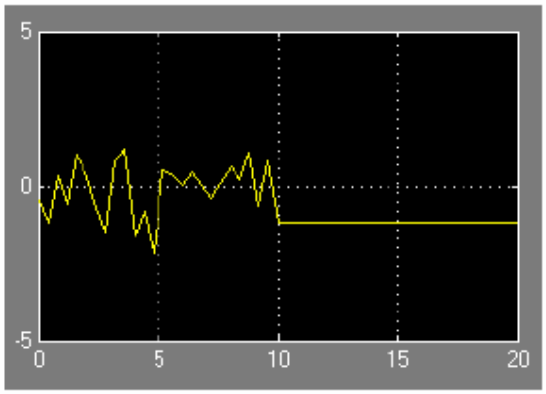


Signal values after final time (signalne vrijednosti nakon krajnjeg vremena)

Setting odredjuje izlaz iz bloka Signal Buildera ako simulacija traje duže nego period definiran blokom. Opcije su:

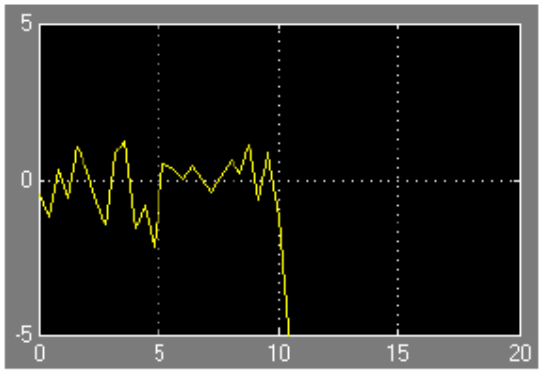
- **Hold final value**

  Izbor ove opcije će prouzročiti da izlaz zadržava zadnju definiranu vrijednost svakog signala u grupi do kraja simulacije , kao na slici:



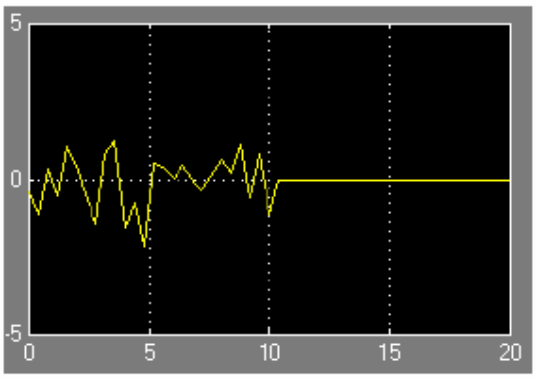
- **Extrapolate**

  Izbor ove opcije uzrokuje da blok Signal Buildera izbacuje vrijednosti ekstrapolirane iz posljednje definirane vrijednosti svakog signala u tekučoj aktivnoj grupi za preostali period simulacije.



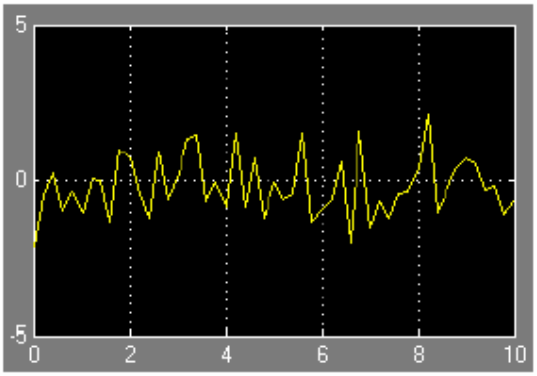
- **Set to zero**

  Izabirući ovu opciju će prouzrokovati da blok SB izbacuje nulu u preostalom intervalu simulacije , kao na slici:

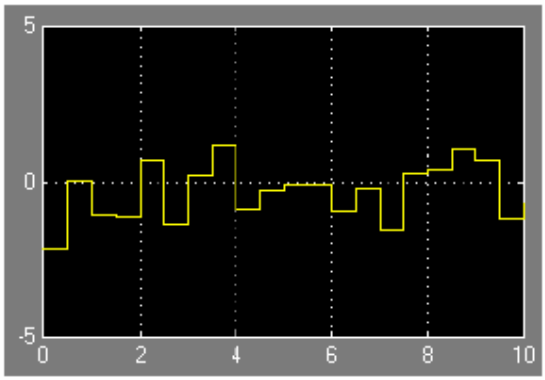


Sample time

Odredjuje da li SB blok izbacuje izlaze kontinualno (default) ili kao diskretni signal. Ako želimo da izbacuje blok kao kontinualni signal treba unjeti **0** u ovo polje. Na slijedećoj slici je SB blok setovan da izbacuje kontinualni valni oblik u periodu od 10 sec. :



Ako želimo da blok izbacuje diskretni signal, unjeti u polje vrijeme sampliranja signala. Na slijedećoj slici je primjer izlaza SB bloka za diskretni Gaussian valni oblik sa 0.5 sec vremenom sampliranja.



Rad sa podatcima

Termin tip podatka (data type) se odnosi na način kako računar predstavlja brojeve u memoriji. Tip podatka odredjuje količinu memorije alocirane broju, na metod koji se koristi da kodira vrijednost broja kao patern binarnih digita (0,1), i operacije koje su na raspolaganju za manipuliranje sa datim tipom podatka.

Simulink, na bazi istih mogućnosti MATLAB-a na kojeg se oslanja, dozvoljava korisniku da specificira tipove podataka Simulink signala i parametara blokova.

Ova mogućnost je naročito korisna kod real-time kontrolnih aplikacija. Ona omogućava da Simulinkov model specificira optimalne tipove podataka koje će koristiti da predstavi signale i parametre blokova u formi generisanoj od koda (code generated form), iz modela koristeći alate za atomatsku generaciju koda (code-generation tools), kao što je Real-Time Workshop.

Simulink izvršava intenzivno provjeravanje prije i za vrijeme simulacije da osigura da je model **typesafe**, tj. da kod generisan iz modela neće preliti ili podliti (overflow or underflow), i time proizvesti pogrešne rezultate.

Simulinkovi modeli koji koriste default tip podatka (double) su inherentno typesafe.

Tipovi podataka koje podržava Simulink

Simulink podržava sve ugradjene (built-in) tipove podataka MATLAB-a izuzev i**nt64** i **uint64**.

Slijedeća tabela pokazuje built-in tipove podataka MATLAB-a koje podržava Simulink

Ime	Opis
double	Pokretni zarez dvostruke preciznosti
single	Pokretni zarez jednostruke preciznosti
Int8	8-bitni cjelobrojni sa predznakom
Uint8	8 bitni cjelobrojni bez predznaka
Int16	16 bitni cjelobrojni sa predznakom
Uint16	16 bitni cjelobrojni bez predznaka
Int32	32 bitni cjelobrojni sa znakom
Uint32	32 bitni cjelobrojni bez znaka

Osim ovih, Simulink definira i boolean (1 ili 0) tip, koji je interno opisan sa **uint8** vrijednostima. Simulink takodjer podržava i tipove podataka sa fiksnim zarezom.

Specificiranje tipova podataka za parametre blokova

Kada se unose parametri blokova čiji su tipovi specificirani od strane korisnika , treba koristiti slijedeću sintaksu:

type (value)

da se specificira parametar, gdje je **type** ime tipa podatka a **value** je vrijednost parametra.

Slijedeći primjeri ilustriraju ovu sintaksu:

Single (1.0)	Specificira vrijednost 1.0 jednostruke preciznosti
Int8(2)	Specificira vrijednost 2 kao 8 bitni cijeli broj
Int32(3 + 2i)	Specificira kompleksni broj čiji realni i imaginarni dio su 32 bitni cijeli brojevi.

Prikazivanje tipova podataka u portovima

Da bi se prikazali tipovi podataka u portovima modela, izabrati **Port Data Types** iz **Format** menija.

Prenošenje tipa signala (typecasting)

Simulink pokazuje grešku svaki put kada otkrije da je signal spojen na blok koji ne prihvata tip podatka signala. Ako želimo da kreiramo takvu konekciju, moramo eksplicitno prenjeti (typecast – convert) signal u tip koji blok prihvata. Za ovu svrhu možemo koristiti **Data type conversion block**.

Rad sa objektima podataka

Simulink objekti podataka (data objects) omogućavaju korisniku da specificira informaciju o podatcima koji se koriste u Simulink modelu, i da pohrani informaciju sa samim podatcima u model.

Klase objekata podataka (data object classes)

Objekat podatka je instanca drugog objekta koji se naziva klasa objekta podatka **(data object class) .** Klasa objekta podatka definira osobine objekata podataka koji su instance i metode za kreiranje i manipuliranje sa instancama. Simulink ima dvije ugradjene (built-in) klase podataka: *Simulink.Parameter i Simulink.Signal*. , koje definiraju parametre i signale objekata podataka.

Paketi objekata podataka (data objects packages)

Simulink organizuje kase u grupe klasa koje se zovu paketi (packages). Simulink je opremljen jednim paketom koji se naziva *Simulink*. Simulink klase *Simulink.Parameter i Simulink.Signal*. pripadaju *Simulink* paketu.

Kvalifikovana imena (qualified names)

Kada referiramo na klasu u MATLAB komandnoj liniji ili u M-file programu, moramo korstiti slijedeću notaciju:

```
PackageName.ClassName
```

Ovo se naziva kvalifikovanim imenom klase.
Dva paketa mogu imati ista imena ali različite klase. Na primjer, paketi A i B mogu imati klasu koja se zove C. Imena klasa i paketa su osjetljiva na mala i velika slova. Ne možemo naprimjer koristiti A.B i a.b izmjenjljivo da se referiše na iste klase.

Kreiranje objekata podataka

Možemo koristiti komande Simulink Data Explorera da kreiramo objekte podataka. Treba koristiti slijedeću sintaksu:

```
h = package.class(arg1, arg2, ...argn);
```

gdje je **h** MATLAB varijabla, **package** je ime paketa kojem klasa pripada, **class** je ime klase, i arg1, arg2,....argn su opcioni argumenti koji se prenose do konstruktora objekta.

Naprimjer, da kreiramo instancu klase Simulink.Parameter, treba unjeti:

```
hGain = Simulink.Parameter;
```

na komandnoj liniji MATLAB-a .

Ova komanda kreira instancu Simulink.Parametar i pohranjuje handle u hGain.

Kreiranje klase objekata podataka

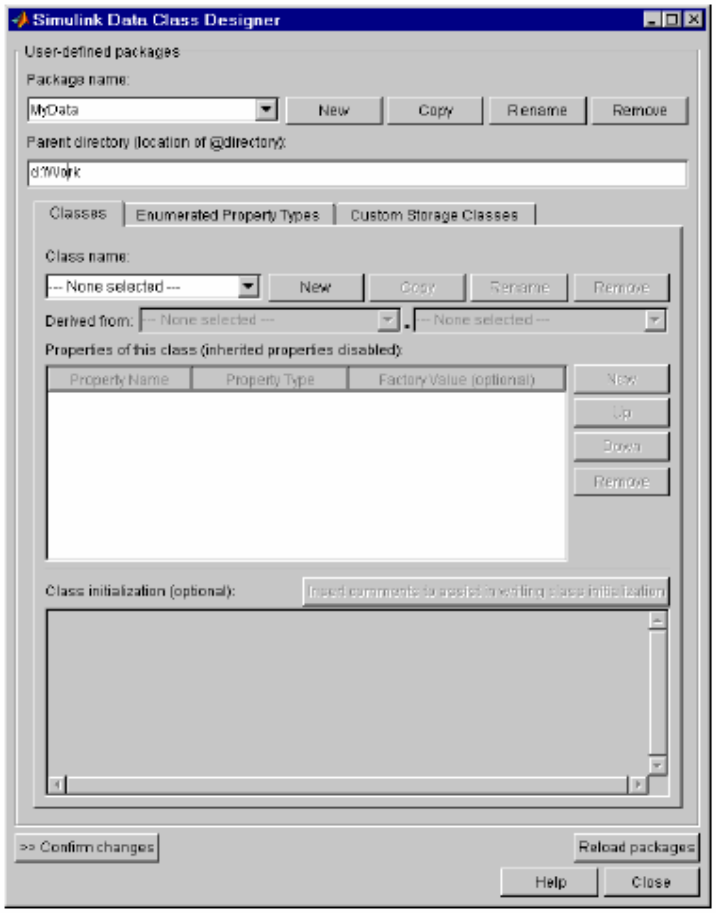
Data Class Designer u okviru Simulinka omogućava korisniku da definira svoje vlastitite klase. Da bi definirao klasu sa **Data Class Designer**, treba da unese paket (package) , ime, klasu roditelja, osobine, i druge karakteristike klase u dijalog boks. **Data Class Designer** će nakon toga generisati P-code koji definira klase koje je kreirao, naprimjer da doda ili otkloni osobine.

Kreiranje klase objekta podataka

Da bi se kreirala klasa sa Data Class Designerom:

1. Izabrati **Data Class designer** iz **Tools** menija.

   Pojaviće se dijalog boks kao na slici:



2. Izabrati ime paketa u kojem želimo da kreiramo klasu iz liste **Package name.**

3. Kliknuti na **New** taster na **Classes** panelu u dijalog boksu **Data Class Designer**.

4. Unjeti ime nove klase u **Classes name** polje na **Classes** panelu.

5. Pritisnuti **Enter** i zatim **Ok** na **Classes** panelu, da se kreira specificirana klasa memorije.

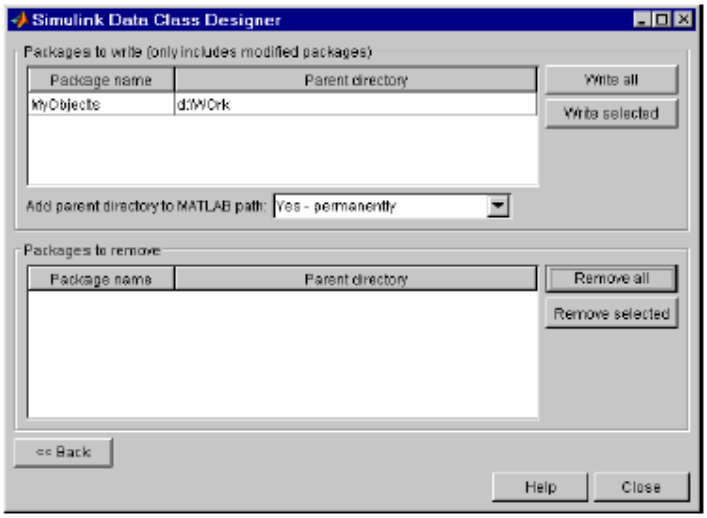
6. Izabrati klasu roditelja za novu klasu.

7. Definirati osobine nove klase.

8. Ako je potrebno , kreirati inicijalizacioni kod za novu klasu.

9. Kliknuti na **Confirm changes**.

Simulink će prikazati **Confirm changes** panel

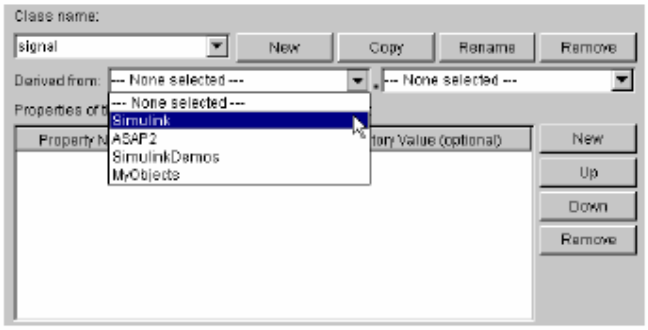


10. Kliknuti **Write all** ili izabrati paket koji sadrži definiciju nove klase i kliknuti **Write selected** da pohrani definiciju nove klase.

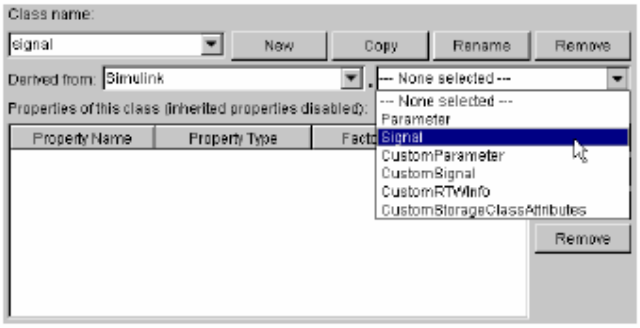
Specificiranje roditelja za klasu

Da se specificira roditelj za klasu treba:

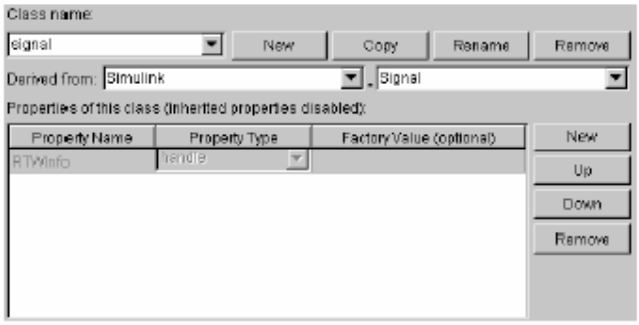
1. Izabrati ime klase iz **Class name** polja u **Classes** panelu.
2. izabrati ime paketa roditeljske klase iz lijevog dijela **Derived from** boksa liste



3. Izabrati roditeljsku klasu iz desnog dijela **Derived from** liste:



Simulink će prikazati osobine izabrane klase izvedene iz roditeljske klase u polju **Propreties of this class.**



Simulink će posiviti naslijedjene osobine da indicira da one ne mogu biti ponovno definirane od strane dijetetove klase.

4. Pohraniti paket koji sadrži klasu

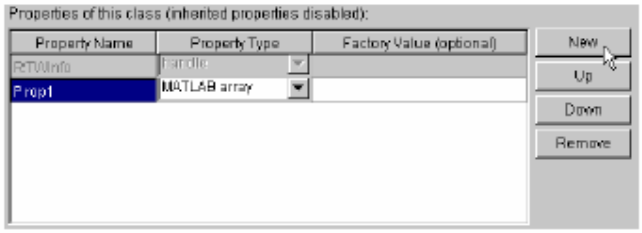
Definiranje osobina klase

Da se doda osobina klase:

1. Izabrati ime klase iz **Class name** polja u **Classes** panelu.

2. izabrati **New** taster koji se nalazi pored polja **Propreties of this class** u **Classes** panelu.

   Simulink kreira osobinu sa default imenom i vrijednošću i prikazuje osobinu u polju **Propreties of this class**.



3. Unjeti ime za novu osobinu u kolonu **Proprety Name.**

4. Izabrati tip podatka osobine iz **Proprety Type** liste.

Lista uključuje ugradjene (built-in) tipove osobina i bilo koji numerirani (enumerated) tip osobina koje je korisnik definisao.

5. Ako želimo da osobina ima default ime, treba unjeti default vrijednost u kolonu **Factory Value**.

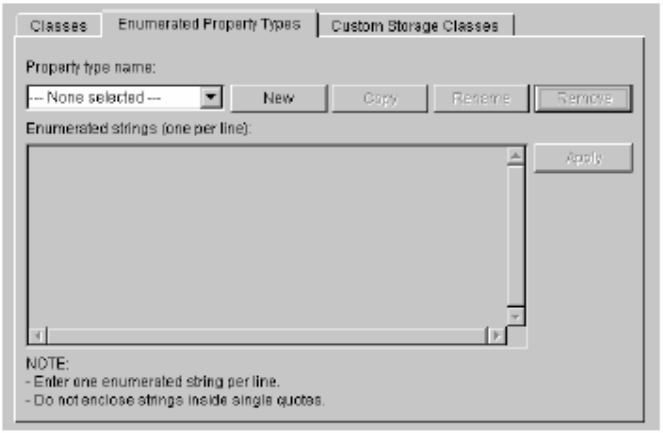
6. Pohraniti paket koji sadrži klasu sa novim ili promjenjenim osobinama.

Definisanje numeriranih tipova osobina

Numerirani tip osobine (**enumerated proprety type**) je tip osobine čija vrijednost mora biti jedna od specificiranih setova vrijednosti , naprimjer: *red, blue, green* . Numerirani tip osobine je validan samo u paketu koji ga je definisao.

Da se kreira numerirani tip osobine:

1. Izabrati **Enumerated Proprety Types** panel u **Data Class Designer**u.



2. Kliknuti na **New** taster pored polja **Proprety type name**.

   Simulink kreira numerirani tip sa default imenom.

3. Promjeniti default ime u polju **Proprety type name** na željeno ime .

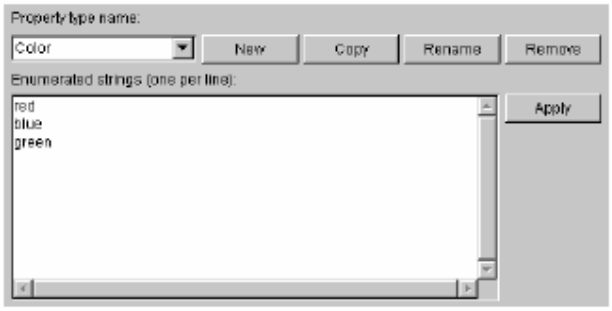
Ime numeriranog tipa osobine mora biti globalno jedinstveno. Ne može biti nikakva druga numerirana osobina sa istim imenom, inače će Simulink prikazati poruku greške.

4. Kliknuti na **OK** taster

Simulink će kreirati novu osobinu u memoriji i omogućiti polje **Enumerated strings** na panelu **Enumerated Proprety Types**.

5. Unjeti dopustive vrijednosti za novi tip osobine u polje **Enumerated strings ,** liniju po liniju.

Naprimjer, slijedeće polje Enumerated strings pokazuje dopustive vrijednosti za numerirani tip osobina koji se zove **Color** (boja).



6. Kliknuti na **Apply** da pohranimo promjene u memoriju.

7. Kliknuti na **Confirm changes.** Nakon toga kliknuti na **Write all** da se pohrane promjene.

   Možemo takodjer koristiti Enumerated proprety Type panel da kopiramo, promjenimo ime i otklonimo numerirane tipove osobina.

Kreiranje inicijalizacionog koda

Korisnik može specificirati kod koji će biti izvršen kada Simulink kreira instancu klase tipa objekta. Da bi se specificirao inicijalizacioni kod za klasu, izabrati klasu iz polja **Class name** u **Data Class Designer** i unjeti inicijalizacioni kod u polje **Class initialization.**

Data Class Designer umeće kod koji je unjet u polje **Class initialization** u funkciju inilijalizacije odogovarajuće klase. Funkcija instantinizacije klase ima oblik:

```
function h = ClassName(varargin)
```

gdje je h – handle ka objektu koji je kreiran i **varargin** je čelija polja koja sadrži ulazne argumente funkcije.

Kreiranje klase paketa

Da bi se kreirao novi paket koji će sadržavati korisnikove klase treba:

1. Kliknuti **New** taster pored polja **Package name** na **Data Class Designeru**.

Simulink prikazuje default ime paketa u polju **Package Name**

2. Editirati **Package name** polje da sadrži ime paketa koje želimo.

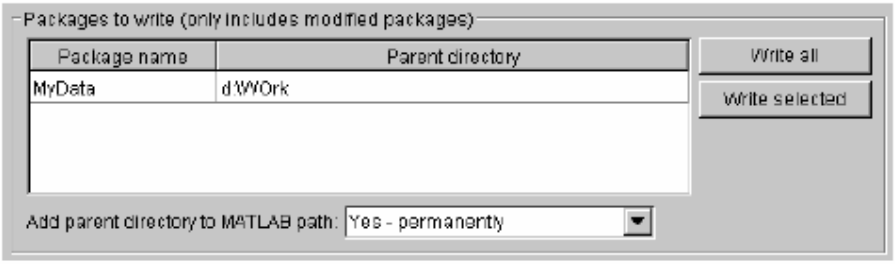
3. Kliknuti **Ok**

4. U polju **Parent directory** , unjeti stazu direktorija gdje želimo da Simulink kreira novi paket.

Simulink kreira specificirani direktorij , ako već ne postoji, kada pohranimo paket u file sistem u narednim koracima.

5. Kliknuti na taster **Confirm changes** na **Data Class Designeru**.

Simulink prikazuje panel **Packages to write**:



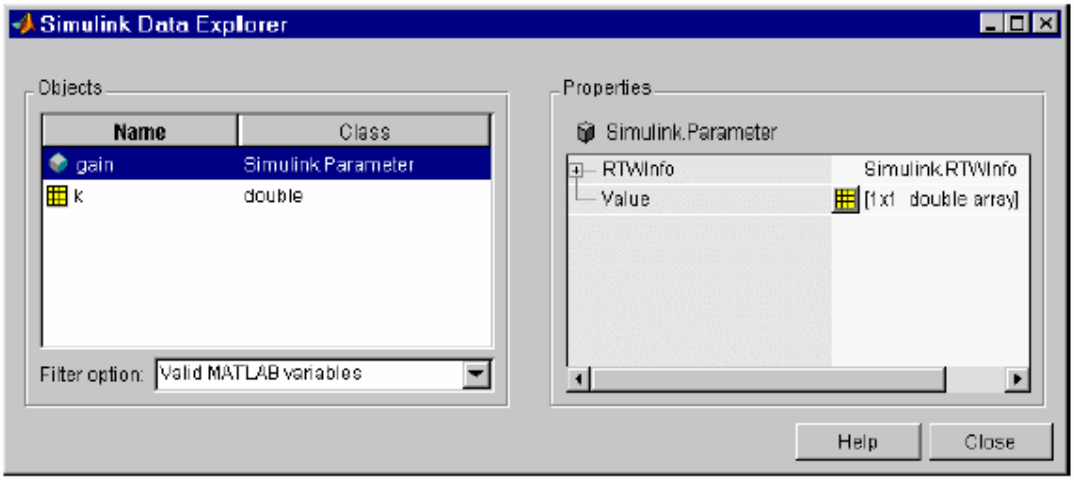
6. Da omogućimo korištenje ovog paketa u tekučoj i budućim sesijama, treba obezbjediti da **Add parent directory to MATLAB path** boks je izabran (default).
   Ovo dodaje stazu roditeljskog direktorija novog paketa na MATLAB stazu.

7. Kliknuti **Write all** ili izabrati novi paket i kliknuti **Write selected** da se pohrani novi paket.

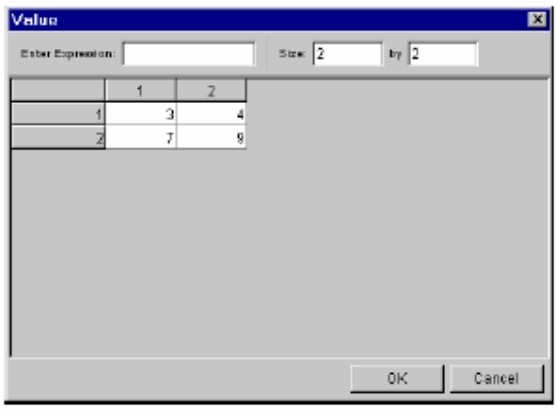
Možemo takodjer koristiti Data Class Designer da kopiramo, promjenimo ime i otklonimo paket.

Simulink Data Explorer

Simulink Data Explorer nam omogućava da setujemo vrijednosti varijabli i objekata podataka u radnom prostoru MATLAB-a. Da se otvori Data Explorer, izabrati **Data Explorer** komandu iz **Tools** menija ili unjeti komandu **slexplr** na MATLAB komandnoj liniji. pojaviće se dijalog boks kao na slici:



Data Exlorer sadrži dva panela. Lijevi panel izlistava varijable koje su definisane u radnom prostoru MATLAB-a. Desni panel prikazuje vrijednost varijabli selektiranih u lijevom panelu. Ako je vrijednost varijabla polja , Data Explorer će prikazati editor polja koji nam dozvoljava da setujemo dimenzije polja (array) i vrijednost svakog elementa.



Pridruživanje korisnikovih podataka sa blokovima

Možemo koristiti Simulinkovu komandu **set_param** da pridružimo svoje podatke sa blokom. Na primjer, slijedeća komanda pridružuje vrijednost varijable **mydata** sa tekuće izabranim blokom:

```
set_param(gcb, 'UserData', mydata)
```

Vrijednost mydata može biti bilo koji tip podatka MATLAB-a, uključujući polja (array), strukture (structures), objekti, i Simulink objekti podataka (data objects). Koristiti **get_param** da dobijemo korisničke podatke pridružene sa blokom:

```
get_param(gcb, 'UserData')
```

Slijedeća komanda će prouzrokovati Simulink da pohrani korisnikove podatke pridružene sa blokom u model file modela koji sadrži blok.

```
set_param(gcb, 'UserDataPersistent', 'on');
```

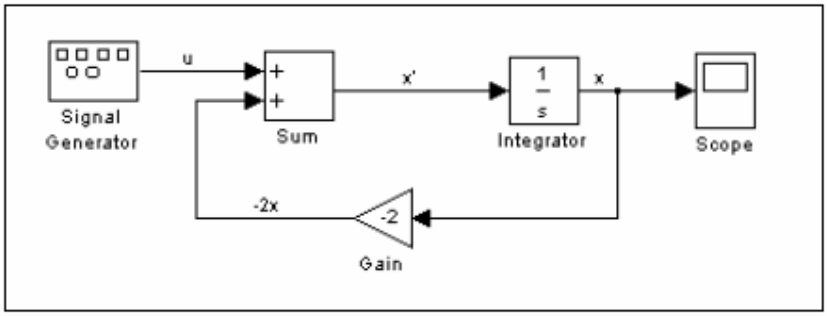
Modeliranje sa Simulinkom

Modeliranje jednostavnog kontinualnog sistema

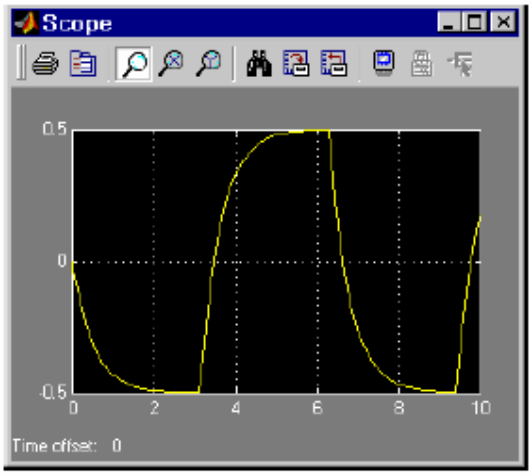
Da se modelira diferencijalna jednaćina:

$$x'(t) = -2x(t) + u(t)$$

gdje je u(t) pravougaoni valni oblik sa amplitudom 1 i frekvencijom 1 rad/sec. Integratorski blok integrira njegov ulaz x' da proizvede x. Kada se kreira kompletan model izgledaće kao na slici:



Osciloskop na izlazu (scope) prikazuje x u svakom trenutku vremena. Za simulaciju od 10 sekundi , izlaz će izgledati kao na slici:



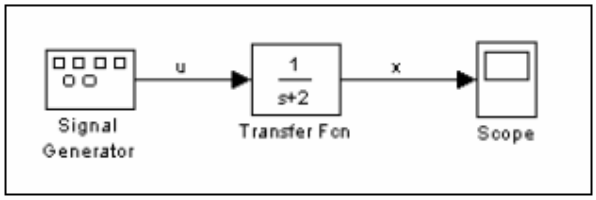
Jednačina koja se modelira može biti takodjer izražena i kao prenosna funkcija. Model koristi blok Transfer Fcn, koji prihvata u kao ulaz i daje x kao izlaz. iz gornje jednačine imamo:

$$sx = -2x + u$$

ili ako sredimo:

$$x/u = 1/(s+2)$$

Blok prenosne funkcije koristi parametre da specificira koeficijente nazivnika i brojnika. U ovom slučaju, brojnik je 1 a nazivnik s+2. Dakle, kao vektori brojnik je [1] a nazivnik [1 2]. Model će sada biti :



Rezultati ove simulacije su identični prethodnoj u vremenskom domenu.

Izbjegavanje pogrešnih kontura (invalid loops)

Simulink nam omogućava da direktno ili indirektno (preko drugih blokova) povezujemo izlaz jednog bloka na njegov ulaz i time kreiramo konturu. Konture su vrlo korisne u nekim slučajevima jer nam naprimjer omogućavaju da dijagramski rješavamo sistem diferencijalnih jednačina ili modeliramo povratno sprežne (feedback) sisteme upravljanja i regulacije.
Medjutim, moguće je takodjer kreirati i konture koje se ne mogu simulirati. Česti slučajevi takvih pogrešnih (invalid) kontura su:

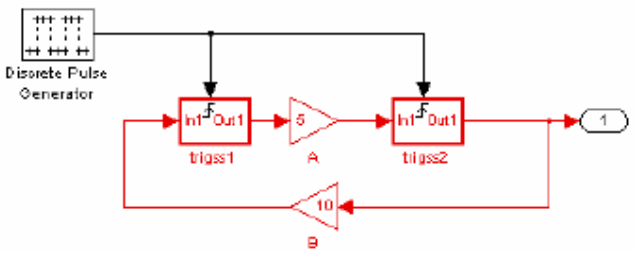
- Konture koje kreiraju pogrešne konekcije poziva funkcija (function-call), ili pokušavaju da modifikuju ulazno/izlazne argumente poziva funkcije
- Samotrigerujući podsistemi i konure koje sadrže nelačevane (non-latched) trigerske podsisteme.
- Konture koje sadrže akcione podsisteme.

Biblioteka primjera blokova podsistema u okviru biblioteka **Ports&Subsystems** sadrži modele koji ilustruju primjere validnih i nevalidnih kontura koje uključuju trigerske i podsisteme poziva funkcije (function-call). Primjeri nevalidnih kontura uključuju slijedeće modele :

- `simulink/Ports&Subsystems/sl_subsys_semantics/Triggered subsystem/sl_subsys_trigerr1`
- `simulink/Ports&Subsystems/sl_subsys_semantics/Triggered subsystem/sl_subsys_trigerr2`
- `simulink/Ports&Subsystems/sl_subsys_semantics/Function-call systems/sl_subsys_fcncallerr3`

Odredjivanje pogrešnih (invalidnih) kontura

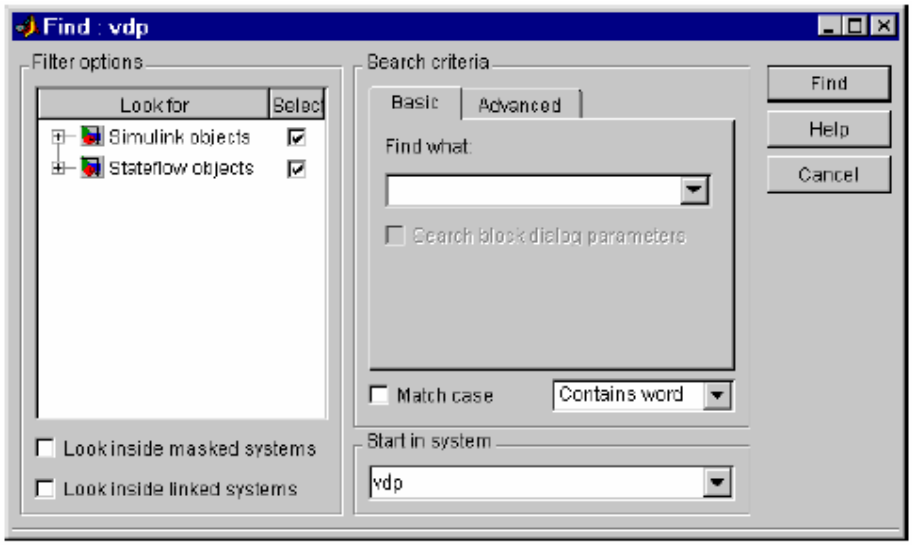
Da bi otkrili da li naš model sadrži invalidne konture, treba izabrati **Update diagram** iz **Edit** menija. Ako model sadrži pogrešnu konturu(e), Simulink će ih naglasiti bojom kao na slijedećoj slici:



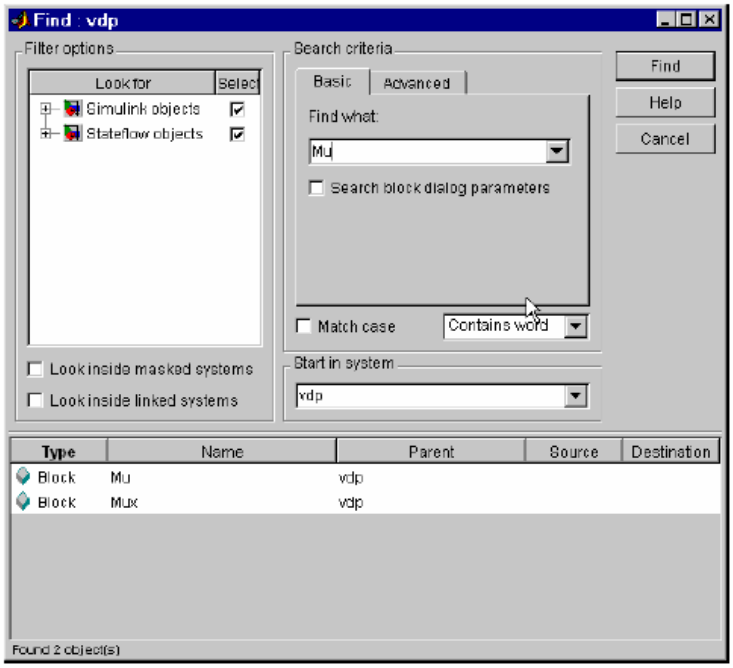
i prikazati poruku greške u **Simulink Diagnostic Viewer**-u.

Nalaženje objekata

za brzo lociranje blokova, signala, stanja, ili drugih objekata u modelu, treba izabrati komandu **Find** iz **Edit** menija. Pojaviće se slijedeći dijalog boks:



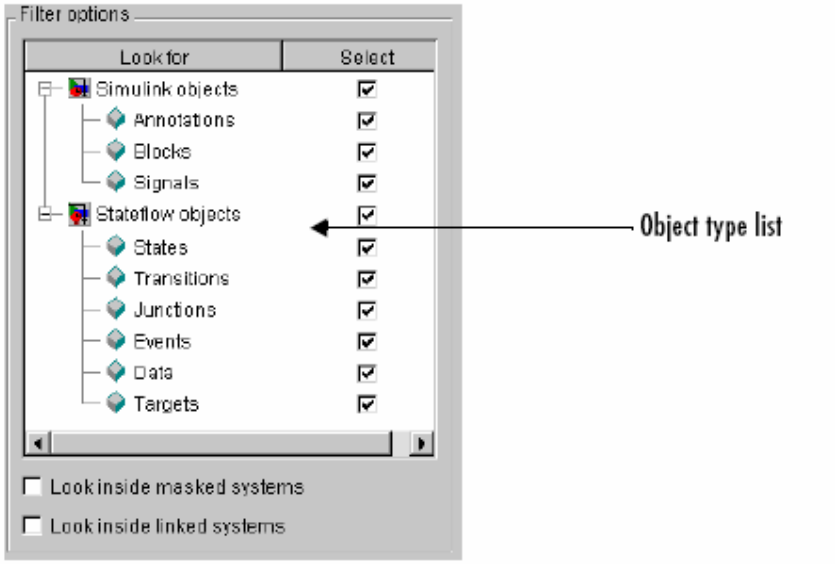
Koristiti filterske opcije **Filter options** i **Search criteria** panele da se specificiraju karakteristike objekata koji se žele naći. Takodjer korisiti i **Start in system** listi da se specificira gdje traženje treba početi.
Bilo koji objekat koji zadovoljava kriterije će se pojaviti u panelu rezultata na dnu dijalog boksa, kao na slici:



Mi možemo prikazati objekat da dva puta kliknemo na njegovo ime u listi rezultata traženja. Simulink će otvoriti sistem ili podsistem koji sadrži objekat i naglasiće bojom traženi objekat.

Filterske opcije

Panel **Filter options** nam omogućava da specificiramo vrste objekata koje tražimo i gdje da ih tražimo.

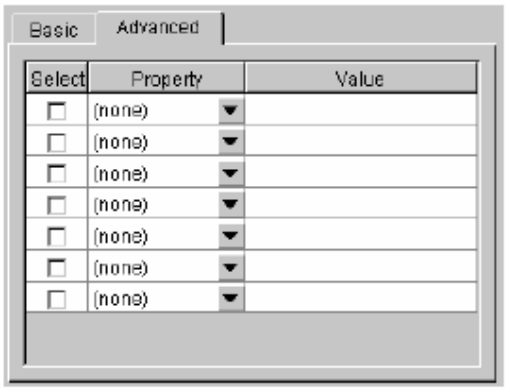


Kriteriji traženja

Search criteria panel nam omogućava da specificiramo kriterije koje objekti moraju ispuniti da zadovolje zahtjev iz traženja.

Basic panel omogućava traženje objekta čije ime i opciono dijalog parametri se uparuju sa specificiranim tekst stringom , koji se unosi u polje **Find what**. izabrati **Search block dialog parameters** ako hoćemo da dijalog parametri budu uključeni u traženje.

Advanced panel omogućava nam da specificiramo set od do sedam osobina koje objekat mora zadovoljiti da bude izabran u traženju.

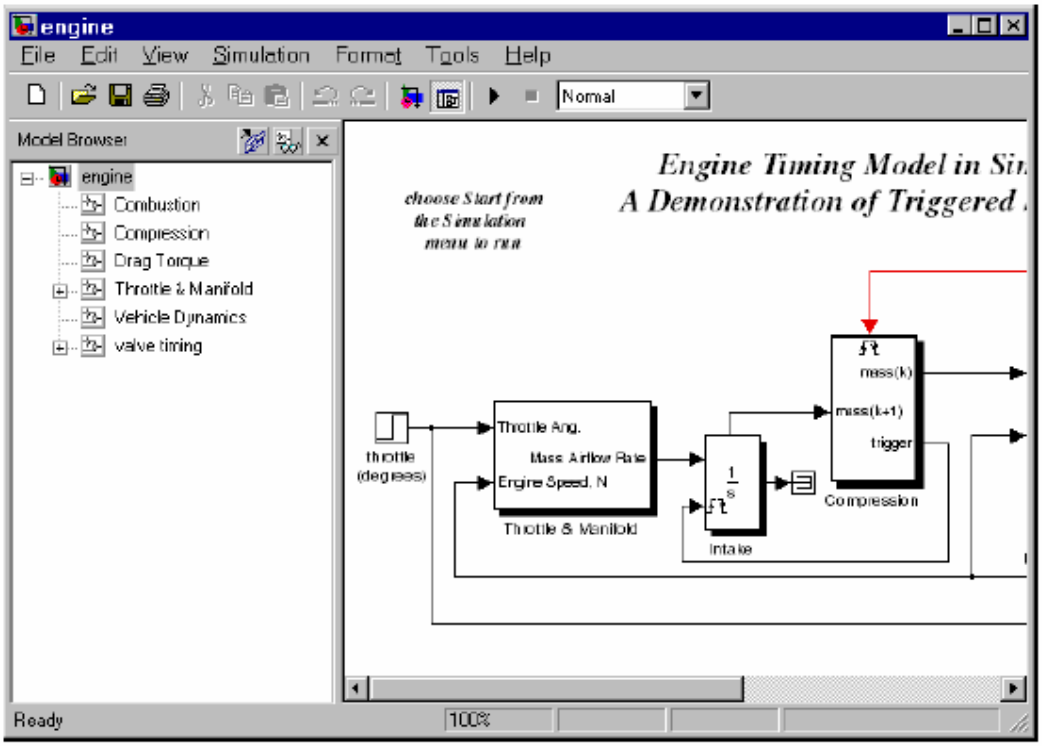


Browser modela

Model browser nam omogućava da

- Putujemo kroz hijerarhiju modela
- Otvorimo sisteme u modelu
- Odredimo blokove koji su sadržani u modelu

Da bi se aktivirao Model Browser treba izabrati **Model browser** iz **View** menija.



Prozor modela je podjeljen u dva panela. Lijevi panel displeja je browser, a drvo struktuirani pogled na blok dijagram je dat u desnom panelu.

Simulacija

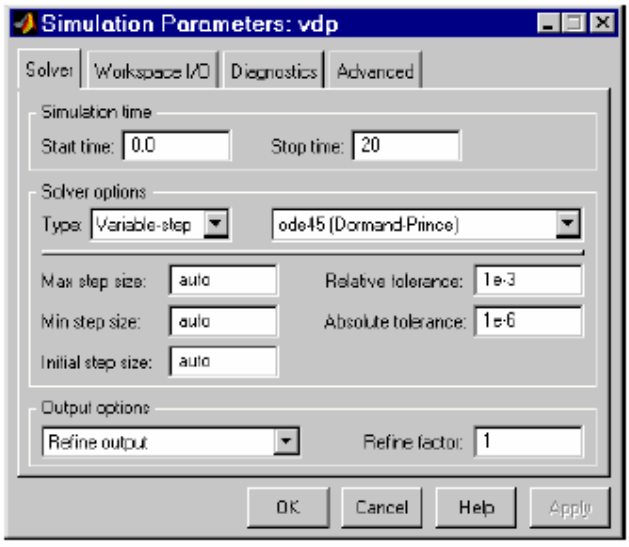
Izvršenje simulacije modela u Simulinku je općenito dvokoračni proces. Prvo, korisnik specificira različite simulacione parametre, kao što je solver koji se koristi da se rješava model u simulaciji, vrijeme početka i kraja simulacije, maksimalna velićina koraka, itd. Nakon toga se u drugom koraku starta simulacija. Simulink izvršava simulaciju od specificiranog startnog vremena do vremena zaustavljanja. Dok se simulacija izvršava, korisnik može interaktirati sa simulacijom na različite načine, zaustaviti ili pauzirati simulaciju, i lansirati simulaciju drugih modela. Ako se greška pojavi za vrijeme simulacije, Simulink zaustavlja simulaciju i iskače dijagnostički viewer koji pomaže korisniku da odredi uzrok greške.

Specifikacija simulacionih parametara

Da bi se koristio dijalog boks **Simulation parameters** treba :

1. Otvoriti ili izabrati model

2. Izabrati **Simulation parameters** iz **Simulation** menija. Otvoriće se **Simulation parameters** dijalog boks kao na slici:



3. Promjeniti setinge koji se žele. Korisnik može specificirati parametre kao validne MATLAB izraze , koji se sastoje iz konstanti, imena varijabli iz radnog prostora, MATLAB funkcija, i matematskih operatora.

4. Kliknuti na **Apply** da se potvrdi promjena a zatim **OK**.

Solver

Simulacija Simulinkovog modela uključuje računanje njegovih ulaza, izlaza i stanja u intervalima od starta simulacije do kraja simulacije. Simulink koristi solver da izvrši ove zadatke. Ni jedan metod rješavanja modela nije pogodan za sve tipove modela. Simulink zbog toga nudi niz solvera, svaki od njih je pogodan za rješavanje specifičnih tipova modela. Panel **Solver** omogućava korisniku da izabere najpogodniji tip za njegov model . Izbor koji mu stoji na raspolaganju uključuje:

- kontinualni solver fiksnog koraka
- kontinualni solver varijabilnog koraka
- diskretni solver fiksnog koraka
- diskretni solver varijabilnog koraka

Kontinualni solveri fiksnog koraka

Ovi solveri izračunavaju kontinualna stanja modela u jednako rasporedjenim vremenskim koracima od startnog vremena do krajnjeg vremena simulacije. Solveri koriste numeričku integraciju da izračunaju kontinualna stanja sistema od izvoda stanja specificiranih od strane modela. Svaki solver koristi različite integracione metode, dozvoljavajući korisniku da izabere najpogodiji metod za njegov model. Da se specificira kontinualni solver fiksnog koraka treba izabrati **fixed-step** iz liste tipa na panelu **Solver.**

Nakon toga treba izabrati jedan od slijedećih integracionih metoda iz liste:

- ode5 , Dormand-Prince formula
- ode4 , RK4, formula Runge-Kuta četvrtog reda
- ode3, Bogacki-Shampine formula
- ode2, Huenov metod, takodjer poznat kao poboljšana Eulerova formula
- ode1, Euler-ov metod

Diskretni solver fiksnog koraka

Simulink obezbjedjuje solver fiksnog koraka koji ne izvršava integraciju. On je pogodan za rješavanje modela koji nemaju kontinualna stanja, uključujući modele bez stanja (stateless models), ili modele koji imaju samo diskretna stanja. Da se specificira ovaj solver , treba izabrati **fixed-step** iz tipa solvera na **Solver** panelu. Nakon toga treba izabrati **discrete** iz liste integracionih metoda.

Kontinualni solver sa varijabilnim korakom

Ovi solveri smanjuju veličinu koraka simulacije da bi povećali tačnost kada se kontinualna stanja sistema brzo mjenjaju a povećavaju veličinu koraka da bi uštedili na

vremenu simulacije kada se stanja sistema sporo mjenjaju. Da bi se specificirao kontinualni solver sa varijabilnim korakom treba izabrati **variable-step** iz liste tipa solvera u **Solver** panelu. Nakon toga treba izabrati jednu od slijedećih opcija za intgracioni metod:

- ode45 je baziran na eksplicitnoj Runge-Kuta (4,5) formuli, paru Dormand-Prince. Opčenito ode45 je najbolji solver za većinu problema, kao prvi pokašaj izbora. Zbog toga je ode45 i default solver koji se koristi u Simulinku za modele sa kontinualnim stanjima.

- ode23 je takodjer baziran na eksplicitnom Runge-Kuta (2,3) paru Bogackija i Shampina. Može biti efikasniji od ode45 kod grubih tolerancija i u prisustvu blagih krutosti. ode23 je jednokoračni solver.

- ode113 je Adams-Bashforth-Moulton PACE solver varijabilnog reda. Može biti efikasniji od ode45 kod strogih tolerancija. ode113 je višekoračni solver , tj. on normalno treba rješenja nekoliko prethodnih trenutaka vremena da izračuna tekuče stanje

- ode15s solver promjenjljivog reda baziran na formuli numeričke diferencijacije (NDF). Kao i ode113 , ode15s je solver sa višekoračnom metodom. Ako sumnjamo da je sistem krut, ili ode45 nije mogao biti korišten ili je vrlo neefikasan, treba pokušati sa ode15s.

- ode23s je baziran na modificiranoj Rosenbrockovoj formuli drugog reda. Pošto je to jednokoračni solver, može biti efikasniji od ode15s kod grubih tolerancija.

- ode23t je implementacija trapezoidalnog pravila koristeći slobodnog interpolanta. Ovaj solver treba koristiti ako je problem samo umjereno krut, i treba nam rješenje bez numeričkog gušenja (damping).

- ode23tb je implementacija TR-BDF2 , implicitne Runge-Kuta formule sa prvim stepenom koji je korak trapezoidalalne formule, a drugi stepen je diferencijalna formula unatrag drugog reda. Kao i ode23s , ovaj solver može biti efikasniji nego ode15s kod grubih tolerancija.

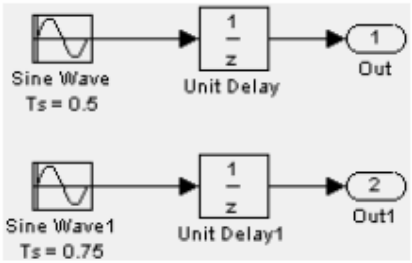
Diskretni solver sa varijabilnim korakom

Simulink obezbjedjuje diksretni solver varijabilnog koraka koji ne izvršava integraciju ali otkriva detekciju prolaska kroz nulu. Treba koristiti ovaj solver za modele koji nemaju kontinualna stanja a koji imaju kontinualne signale koji zahtjevaju detekciju prolaska kroz nulu i/ili imaju diksretne blokove koji rade u različitim vremenima sampliranja. Simulink koristi ovaj solver po defaultu , ako korisnik ne specificira diskretni solver fiksnog koraka a model nema kontinualnih stanja.

Diksretni solver fiksnog koraka poboljšava simulaciju zbog vremenskih koraka fiksne velićine. Kao rezultat toga, može učiniti korak čak i kada se ništa ne dešava u modelu. Nasuprot ovoga, diskretni solver varijabilnog koraka ne mora da napravi vremenski korak kada se ništa ne dešava u modelu. Umjesto toga, on može podesiti velićinu

koraka da napreduje simulaciju do slijedeće tačke gdje se nešto značajno može desiti. Zavisno od modela, ovo može značajno reducirati broj koraka i time i vrijeme potrebno da se simulira model.

Slijedeći višebrzinski model ilustrira kako solver varijabilnog koraka može skratiti vrijeme simulacije:



Ovaj model generira izlaze kod dvije različite brzine, svakih 0.5 sekundi i svakih 0.75 sekundi. Da bi se hvatala obadva izlaza , solver fiksnog koraka mora uzimati vremenski korak svakih 0.25 sekundi pa je (fundamentalno vrijeme sampliranja modela):

```
[0.0 0.25 0.5 0.75 1.0 1.25 ...]
```

Nasuprot ovome, solver varijabilnog koraka, treba da uzme korak samo kada model stvarno generiše izlaz:

```
[0.0 0.5 0.75 1.0 1.5 2.0 2.25 ...]
```

Ovo značajno reducira broj vremenskih koraka koji su potrebni da se simulira model.

Multitasking opcije

Ako izaberemo solver sa fiksnim korakom, panel **Solver** u dijalog boksu **Simulation Parameters** prikazaće **Mode** listu opcija. Lista nam omogućava da izaberemo jedan od slijedećih simulacionih modova rada:

Multitasking

Ovaj mod izdaje grešku ako otkrije ilegalnu tranziciju u brzini sampliranja izmedju blokova, tj. , direktnu vezu izmedju blokova koji rade na različitim brzinama samplovanja. U multitasking sistemima u realnom vremenu, ilegalna tranzicija u brzini samplovanja izmedju taskova može rezultirati u tome da izlaz iz taska nije raspoloživ kada je potreban nekom drugom tasku. Provjerevanjem takvih tranzicija, multitasking mod pomaže nam da kreiramo validne modele multitasking sistema iz realnog svijeta, gdje sekcije modela predstavljaju taskove koji se istovremeno izvršavaju (concurrently).

Treba koristiti **Rate Transition** blok da se eliminiraju ilegalne tranzicije u brzinama unutar modela.

Single tasking

Ovaj mod ne provjerava tranzicije u brzinama sampliranja medju blokovima. Ovaj mod rada je koristan kada se modelira sistem sa samo jednim taskom. U takvim sistemima sinhronizacija izmedju taskova se ne korsti.

Auto

Ova opcija prouzrokuje da Simulink koristi mod single taskinga ako svi blokovi rade sa istom brzinom sampliranja , a multitasking mode ako model sadrži blokove koji rade sa različitim brzinama.

Izlazne opcije

Panel **Output options** u dijalog boksu omogućava nam da kontrolišemo koliko izlaza simulacija generiše. Možemo izabrati izmedju tri opcije:

- Refine output
- Produce additional output
- Produce specified output only

Rafinacija izlaza

Opcija **Refine output** obezbjedjuje dodatne izlazne tačke kada je simulacioni izlaz isuviše grub. Ovaj parametar obezbjedjuje cjelobrojni broj izlaznih tačaka izmedju vremenskih koraka: naprimjer faktor rafinacije od 2 obezbjedjuje izlaz na polovini izmedju dva vremenska koraka , kao i na samim koracima sampliranja. Default refine faktor je 1.
Da bi se dobio gladki izlaz, mnogo je brže promjeniti faktor rafiniranja umjesto da se smanjuje velićina koraka simulacije . Kada se promjeni faktor rafinacije, solveri generišu dodatne tačke evaluirajući kontinualnu formulu proširenja u ovim tačkama. Mjenjajući faktor rafinacije ne mjenja korake koje koristi solver.

Faktor rafinacije se može primjeniti na solvere sa promjenljivim korakom i najkorisniji je kada koristimo ode45. Solver ode45 je u stanju da uzima velike korake, i kada iscrtavamo simulacione izlaze, možemo naći da izlaz iz ovog solvera nije dovoljno gladak. Ako je ovo slučaj, treba ponovo izvršiti simulaciju sa većim faktorom rafinacije. Vrijednost od 4 treba obezbjediti mnogo gladje rezultate.

Proizvodjenje dodatnog izlaza

Opcija **Produce additional output** omogućava korisniku da specificira direktno ona dodatna vremena kod kojih solver treba da generiše izlaze. Kada se izabere ova opcija, Simulink prikazuje polje **Output Times** na panelu **Solver**. Unjeti MATLAB izraz u ovo polje koji se evaulira u dodatno vrijeme ili vektor dodatnih vremena. Dodatni izlazi se proizvode koristeći kontinualne formule za proširenje u dodatnim trenutcima vremena. Za razliku od **refine factor**a, ova opcija mjenja velićinu koraka simulacije tako da

vremenski koraci koincidiraju sa vremenima koja smo specificirali za dodatne vrijednosti izlaza.

Proizvodjenje samo specificranih izlaza

Ova opcija **Produce specified output only** , obezbjedjuje simulacione izlaze samo u specificiranim trenutcima vremena. Ova opcija mjenja velićinu koraka simulacije tako da vremenski koraci koincidiraju sa vremenima koja smo specificirali za proizvodjenje izlaza. Ovaj izbor je koristan kada poredimo različite simulacije da obezbjedimo da simulacije proizvedu izlaze u istim trenutcima vremena.

Poredjenje izlaznih opcija

Predpostavimo simulaciju koja generiše izlaze u ovim trenutcima vremena:

```
0, 2.5, 5, 8.5, 10
```

izabirući **Refine output** i specificirajući faktor rafinacije od 2 , generisaće se izlaz u slijedećim vremenskim trenutcima:

```
0, 1.25, 2.5, 3.75, 5, 6.75, 8.5, 9.25, 10
```

Izabirući **Produce additional output** opciju i specificirajući [0:10], generiše izlaze u slijedećim vremenima :

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

i možda i u dodatnim vremenima, zavisno od velićine stepa koja je izabrana od strane solvera sa varijabilnim korakom.

Izabirući opciju **Produce Specified Output Only** i specificirajući [0:10], generisaće izlaz u slijedećim vremenima :

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

Opčenito, trebamo specificirati izlazne tačke kao cjelobrojna vremena od osnovnog koraka. Na primjer,

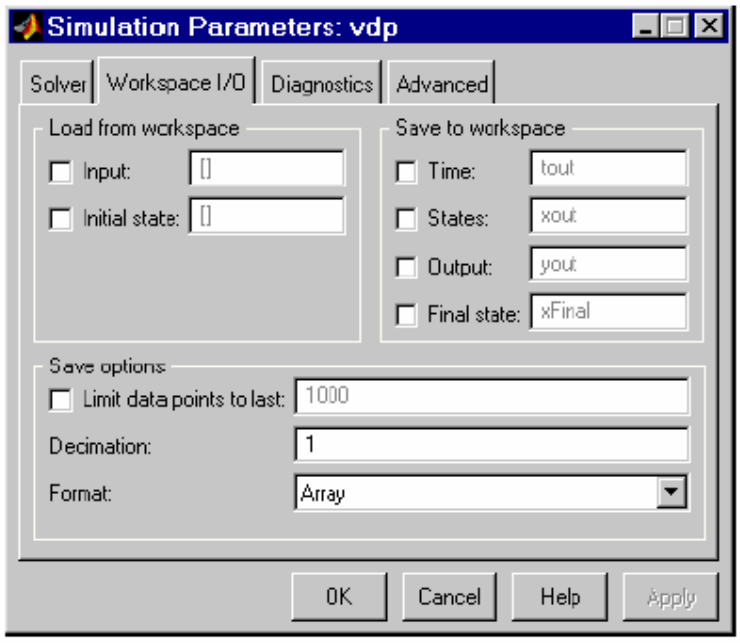
```
[1:100]*0.01
```

je tačnije nego :

```
[1:0.01:100]
```

Radni prostor I/O panela

Korisnik može usmjeriti simulacione izlaze na varijable u radnom prostoru kao i dobiti ulaze i početna stanja iz radnog prostora. Na dijalog boksu **Simulation parameters** izabrati Workspace I/O tab. Pojaviće sa slijedeći panel:



Punjenje ulaza iz baznog radnog prostora

Simulink može primjeniti ulaz iz baznog radnog prostora modela, na modelove inporte najvišeg nivoa za vrijeme izvršenja simulacije. Da bi se specificirala ova opcija , izabrati **Input** boks u **Load from workspace** zoni od **Workspace I/O** panela. Nakon toga , unjeti specifikaciju vanjskog ulaza u edit boks i izabrati **Apply.**

Eksterni (tj. iz radnog prostora) ulaz može uzeti jedan od slijedećih oblika:

Array . Da bi se koristio ovaj format, izabrati **Input** u **Load from workspace** panel i izabrati array opciju iz **Format** liste na **Workspace I/O** panelu. Izabirući ovu opciju prouzrokovaće da Simulink evaulira izraz pored **Input** check boksa i koristi rezultat kao ulaz u model.

Izraz se mora evaluirati u realnu (nekompleksnu) matricu tipa podatka **double**. Prva kolona matrice mora biti vektor vremena u rastučem redoslijedu. Ostale kolone specificiraju ulazne vrijednosti. Naročito, svaka kolona predstavlja ulaz za različite Inport blok signale (u sekvencijalnom redoslijedu) i svaki red je ulazna vrijednost za odgovarajuću vremensku tačku. Simulink linearno interpolira ili ekstrapolira ulazne vrijednosti prema potrebi ako se izabere opcija **Interpolate data** , za odgovarajući Inport.
Totalni broj kolona ulazne matrice mora biti jednak n + 1, gdje n je totalni broj signala koji ulaze u inporte modela.

Default ulazni izraz za model je **[t, u]** a default ulazni format je **array**. Tako, ako definiramo **t** i **u** u baznom radnom prostoru, treba samo da izaberemo **Input** opciju da unesemo podatke iz baznog radnog prostora modela. Na primjer, predpostavimo da model ima dva inporta, od kojih jedan prihvata dva signala a drugi prihvata jedan signal. Takodjer, predpostavimo da bazni radni prostor definira **u** i **t** kako slijedi:

```
t = (0:0.1:1)';
u = [sin(t), cos(t), 4*cos(t)];
```

Opaska: Ulazni format polja (array) nam dozvoljava da unesemo samo realne (nekompleksne) skalarne ili vektorske podatke tipa **double**. Treba koristiti format strukture (structure) da unesemo kompleksne podatke, matrične (2-D) podatke, i/ili druge tipove osim double.

Struktura sa vremenom

Simulink može čitati podatke iz radnog prostora u formi strukture čije ime je specificirano u **Input** polju teksta. Ulazna struktura mora imati dva polja najvišeg nivoa: **time** i **signals**. Polje time sadrži vektor kolonu vremena simulacije. Polje **signals** sadrži polje substruktura , svaka od kojih korespondira ulaznom portu modela.

Svaka podstruktura **signals** mora sadržavati dva polja koja se zovu **values** i **dimensions** . Polje **values** mora sadržavati polje (array) ulaza za odgovarajući ulazni port gdje svaki ulaz korespondira sa vremenskom tačkom koja je specificirana sa poljem **time**. Polje **dimensions** specificira dimenzije ulaza. Ako je svaki ulaz skalar ili vektor (1-D polje) vrijednost, polje **dimensions** mora biti skalarna vrijednost koja specificira dužinu vektora (1 za skalar). Ako svaki ulaz je matrica (2-D polje), polje **dimensions** mora biti dvo elementni vektor čiji prvi element specificira broj redova u matrici i čiji drugi element specificira broj kolona.

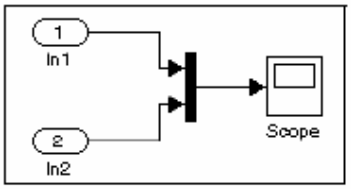
Ako ulazi za port su skalarne ili vektorske vrijednosti, polje values mora biti M x N polje gdje je **M** broj vremenskih tačaka specificiranih sa time poljem a **N** je dužina svake vektorske vrijednosti. Na primjer, slijedeći kod kreira ulaznu strukturu za punjenje 11 vremenskih samplova dvo elementnog signalnog vektora tipa **int8** u model sa jednim ulaznim portom.

```
a.time = (0:0.1:1)';
c1 = int8([0:1:10]');
c2 = int8([0:10:100]');
a.signals(1).values = [c1 c2];
a.signals(1).dimensions = 2;
```

Da bi se napunili ovi podatci u inport modela, treba izabrati **Input** opciju na **Workspace I/O** panelu i unjeti **a** u polje ulaznog izraza.

Ako su ulazi za port matrice (2-D polja) , polje **values** mora biti **M x N x T** polje gdje **M** i **N** su dimenzije svakog matričnog ulaza a **T** je broj vremenskih tačaka. Na primjer,

predpostavimo da želimo da unesemo 51 vremenski sample od 4 x 5 matrice signala u jedan od ulaznih portova modela. Tada, odgovarajuće polje **dimensions** od strukture radnog prostora mora biti jednako [4 5] a polje **values** mora imati dimenzije 4 x 5 x 51. Kao drugi primjer , posmatrajmo slijedeći model, koji ima dva ulaza:



Predpostavimo da želimo da unesemo sinusni valni oblik u prvi port a kosinusni u drugu port. Da bi ovo uradili, definisaćemo vektor **a** , kako slijedi, u baznom radnom prostoru:

```
a.time = (0:0.1:1)';
a.signals(1).values = sin(a.time);
a.signals(1).dimensions = 1;
a.signals(2).values = cos(a.time);
a.signals(2).dimensions = 1;
```

Izabrati **Input** boks za ovaj model, i unjeti **a** u susjedno polje teksta, i izabrati **StructureWithTime** kao I/O format.

Struktura

Format Structure je isti kao **Structure with time** format izuzev da je polje **time** prazno. Na primjer, u slijedećem primjeru, možemo postaviti polje **time** kako slijedi:

```
a.time = []
```

U ovom slućaju, Simulink očitava ulaz po prvi put iz prvog elementa iz polja vrijednosti inporta, vrijednost za drugi vremenski korak iz drugog elementa iz polja vrijednosti , itd.

Per-port strukture

Per-port structure format se sastoji od odvojene strukture –sa-vremenom **(structure-with-time)** ili strukture –bez-vremena (**structure-without-time**) za svaki port. Ulazna struktura podataka svakog porta ima samo jedno polje **signals**. Da se specificira ova opcija, unjeti imena struktura u **Input** tekst polje kao listu odvojenu zarezom (comma-separated list) , in1, in2, ..., inN , gdje in1 su podatci za prvi port modela, in2 za drugi port itd.

Time expression

Ovaj vremenski izraz može biti bilo koji MATLAB izraz koji se evaluira u vektor red jednak po dužini sa brojem signala koji ulaze u inportove modela. Nadalje, predpostavimo da **timefcn** je funkcija koju je definirao korisnik koja vraća vektor red dužine dva elementa. U nastavku ćemo dati validan ulaz vremenskih izraza za takav model:

```
'[3*sin(t), cos(2*t)]'

'4*timefcn(w*t)+7'
```

Simulink evaluira izraz u svakom koraku simulacije, primjenjujući rezultirajuće vrijednosti na inportove modela. Primjetimo da Simulink definira varijablu **t** kada se simulacija izvršava. Takodjer, možemo izlistaviti vremensku varijablu u izrazima za funkcije od jedne varijable. Naprimjer, Simulink interpretira izraz **sin** kao **sin(t)** .

Pohranjivanje izlaza u radni prostor

Možemo specificirati povratne varijable izabirući **Time, States,** i/ili **Output** check boksove u zoni **Save to workspace** u panelu dijalog boksa. Specificirajući povratne varijable prouzrokovaće da Simulink upisuje vrijednosti za vrijeme, stanje, i izlazne trajektorije (koliko ih je izabrano) u radni prostor.

Da bi se doznačile vrijednosti na različite varijable, specificirati imena onih varijabli u poljima na desno od check boksova. Da bi se upisivao izlaz na više od jedne varijable, specificirati imena varijabli u listu sa zapetom kao razdjelnikom. Simulink pohranjuje simulaciona vremena u vektor specificiran u zoni **Save to Workspace .**

Structure with time

Ako izaberemo ovaj format, Simulink pohranjuje stanja modela i izlaze u strukture koje imaju imena specificirana u **Save to Workspace** zoni (naprimjer **xout** i **yout**).

Struktura koja se korsti da pohrani izlaze ima dva polja najvišeg nivoa: **time** i **signals**. Polje **time** sadrži vektor simulacionih vremena. Polje **signals** sadrži polje podstruktura, svaka od kojih korespondira sa outportom modela. Svaka podstruktura ima četiri polja: **values, dimensions, label i blockName**. Polje **values** sadrži izlaze za odgovarajući outport. Ako su izlazi skalari ili vektori, polje **values** je matrica kod koje svaki red predstavlja izlaz u istom vremenu specificiranom sa odgovarajućim elementom vremenskog vektora. Ako su izlazi matrice (2-D) vrijednosti, polje values je 3-D polje dimenzije **M x N x T** , gdje **M x N** je dimenzija izlaznog signala a **T** je broj izlaznih samplova. Ako je T = 1, MATLAB izostavlja posljednju dimenziju. Zbog toga, polje values je **M x N** matrica. Polje **dimensions** specificira dimenzije izlaznog signala. Polje **label** specificira labelu signala koji je spojen na outport ili tip stanja (kontinualni ili diskretni). Polje **blockName** specificira ime odgovarajućeg outporta ili blok sa stanjima.

Struktura koja se koristi da pohrani stanja ima slićnu organizaciju. Struktura stanja ima dva polja najvišeg nivoa: **time i signals**. Polje **time** sadrži vektor vremena simulacije. Polje **signals** sadrži polje podstruktura , svaka od kojih odgovara sa jednim od stanja modela. Svaka struktura **signals** ima četiri polja: **values**, **dimensions**, **label** i **blockName**. Polje **values** sadrži vremenske samplove stanja bloka specificirane sa poljem **blockName**. Polje label za ugradjene blokove indicira tip stanja: bilo **CSTATE** (kontinualno stanje) ili **DSTATE** (diskretno stanje). Za blokove S-funkcija, polje label sadrži bilo koje ime koje je doznačeno sa stanjem sa blokom S-funkcije.

Vremenski samplovi stanja se pohranjuju u polje **values** kao matrica vrijednosti. Svaki red korespondira sa vremenskim samplom. Svaki element reda korespondira jednom elementu stanja. Ako je stanje matrica, matrica se pohranjuje u **values** polje u redoslijedu kolone (**column-major order**). Na primjer, predpostavimo da model uključuje 2 x 2 matricu stanja i da Simulink loguje 51 uzorak stanja za vrijeme trajanja simulacije. Polje **values** za ovo stanje će sadržavati 51 x 4 matricu gdje svaki red korespondira sa vremenskim samplom stanja, i gdje prva dva elementa svakog reda odogovaraju prvoj koloni sampla a posljednja dva elementa svakog reda odgovaraju drugoj koloni sampla.

Structure

Ovaj format je isti kao i prethodni izuzev što Simulink ne pohranjuje simulaciona vremena u polje time od pohranjene strukture.

Per-port structures.

Ovaj format se sastoji od odvojenih **structure-with-time** ili **structure-without-time** za svaki izlazni port. Svaka struktura izlaznih podataka ima samo jedno polje **signals**. Da bi se specificirala ova opcija, unjeti imena struktura u Output tekst polje kao listu odjelejnu zarezima , out1, out2, ..., outN, gdje out1 su podatci za prvi port modela, out2 za drugi port , itd.

Pohranjivanje podataka u radni prostor može usporiti simulaciju i trošiti memoriju. Da bi se ovo izbjeglo, možemo ograničiti broj uzoraka koji se pohranjuju samo na najsvježije uzorke, ili možemo preskakati samplove primjenjujući faktor prorjedjivanja (decimation factor). Da bi se postavile granice na broj samplova podataka koji se pohranjuje, izabrati **check box** označen kao **Limit data points to last** i specificirati broj samplova koji se pohranjuju. Da se promjeni faktor prorijedjivanja , unjeti vrijednost u polje na desno od **Decimation** labele . Na primejr, vrijednost 2 će pohraniti svaku drugu tačku koja se generiše u simulaciji.

Punjenje i pohranjivanje stanja

Početni uslovi, koji se primjenjuju na sistem kod starta simulacije, su općenito setovani u blokovima. Možemo nadjahati početne uslove koji se postavljaju u blokovima specificirajući ih u **States** zoni ovog panela.

Korisnik može takodjer pohraniti finalna stanja za tekuću simulaciju i primjeniti ih na naredne simulacije. Ova osobina može biti korisna kada želimo da pohranimo rješenja

stacionarnog stanja i ponovno startamo simulaciju kod tog poznatog stanja. Stanja se pohranjuju u formatu koji izaberemo u zoni **Save options** u panelu **Workspace I/O.**

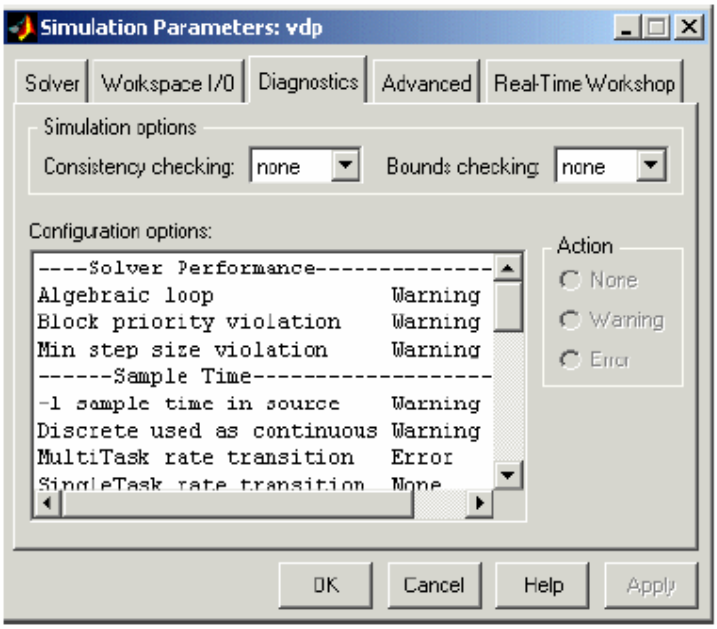
Da se pohrane konačna stanja (tj. stanja na završetku simulacije), izabrati check boks **Final state** i unjeti varijablu u susjedno edit polje.

Da se napuni (load) stanje, izabrati **Initial State** check boks i specificirati ime varijable koja sadrži vrijednosti početnog stanja. Ova varijabla može biti matrica ili struktura iste forme koja je korištena i za finalna stanja. Ovo omogućava Simulinku da postavi početna stanja za tekuću sesiju na finalna stanja pohranjena iz prethodne sesije, koristeći **Structure** ili **Structure with time** format.

Ako check boks nije izabran ili je polje stanja prazno ([ ]) , Simulink koristi početne uslove definirane u blokovima.

Dijagnostički panel

Možete indicirati željenu akciju za mnoge tipove dogadjaja ili uslova koji se mogu sresti za vrijeme simulacije , izabirući **Diagnostics** tab u dijalog boksu **Simulation Parameters.** Pojaviće se dijalog boks kao na slici:



Dijalog boks uključuje slijedeće opcije:

Consistency checking

Čekiranje konzistentnosti je alat za debagiranje (debugging tool) koji validira izvjesne predpostavke koje prave Simulinkovi ODE solveri. Njihovo glavno korištenje je da se obezbjedi da S-funkcije poštuju ista pravila kao i Simulinkovi ugradjeni blokovi. Pošto provjera konzistentnosti rezultira u značajnom smanjenju performansi (i do 40 %),

treba u opštem slučaju biti setovana na **off.** Treba je korsititi samo da se validiraju S-funkcije i da se odrede uzroci neoćekivanih rezultata simulacije.

Da bi se ostvarila efikasna integacija, Simulink pohranjuje (u cache memoriji), izvjesne vrijednosti iz jednog vremenskog koraka da bi koristio u narednom koraku. Na primjer, izvodi na kraju vremenskog koraka mogu u opštem slučaju biti ponovno korišteni na startu slijedećeg vremenskog koraka. Solveri koriste ovo da izbjegnu redundantno izračunavanje izvoda.

Druga namjena provjere konzistentnosti je da obezbjedi da blokovi proizvode konstantan izlaz kada se pozivaju sa datom vrijednošću vremena (t). Ovo je važno za krute solvere (ode23s i ode15s), pošto, dok izračunavaju Jacobian, izlazne funkcije bloka mogu biti pozivane više puta pri istoj vrijednosti od t.

Kada je provjera konzistentnosti omogućena , Simulink ponovno računa odgovarajuće vrijednosti i poredi ih sa kaširanim vrijednostima . Ako vrijednosti nisu iste, pojaviće se greška konzistentnosti. Simulink poredi izračunate vrijednosti za ove velićine:

- izlaze
- prolaske kroz nulu (zero crossing)
- izvode (derivatives)
- stanja (states)

Provjera granica (bounds checking)

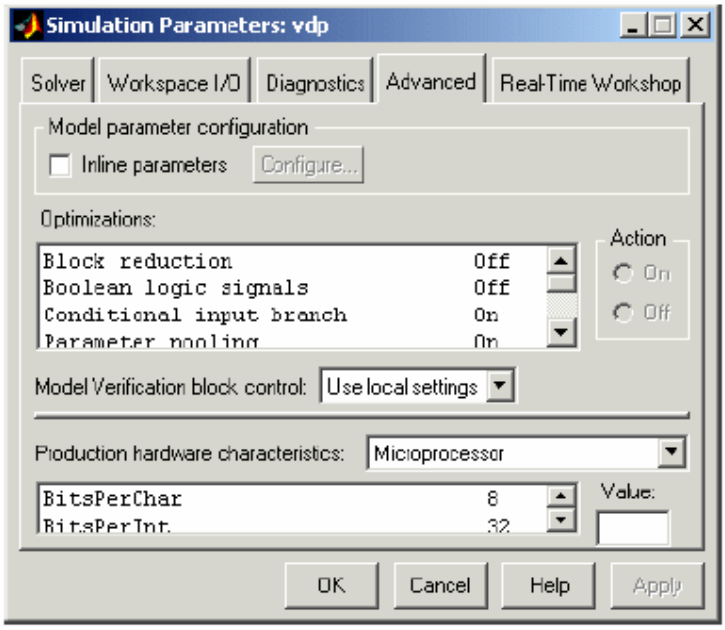
Ova opcija prouzrokuje da Simulink provjerava da li blok piše van memorije koja je dodjeljena njemu za vrijeme izvršenja simulacije. Tipično, ovo se može desiti samo ako model uključuje korisnikovu **S-funkciju** koja ima grešku (bug). Ako je omogućena, ovo će biti provjereno za svaki blok u modelu svaki put kada se blok izvršava. Kao rezultat, omogućenje ove opcije značajno usporava brzinu izvršenja modela. Zbog toga, da se izbjegne nepotrebno usporenje modela, treba omogućiti ovu opciju samo ako sumnjamo da model sadrži S-funkciju napisanu od strane korisnika koja ima bug.

Konfiguracione opcije

Kontrolna lista **Configuration options** izlistava nenormalne tipove dogadjaja koji se mogu pojaviti za vrijeme izvršenja modela. Za svaki tip dogadjaja možemo izabrati da li želimo da ne dobijemo poruku, ili pak poruku upozorenja ili poruku greške. Poruka upozorenja neće završiti simulaciju , ali poruka greške hoće.

Napredni panel (advanced pane)

Napredni panel (advanced pane) nam omogućava da postavimo različite opcije koji utiču na simulacione performanse.



Inline parametri

Po defaultu , korisnik može modificirati mnoge parametre blokova za vrijeme simulacije. Izabirući ovu opciju, čini sve parametre nepodesivim po defaultu. Čineći ih ne podesivim, dozvoljava Simulinku da pomjera blokove čiji su izlazi zavisni samo od vrijednosti parametara bloka van simulacione konture, i time ubrzavajući simulaciju modela i izvršenje koda iz modela. Kada je ova opcija izabarana , Simulink onemogućava kontrolu parametara bloka putem njihovih dijalog boksova, da bi spriječio da korisnik greškom ne modificira parametre blokova u modelu.

Simulink dozvoljava nadjahivanje (override) opcije **Inline parameters** za parametre čije su vrijednosti definirane sa varijablama u MATLAB radnom prostoru. Da specificira da takav parametar ostaje podesiv, treba specificirati parametar kao globalan u dijalog boksu **Model Parameter Configuration**. Da se prikaže dijalog, treba izabrati susjedni **Configure** taster. Da bi se podesio globalni parametar , promjeniti vrijednost odgovarajuće varijable u radnom prostoru i izabrati **Update Diagram** (Ctrl+D) iz **Edit** menija.

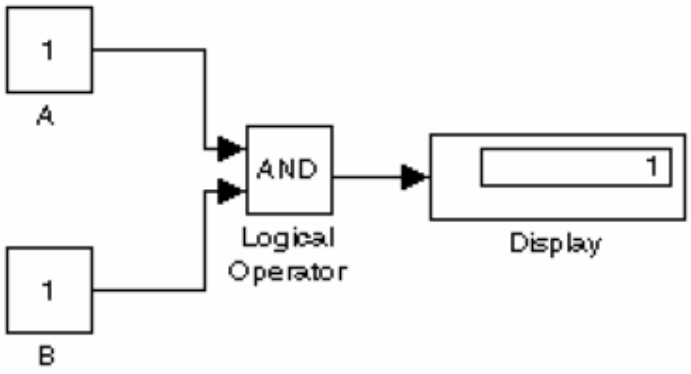
Optimizacije

Redukcija blokova (block reduction)

Zamjenjuje grupu blokova sa sintetiziranim blokom, i time ubrzava izvršenje modela.

Boolovi logički signali (boolean logic signals)

Prouzrokuje da blokovi koji prihvataju Boolove signale zahtjevaju Booleove signale. Ako je ova opcija isključena (**off**) , blokovi koji prihvataju ulaze tipa **boolean** takodjer prihvataju ulaze tipa **double**. Na primjer, posmatrajmo slijedeći model:



Ovaj model spaja signale tipa double na blok **Logical Operator**, koji prihvata ulaze tipa **boolean**. Ako opcija **Boolean logic signal** je uključena (**on**), ovaj model generiše signal greške kada se izvršava. Ako je pak **Boolean logic signal** opcija isključena (**off**) , ovaj model se izvršava bez greške.

Granjanje na uslovni ulaz (conditional input branch)

Ova optimizacija se primjenjuje na modele koji sadrže **Switch i Multiport Switch** blokove. Kada je omogućena, ova optimizacija izvršava samo blokove koji su potrebni da se izračunaju kontrolni ulazi i ulazi podataka izabrani od strane kontrolnog ulaza u svakom vremenskom koraku za svaki Switch i Multiple Switch blok u modelu. Slično, kod generisan iz modela sa RTW izvršava samo kod koji je potreban da se izračuna kontrolni ulaz i izabrani ulazi podataka. Ova optimizacija ubrzava simulaciju i izvršenje koda generisanog iz modela.

Na početku simulacije ili generacije koda, Simulink ispituje svaku stazu signala koji napaja ulaz podataka **switch** bloka da odredi dio staza koji može biti optimiziran. Ovaj optimizacioni dio staza je sada onaj dio staza signala koji se prostire unatrag od ovih ulaza podataka do prvog bloka koji je nevirtualni podsistem, i ima kontinualna ili diskretna stanja ili detektuje prolaske kroz nulu.

Simulink zatvara ovaj optimizabilni dio staza signala u nevidljivi **atomic** podsistem. Za vrijeme simulacije, ako ulaz podataka od **switch** nije izabran, Simulink izvršava samo neoptimizabilni dio staza signala koji napajaju ulaze. Ako je izabran ulaz podataka, Simulink izvršava obadva i optimizabilni i neoptimizabilni dio staza ulaznih signala.

Izbor parametara (parameter pooling)

Ova opcija se koristi za generisanje koda. Ako se ne vrši generisanje koda treba ostaviti ovu opciju uključenu.

Ponovno korištenje memorije za pohranjivanje signala (signal storage reuse)

Prouzrokuje da Simulink ponovno koristi memorijske bafere alocirane da se pohrane ulazni i izlazni signali bloka. Ako je ova opcija isključena (**off**), Simulink alocira poseban memorijski bafer za svaki izlaz bloka. Ovo može značajno povečati količinu potrebne memorije da se simuliraju veliki modeli, tako da treba izabrati ovu opciju samo onda kada treba da debagiramo model.

Naročito je potrebno da onemogućimo ovo ponovno korištenje memorije ako treba da :

- Debagiramo C-MEX S-funkciju
- Koristimo floating Scope ili blok displaya da pregledamo signale u modelu kojeg debagiramo.

Simulink će otvoriti dijalog greške ako je **Signal storage reuse** omogućen , i korisnik pokušava da koristi floating Scope ili blok displaya da prikaže signal čiji se bafer ponovo koristi.

Otkrivanje prolaska kroz nulu (zero crossing detection)

Omogućava otkrivanje prolaska kroz nulu za vrijeme simulacije modela sa promjenjlivim korakom. Za većinu modela, ovo ubrzava simulaciju omogučujući solveru da uzima veće vremenske korake. Ako model ima ekstremno velike dinamičke promjene, onemogućavanje ove opcije može ubrzati simulaciju ali može takodje smanjiti tačnost rezultata simulacije.

Korisnik može nadjahati ovu optimizaciju na nivou pojedinačnih blokova za slijedeće tipove blokova:

Abs	Integrator	Step
Backlash	MinMax	Switch
Dead Zone	Relay	Switch Case
Enable	Relational Operator	Trigger
Hit Crossing	Saturation	
If	Sign	

Na nadjaše detekciju prolaska kroz nulu za jednu instancu jednog od pobrojanih blokova, treba otvoriti dijalog boks parametara bloka i dečekirati opciju **Enable zero crossing detection**. Korisnik može omogućiti ili onemogućiti selektivno detekciju prolaska kroz nulu za ove blokove samo ako je detekcija globalno omogućena , tj. izabrana je **Zero-crossing optimization** na **Advanced** panelu u dijalog boksu **Simulation Parameters**.

Kontrola bloka verifikacije modela (model verification block control)

Ovaj parametar dozvoljava korisniku da omogući ili onemogući verifikaciju blokova modela u tekućem modelu bilo globalno ili lokalno. Treba izabrati jednu od slijedećih opcija:

- **Use local settings**

Omogućava ili onemogućava blokove bazirane na vrijednosti parametra **Enable Assertion** u svakom bloku. Ako je ovaj parametar omogučen (**on**) , blok je omogućen , inače blok je onemogućen.

- **Enable all**

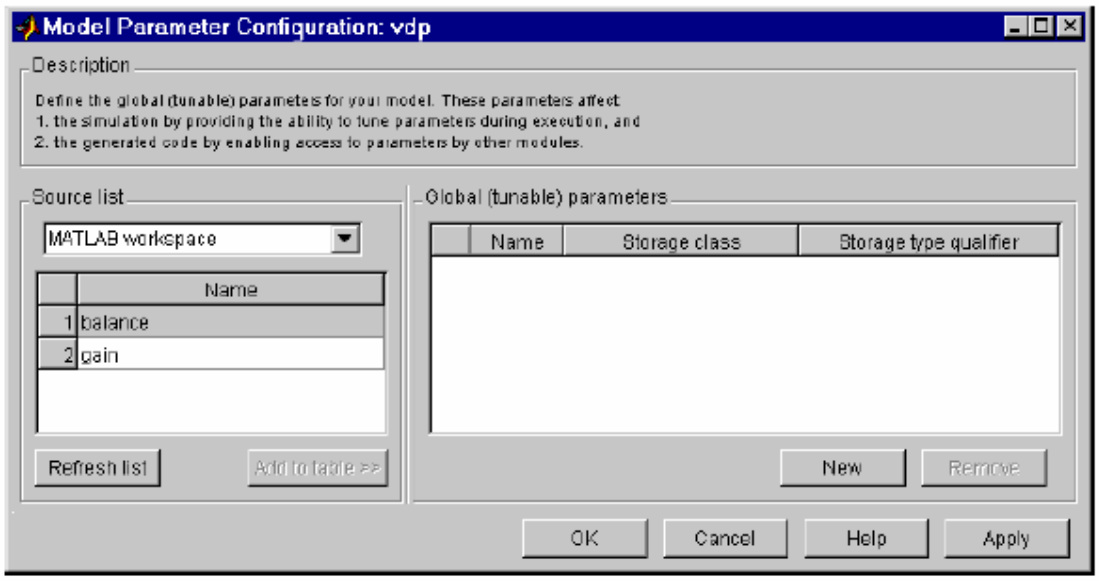
Omogućuje verifikaciju svih blokova u modelu bez obzira na setinge parametara **Enable Assertion**.

- **Disable all**

Onemogućuje verifikaciju svih blokova modela u modelu bez obzira na setinge parametara **Enable Assertion**.

Dijalog boks za konfigurisanje parametara modela

Dijalog boks **Model Parameter Configuration** omogućava korisniku da nadjahuje opciju **Inline parameters** za izabrane parametre.



Ovaj dijalog boks ima slijedeće kontrolne elemente :

Source llist - Prikazuje listu varijabli u radnom prostoru. Opcije su :

- **MATLAB workspace** – Izlistava sve varijable u MATLAB radnom prostoru koje imaju numeričke vrijednosti.

- **Referenced workspace variables** – Izlistava samo one varijable koje su referencirane od strane modela

Refresh list - Ažurira izvornu listu.

Add to table – Dodaje varijable izabrane u izvornoj listi u susjednu tabelu podesivih parametara

New – Definira novi parametar i dodaje ga listi podesivih parametara

Storage Class - Koristi se kod generacije koda u RTW

Karakteristike proizvodnog hardwarea (production hardware characteristics)

Ovo podešenje je namjenjeno za korištenje kod modeliranja, simulacije i generisanja koda za digitalne sisteme. Omogućava korisniku da specificira veličine tipova podataka koli su podržani od sistema koji se modelira. Simulink koristi ovu informaciju da automatizira izbor tipova podataka za izlazne signale kod nekih blokova, napr. **Product** i **Gain** blokova.

Treba izabrati jedno od slijedećih podešenja iz liste:

- **Microprocessor**

  Specificira da taj model predstavlja mikroprocesorski bazirani digitalni sistem čije dužine cjelobrojnih riječi odgovaraju C tipovima podataka navedenih u kontrolnom polju ispod **Production hardware characteristics** liste :

  Ulaz za svaki C tip podatka specificira dužinu u bitima od odgovarajuće mikroprocesorske riječi. Korisnik može promjeniti dužinu izabirući tip podatka i unoseći novu dužinu u susjedno **Value** polje .

- **Unconstrained integer size**

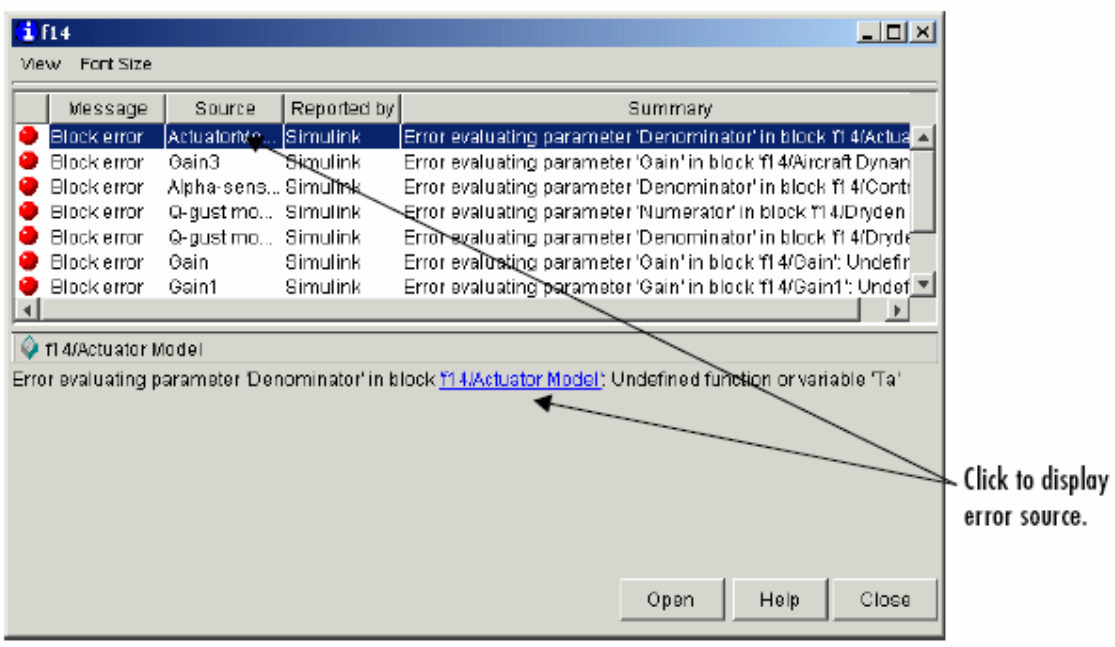
  Specificira da hardwareska implementacija modeliranog sistema ne postavlja ograničenja na izbor tipova cjelobrojnih podataka koji se koriste da izvrše izračunavanja. Ovo može biti slučaj , naprimjer, ako proizvodni hardware je namjenjen da bude kastomizirano integralno kolo.

Dijagnoza grešaka simulacije

Ako se pojave greške za vrijeme simulacije, Simulink zadržava simulaciju, otvara podsisteme koji su prouzrokovali grešku (ako je neophodno) , i prikazuje greške u **Simulation Diagnostics Viewer**u (prozoru dijagnostike simulacije).

Simulation Diagnostic Viewer

Prozor (viewer) se sastoji od panela sumarnih grešaka (Error Summary pane) i panela poruka grešaka (Error message pane).



Panel sumarnih grešaka (Error Summary pane)

Gornji panel izlistava greške koje su prouzrokovale da Simulink okonča simulaciju. Panel prikazuje slijedeće informacije o svakoj grešci:

Message – Tip greške (napr. greška bloka, upozorenje, log)

Reported by – Komponenta koja je izvjestila o grešci (naprimjer : Simulink, Stateflow , RTW , itd .)

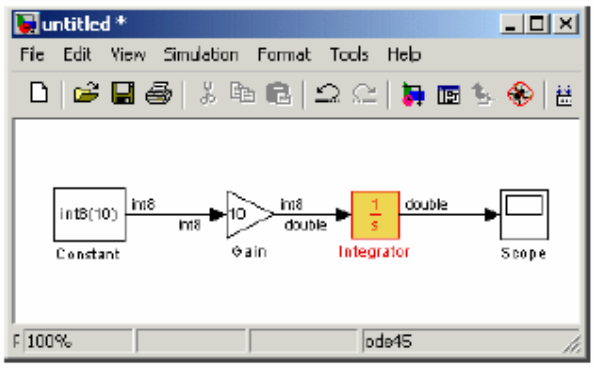
Source - Ime elementa modela koji je prouzrokovao grešku.

Summary - Poruka greške , skraćena da stane u kolonu

Panel poruke greške (error message pane)

Ovaj donji panel inicijalno sadrži sadržaj poruke prve greške izlistane u gornjem panelu. Možemo pokazati sadržaje drugih poruka klikanjem na njihove ulaze u gornjem panelu.

Osim prikazivanja Diagnostic Viewera, Simulink takodjer otvara (ako je neophodno) , podsistem koji sadrži izvor prve greške i naglašava izvor:

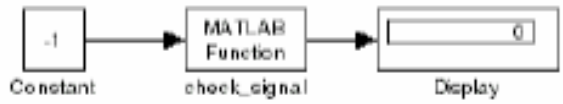


Možemo pokazati izvore drugih grešaka klikanjem na poruku greške u gornjem panelu.

Kreiranje poruka greški iz kastom simulacija

Simulink dijagnostički Viewer prikazuje izlaz bilo koje instance greške MATLAB funkcije izvršene u toku simulacije, uključujući instance koje poziva blok ili model tipa callbacks ili S-funkcije koje je korisnik kreirao ili koje su izvršene od strane MATLAB **Function** bloka. Na taj način, korisnik može koristiti MATLAB funkciju greške u callbackovima i S-funkcijama ili u MATLAB **Function** bloku da kreira poruke greške simulacije specifične za njegovu aplikaciju.

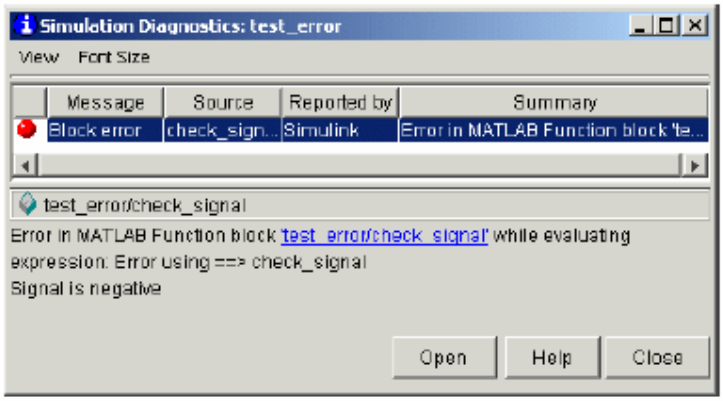
Naprimjer, u slijedećem modelu:



MATLAB Fcn blok poziva slijedeću funkciju:

```
function y=check_signal(x)
  if x<0
    error('Signal is negative.');
  else
    y=x;
  end
```

Izvršenje ovog modela prikazuje poruku greške u Simulation Diagnostic Vieweru:



Uključenje hyperlinkova u poruke greške

Korisnik može uključiti hyperlinkove u blokove, tekst fileove i direktorije.

Da se uključi hyperlink u blok, stazu (path) ili direktorij, treba uključiti stazu detalja zatvorenu sa znacima navoda u poruku greške , kao naprimjer:

```
• error ('Error evaluating parameter in block "mymodel/Mu"')
```

će prikazati tekst hyperlink na blok Mu u tekućem modelu u poruci greške. Klikanjem na hyperlink prikazaće se blok u prozoru modela.

```
• error ('Error reading data from"c:/work/test.data"')
```

prikazaće tekst hyperlink na file **test.data** u poruci greške. Klikanjem na link pokazaće se file u izabranom MATLAB editoru.

```
• error ('Could not find data in directory "c:/work"')
```

prikazaće tekst hyperlink na **c:/work** direktorij. Klikanjem na link otvoriće se komandni prozor sistema (shell – jezgro) i setovati svoj direktorij na **c:/work** .

Poboljšanje tačnosti i performanse simulacije

Poboljšanje performanse i tačnosti simulacije se može ostvariti na mnogo načina , uključivo i kroz dizajn modela i izbor simulacionih parametara.

Solveri upravljaju sa većinom simulacija tačno i efikasno i sa njihovim default parametrima. Ipak, neki modeli daju bolje rezultate ako podesimo solver i simulacione parametre. Takodjer, ako znamo informacije o ponašanju modela, rezultati simulacije se mogu poboljšati ako obezbjedimo ovu informaciju solveru.

Ubrzanje simulacije

Spora brzina simulacije ima mnogo razloga. Neki od njih su:

- Model uključuje MATLAB **Fcn** blok. Kada model uključuje ovaj blok, MATLAB interpreter se poziva u svakom vremenskom koraku, i time drastično usporava simulaciju. Zbog toga treba koristiti ugradjeni Fcn blok ili Math Function blok kada je to god moguće.

- Korisnikov model uključuje M-file S-funkciju. M-file S-funkcije takodjer prouzrokuju da se poziva MATLAB interpreter u svakom vremenskom koraku. Treba razmatrati ili da se konvertuje S-funkcija u podsistem ili u C-MEX S-funkciju.

- Korisnički model uključuje Memory blok. Korištenje Memory bloka prouzrokuje da solveri varijabilnog reda (ode15s i ode113) se resetuju na red 1 u svakom vremenskom koraku.

- Maksimalni korak je suviše mali. Ako promjenimo maksimalnu velićinu koraka, treba ponovno pokušati simulaciju sa default velićinom (auto).

- Korisnik traži suviše veliku tačnost. Default relativna tolerancija (0.1% tačnost), je obićno dovoljna. Za modele sa stanjima koja idu na nulu, ako je parametar absolutne tolerancije isuviše mali, simulacija može praviti isuviše mnogo koraka oko vrijednosti stanja bliskim nuli.

- Vrijeme trajanja simulacije može biti isuviše dugačko. Treba smanjiti taj interval

- Problem može biti u krutosti sistema a mi koristimo solver koji nije dovoljno krut. Treba pokušati koristiti ode15s.

- Model koristi vremena uzorkovanja koja nisu multipli jedan drugoga. Mješanje vremena samplovanja koji nisu multipli jedan drugoga prouzrokovaće da solver uzima vrlo male korake da bi obezbjedio pogadjanja svih vremena samplovanja.

- Model sadrži algebarsku konturu. Riješenja za algebarsku konturu se iterativno računaju u svakom vremenskom koraku. Zbog toga, oni značajno degradiraju performansu.

- Model snadbjeva signale iz Random number bloka u Integratorski blok. Za kontinualne sisteme , koristiti Band-Limited White noise block u biblioteci izvora.

Poboljšanje tačnosti simulacije

Da bi se provjerila tačnost simulacije, treba izvršiti simulaciju u razumnom vremenskom opsegu. Nakon toga, treba ili reducirati relativnu toleranciju na 1e-4 (default je 1e-3) ili

reducirati absolutno toleranciju i izvršiti ponovno simulaciju. Porediti rezultate obadvije simulacije. Ako se rezultati značajnije ne razlikuju, možemo se osjećati sigurnim da je rješenje konvergiralo ka konačnim vrijednostima.

Ako simulacija značajno odstupa od očekivane na svom startu, reducirati veličinu početnog koraka da osiguramo da simulacija ne prekoračuje preko značajnog ponašanja.

Ako simulacioni rezultati postanu vremenom nestabilni:

- Možda se radi o nestabilnom sistemu čiji je to model
- Ako koristimo ode15s, možda moramo ograničiti maksimalni red na 2 (maksimalni red za koji je solver A-stabilan), ili pokušati koristiti ode23s solver
- Za model koji ima stanja čije se vrijednosti približuju nuli, ako je parametar absolutne tolerancije isuviše velik, simulacija uzima isuviše koraka oko oblasti gdje su vrijednosti stanja bliske nuli.
- Ako reduciranje absolutnih tolerancija ne poboljšava značajnije tačnost, reducirati veličinu parametra relativne tolerancije, da se reducira prihvatljiva greška i prisili manja velićina koraka i više koraka.

Programsko izvršenje simulacije

Unošenje simulacionih komandi u MATLAB komandni prozor ili iz M-filea omogućava nam da izvršavamo simulaciju bez prisustva korisnika.

Možemo izvršiti Monte Carlo analizu slučajnom promjenom parametara i izvršiti simulacije u konturi. Možemo izvršiti simulaciju sa komandne linije koristeći **sim** komandu ili **set_param** komandu.

Analiza rezultata simulacije

Vizuelizacija izlaznih trajektorija

Izlazne trajektorije iz Simulinka se mogu crtati koristeći jedan od tri metoda:

- Vodjenje signala u Scope ili XY blok
- Upisivanje izlaza u povratne varijable i korištenje MATLAB komandi za iscrtavanje
- Upisivanje izlaza u radni prostor koristeći To Workspace blokova i iscrtavanje rezultata koristeći MATLAB komande za crtanje

Koristeći povratne varijable

Vračajući vrijeme i izlazne historije , korisnik može koristiti MATLAB komande za crtanje da prikaže i anotira izlazne trajekorije.

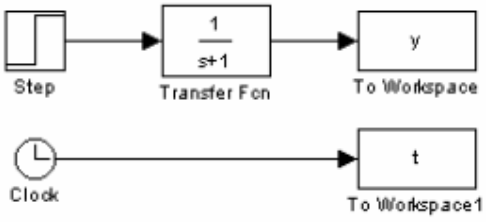
Blok koji je labeliran Out je izlazni blok iz biblioteke **Signals&Systems** . Izlazna trajektorija, **yout** se vraća od strane integracionog solvera.

Možemo izvršiti ovu simulaciju i iz menija **Simulation** specificirajući varijable za vrijeme, izlaz, i stanja na **Workspace I/O** strani dijalog boksa **Simulation parameters.** Nakon toga možemo iscrtati ove rezultate koristeći komandu **:**

plot (tout, yout)

Koristeći To Workspace blok

Blok To Workspace se može koristiti da vrati izlazne trajektorije u MATLAB radni prostor. Model koji slijedi ilustrira ovaj način rada :



Varijable **y** i **t** se pojavljuju u radnom prostoru kada je simulacija kompletna. Pohranjuje se vremenski vektor sa slanjem iz Clock bloka u To Workspace blok. Možemo takodjer prikupiti vremenski vektor unošenjem imena varijable za vrijeme u panel **Workspace I/O** u **Simulation Parameters** dijalog boksu, za menu upravljane simulacije, ili vračajući ga koristeći **sim** komandu.

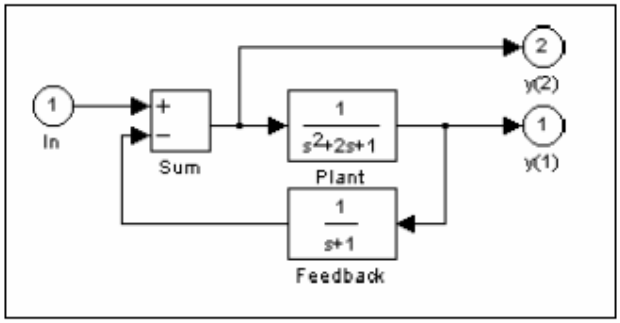
Blok To Workspace može prihvatiti polje ulaza , sa svakom trajektorijom ulaznog elementa pohranjenom u rezultirajuću varijablu radnog prostora.

Linearizacija modela

Simulink obezbjedjuje funkcije **linmod** i **dlinmod** da dobije linearne modele u formi matrica u prostoru stanja A,B,C i D. Matrice prostora stanja opisuju linearnu ulazno-izlaznu relaciju kao :

$$\begin{aligned} x &= Ax + Bu \\ y &= Cx + Du \end{aligned}$$

gdje x, u i y su stanje, ulazni i izlazni vektori respektivno. Na primjer, slijedeći model se zove **lmod**:



Da dobije linearni model iz ovog Simulink sistema, unjeti ovu komandu.

```
[A,B,C,D] = linmod('lmod')
A =
    -2    -1    -1
     1     0     0
     0     1    -1
B =
     1
     0
     0
C =
     0     1     0
     0     0    -1
D =
     0
     1
```

Ulazi i izlazi moraju biti definirani koristeći Inport i Outport blokove iz biblioteke **Signals&Systems**. Blokovi izvora i ponora (sourcs i sink) ne djeluju kao ulazi i izlazi. Inport blokovi se mogu koristiti u sprezi sa izvornim blokovima, koristeći **Sum** blok. Kada se podatci u formi prostora stanja konvertuju u LTI objekat, možemo primjeniti funkcije u CST toolboksu (Control systems toolbox) , za dalju analizu , kao:

- Konverzija u LTI objekat :

```
sys = ss(A,B,C,D);
```

- Bodeov plot amplitude i faze :

```
bode(A,B,C,D) or bode(sys)
```

- Linearizirani vremenski odziv:

```
step(A,B,C,D) or step(sys)
impulse(A,B,C,D) or impulse(sys)
lsim(A,B,C,D,u,t) or lsim(sys,u,t)
```

Možemo koristiti i druge funkcije u CST i RCT (robust control toolbox) za dizajn linearnih sistema upravljanja.

Kada je model nelinearan, radna tačka se može izabrati kod koje treba izvući linearizirani model. Nelinearni model je takodjer osjetljiv na velićine perturbacija sa kojima je model izveden. Ove perturbacije moraju biti izabrane tako da balansiraju kompromis izmedju grešaka skraćenja i grešaka zaokruženja. Dodatni argumenti za linmod specificiraju radnu tačku i perturbacione tačke.

```
[A,B,C,D] = linmod('sys', x, u, pert, xpert, upert)
```

Za diskretne sisteme ili mješane kontinualne i diskretne sisteme, koristiti funkciju **dlinmod** za linearizaciju. Ovo ima istu pozivajuću sintaksu kao **linmod** izuzev što drugi argument sa desne strane mora sadržavati vrijeme samplovanja kod kojeg treba izvesti linearizaciju.

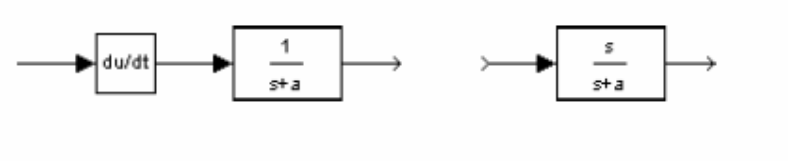
Korištenje linmod funkcije da linearizira model koji sadrži blokove izvoda i transportnog kašnjenja može biti problematično. Prije linearizacije, treba zamjeniti ove blokove sa specijalno dizajniranim blokovima koji izbjegavaju ove probleme. Ovi blokovi su u Simulinkovoj dodatnoj biblioteci (extra library) , u okviru linearizacione podbiblioteke (**linearization sublibrary**). :

- za derivative blok, treba koristiti **Switched derivative** za linearizaciju
- za Transport delay blok, treba koristiti **Switched transport delay** blok za linearizaciju.

Kada koristimo blok izvoda (derivative) , možemo takodjer pokušati da inkorporiramo derivativni član i u drugim blokovima. Na primjer, ako imamo derivative blok u seriji sa **Transfer Fcn** blokom, bolje je implementirati (mada to nije uvjek i moguće) sa jednom prenosnom funkcijom oblika:

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

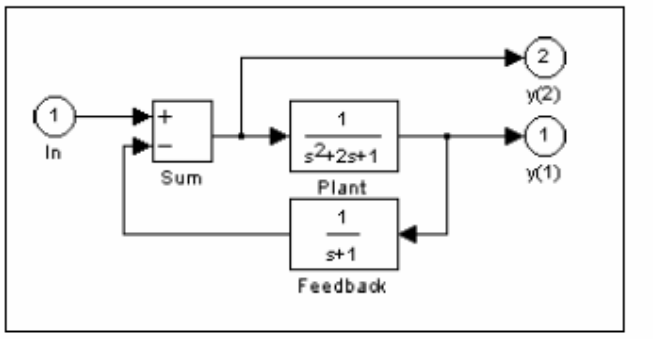
U ovom primjeru , blokovi na lijevoj strani slike se mogu zamjeniti sa jednim blokom na desnoj strani:



Nalaženje tačaka stacionarnog (steady – state) stanja

Simulinkova **trim** funkcija koristi Simulinkov model da odredi tačke stacionarnog (steady-state) stanja dinamičkog sistema koji zadovoljava ulaz, izlaz i uslove stanja koje je korisnik specificirao.

Razmatrajmo , naprimjer, ovaj model, nazvan **lmod**:



Možemo koristiti **trim** funkciju da nadjemo vrijednosti ulaza i stanja koja setuju obadva izlaza na 1. Prvo, napravićemo početne procjene za varijable stanja (**x**) i ulazne vrijednosti (**u**) , a zatim setovati željenu vrijednost za izlaz (y).

```
x = [0; 0; 0];
u = 0;
y = [1; 1];
```

Koristiti indeksne varijable da indiciramo koje varijable su fiksne a koje mogu varirati.

```
ix = [];      % Don't fix any of the states
iu = [];      % Don't fix the input
iy = [1;2];   % Fix both output 1 and output 2
```

Pozivajući **trim** komandu vratiće nam rješenje. Rezultati koje dobijemo se mogu razlikovati zbog grešaka zaokruženja.

```
[x,u,y,dx] = trim('lmod',x,u,y,ix,iu,iy)

x =
    0.0000
    1.0000
    1.0000
u =
    2
y =
    1.0000

    1.0000
dx =
    1.0e 015 *
     -0.2220
     -0.0227
      0.3331
```

Primjetimo da možda ne bude rješenja za probleme tačaka ravnoteže. Ako je to slučaj **trim** će vratiti rješenje koje minimizira maksimalno odstupanje od željenog rezultata nakon što prvo pokuša da postavi izvode na nulu.

Kreiranje maskiranih podsistema

O maskama

Maska je kastomizirani korisnički interfejs za podsistem koja krije sadržaj podsistema, tako da se on pojavljuje korisniku kao atomic blok (jedan jedinstveni nedjeljiv blok), sa svojom ikonom i dijalog boksom za parametre.

Simulink Mask Editor omogućava korisniku da kreira masku za bilo koji podsistem. Maskiranje podsistema omogućava :

- da zamjenimo dijalog boks za parametre podsistema i za njegove blokove koje sadrži , sa samo jednim dijalog boksom i njegovim vlastitim opisom bloka, promta za parametere i help tekstom
- da zamjeni standardnu ikonu podsistema sa kastomiziranom ikonom koja odslikava njegovu namjenu.
- spriječi neželjene izmjene podsistema krijući njegov sadržaj iza maske.
- kreira kastom blok da enkapsulira blok dijagram koji definira ponašanje bloka u maskiranom podsistemu i zatim postavi maskirani podsistem u biblioteku.

Osobine maske

Maske mogu uključiti jednu od slijedećih osobina:

Maska ikone

Maska ikone zamjenjuje standardnu ikonu podsistema , tj. ona se pojavljuje u blok dijagramu umjesto standardne ikone za blok podsistema.

Parametri maske

Simulink omogućava korisniku da definiše set parametara podesivih od korisnika za maskirane podsisteme. Simulink pohranjuje vrijednost parametara u radni prostor maske kao vrijednost varijable čije ime specificira korisnik.

Ove pridružene varijable omogućavaju korisniku da poveže parametre maske sa specifičnim parametrima blokova unutar maskiranog podsistema (interne parametre) tako da postavljanje parametara maske postavlja pridružene parametre bloka.

Dijalog boks parametara maske

Dijalog boks maske parametara sadrži kontrolne elemente koji omogućuju korisniku da postavi vrijednosti prametara maski i time i vrijednosti bilo kojih internih parametara povezanih sa parametrima maski.

Dijalog boks parametara maske zamjenjuje standardni dijalog boks podsistema, tj. klikanjem na ikonu maskiranog podsistema prouzrokovaće da se pojavi dijalog boks maske umjesto standardnog dijalog boksa parametara bloka podsistema.

Inicijalizacioni kod maske

Inicijalizacioni kod je MATLAB kod koji specificira korisnik i Simulink izvršava da inicijalizira maskirani podsistem na početku simulacije. Korisnik može koristiti inicijalizacioni kod da postavi inicijalne vrijednosti parametara maskiranog podsistema.

Maskirani radni prostor

Simulink pridružuje MATLAB radni prostor sa svakim maskiranim podsistemom koji korisnik kreira. Simulink pohranjuje tekuče vrijednosti parametara podsistema u radni prostor kao i bilo koje varijable kreirane od strane inicijalizacionog koda bloka i callback parametara. Korisnik može koristiti varijable modela i radnog prostora maske da inicijalizira maskirani podsistem i da setuje vrijednosti blokova unutar maskiranog podsistema, pod slijedećim pravilima:

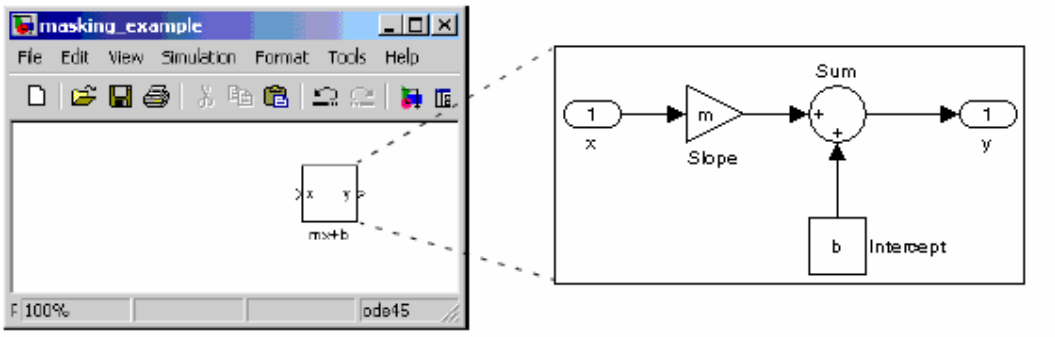
- Izraz blok parametra može se odnositi samo na varijable definirane u podsistemu ili ugnjezdenom (nested) podsistemu koji sadrži blok ili u radnom prostoru modela.

- Validna referenca na varijablu definiranu na više od jednog nivoa u hijerarhiji modela razlučuje se na lokalnu definiciju.
  Na primjer, predpostavimo da model **M** sadrži maskirani podsistem **A** , koji sadrži maskirani podsistem **B**. Nadalje, predpostavimo da **B** referira na varijablu **x** koja egzistira i u **A** i **M** radnim prostorima. U ovom slučaju, referenca se razlučuje na vrijednost u A-ovom radnom prostoru.

- Inicijalizacioni kod maskiranog podsistema može referencirati samo varijable u svom lokalnom radnom prostoru.

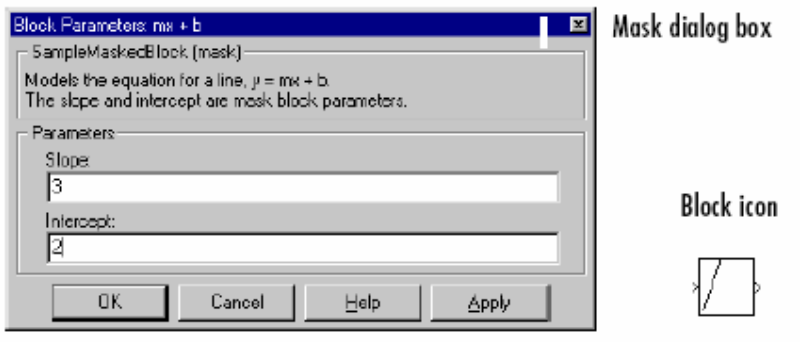
Primjer maskiranog podsistema

Ovaj jednostavni podsistem modelira jednačinu za liniju, y = mx + b.



Normalno, mi dvaput kliknemo na blok podsistema, otvori se blok podsistema, prikazujući svoje blokove u posebnom prozoru.

Ovaj primjer kreira kastom dijalog boks i ikonu za podsistem. Dijalog boks sadrži prompt ove i za nagib i za konstantu b. Nakon što kreiramo masku, treba dvaput kliknuti na blok podsistema da se otvori maska dijalog boksa. Maska diajlog boksa i ikona izgledaju kao na slijedećoj slici:



Korisnik unosi vrijednosti nagiba (**slope**) i konstante b (**intercept**) u dijalog boksu maske. Simulink čini ove vrijednosti raspoloživim svim blokovima u podsistemu.

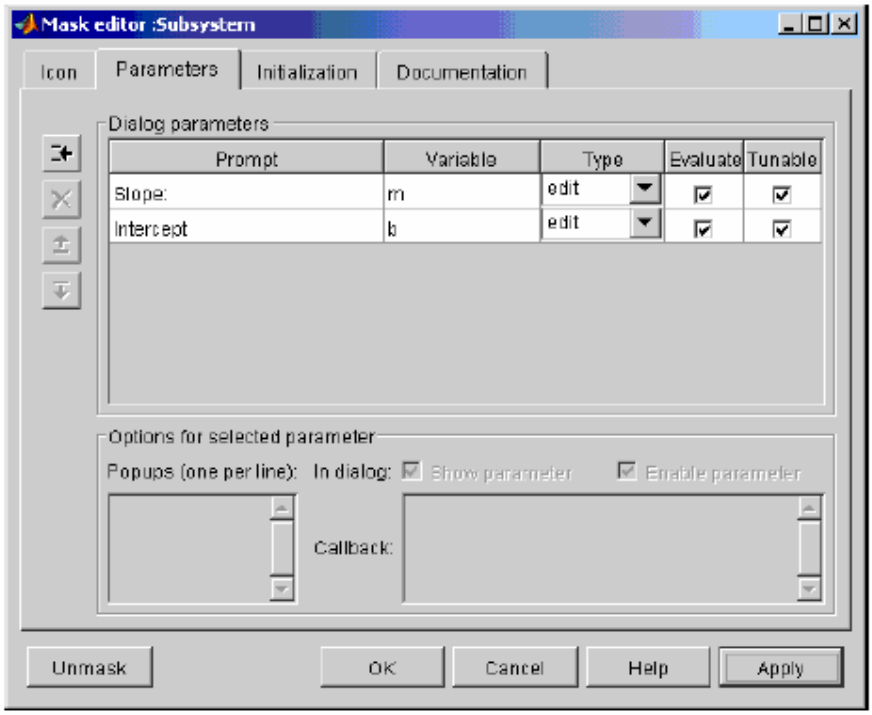
Maskiranjem ovoga podsistema kreira se samo sadrživa funkcionalna jedinica sa svojim vlastitim aplikaciono specifičnim parametrima, **Slope i Intercept**. Maska mapira ove **mask parameters** na generičke parametra od blokova koji su uključeni u podsistem. Kompleksnost podsistema je enkapsulirana sa novim interfejsom koji ima izgled ugradjenog Simulink bloka.

Da bi kreirali masku za ovaj podsistem, korisnik treba da:

- specificira promptove za masku dijalog boksa parametara. U ovom primjeru, dijalog boks maske ima promptove za slope i intercept.
- specificira ime varijable korištene za pohranjivanje vrijednosti svakog parametra
- unese dokumentaciju bloka, koja se sastoji od opisa bloka i teksta za help file.
- specificira komande crtanja koje kreiraju ikonu bloka
- specificira komande koje obezbjedjuju varijable potrebne od strane komandi za crtanje

Kreiranje promptova za dijalog boks maske

Da bi se kreirala maska za ovaj podsistem, izabrati blok podsistema i zatim izabrati **Mask Subsystem** iz **Edit** menija.
Ovaj primjer primarno koristi panel **Parameteres** od editora maske da kreira dijalog boks maskiranog podsistema.

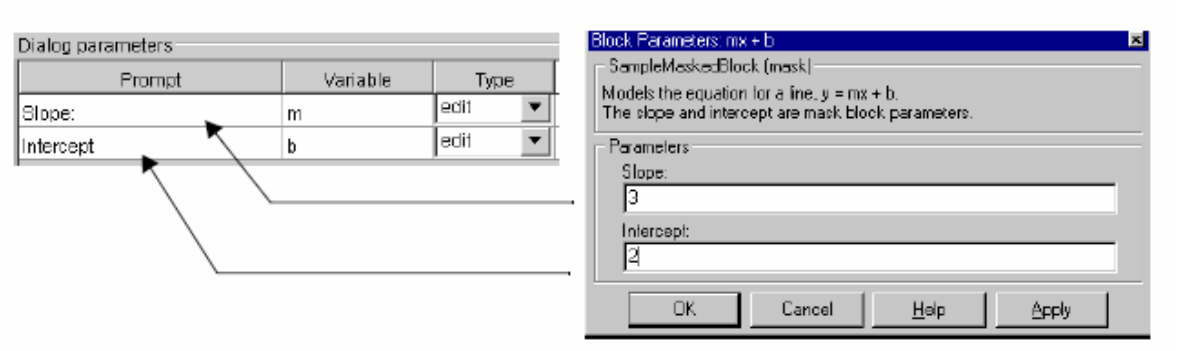


Editor maske omogućava korisniku da specificira slijedeće atribute parmetara maske:

- prompt , tekst labela koja opisuje parametar
- kontrolni tip, i stil korisničkog interfejsa koji odredjuje kako vrijednosti parametara su unesene i izabrane
- varijabla, ime varijable koja pohranjuje vrijednost parametra

Slope i intercept se definirani kao edit kontrolni elementi. To znači da korisnik unosi vrijednosti u edit polja u mask dijalog boksu. Ove vrijednosti su pohranjene u varijable u radnom prostoru maske (mask workspace).

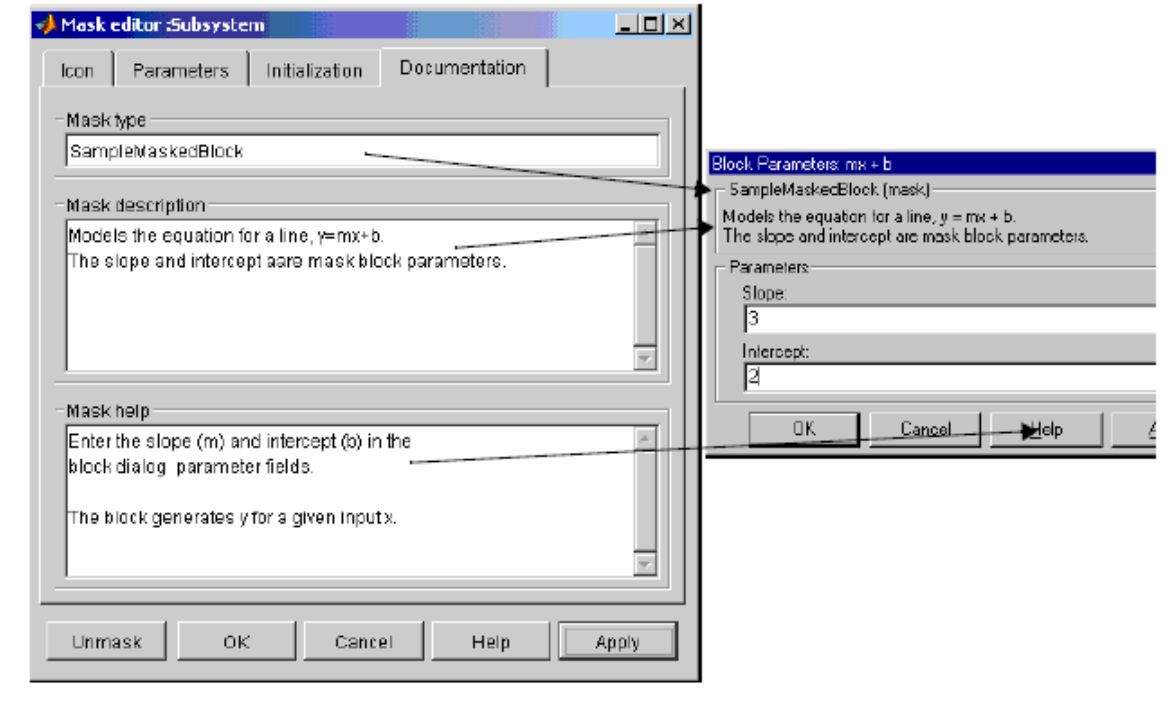
Naredna slika pokazuje kako parametar nagiba (slope) definicije u mask editoru se mapiraju na parametre stvarne maske dijalog boksa:



Nakon što kreiramo parametre maske za nagib i intercept (b) , kliknuti na OK taster. Nakon toga dvaput kliknuti na blok podsistema, da se otvori novo konstruirani dijalog boks. Unjeti 3 za **Slope** (nagib) i 2 za parametar **Intercept**.

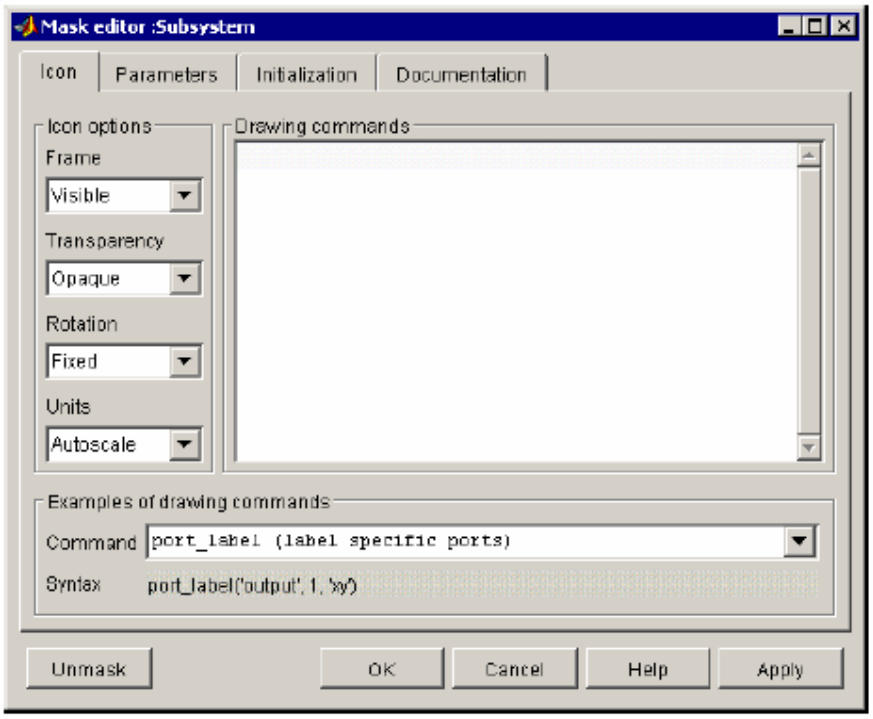
Kreiranje opisa bloka i help teksta

Tip maske, opis bloka, i tekst helpa su definirani na **Documentation** panelu. Za naš jednostavni primjer maskiranog podsistema, panel će izgledati kao na slijedećoj slici:



Editor maske

Editor maske omogućava korisniku da kreira ili editira masku podsistema. Da se otvori editor Maske, izabrati ikonu bloka podsistema i zatim izabrati **Edit Mask** iz **Edit** menija prozora modela koji sadrži blok podsistema. Pojaviće se Mask editor kao na slici:



Mask Editor sadrži set panela , od kojih svaki omogućava korisniku da definiše karakteristike maske:

- **Icon** panel omogućava mu de definiše ikonu bloka
- **Parameters** panel omogućava da definiše i opiše promptove za parametre diajlog boksa maske i imenuje varijable pridružene sa parametrima
- **Initialization** panel omogućava specificiranje inicijalizacionih komandi
- **Documentation** panel omogućava definiranje tipa maske i specificiranje opisa bloka i helpa za blok,

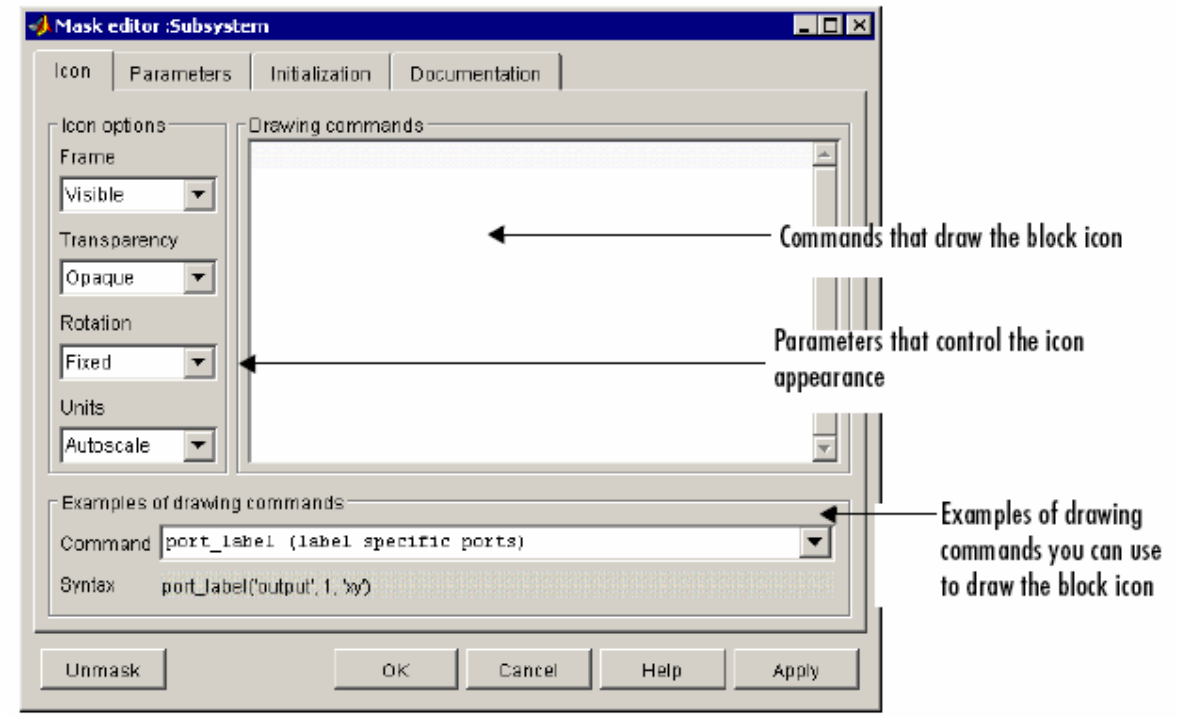
Pet dugmadi se pojavljuje na dnu Mask Editora:

- **Unmask** taster deaktivira masku i zatvara Mask Editor. Informacija o maski je zadržana tako da se maska može ponovno aktivirati. Da bi se reaktivirala maska, izabrati blok i onda **Create Mask**. Otvora se **Mask Editor** , prikazujući prethodna podešenja. Informacije neaktivne maske se odbacuju kada se model zatvori i više se neće moći oporaviti.
- **OK** taster se primjenjuje na podešenja u svim panelima i zatvara Mask Editor.
- **Cancel** taster zatvara Mask Editor bez primjene bilo kojih promjena .
- **Help** taster prikazuje ovaj help tekst
- **Apply** taster kreira ili mjenja masku koristeći informacije koje se pojavljuju na svim panelima za masku. Mask Editor će ostati otvoren.

Da bi se vidjeo sistem i pod maskom bez da se demaskira, treba izabrati blok podsistema, zatim izabrati **Look under Mask** iz **Edit** menija.

Panel Ikone

Panel **Icon** u Editoru maske omogućava korisniku da kreira ikone koje mogu sadržavati opisni tekst , jednačine stanja, slike, i grafike.



Panel sadrži slijedeće komande:

Komande za crtanje

Ovo polje dozvoljava da unesemo komande koje crtaju ikonu bloka. Simulink obezbjedjuje set komandi koje mogu prikazati tekst, jedan ili više plotova, ili prikazati prenosnu funkciju. Ove komande trebaju biti korištene da se nacrta ikona. Simulink izvršava komande za crtanje u redoslijedu u kojem se pojavljuju u ovom polju. Komande za crtanje imaju pristup svim varijablama u radnom prostoru maske.

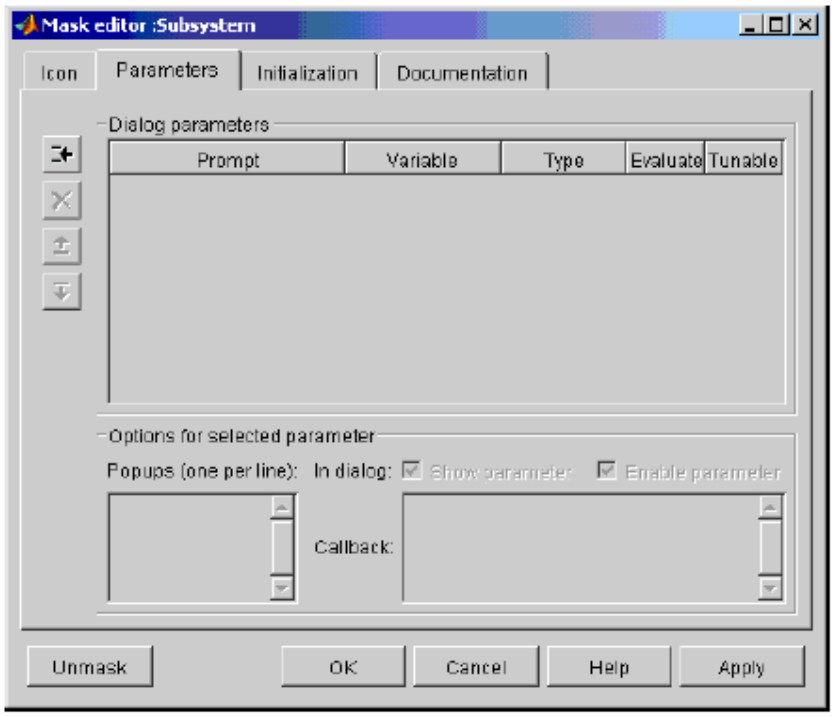
Naredni primjer demonstrira kako kreirati ikonu za mx + b primjer maskiranog podsistema :

```
pos = get_param(gcb, 'Position');
width = pos(3)   pos(1); height = pos(4)   pos(2);
x = [0, width];
if (m >= 0), y = [0, (m*width)]; end
if (m < 0),  y = [height, (height + (m*width))]; end
```

Ove inicijalizacione komande definiraju podatke koji omogućuju komandama crtanja da proizvedu tačnu ikonu bez obzira na oblik bloka. Komanda koja generiše ovu ikonu je plot (x, y).

Parametarski panel

Parameters panel dozvoljava korisniku da kreira i modificira parametre maskiranog podsistema (maskirane parametre) koji odredjuju ponašanje maskiranog podsistema



Panel **Parameters** sadrži slijedeće elemente:

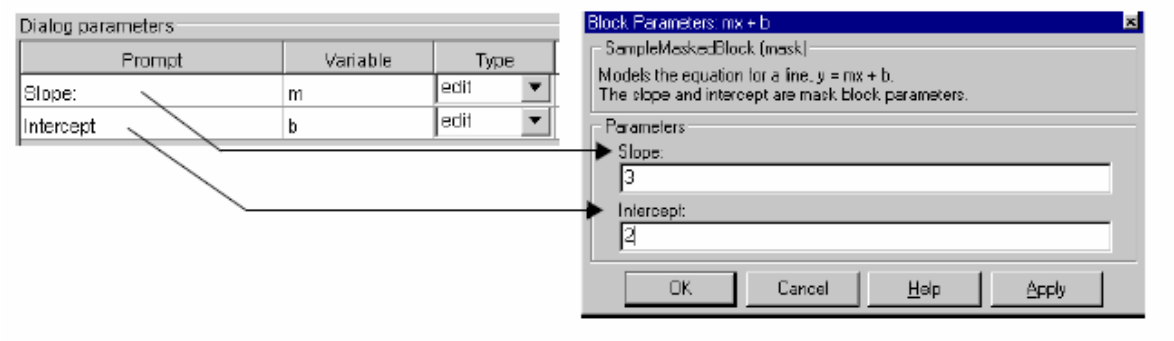
- **Dialog parameters** panel
  Izlistava parametre maske u tabularnoj formi. Svaki red prikazuje glavne atribute jednog od parametara maske

Prompt - tekst koji identificira parametar na dijalog boksu maskiranog podsistema

Variable - Ime varijable koja pohranjuje vrijednost parametra u radni prostor maske

Type - Tip kontrolnog elementa koji se koristi da editira vrijednost parametra.

Evaluate – Ako je chekiran , ova opcija će prouzrokovati da Simulink evaluira izraz unesen od strane korisnika prije doznačavanja varijable.



- **Options for selected parameter**

Ovaj panel omogućava korisniku da setuje dodatne opcije za parametar koji je izabran u tabeli **Dialog parameters.**

Show parameter – Izabrani parametar će se pojaviti u dijalog boksu parametra maskiranog bloka, samo ako je ova opcija čekirana.

Enable parameter – Dečekiranjem ove opcije za izabrani parametar posivi (onemogući) prompt i onemogućava njegovo editiranje. To znači da korisnik ne može postaviti vrijednost ovog parametra.

Popups – Ovo polje je samo omogućeno ako edit kontrola za izabrani parametar je iskočila (pop-up). Unjeti vrijednost pop-up kontrole u ovo polje, svaki na posebnu liniju.

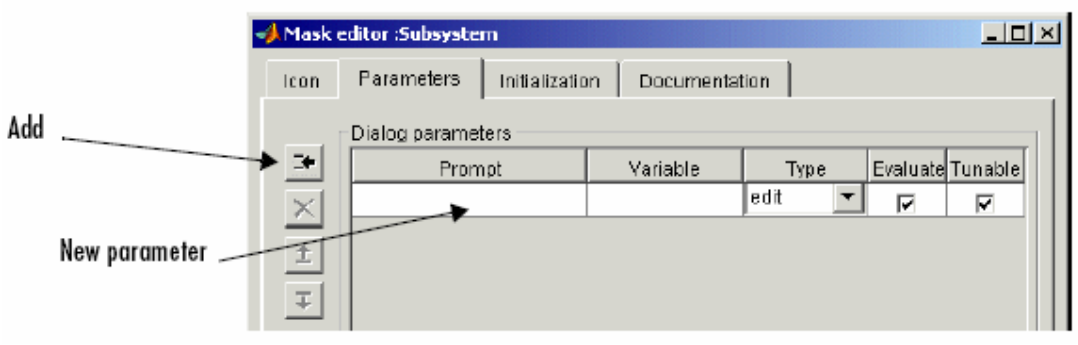
Callback – Unjeti MATLAB kod, koji želimo da Simulink izvrši kada korisnik editira izabrani parametar. Ako callback treba vrijednost parametra maske, on je može dobiti koristeći get_param , tj. :

```
if str2num(get_param(gcb, 'g'))<0
   error('Gain is negative.')
end
```

Parametarski tasteri

Na **Parameters** panelu pojaviće se slijedeći tasteri i kontrolni elementi :

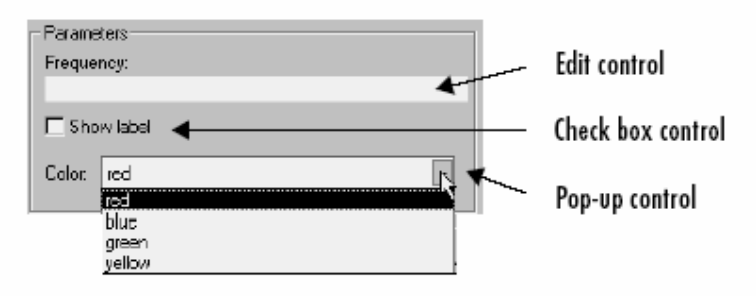
Add buttons – Dodaje parametar na listu prametara maske. Novo kreirani parametar će se pojaviti u susjednoj tabeli **Dialog parameters** :



Kontrolni tipovi

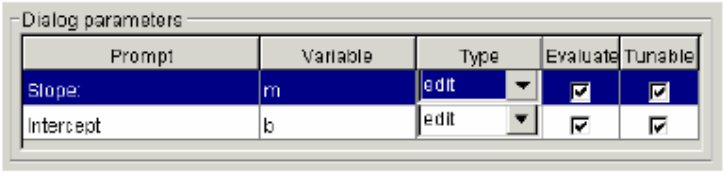
Simulink omogućuje korisniku da izabere kako su vrijednosti parametara unesene i izabrane. Možemo kreirati tri stila kontrolnih elemenata: edit polja, ček boksove i pop-up kontrolne elemente.

Naprimjer, naredna slika pokazuje zonu parametara dijalog boksa maske koja koristi sva tri stila kontrolnih elemenata:



Edit kontrolni element

Polje edit omogućava korisniku da unese vrijednost parametra ukucavanjem u polje, kao na slijedećoj slici:

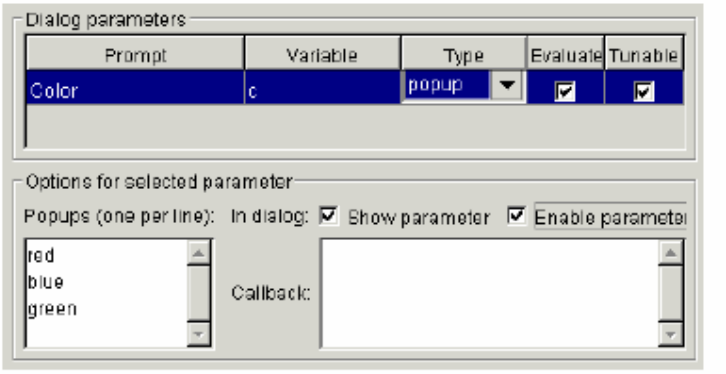


Vrijednost varijable pridružene sa parametrom je odredjena sa **Evaluate** opcijom.

Evaluate	Vrijednost
On	Rezultat evaluacije izraza se unosi u polje
Off	Stvarna vrijednost stringa se unosi u polje

Pop-up kontrola

Pop-up omogućava korisniku da izabere vrijednost parametra iz liste mogućih vrijednosti. Specificirati vrijednosti u Popups polju na Parameters panelu. Slijedeći primjer pokazuje pop-up parametar:

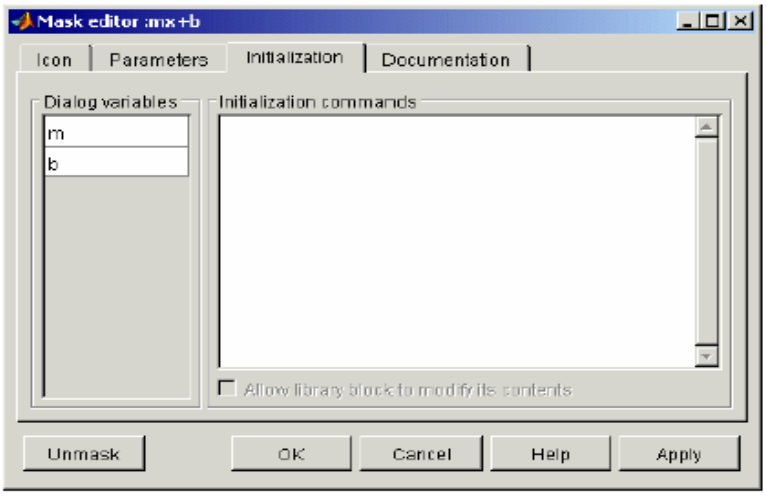


Vrijednost varijable pridružena sa parametrom (color) zavisi od detalja izabranog iz pop-up liste i da li je **Evaluate** opcija čekirana:

Evaluate	Vrijednost
on	Indeks vrijednosti izabrane iz liste, starta sa 1. Na primjer, ako je izabran treći detalj, vrijednost parametra je 3.
off	String čija je vrijednost izabrana. Ako je treći detalj izabran, vrijednost parametra je 'green'

Inicijalizacioni panel

Initialization panel dozvoljava da korisnik unese MATLAB komande koje inicijaliziraju maskirani podsistem.

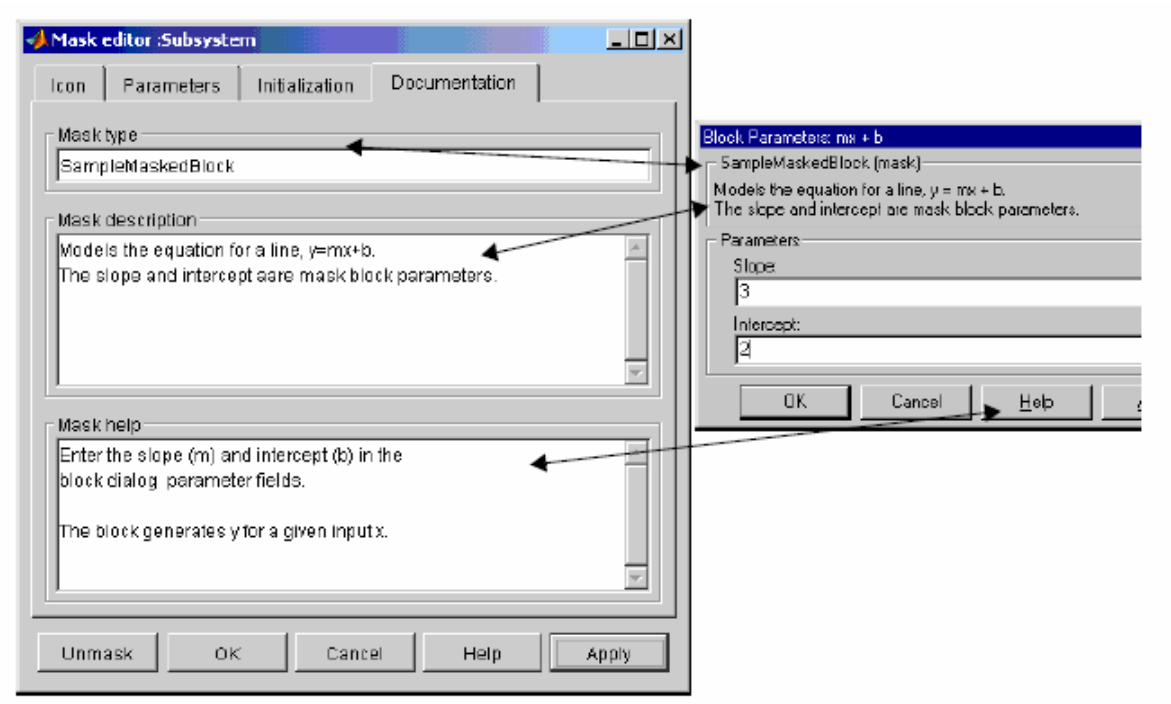


Simulink izvršava inicijalizacione komande kada:

- Napuni model u radnu memoriju
- Starta simulaciju ili ažurira blok dijagram
- Rotira maskirani blok
- Ponovno iscrta ikonu bloka

Dokumentacioni panel

Documentation panel omogućava korisniku da definira ili modificira tip, opis , i help tekst za maskirani blok. Naredna slika pokazuje kako polja u Documentation panelu korespondiraju sa dijalog boksom mx + b maske.

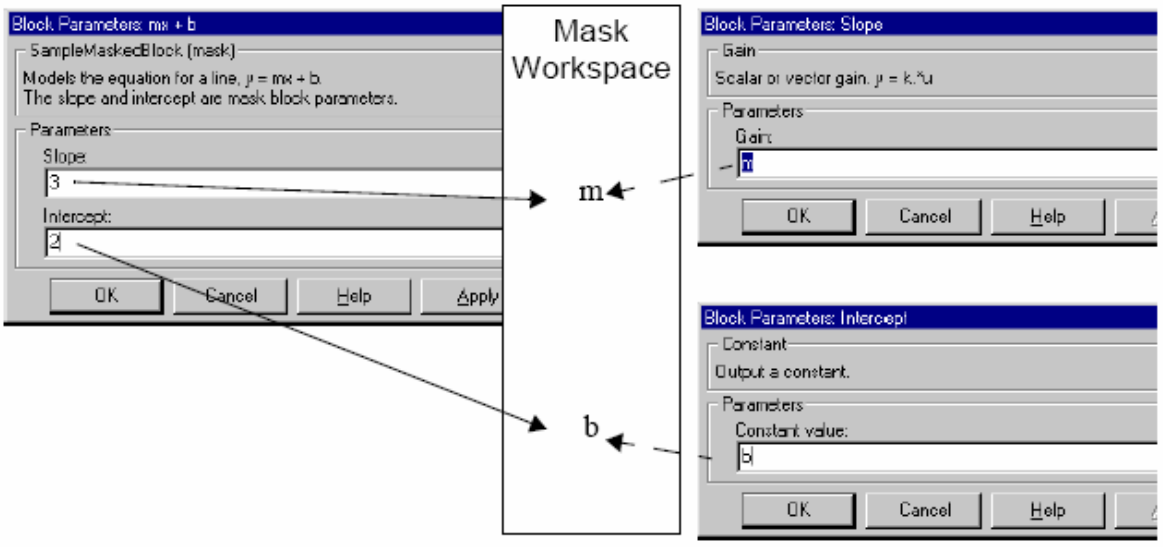


Povezivanje (linking) parametara maske sa parametrima bloka.

Varijable pridružene sa parametrima maske dozvoljavaju korisniku da linkuje parametre maske sa parametrima bloka. Ovo sa druge strane dozvoljava korisniku da koristi masku da postavi vrijednosti parametara bloka unutar maskiranog podsistema.

Da bi linkovao parametre, treba otvoriti dijalog boks parametara bloka, i unjeti izraz u polje vrijednosti paramtera bloka, koje koristi maskirani parametar.

mx + b maskirani podsistem, ranije opisan, koristi ovaj pristup da poveže parametre maske Slope (**m**) i Intercept (**b**) sa odgovarajućim parametrima blokova pojačanja (Gain) i konstante (Constant), koji se nalaze unutar podsistema



Možemo koristiti inicijalizacioni kod maskiranog bloka da linkujemo maskirane parametre indirektno sa blok parametrima. U ovom pristupu, inicijalizacioni kod kreira varijable u maskiranom radnom prostoru čije vrijednosti su funkcije parametara maske i koji se pojavljuju u izrazima koji setuju vrijednost parametara bloka koje su sakrivene maskom.

Kreiranje dinamičkih dijaloga za maskirane blokove

Simulink omogućava kreiranje dijaloga za maskirane blokove čiji vanjski izgled se mjenja kao odziv na ulaze od korisnika. Osobine maskiranih dijalog boksova koji se mogu promjeniti na ovaj način uključuju:

- vidljivost kontrolnih parametara
- omogućavanje stanja kontrolnih parametara
- vrijednosti parametara

Setovanje dijalog parametara maskiranog bloka

Simulink definira set parametara maskiranog bloka koji definišu tekuće stanje maskiranog dijalog boksa. Možemo koristiti editor maske da pregleda i setuje mnoge od ovih parametara.

Predefinisani parametri maskiranog dijaloga

Simulink pridružuje slijedeće predefinisane parametre sa maskiranim dijalozima.

Mask callbacks

Vrijednost ovog parametra je čelija polja stringova koja specificira callback izaraze za dijaloge korisnički definisanih kontrolnih parametara.

Masked Description

Vrijednost ovog parametra je string koji specificira opis ovog bloka. Korisnik može promjeniti opis maskiranog bloka dinamički setujuči ovaj parametar.

MaskEnables

Vrijednost ovog parametra je čelija polja stringova koja definira omogućeno stanje korisnički definisanih parametara kontrole za ovaj dijalog.

Možemo omogućiti ili onemogućiti korisnički ulaz dinamički setujući ovaj parametar u callbacku. Na primjer, slijedeća komanda u callbacku:

```
set_param(gcb,'MaskEnables',{'on','on','off'});
```

će onemogućiti treči kontrolni element od tekućeg otvorenog dijaloga maskiranog bloka. Simulink će obojiti onemogućene kontrolne elemente sivom bojom da vizuelno indicira da su onemogućeni.

MaskPrompts

Vrijednost ovog parametra je čelija polja stringova koja specificira promptove za korisnički definirane parametre. Prva čelija definira prompt za prvi parametar, druga za drugi parametar itd.

Masktype

Vrijednost ovog parametra je tip maske bloka pridruženog sa ovim dijalogom.

MaskValues

Vrijednost ovog parametra je čelija polja stringa koja specificira vrijednosti korisnički definiranih parametara za ovaj dijalog. Prva čelija definira vrijednostt za prvi , itd

MaskVisibility

Vrijednost ovog parametra je čelija polja stringova koja specificira vidljivost (visibility) korisnički definiranih parametara kontrole za ovaj dijalog.

Možemo sakriti ili pokazati korisnički definirane parametre kontrole dinamički setujući ovaj parametar u callbacku za kontrolni element. Na primjer, slijedeća komanda u callbacku:

```
set_param(gcb,'MaskVisibilities',{'on','off','on'});
```

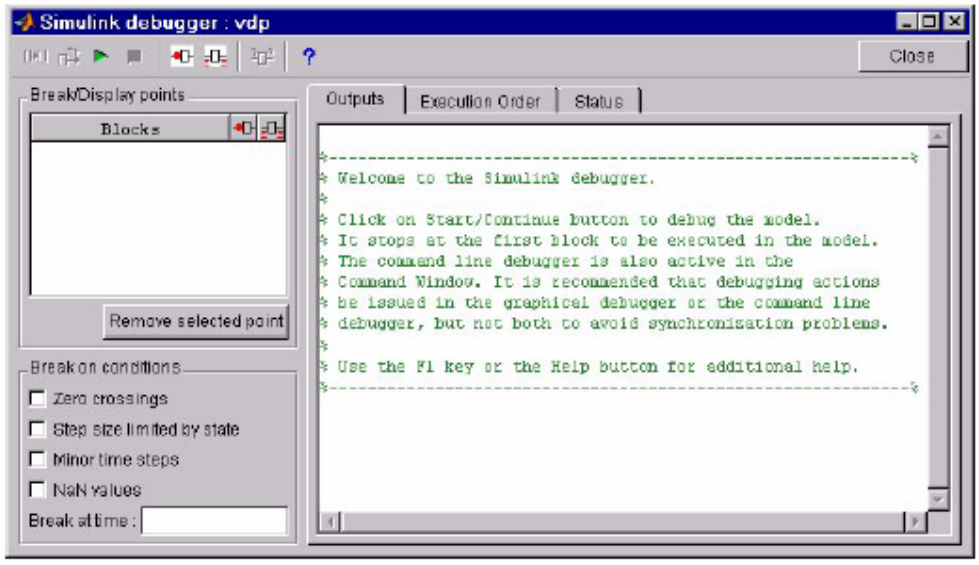
će sakriti kontrolni element za tekuče izabrani drugi korisnički definirani parametar maske bloka. Simulink će ekspandirati ili sažeti dijalog da pokaže ili sakrije kontrolni element.

Simulink debager (debugger)

Simulink debager je alat za lokalizaciju i dijagnostiku bagova u modelu Simulinka. On omogućava korisniku da uočava probleme izvršavajući simulaciju korak po korak i prikazujući trenutačna stanja blokova kao i ulaza i izlaza. Simulink debager ima i grafički i interfejs sa korisnikom putem komandne linije.

Startanje debagera

Da se starta debager, treba otvoriti model koji želimo debagirati i izabrati **Debugger** iz **Tools** menija. Pojaviće se prozor debagera kao na slici:



Možemo startati debager i sa komandne linije MATLAB-a , koristeći komandu **sldebug**, ili debug opciju sim komande. Naprimjer:

```
sim('vdp',[0,10],simset('debug','on'))
```

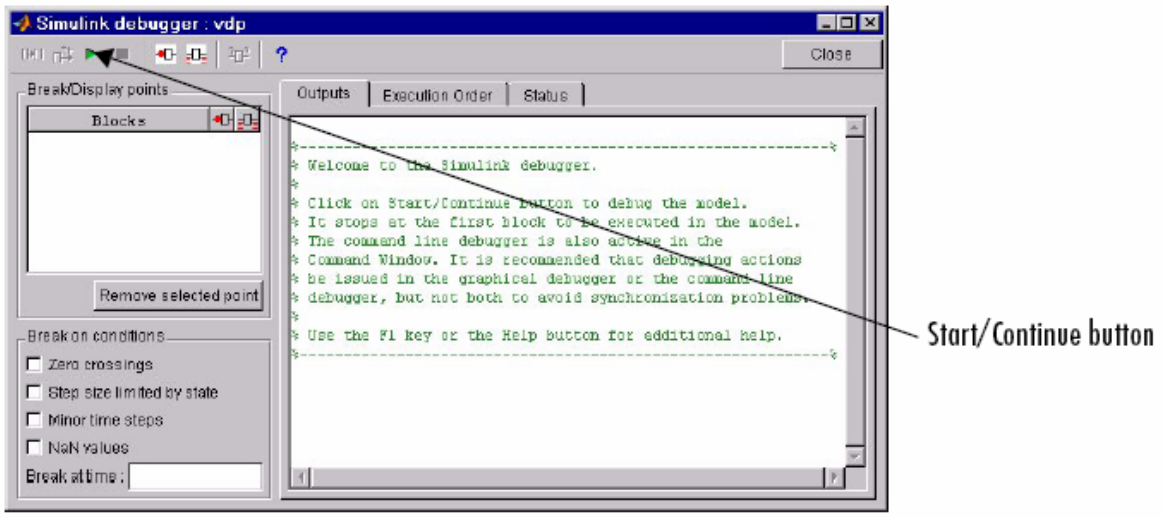
ili komanda:

```
sldebug 'vdp'
```

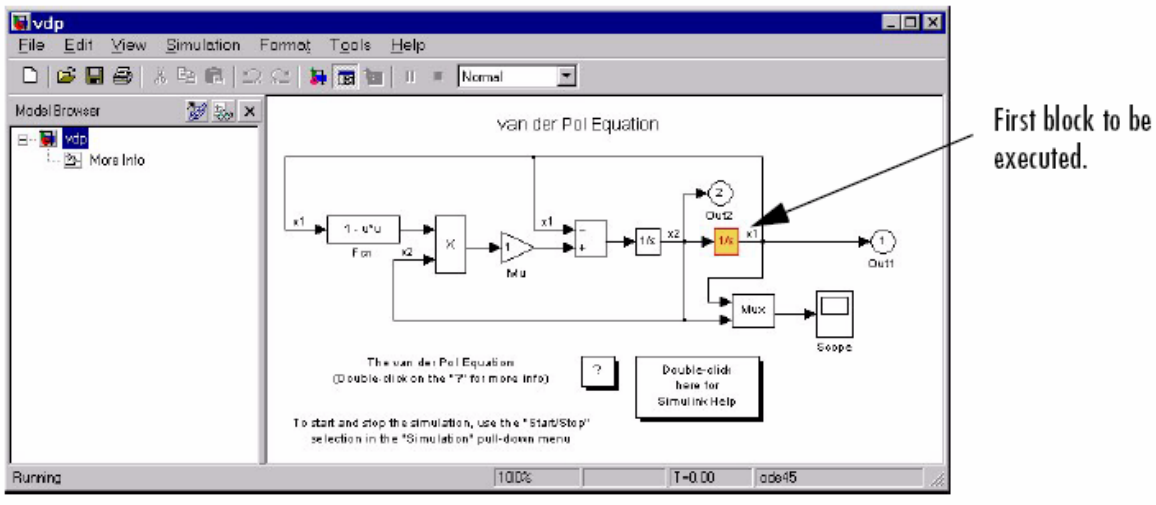
puni Simulink demo model vdp u memoriju, starta simulaciju , i zaustavlja simulaciju na prvom bloku u listi izvršenja modela.

Startanje simulacije

Da se starta simulacija, izabrati Start/Continue taster na toolbaru debagera, kao na slici:



Simulacija starta i zaustavlja se na prvom koraku koji će se izvršiti. Debager će otvoriti browser panel modela i naglasiti bojom blok kod kojeg se zaustavilo izvršenje simulacije.



U ovom momentu korisnik može postaviti brejkpointe (breakpoint-tačke preloma), nastaviti izvršavati simulaciju korak po korak, nastaviti simulaciju do slijedeće prelomne tačke ili kraja, ispitati podatke, ili izvršiti druge debaging aktivnosti.

Korištenje debagerovog interfejsa sa komandne linije

U modu rada komandne linije, korisnik kontrolira debager unoseći komande na komandnoj liniji debagera u MATLAB komandnom prozoru.

Pristupanje radnom prostoru MATLAB-a

Korisnik može unjeti bilo koji MATLAB izraz na **sldebug** promptu. Naprimjer, predpostavimo da smo na brejkpointu i da želimo da pohranimo (log) , vrijeme i izlaz iz modela kao **tout** i **yout**. Slijedeća komanda:

```
(sldebug ...) plot(tout, yout)
```

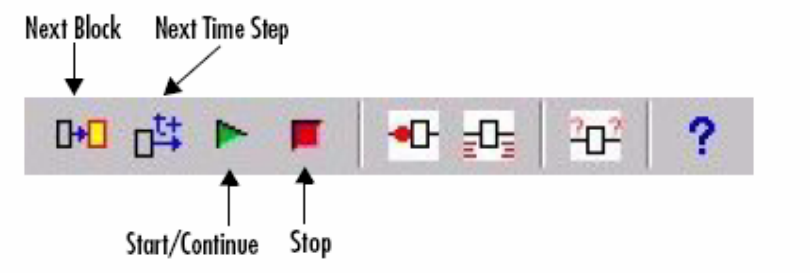
će kreirati plot.

Izvršavanje simulacije

Simulink degager omogućava nam da izvršavamo simulaciju od tačke gdje je trenutačno zaustavljena (suspended) , do slijedećih tačaka:

- kraja simulacije
- slijedećeg brejkpointa
- slijedećeg bloka
- slijedećeg vremenskog koraka

Korisnik može izabrati koliko će napredovati u simulaciji izabirući odgovarjući taster na toolbaru debagera



ili unošenjem odgovarajuće debagerske komande na komandnoj liniji.

Komanda	Napredovanje u simulaciji
step	Jedan blok
next	Jedan vremenski korak (integracioni)
continue	Do slijedećeg brejkpointa
run	Do kraja simulacije, ignorišući brejkpointe

Stepovanje sa minimalnim vremenskim koracima

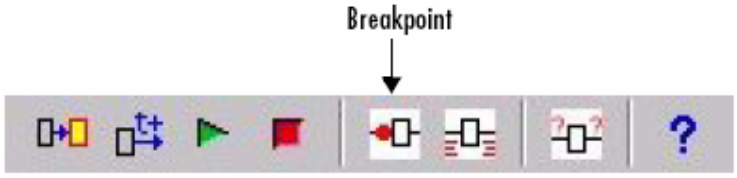
Možemo stepovati kroz blokove sa minimalnim vremenskim koracima (minor time steps) ili sa najvećim koracima (major time steps). Da bi išli sa najmanjim treba izabrati **Minor time steps** opciju na panelu debagera **Break on conditions** ili unjeti **minor** na komandnom promptu debagera.

Setovanje brejkpointa

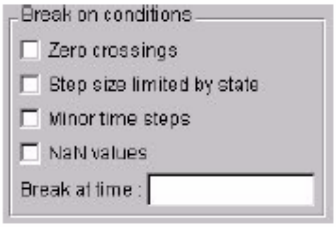
Simulink debager dozvoljava korisniku da definira zaustavne tačke koje se zovu breakpoints. Korisnik sa njima može izvršavati simulaciju od jedne prekidne tačke do druge, koristeći komandu **continue.** Debager nam omogućava da definiramo dva tipa tačaka prekida: bezuslovne i uslovne.

Bezuslovna tačka prekida pojaviće se svaki put kada simulacija dostigne blok ili vremenski korak koji je prethodno specificiran.
Uslovna tačka prekida će se pojaviti kada uslov koji smo specificirali unaprijed se pojavi u toku izvršavanja simulacije.

Tačka prekida se postavlja klikanjem na taster brakpoint na toolbaru debagera:



ili izborom odgovarajućeg brejkpoint uslova iz bloka:



ili unošenjem odgovarajuće brejkpoint komande na komandnoj liniji debagera:

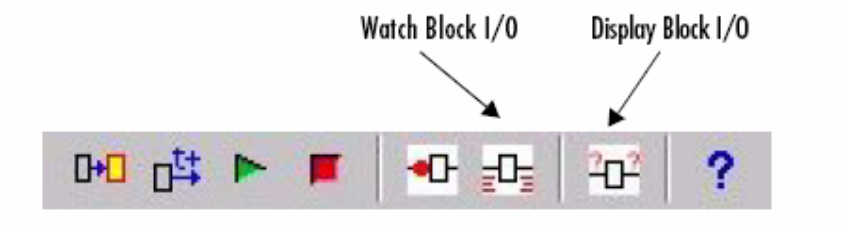
Komanda	Prouzrokuje da se simulacija zaustavi na
break <gcb | s:b>	Na početku naznačenog bloka
bafter <gcb | s:b >	Na kraju bloka
tbreak [t]	Kod pojave podliva ili preliva (NaN) ili beskonačne vrijednosti (Inf)
xbreak	Kada simulacija dostigne stanje koje odredjuje velićinu koraka simulacije
zcbreak	Kada se pojavi prolazak kroz nulu izmedju vremenskih koraka simulacije

Prikazivanje informacija o simulaciji

Simulink debager obezbjedjuje set komandi koje dozvoljavaju korisniku da prikaže stanja blokova, ulaze i izlaze blokova, i druge informacije dok se izvršava simulacija.

Prikazivanje ulaza i izlaza (I/O) blokova

Debager omogućava pikazivanje I/O vrijednosti izabirući odgovarajuće tastere na toolbaru debagera:



ili unošenjem odgovarajučih komandi debagera:

Komanda	Prikazivanje I/O bloka (ulaza/izlaza)
probe	odmah
disp	Kod svake tačke prekida
trace	Kad god se blok izvrši

Sumarni pregled komandi debagera

Naredna tabela daje listu svih komandi debagera sa komandne linije MATLAB-a. Kolona "Repeat" specificira da li pritiščući Return taster na keyboardu, posljednja komanda sa komandne linije će se automatski ponoviti.

Komanda	Kratki oblik	Repeat	Opis
ashow	as	ne	Pokazuje algebarsku konturu
atrace	at	ne	Postavlja nivo trace komande za algebarsku konturu
bafter	ba	ne	Unosi tačku prekida nakon izvršenja bloka
break	b	ne	Unosi tačku prekida prije izvršenja bloka
bshow	bs	ne	Pokazuje specificirani blok
clear	cl	ne	Otklanja tačku prekida iz bloka
continue	c	da	Nastavlja simulaciju
disp	d	da	Prikazuje I/O bloka kada se simulacija zaustavi
help	? ili h	ne	Prikazuje help za debagerske komande
ishow	i	ne	Omogućuje ili onemogućuje prikaz informacije o integraciji
minor	m	ne	Omogućuje ili onemogućuje mod minimalnog koraka (minor)

Komanda	Kratki oblik	Repeat	Opis
Nanbreak	na	ne	Setuje ili čisti prekid na beskonačnu vrijednost (NaN)
next	n	da	Ide do početka slijedećeg vremenskog koraka
probe	p	ne	Prikazuje I/O bloka
quit	q	ne	Završava simulaciju
run	r	ne	Izvrši simulaciju do kraja
slist	sli	ne	Izlistava nevirtualne blokove modela
states	state	ne	Prikazuje tekuče vrijednosti stanja
status	stat	ne	Prikazuje aktualne opcije debagera
step	s	da	Stepuje do slijedećeg bloka
stop	sto	ne	Zaustavlja simulaciju
systems	sys	ne	Izlistava nevirtualne podsisteme u modelu
tbreak	tb	ne	Setuje ili čisti vremensku tačku prekida
trace	tr	da	Prikazuje I/O bloka svaki put kada ga izvrši
undisp	und	da	Otklanja blok iz debagerske liste prikazanih tačaka
untrace	unt	da	Otklanja blok iz debagerske liste trace tačaka
xbreak	x	ne	Prekida kada debager naidje na stanje koje limitira velićinu stepa
zcbreak	zcb	ne	Prekida kod nesampliranih dogadjaja prolaska kroz nulu
zclist	zcl	ne	Izlistava blokove koji sadrže nesamplirane prolaske kroz nulu

Opcije alata za mjerenje performansi Simulinka

Simulink alati za poboljšanje performanse uključuju slijedeće :

- Simulink ubrzavač (accelerator)
- Alat za grafičko objedinjavanje (graphical merge tool)
- profiler
- alat za praćenje modela (model coverage tool)

Ubrzavač Simulinka

Simulink ubrzavač ubrzava izvršenje Simulink modela. Ubrzavač koristi dijelove Real-Time radione (real-time workshop - RTW) , koji je softwareski modul u okviru MATLAB-a koji automatski generira C kod iz modela Simulinka, što uz C kompajler omogućava kreiranje direktno izvršivih (.exe) programa. Primjetimo da , mada Simulink ubrzavač koristi tehnologiju RTW , on nije potreban da bi se koristio Simulink ubrzavač.

Takodjer ako nemamo raspoloživ C kompajler instaliran na PC-ju, možemo koristiti **lcc** kompajler u okviru MATLAB-a.

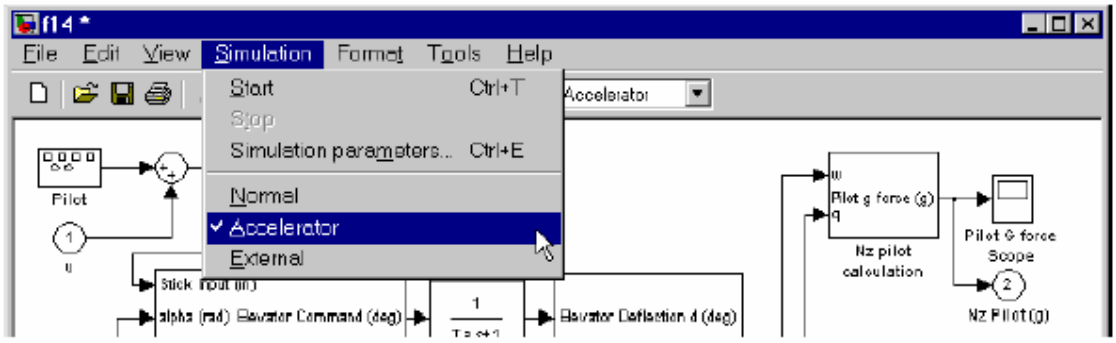
Primjetimo da ubrzavač ne podržava modele koji imaju algebarske konture. Ako ubrzavač otkrije algebarsku konturu u modelu, on zaustavlja simulaciju i pokazuje poruku greške.

Simulink ubrzavač radi na taj način što kreira i kompilira C kod koji zauzima mjesto umjesto koda interpretera koji Simulink koristi u normalnom modu rada (tj kada nije u modu ubrzavača).
Ubrzavač generira C kod iz Simulink modela a Mex funkcija MATLAB-a poziva kompajler i dinamički povezuje generirani kod sa Simulinkom.

Simulink ubrzavač otklanja većinu računarske redudancije koju zahtjevaju Simulinkovi modeli u normalnom modu izvršenja. To radi na taj način što zamjenjuje blokove koji su dizajnirani da mogu da rade u bilo kojoj konfiguraciji u Simulinku sa kompiliranim verzijama prilagodjenim za specifičnu konfiguraciju modela. Putem ovoga modela, ubrzavač je u stanju da postigne značajna poboljšanja u performansi kod velikih modela Simulinka. Dobitak u performansi zavisi od velićine i kompleksnosti modela. Općenito, kako velićina i kompleksnost rastu, tako se povećava i performansa. Tipično, možemo očekivati 2 do 6 – tostruko povećanje u performansi modela koji koristi ugradjene blokove Simulinka.

Rad sa ubrzavačem Simulinka

Da se aktivira ubrzavač Simulinka, izabrati **Accelerator** komandu iz menija **Simulation** u modelu u kojem radimo.



Da počnemo simulaciju treba izabrati **Start** iz menija **Simulation**. Kada startamo simlaciju, akcelerator će geneririati C kod i kompilirati ga. Nakon toga, akcelerator će uraditi slijedeće:

- postaviti generirani kod u poddirektorij koji će se zvati modelname_accel_rtw
- postaviti kompilirani MEX-file u tekući radni direktorij
- izvršiti kompilirani model

Akcelorator koristi tehnologiju RTW da generira kod koji će ubrzati izvršenje simulacije modela. Medjutim, ovaj kod je pogodan samo za ubrzavanje modela. Ako želimo da generiramo kod za druge namjene (napr. download u ciljni hardware), za to moramo koristiti RTW.

Rukovanje sa promjenama u strukturi modela

Ako koristimo Simulink akcelerator da simuliramo model, MEX-file koji sadrži kompiliranu verziju modela ostaje raspoloživ i za korištenje u kasnijim simulacijama.
Ako promjenimo strukturu Simulink modela, naprimjer, dodavajući ili brišuči neke blokove, akcelerometar automatski regenerira C kod i ažurira postojeći MEX-file.

Primjeri promjene strukture modela koji će zahtjevati da akcelerometar ponovno kompilira model su:

- promjena metoda integracije
- dodavanje ili brisanje blokova ili konekcija izmedju blokova
- promjena broja ulaza ili izlaza bloka, čak i kad je konektivnost vektorisana
- promjena broja stanja u modelu
- promjena funkcije u trigonometrijskom funkcionalnom bloku
- promjena znaka u Sum bloku
- Dodavanje S-funkcije u TLC (target language compiler) file u inline

Inače, Simulink akcelerator će pokazati upozorenje kada pokušamo da učinimo bilo koju nedopustivu promjenu u modelu za vrijeme simulacije. Da bi proveli promjenu u modelu, treba zaustaviti simulaciju, napraviti promjene i ponovno startati.

Povećanje performanse akceleratorskog moda rada

Općenito, Simulink akcelerator kreira kod optimiziran za brzinu, za većinu blokova u Simulinku. Medjutim, postoje situacije kada možemo još dodatno poboljšati performansu, kao:

- **Simulation parameters** dijalog boks. Ova opcija u Diagnostics i Advanced panelima može uticati na performansu akceleratora. Da bi je povećali treba:
  - Onemogučiti **Consistency checking** i **Bounds checking** na Diagnostics panelu
  - postaviti Signal storage reuse na **Advanced** panelu
- **Stateflow** – Akcelerometar je u potpunosti kompatibilan sa Stateflow, ali ne poboljšava performansu od Stateflow dijelova modela. Zato treba onemogućiti

debagiranje i animaciju za Stateflow, da bi se povećala performansa modela koji uključuju Stateflow blokove.

- S-funkcije koje je napisao korisnik – Akcelerator ne može poboljšati simulacionu brzinu za S – funkcije izuzev ako ih inlajniramo koristeći TLC. Inlajniranje se odnosi na postupak kreiranja TLC fajlova koji usmjeravaju RTW da kreira C kod za S-funkciju. Ovo će eliminirati nepotrebne pozive ka Simulinkovom API interfejsu.

- S-funkcije koje obezbjedjuje Simulink i blocksetovi. – Mada Simulink akcelerator je kompatibilan sa svim blokovima koji su dio isporuke sa Simulinkom i bloksetovima, to neće poboljšati brzinu simulacije za M-file ili C-MEX S-funkcijske blokove, koji nemaju pridružen inlajning TLC file.

- Logiranje velike količine podataka - Ako koristimo blokove Workspace I/O , To Workspace, To File, ili Scope, velike količine podataka će usporiti akcelerator. Treba pokušati prorjedjivanje (decimation) ili limitiranje izlaza na posljednjih N tačaka.

- Veliki modeli - i u slućaju akceleratorskog kao i normalnog moda rada, Simulink može potrošiti mnogo vremena na inicijalizaciju velikih modela. U takvim slučajevima akceleratorsko ubrzanje može biti skromno, ako je vrijeme inicijalizacije srazmjerno vremenu trajanja preostalog dijela simulacije od početnog do krajnjeg vremena.

Blokovi koji ne pokazuju poboljšanje brzine

Simulink akcelerator ubrzava izvršenje samo blokova iz Simulinka, Fixed point i DSP bloksetova. Medjutim odredjeni blokovi i medju ovim se ne ubrzavaju i to:

Simulink blokovi

- Display
- From File
- From Workspace
- Inport (na osnovnom nivou sistema)
- MATLAB Fcn
- Outport (na osnovnom nivou sistema)
- Scope
- To File
- To Workspace

- Transport delay
- Variable Transport delay
- XY Graph

DSP Blockset blokovi

- Biquadratic filter
- Convolution
- Direct-form II Transpose Filter
- Dyadic Analysis Filter Bank
- Dyadic Synthesis Filter Bank
- FIR decimation
- FIR Interpolation
- FIR Rate Conversion
- From Wave device
- From Wave file
- Integer delay
- Variable Integer Delay
- Matrix Multiply
- Matrix to Workspace
- Triggered Signal To Workspace
- Triggered Signal from Workspace
- Time –Varying Direct –Form II Transpose Filter
- To Wave file
- To Wave device
- Wavelet Analysis
- Wavelet synthesis
- Zero Pad

Korištenje Simulink akceleratora sa Simulink debagerom

Ako imamo velike i kompleksne modele koje treba da debagiramo, Simulink može skratiti dužinu debaging sesija. Naprimjer, ako treba da setujemo vremenski prekid koji je vrlo veliki, možemo koristiti akcelerator da dostignemo tačku prekida mnogo brže.
Da bi izvršavali Simulink debager dok smo u akceleratorskom modu:

1. Izabrati Accelerator iz Simulink menija , a onda unjeti:

   sldebug modelname , na MATLAB promptu

2. Na debugger promptu, postaviti vremenski prekid:

   tbreak 1000
   continue

3. Kada dostignemo tačku prekida, koristiti komandu deguggera **emode** (excution mode) da mjenjamo izmedju **Accelerator** i **Normal** mode.

Primjetimo da moramo preći u Normal mod da stepujemo simulaciju po blokovima. Takodjer moramo preći u Normal mod da koristimo slijedeće debug komande:

- trace
- break
- zcbreak
- nanbreak
- minor

Poredjenje performansi

Ako je potrebno da poredimo performansu Simulink akceleratora sa Normal modom rada, treba koristiti **tic**, **toc** i **sim** komande, Naprimjer , ako smo koristili f14 primjer , treba koristiti slijedeći kod (pri tome treba biti u Normal modu):

```
tic [t,x,y] = sim ( 'f14', 1000 ) ; toc

elapsed_time = 13.78
```

U akceleratorskom modu , dobićemo slijedeći rezultat:

```
elapsed_time = 1.96
```

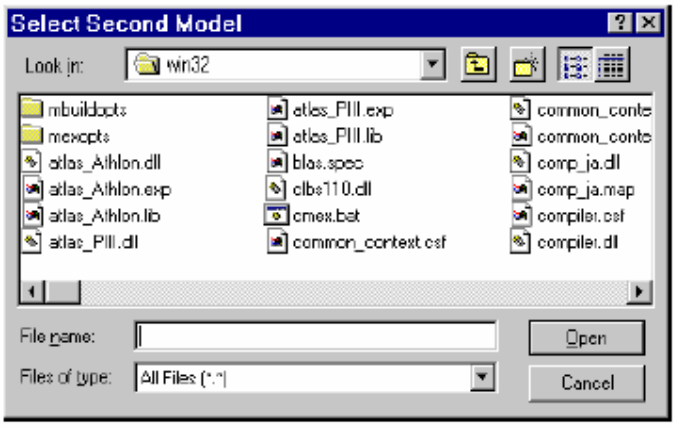
Dakle ubrzanje je preko šest puta

Grafički alat za objedinjavanje (graphical merge tool)

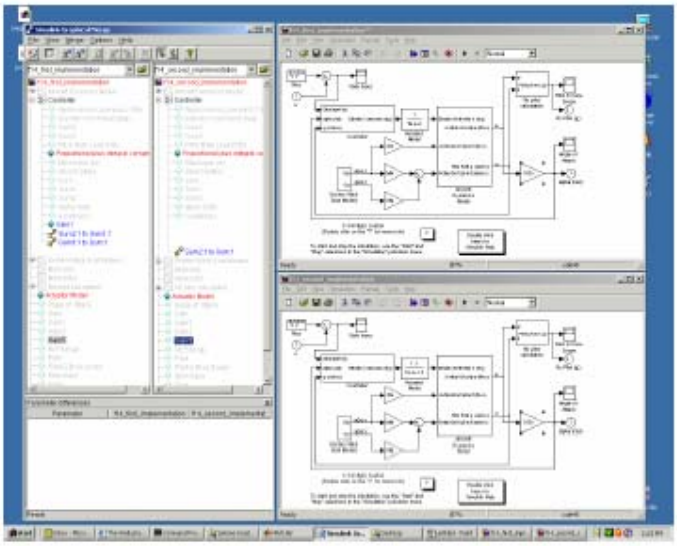
Ovaj alat pomaže korisniku da nadje i objedini razlike izmedju više verzija Simulink modela, uključujući i modele koji sadrže Stateflow chartove.

Ovaj alat pojednostavljuje zajednički (colaborative) rad na razvoju modela. Naprimjer, dva saradnika rade na razvoju modela, i kada su završili sa radom, koristiće ovaj alat da objedine dvije verzije.

Prozor **Graphical Merge Tool** se pojavljuje zajedno sa **Select Second Model** dijalog boksom:



Treba koristiti **Select model to compare** to dijalog boks da se izabere drugi model koji će se porediti. Simulink otvara drugi model, ako već nije otvoren , i aranžira **Graphical Merge Tool** prozor i dva modela, kao što je pokazano na narednoj slici:



Simulink takodjer povećava ili smanjuje blok dijagrame modela tako da oni mogu upotpunosti se uklopiti u njihove odgovarajuće prozore modela.

Profiler

Simulink profiler simulacije skuplja informacije o performansi dok traje simulacija modela i generira izvještaj , koji se naziva simulacioni profil (simulation profile), na bazi prikupljenih podataka.
Simulacioni profil generiran od strane profilera pokazuje koliko je vremena Simulink potrošio izvršavajući svaku funkciju , potrebnu da se simulira model. Ovaj profil

omogučuje korisniku da odredi dijelove modela koji zahtjevaju najviše vremena da se simulira i time odredjuje gdje treba koncentrirati napore na optimizaciji modela.

Slijedeći pseudokod sumira izvršni model na kojem se bazira profiler.

```
Sim()
   ModelInitialize().
   ModelExecute()
       for t = tStart to tEnd
       Output()
       Update()
       Integrate()
           Compute states from derivs by repeatedly calling:
               MinorOutput()
               MinorDeriv()
           Locate any zero crossings by repeatedly calling:
               MinorOutput()
               MinorZeroCrossings()
       EndIntegrate
       Set time t = tNew.
   EndModelExecute
   ModelTerminate
EndSim
```

U skladu sa ovim konceptualnim modelom, Simulink izvršava model pozivajući slijedeće funkcije jedanput ili više puta, zavisno od funkcije i modela:

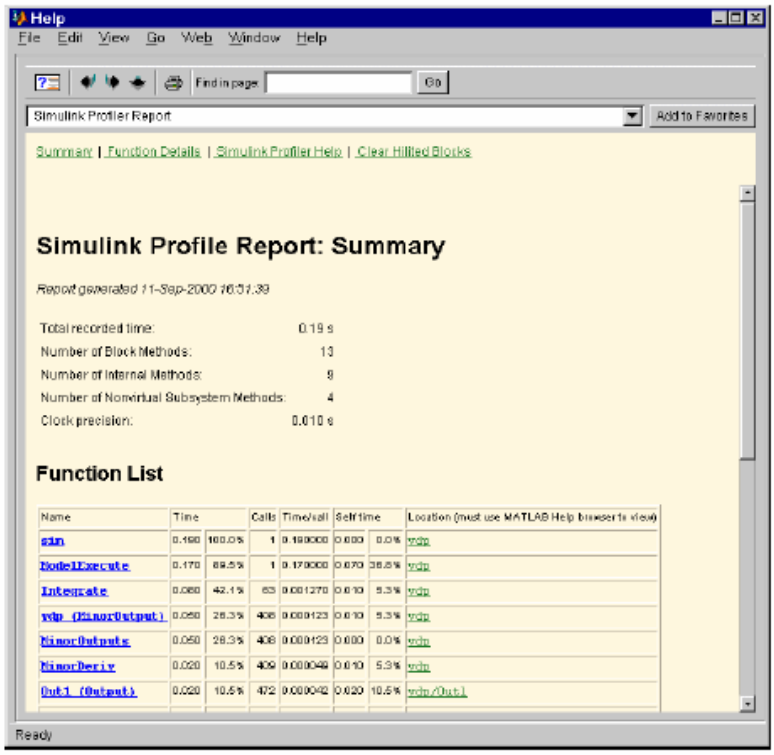
Funkcija	Namjena	Nivo
sim	Simulira model. Vrijeme provedeno u ovoj funkciji je totalno vrijeme potrebno za simulaciju modela	sistem
ModelInitialize	Setuje model za simulaciju	sistem
ModelExecute	Izvršava model pozivajući funkcije :izlaz (output), update, integrate, itd, za svaki blok u svakom vremenskom koraku od starta do kraja simulacije.	sistem
Output	Izračunava izlaze bloka kod tekućeg vremenskog koraka	blok
update	Ažurira stanje bloka kod tekučeg vremenskog koraka	blok
Integrate	Izračunava kontinualna stanja bloka integrišući izvode stanja kod tekučeg vremenskog koraka	blok
MinorOutput	Izračunava izlaz bloka kod najmanjeg (minor) vremenskog koraka	blok

Funkcija	Namjena	Nivo
MinorDeriv	Izračunava izvode stanja bloka kod najmanjeg (minor) vremenskog koraka	blok
MinorZeroCrossing	Izračunava vrijednosti prolaska kroz nulu bloka kod najmanjeg (minor) vremenskog koraka	blok
ModelTerminate	Oslobadja memoriju i izvršava i sva druga čišćenja na kraju simulacije	sistem
Nonvirtual subsystem	Izračunava izlaz nevirtualnog podsistema , kod tekućeg vremenskog koraka pozivajući funkcije output, update, integrate, itd. za svaki blok koji sadrži. Vrijeme provedeno u ovoj funkciji je vrijeme koje je potrebno da se izvrši nevirtualni sistem.	blok

Profiler mjeri vrijeme potrebno da se izvrši svako pozivanje ovih funkcija i generiranje izvještaja na kraju , da se pokaže koliko je vremena provedeno sa svakom funkcijom.

Omogućavanje profilera

Da bi se profilirao model, treba otvoriti model i izabrati **Profiler** iz menija **Tools**. Nakon toga startati simulaciju. Kada se simulacija završi, Simulink će generirati i prikazati simulacioni profil za model u MATLAB help browseru.
Simulink pohranjuje simulacioni profil u radnom direktoriju MATLAB-a.



Izvještaj ima dvije sekcije : sumarni i detaljni izvještaj

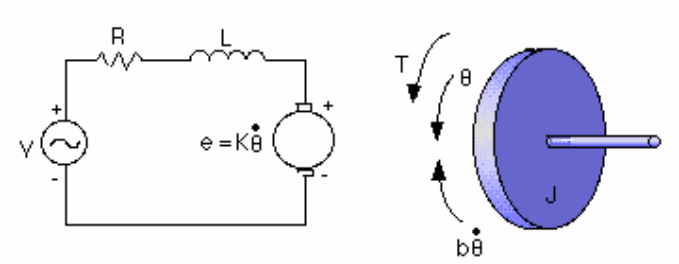
Alat praćenja modela (model coverage tool)

Ovaj alat odredjuje obim koliko test case modela ostvaruje praćenje simulacije kroz model. Procenat staza koje test case ostvaruje se naziva praćenjem modela (**model coverage**). Ovaj pokazatelj je dakle mjera koliko temeljito se testira model. Zato ovaj alat pomaže korisniku da validira njegove test rezultate sa modelom.

Ovaj alat radi na taj način što analizira izvršenje blokova koji direktno ili indirektno odredjuju simulacione staze prolaza kroz model. Ako model uključuje Stateflow chartove, alat takodjer analizira stanja i tranzicije ovih chartova. Za vrijeme izvršenja simulacije, alat bilježi ponašanje blokova, stanja i tranzicija koji se prate. Na kraju simulacije, alat izvještava obim do kojeg je simulacija ostvarila potencijalne staze prolaska kroz svaki blok koji se prati.

Primjer gradnje modela u Simulinku i simulacije

Kao primjer gradnje matematskog modela, unošenja modela u Simulink i simulacije , analiziraćemo na primjeru istosmjernog (DC) motora, koji je prisutan i u mnogim aktuatorima i izvršnim organima ugaonog ili translatornog (linearnog) izlaza prema izvršnom organu.
On direktno obezbjedjuje rotaciono kretanje a spregnut sa zupčanicima sa nazubljenom letvom ili bubnjem i kablom, obezbjedjuje i linearno kretanje na izlaznoj osovini.
Električno kolo armature (rotora) kao i mehanički podsistem rotora pokazani su na narednoj slici:



Za ovaj primjer, predpostavićemo slijedeće vrijednosti fizikalnih parametara:

- moment inercije rotora – J = 0.01 kg m^2/s^2
- prigušenje mehaničkog podsistema – b = 0.1 Nms
- konstanta elektromagnetne sile - (Ke=Kt=K) = 0.01 Nm/A
- električni otpor namotaja statora – R = 1 ohm
- električna induktanca namotaja na statoru- L = 0.5 H

Ulaz

- napon napajanja statora : V

Izlaz

- ugaona pozicija rotora : theta

Rotor i njegova osovina su predostavljeni idealno krutim tako da nema akumulacije atraktivne energije u rotorskom podsistemu.

Mehanički momenat motora **T** je vezan sa strujom armature (rotora) **i** , preko konstante **Kt**. kontra elektromotorna sila , e , je vezana sa ugaonom brzinom obrtanja rotora preko slijedeće jednačine:

$$\mathbf{T = Kt\ i}$$

$$\mathbf{e = Ke}\ \frac{d\theta}{dt}$$

U SI sistemu mjernim jedinica armaturna konstanta **Kt** je jednaka motornoj konstanti **Ke**.

Gradnja matematskog modela

Ovaj sistem će se modelirati na taj način što ćemo uzeti jednačinu momenata za mehanički podsistem rotora i kirhofove zakone za elektromagnetni podsistem.

Startaćemo Simulink i otvoriti novi model .

Dodaćemo prvo dva integratora da integralino ugaono ubrzanje da bi dobili ugaonu brzinu i takodjer da integralimo brzinu promjene struje armature da dobijemo samu struju:

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

$$\int \frac{di}{dt} = \mathrm{i}$$

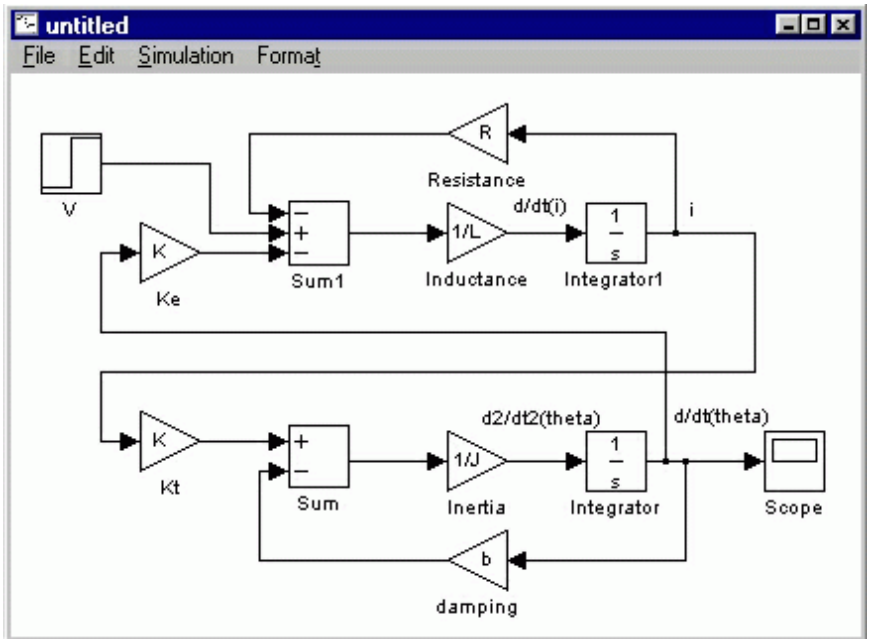
Nakon toga , poćećemo izvodjenje modela koristeći Newtonov zakon za mehanički podsistem i Kirchoffov zakon za elektromagnetni podsistem.
Za mehanički podsistem imaćemo:

$$\mathrm{J}\frac{d^2\theta}{dt^2} = T - b\frac{d\theta}{dt} \quad \Rightarrow \frac{d^2\theta}{dt^2} = \frac{1}{J}(K_t i - b\frac{d\theta}{dt})$$

za elektromagnetni podsistem:

$$L\frac{di}{dt} = -Ri + V - e \quad \Rightarrow \frac{di}{dt}\frac{1}{L}(-Ri + V - K_e\frac{d\theta}{dt})$$

Kada unesemo ove jednačine u Simulink koristeći odgovarajuće blokove iz biblioteka, dobićemo slijedeći Simulink model kao na slici:

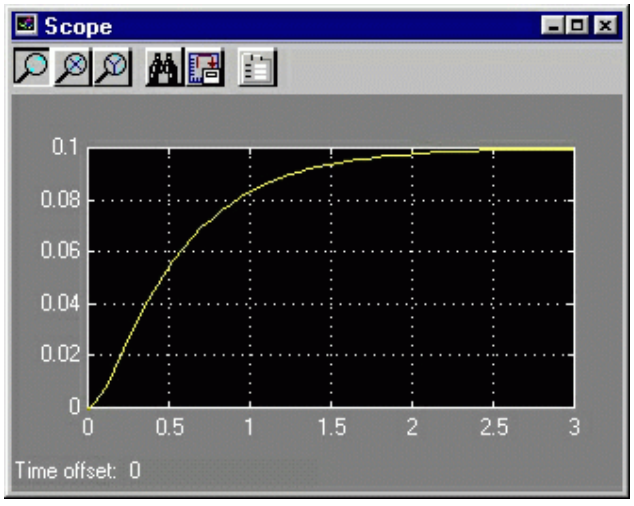


Da bi simulirali sistem, treba postaviti vrijeme trajanja simulacije u unjeti vrijednsti parametara u radni prostor MATLAB-a bilo direktno sa komandne linije ili preko S funkcije.

Unesimo slijedeće vrijednosti parametara:

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

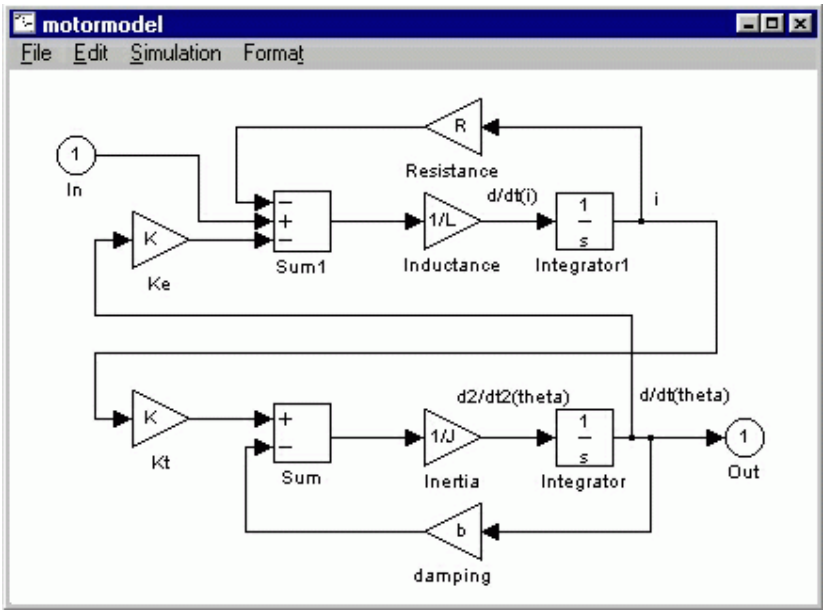
Nakon završetka simulacije , dobićemo na ekranu Scope, slijedeću sliku za promjenu ugaone brzine u vremenu:



Ekstrakcija linearnog modela u MATLAB

Linearni model sistema (u formi prostora stanja i prenosne funkcije), može se dobiti iz Simulink modela u MATLAB.
Ovo se može ostvariti korištenjem In i Out konekcija u Simulink modelu i MATLAB funkcijom **linmod**.

Zamjenićemo blokove Step i Scope sa In Connection blokom i Out Connection blokom, kao na narednoj slici:



Ovako modifikovan Simulink model ćemo pohraniti kao "motormod.mdl".
MATLAB će ektrahirati linearni model iz pohranjenog filea modela. Na komandnom promptu MATLAB-a treba unjeti slijedeće komandu:

```
[A,B,C,D]=linmod('motormodel')
```

Dobićemo slijedeće izlaze, za model u prostoru stanja:

```
A =

  -10.0000    1.0000
   -0.0200   -2.0000


B =

     0
     2
```

C =

1 0

D =

0

Unesimo sada slijedeću komandu:

[num,den]=ss2tf(A,B,C,D)

Dobićemo :

num =

0 0.0000 2.0000

den =

1.0000 12.0000 20.0200

Da verifikujemo ekstrakciju modela, generiraćemo step odziv u otvorenoj konturi (open-loop) od izvadjene prenosne funkcije u MATLAB-u.
Unjećemo slijedeću komandu u MATLAB-u:

step (num, den) ;

Dobićemo slijedeći plot na ekranu, koji je u potpunosti ekvivalentan onome što smo ranije prikazali kao odziv ugaone brzine na step promjenu ulaznog napona na armaturi u okviru Simulink simulacije: