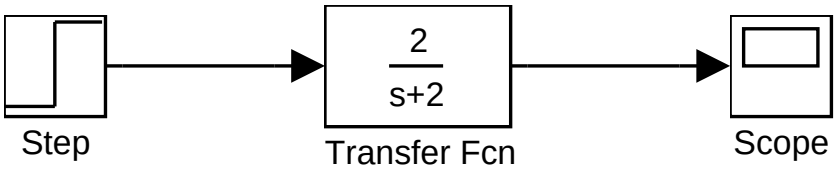
Introduktion till SIMULINK

Augusti 2009



Inledning

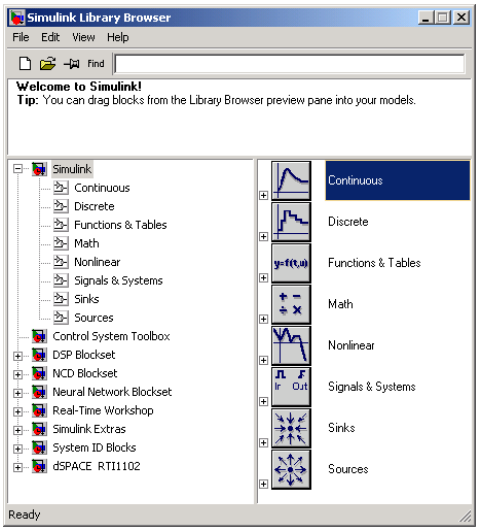
SIMULINK är en simuleringsmiljö som körs under MATLAB. Syftet med SIMULINK är att en användare ska kunna beskriva och simulera ett system på ett enkelt sätt. För att underlätta beskrivningen finns ett signalbaserat grafiskt användargränssnitt. I denna introduktion ges en kortfattad beskrivning av de mest använda funktionaliteterna. För en mer heltäckande beskrivning rekommenderar vi att man tittar i manualen som följer med programvaran. *Observera att det kan skilja något mellan olika versioner av* SIMULINK, så bli inte förvirrad om introduktionen skiljer sig från det på datorskärmen.

Vi rekommenderar att du verkligen kör SIMULINK och följer med i exemplen medan du läser introduktionen. Det bästa sättet att få en känsla för hur SIMULINK fungerar är sedan att modifiera och experimentera på egen hand.

Grundläggande användning

Att komma igång

För att starta SIMULINK måste man först starta MATLAB och sedan ge kommandot `simulink` vid MATLAB-prompten. Då kommer SIMULINK-biblioteket i figur 1 upp på skärmen. För att börja bygga en modell, gå via menyn `File` och välj därefter `New`. I det nya fönster med namnet `untitled` som då öppnas kommer vi att bygga en modell med hjälp av komponenter från biblioteket.



Figur 1: SIMULINK-biblioteket.

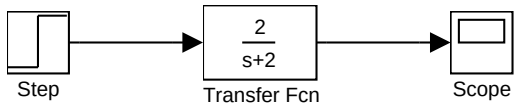
Bygga ett system

Antag nu att man vill studera stegsvaret till det linjära systemet

$$G(s) = \frac{2}{s+2}$$

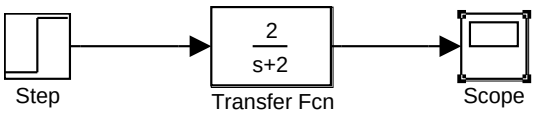
genom att simulera det. För att skapa en överföringsfunktion, dubbelklickar man på blocket `Continuous` i högra delen av fönstret `Simulink Library Browser` och sedan blocket `Transfer Fcn`. Genom att peka på `Transfer Fcn` med musen, hålla vänster mustangent nere och dra över det till fönstret `untitled` förs en kopia av blocket in i den nya modellen. Dubbelklicka på blocket med vänster mustangent och ändra därefter på raderna `Numerator` och `Denominator`. Observera att man bara anger koefficienterna till polynomen, det vill säga $s + 2$ anges som [1 2] och 2:an i täljaren anges som [2].

För att hitta ett steg som insignal går man till blocket `Sources` i `Simulink Library Browser` och väljer `Step`. Utsignalblocket tas från `Sinks` och där är `Scope` lämpligt, eftersom det visar utsignalen från det simulerade systemet. Blocken förbinds nu genom att man pekar på en utport, håller ned vänster mustangent och drar iväg till en lämplig inport. I figuren dras då en pil från ett block till ett annat. Efter ihopkoppling ser systemet ut som i figur 2. Genom att följa pilarna kan man se hur signalflödet går.



Figur 2: En överföringsfunktion med hjälp av blocket `Transfer Fcn`.

Om man skulle råka göra något misstag kan man ta bort block (eller exempelvis pilar) på följande sätt. Antag att man vill ta bort blocket `Scope`. Först markerar man det genom att klicka på det med vänster mustangent. Blocket kommer då att se ut enligt figur 3. Vill man plocka bort det väljer man `Cut` som finns i `Edit`-menyn eller trycker på `delete` på tangentbordet.



Figur 3: Ett markerat element.

Simulera ett system

Innan man simulerar systemet bör man nu bestämma simuleringstiden, som per default är satt till 10.0 s. Välj `Configuration Parameters` i menyn `Simulation`. Där kan man sätta ett antal parametrar som bestämmer hur simuleringen ska utföras. De som är intressanta nu finns under fliken `Solver`.

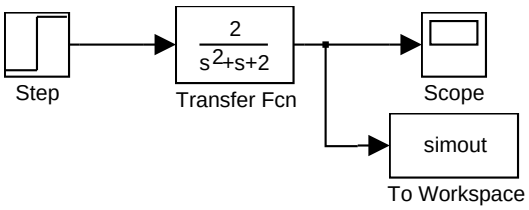
Gå in under fliken `Solver` och ändra `Stop time` från 10 s till exempelvis 20 s. Nu kan simuleringen startas genom att välja `Start` i menyn `Simulation`. SIMULINK piper till när simuleringen är färdig. För att se resultatet dubbelklickar man på `Scope`. Genom att välja något av de tre förstoringsglasen i `Scope`-fönstret och markera ett område i figuren med hjälp av musen kan figuren skalas. Prova dig fram och se vad som skiljer förstoringsglasen åt.

Variabler/parametrar i MATLAB

Nu kan man testa lite olika överföringsfunktioner. Prova till exempel med

$$G(s) = \frac{2}{s^2 + s + 2}$$

genom att ändra `Numerator` och `Denominator`. Vill man manipulera utsignalen mer i MATLAB kan man spara SIMULINK-signaler med hjälp av blocket `To Workspace` som finns i `Sinks`. Detta block kan kopplas till utsignalen från överföringsfunktionen genom att peka på inporten till `To Workspace`-blocket och hålla ned någon mustangent samtidigt som man drar korset till pilen mellan `Transfer Fcn` och `Scope`, så att en ny pil bildas utgående från den redan existerande pilen som i figur 4. Dubbelklicka på `To Workspace`-blocket och ändra `Save format` från `Structure` till `Array`. Kör nu en simulering och plotta resultatet i MATLAB genom att skriva `plot(tout,simout)`.

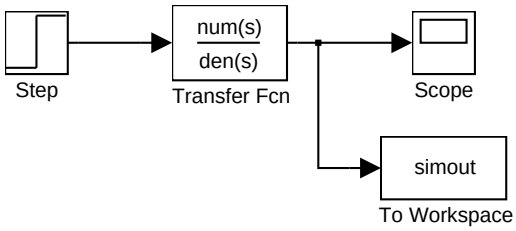


Figur 4: Signal till MATLABs workspace med hjälp av blocket `To Workspace`.

Man kan också definiera systemet som ska simuleras direkt i MATLAB genom

```
>> num=[2];
>> den=[1 1 2];
```

och skriv in variabelnamnen `num` och `den` i blocket `Transfer Fcn` på respektive plats. Blockschemat i SIMULINK blir då som i figur 5.



Figur 5: Variabler i täljare (`num`) och nämnare (`den`) i `Transfer Fcn`.

Fler byggstenar

Vill man simulera ett olinjärt system, till exempel

$$\dot{x}(t) = -x(t)^2 + u(t)$$

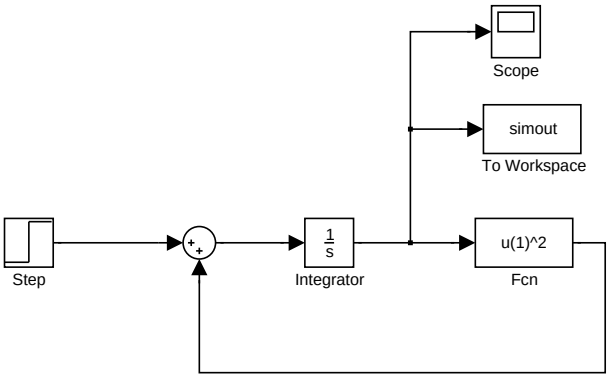
kan man göra så här istället. Om båda sidorna av tillståndsekvationen integreras får man istället

$$x(t) = \int_0^t (-x(\tau)^2 + u(\tau))d\tau$$

så vad som behövs är en numerisk integrator. Denna finns under `Continuous` och heter `Integrator`. Dessutom måste x^2 beräknas, vilket exempelvis kan göras med hjäp av blocket `Fcn` som finns under `Functions & Tables`. Allt man behöver göra är att gå in och ändra det förinskrivna exempeluttrycket `sin(u(1)*exp(2.3*(-u(2))))` till det nu aktuella uttrycket `u(1)^2`. Funktionen beskrivs alltid i termer av u, som är en vektor med insignalerna $u[1], u[2], \ldots$. I det här exemplet är u en vektor med en enda signal $u[1]$. Ett blockschema i SIMULINK kan då se ut som i figur 6, där också blocket `Sum` använts som finns under `Math`. För att byta ut de två plustecknen i `Sum`, dubbelklicka och byt ut ett av plustecknen till ett minustecken. Övertyga dig själv om varför figuren simulerar det system vi vill och att du själv kan återskapa motsvarande blockschema i SIMULINK. Kontrollera speciellt att du förstår varför man plockar ut x efter integratorn. Prova också att simulera.

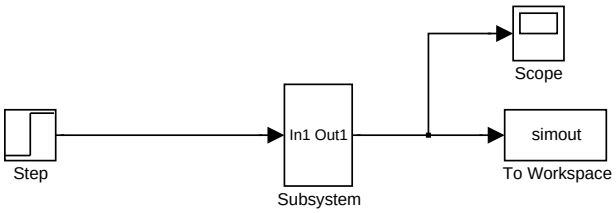
Strukturera genom att skapa delmodeller

Större SIMULINK-modeller är svåröverskådliga om man inte strukturerar dem. Ett bra sätt är då att skapa delmodeller. I figur 6 ovan utgör blocken

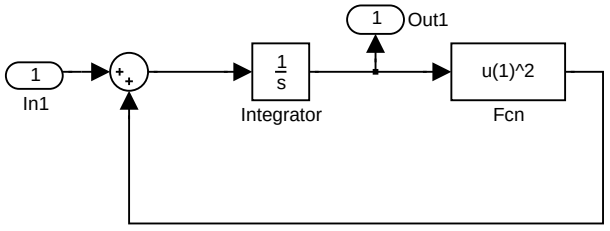


Figur 6: Ett olinjärt system.

`Sum`, `Integrator` och `Fcn` en överföringsfunktion. Låt oss göra en delmodell av de tre blocken. Markera blocken genom att trycka ner vänster musknapp och innesluta dem i den rektangel som bildas då man drar musen med knappen nedtryckt. Gå in i menyn `Edit` och välj `Create Subsystem`, vilket ger figur 7. Dubbelklickar man på `Subsystem` öppnas ett nytt fönster med delmodellen, se figur 8. Detta nya block kan nu betraktas som "vilket annat SIMULINK-block som helst" och kan till exempel kopieras och användas i en annan modell.



Figur 7: Ett system med en delmodell.



Figur 8: Innehållet i blocket "Subsystem".

Val av numerisk lösare

Ett svårighet vid simuleringar är att välja rätt numerisk lösare. I SIMULINK bestäms lösare genom att välja `Simulation` och därefter `Configuration Parameters`. Lösarna finns under fliken `Solver`. De delas först upp i lösare med fix eller variabel steglängd. Om lösare med variabel steglängd används kan de maximala felen specificeras. De med fix steglängd är något snabbare.

Lösarna kan också delas in i enstegs- och flerstegslösare. En enstegslösare använder vid varje iterationssteg endast de värden den räknade ut vid föregående steg. En flerstegslösare även värden den räknat ut tidigare. Generellt kan man säga att en enstegslösare är snabbare men mindre noggrann.

Lösare med variabel steglängd:

- Standardlösaren `ode45` är en enstegslösare, baserad på en explicit Runge-Kutta-algoritm. Det är generellt en mycket bra lösare och vi rekommenderar att den används i första hand. `ode23` är likvärdig.
- `ode113` är en Adams-Bashforth-Moulton-lösare. Det är en flerstegslösare och är bra om man vill ha små feltoleranser. Den är långsammare än `ode45` och `ode23`.
- `ode15s` är också en flerstegslösare. Den använder en numerisk deriveringsalgoritm, NDF, och är effektiv för styva problem.
- `ode23s` är en enstegslösare, baseras på en Rosenbrockalgoritm. Även denna lösare är bra för styva problem. Då den är en enstegslösare är den snabbare men mindre noggrann än `ode15s`.
- För problem som saknar kontinuerliga tillstånd finns lösaren `discrete`.

Lösare med fix steglängd:

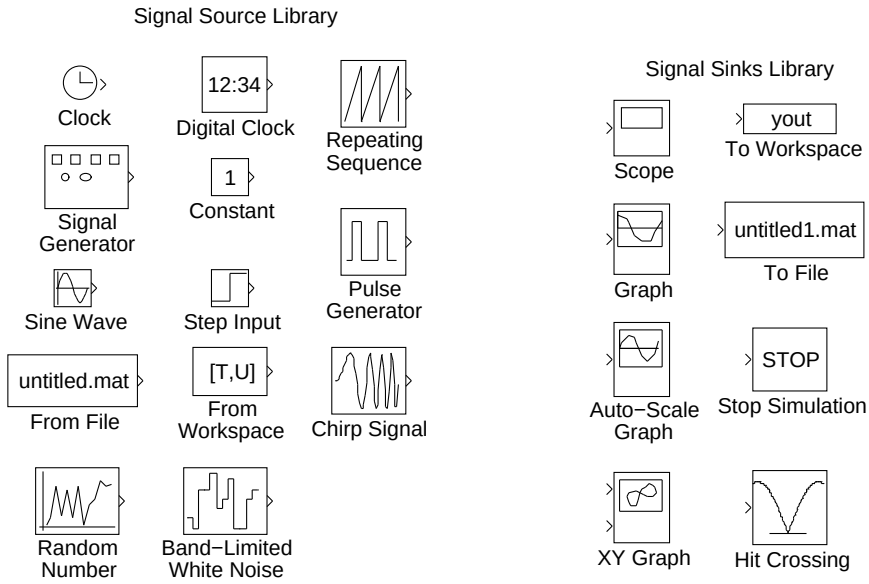
- `ode5` är en variant av `ode45` med fix steglängd. `ode4` är precis som `ode5` en Runge-Kuttalösare, fast av ordning 4. `ode3` är `ode23` med fixerad steglängd.
- `ode1` använder Eulers metod. Det är en enstegslösare som är användbar för små, enkla problem. `ode2` använder en något förbättrad version av Eulers metod.
- Även lösaren `discrete` finns med fix steglängd.

Block i Simulink

Vi går nu igenom de block som finns tillgängliga och ger korta kommentarer till speciellt användbara block. Funktionen hos de block som inte nämns kan man själv lätt kolla genom att dubbelklicka på dem. Varje block har en hjälptext som kort beskriver vad blocket gör och hur det används.

Först ut är signalkällorna, det vill säga `Sources`, se figur 9. De mest användbara blocken här är förutom de som redan nämnts `Constant`, `Sine Wave` och `Signal Generator`. Det sistnämnda blocket innehåller möjligheter att välja ett antal olika insignaler, som exempelvis fyrkantvåg.

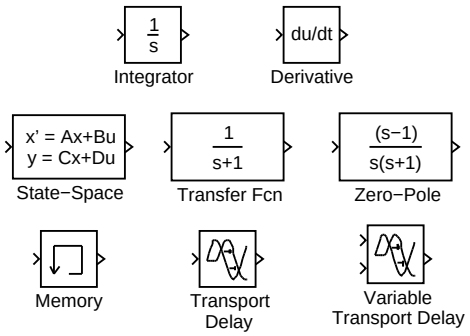
Studerar vi sedan "sänkorna" i `Sinks`, ger de oss möjlighet att titta på olika signaler i de system som simuleras. Blocken `To Workspace` och `Scope` har redan undersökts i exemplen. I `XY Graph` kan man plotta två signaler mot varandra, vilket är speciellt användbart då man ska rita fasplan.



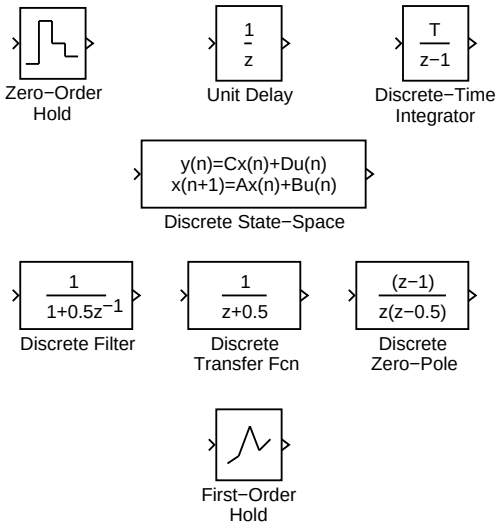
Figur 9: Block i biblioteken `Sources` och `Sinks`.

I biblioteket `Continuous`, figur 10, finns block med vilkas hjälp olika former av tidskontinuerliga system kan simuleras. Här finns ett antal sätt att beskriva tidskontinerliga överföringsfunktioner och tillståndsbeskrivningar. I biblioteket `Discrete`, figur 11, finns i princip samma sak för tidsdiskreta system.

I `Math` i figur 12 finns det mycket användbara block som `Sum` för summering och `Gain` för förstärkning (multiplikation med en konstant). Observera att

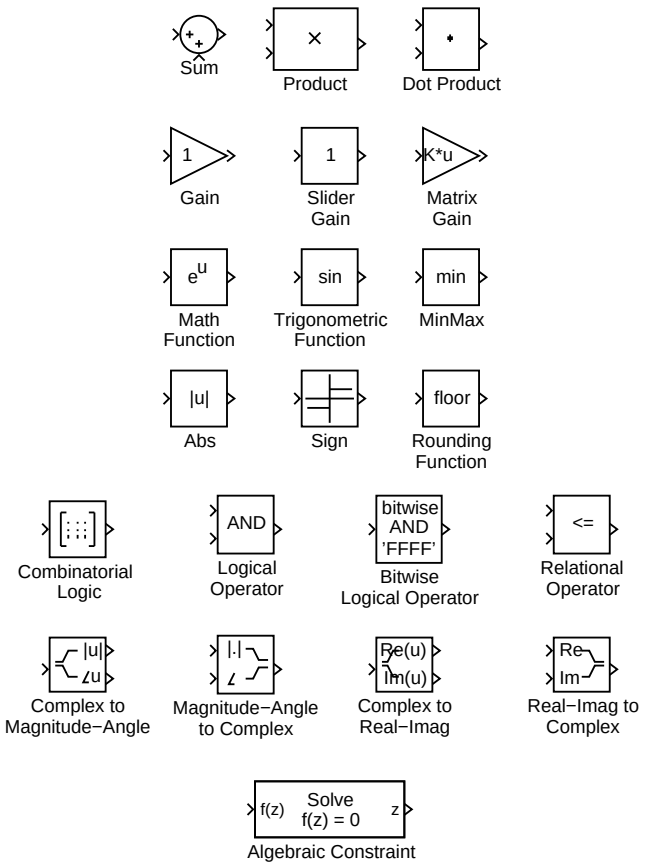


Figur 10: Block i biblioteket `Continuous`.



Figur 11: Block i biblioteket `Discrete`.

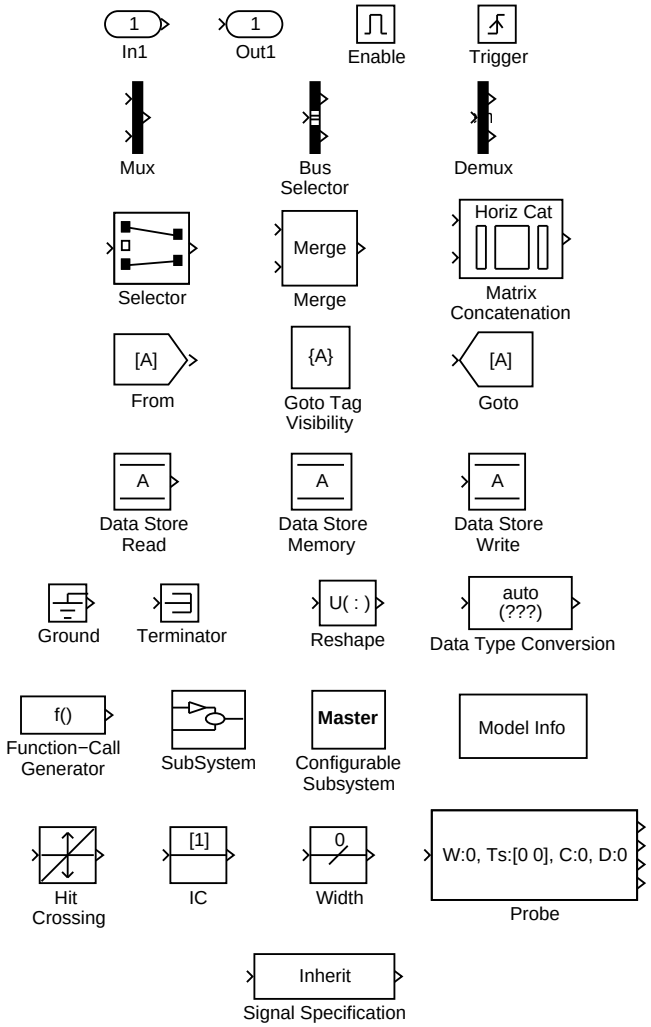
genom att lägga till fler plus- eller minustecken i `Sum` kan man addera fler signaler är bara två. `Product` utför multiplikation mellan två signaler är också mycket användbar och ändras på samma sätt som `Sum`. I biblioteket hittar man också mer avancerade funktioner, som till exempel `Sign` (signum).



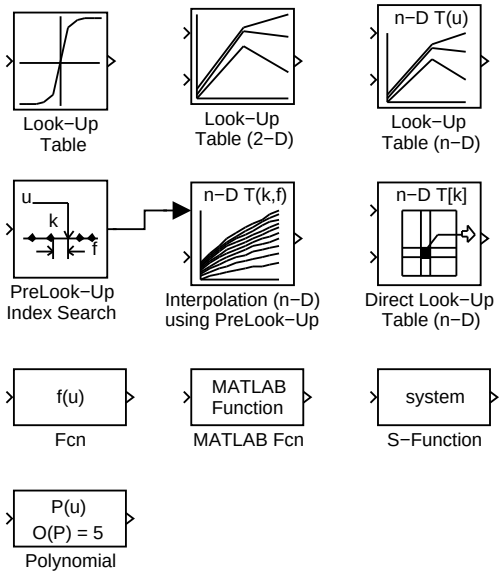
Figur 12: Block i biblioteket `Math`.

Hur slår man ihop flera insignaler till en vektor som bara beskrivs med hjälp av en pil? Detta tas hand om av biblioteket `Signal & Systems` i figur 13. Vill man beskriva en vektor av flera signaler används `Mux` och vill man dela upp en vektor i sina komponenter använder man `Demux`. Givetvis kan man specificera hur många in- respektive utsignaler man vill ha. `Signals & Systems` innehåller även annat blocken `In1` och `Out1` som används om man vill definiera egna block, till exempel delsystem av ett större system.

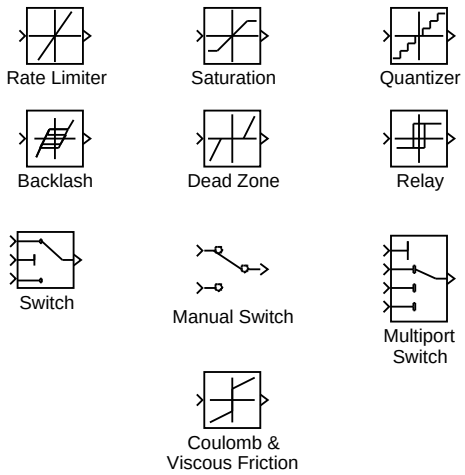
I `Functions & Tables`, figur 14, hittar man blocket `Fcn` vilket är det mest generella. Där kan man ange i princip vilken funktion som helst. Det enda man måste komma ihåg är att funktionerna måste vara i termer av `u`. Om man har flera insignaler anges de som `u(1),u(2)` och så vidare.



Figur 13: Block i biblioteket `Signals & Systems`.



Figur 14: Block i biblioteket `Functions & Tables`.



Figur 15: Block i biblioteket `Nonlinear`.

I `Nonlinear` i figur 15 kan man hitta olika olinjära fenomen, som exempelvis mättning och dödzon. Här finns också en manuell switch, som kan vara bra att ha när man vill växla manuellt mellan två olika insignaler.

Den noggranne teknologen har antagligen redan noterat att det finns ytterligare blockbibliotek. I det som heter `Blocksets & Toolboxes` finns block för mer avancerade användningar av SIMULINK som inte tas upp här. Den som är intresserad uppmanas att botanisera bland dessa block på egen hand. I `Demos` kan du se exempel på hur SIMULINK kan användas.