Scuola Universitaria Professionale
della Svizzera Italiana

Dipartimento
di informatica
ed elettronica

Scrittura e utilizzo di S-Function con SIMULINK

Ing. Roberto Bucher

7 aprile 2003

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Le S-Function mettono a disposizione dell'utente di Simulink un potente mezzo per ampliare notevolmente il sistema. In effetti si potrebbe dire che ogni singolo blocco di Simulink in effetti altro non è che una S-Function.
Le S-Function utilizzano una sintassi particolare che permette loro di integrarsi perfettamente nei metodi di simulazione di Simulink.
Una S-Function non è altro che un blocco con un certo numero di entrate (anche nulle) e un certo numero di uscite (anche nulle). Al suo interno possono trovarsi degli stati continui o discreti che possono poi essere risolti e integrati dal "Solver" di Simulink secondo lo schema di figura 1.1.
Durante la simulazione Simulink chiama ripetutamente le procedure all'interno della S-Function secondo le necessità:

Inizializzazione chiamata all'inizio della simulazione

- inizializzazione della struttura di simulazione (entrate, uscite, stati ecc.)
- inizializzazione dei tempi (campionamento ecc.
- inizializzazione delle matrici e delle variabili temporanee

Calcolo del prossimo tempo di elaborazione per i sistemi a passo variabile di integrazione

Calcolo delle uscite al "major time step"

Update degli stati discreti al "major time step"

Integrazione nel caso di sistemi con stati continui

Il sistema lavora sempre in questo modo, indipendentemente dal fatto che le S-Function siano state implementate come M-Files o come C-MEX.
Nella realizzazione tramite C-MEX tutte queste attività vengono realizzate tramite funzioni distinte, mentre nel caso di M-Files, la differenziazione viene fatta tramite flags (vedi tabella 1.1).

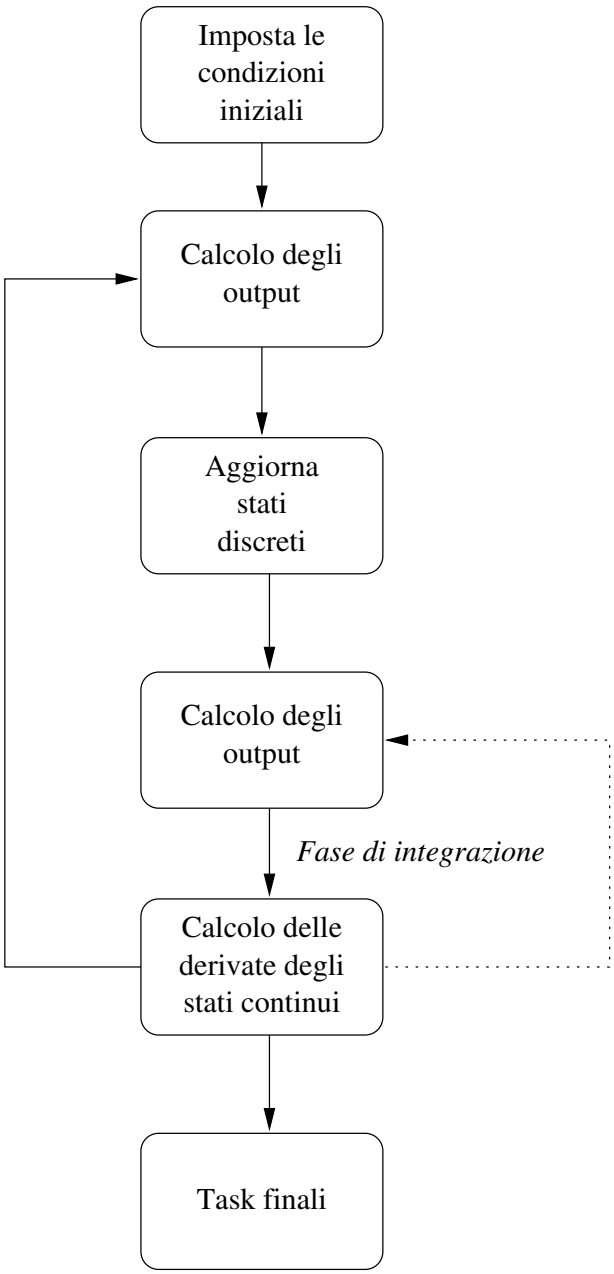

Figura 1.1: Schema generale del funzionamento di una S-Function

Fase di simulazione	**C-MEX S-Function**	**M-File S-Function**
Inizializzazione	mdlInitializeSizes	flag=0
Prossimo passo	mdlGetTimeOfNextVarHit	flag=4
Output	mdlOutputs	flag=3
Update stati discreti	mdlUpdate	flag=2
Calcolo derivate	mdlDerivatives	flag=1
Fine simulazione	mdlTerminate	flag=9

Tabella 1.1: Chiamata alle fasi di simulazione

Capitolo 2

M-File e C-MEX S-Function

2.1 Introduzione

Le S-Function possono essere programmate senza problemi tramite M-Files o C-MEX Files. In entrambi i casi, le funzioni create vanno integrate in un modulo di Simulink chiamato "S-Function", passando eventuali parametri supplementari nella finestra di dialogo del blocco.
In generale un blocco con una S-Function viene poi mascherato (vedi script di Simulink) generando cosìuna sua maschera specifica per l'immissione dei parametri.

2.2 M-File S-Function

Una S-Function di questo tipo viene creata utilizzando unicamente funzioni da Matlab, ottenendo n file di tipo ASCII con tutte le funzioni necessarie.
La funzione ha sempre la chiamata seguente:

```
function [sys,x0,str,ts] = my-sfunction(t,x,u,flag,par1,par2,...)
```

dove $par1, par2, ...$ sono parametri opzionali che si possono passare alla funzione.
Nella funzione principale viene fatta la distinzione tramite il parametro $flag$ sulle attività da svolgere, tramite un controllo di tipo *switch*.
Analizziamo in dettagli l'esempio seguente che si trova sotto "../toolbox/simulink/blocks/timestwo.m".

```
function [sys,x0,str,ts] = timestwo(t,x,u,flag)
%TIMESTWO S-function whose output is two times its input.
%   This M-file illustrates how to construct an M-file S-function that
%   computes an output value based upon its input.  The output of this
%   S-function is two times the input value:
%
%     y = 2 * u;
%
%   See sfuntmpl.m for a general S-function template.
%
%   See also SFUNTMPL.

%   Copyright 1990-2001 The MathWorks, Inc.
%   $Revision: 1.6 $

%
% Dispatch the flag. The switch function controls the calls to
% S-function routines at each simulation stage of the S-function.
```

```
%
switch flag,
  %%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%
  % Initialize the states, sample times, and state ordering strings.
  case 0
    [sys,x0,str,ts]=mdlInitializeSizes;

  %%%%%%%%%%
  % Outputs %
  %%%%%%%%%%
  % Return the outputs of the S-function block.
  case 3
    sys=mdlOutputs(t,x,u);

  %%%%%%%%%%%%%%%%%%
  % Unhandled flags %
  %%%%%%%%%%%%%%%%%%
  % There are no termination tasks (flag=9) to be handled.
  % Also, there are no continuous or discrete states,
  % so flags 1,2, and 4 are not used, so return an emptyu
  % matrix
  case { 1, 2, 4, 9 }
    sys=[];

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Unexpected flags (error handling)%
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Return an error message for unhandled flag values.
  otherwise
    error(['Unhandled flag = ',num2str(flag)]);

end

% end timestwo

%
%=============================================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function.
%=============================================================================
%
function [sys,x0,str,ts] = mdlInitializeSizes()

sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = -1;  % dynamically sized
sizes.NumInputs      = -1;  % dynamically sized
sizes.DirFeedthrough = 1;   % has direct feedthrough
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);
str = [];
x0  = [];
ts  = [-1 0];   % inherited sample time

% end mdlInitializeSizes

%
%=============================================================================
% mdlOutputs
% Return the output vector for the S-function
%=============================================================================
%
function sys = mdlOutputs(t,x,u)
```

```
sys = u * 2;

% end mdlOutputs
```

I parametri in entrata sono:

t L'attuale tempo di simulazione

x I valori attuali degli stati

u Il valore dell'entrata

flag Il flags che determina l'attività attuale da svolgere nella funzione.

Questi 4 parametri sono passati in entrata da parte di simulink ogni volta che la funzione deve essere effettuata.
I valori in uscita dipendono dalla funzione svolta. Ad esempio "sys" è un valore in uscita generico dipendente dal flag. Con $flag = 3$ ad esempio "sys" contiene i valori in uscita.
Come si può vedere nell'esempio la funzione principale "smista" le attività da effettuare in base al flag ricevuto da Simulink, chiamando le varie funzioni necessarie. All'inizio ($flag = 0$) viene chiamata la funzione "mdlInitializeSizes" che inizializza il sistema memorizzando il numero di entrate e uscite, il numero di stati continui e discreti, le condizioni iniziali e i tempi di campionamento.
In seguito viene chiamata la funzione "mdlOutputs" che calcola le uscite. Questa chiamata viene fatta da Simulink tramite il $flag = 3$ a ogni passo maggiore di integrazione. Non essendoci stati discreti o continui, la chiamata ad altre funzioni non è necessaria.
Il prossimo esempio rappresenta un sistema dato nella forma generica nello spazio degli stati.

$$\begin{aligned} \dot{\mathbf{x}} &= A\mathbf{x} + B\mathbf{u} \\ \mathbf{y} &= C\mathbf{x} + D\mathbf{u} \end{aligned}$$

Si può ben vedere come in questo esempio vengano chiamate tutte le varie funzioni durante l'integrazione. Le 4 matrici del sistema vengono passate come parametri supplementari.

```
function [sys,x0,str,ts]=simom2(t,x,u,flag,A,B,C,D)
%SIMOM2 Example state-space M-file S-function with external A,B,C,D matrices
%   This S-function implements a system described by state-space equations:
%
%           dx/dt = A*x + B*u
%               y = C*x + D*u
%
%   where x is the state vector, u is vector of inputs, and y is the vector
%   of outputs. The matrices, A,B,C,D are provided externally as parameters
%   to this M-file.
%
%   See sfuntmpl.m for a general S-function template.
%
%   See also SFUNTMPL.

%   Copyright 1990-2001 The MathWorks, Inc.
%   $Revision: 1.19 $

switch flag,
```

```
  %%%%%%%%%%%%%%%%%%
  % Initialization %
  %%%%%%%%%%%%%%%%%%
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes(A,B,C,D);

  %%%%%%%%%%%%%%%
  % Derivatives %
  %%%%%%%%%%%%%%%
  case 1
    sys = mdlDerivatives(t,x,u,A,B,C,D);

  %%%%%%%%%%
  % Outputs %
  %%%%%%%%%%
  case 3
    sys = mdlOutputs(t,x,u,A,B,C,D);

  %%%%%%%%%%%%
  % Terminate %
  %%%%%%%%%%%%

  case 9
    sys = []; % do nothing

end

%end simom2

%
%=============================================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-function.
%=============================================================================
%
function [sys,x0,str,ts] = mdlInitializeSizes(A,B,C,D)

sizes = simsizes;

sizes.NumContStates  = size(A,1);
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = size(D,1);
sizes.NumInputs      = size(D,2);
sizes.DirFeedthrough = any(any(D~=0));
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);
x0  = ones(size(A,1),1);
str = [];
ts  = [0 0];

% end mdlInitializeSizes

%
%=============================================================================
% mdlDerivatives
% Return the derivatives for the continuous states.
%=============================================================================
%
function sys = mdlDerivatives(t,x,u,A,B,C,D)

sys = A*x + B*u;

% end mdlDerivatives

%
```

```
%=============================================================================
% mdlOutputs
% Return the output vector for the S-function
%=============================================================================
%
function sys = mdlOutputs(t,x,u,A,B,C,D)

sys = C*x + D*u;

% end mdlOutputs
```

2.3 C-MEX S-Function

Una C-MEX S-Function si differenzia da una M-File S-Function per il fatto che viene programmata in "C" e in seguito compilata, con il vantaggio di poter essere eseguita molto più velocemente durante l'elaborazione.
Per la compilazione sotto "Unix" si utilizza il normale GNU C Compiler, mentre la compilazione in ambiente "Windows" può essere fatta utilizzando il compilatore all'interno di Matlab, o un altro compilatore esterno (Visual C++, Borland C++ o Watcom C++). La configurazione del compilatore viene fatta tramite il comando di Matlab

```
mex -setup
```

Una C-MEX S-Function viene compilata tramite il comando

```
mex my-sfunction.c
```

e crea un file che campia estensione a seconda del sistema operativo utilizzato. Sotto Windows, ad esempio, viene creata una "DLL". Il file compilato deve trovarsi nel "PATH" di Matlab e Simulink, e può quindi essere utilizzato in un blocco.
Vediamo come la funzione "timestwo" viene implementata come C-MEX S-Function.

```
/*
 * File : timestwo.c
 * Abstract:
 *       An example C-file S-function for multiplying an input by 2,
 *       y  = 2*u
 *

#define S_FUNCTION_NAME  timestwo
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"

/* Function: mdlInitializeSizes ===============================================
 * Abstract:
 *   Setup sizes of the various vectors.
 */
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return; /* Parameter mismatch will be reported by Simulink */
    }

    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, DYNAMICALLY_SIZED);
    ssSetInputPortDirectFeedThrough(S, 0, 1);
```

```
    if (!ssSetNumOutputPorts(S,1)) return;
    ssSetOutputPortWidth(S, 0, DYNAMICALLY_SIZED);

    ssSetNumSampleTimes(S, 1);

}

/* Function: mdlInitializeSampleTimes =========================================
 * Abstract:
 *    Specifiy that we inherit our sample time from the driving block.
 */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}

/* Function: mdlOutputs =======================================================
 * Abstract:
 *    y = 2*u
 */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    int_T             i;
    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
    real_T            *y    = ssGetOutputPortRealSignal(S,0);
    int_T             width = ssGetOutputPortWidth(S,0);

    for (i=0; i<width; i++) {
        /*
         * This example does not implement complex signal handling.
         * To find out see an example about how to handle complex signal in
         * S-function, see sdotproduct.c for details.
         */
        *y++ = 2.0 *(*uPtrs[i]);
    }
}

/* Function: mdlTerminate =====================================================
 * Abstract:
 *    No termination needed, but we are required to have this routine.
 */
static void mdlTerminate(SimStruct *S)
{
}

#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

Il numero delle entrate, delle uscite, e il valore del tempo (continuo o campionato), vengono direttamente lette dallo schema ("INHERITED").
Vediamo ora un esempio più complesso, rappresentato da una S-Function utilizzata per dialogare con una scheda di acquisizione tramite un driver. In questo caso ci sono alcuni parametri da passare (numero di entrate, numero di uscite, tempo di campionamento).

```
#define S_FUNCTION_NAME  daqepp
#define S_FUNCTION_LEVEL 2

#ifdef MATLAB_MEX_FILE
#include "mex.h"      /* needed for declaration of mexErrMsgTxt */
#endif
```

```
/*
#include "eppio.h"
*/

#include "simstruc.h"

#define n_out   ssGetSFcnParam(S,0)   /* Number of DAQ inputs */
#define n_in    ssGetSFcnParam(S,1)   /* Number of DAQ inputs */
#define device  1
#define gain    1

static void mdlInitializeSizes(SimStruct *S)
{
int_T n1,n2;

    ssSetNumSFcnParams(S, 2);
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return; /* Parameter mismatch will be reported by Simulink */
    }

    n1=(int_T) mxGetPr(n_out)[0];
    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, n1);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    n2=(int_T) mxGetPr(n_in)[0];
    if (!ssSetNumOutputPorts(S,1)) return;
    ssSetOutputPortWidth(S, 0, n2);

    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 0);
    ssSetNumSampleTimes(S, 1);

    InitializeEPP();
}

static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}

static void mdlOutputs(SimStruct *S, int_T tid)
{
    int_T             i;
    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
    real_T            *y    = ssGetOutputPortRealSignal(S,0);
    int_T       n1    = ssGetInputPortWidth(S,0);
    int_T             n2    = ssGetOutputPortWidth(S,0);
    real_T       volt_in;

   for(i=0;i<n1;i++) {
     volt_in=*uPtrs[i];
      if(WriteEPP(i,volt_in)!=0)
  mexErrMsgTxt("Error by writing to EPP");
    }

    for(i=0;i<n2;i++) {
      if(ReadEPP(i,&volt_in)!=0)
          mexErrMsgTxt("Error by reading from EPP");
      else
  *y++=volt_in;
    }
}

static void mdlTerminate(SimStruct *S)
```

```
{
int_T err,i;

    for(i=0;i<2;i++) {
        if(WriteEPP(i,0.0)!=0) mexErrMsgTxt("Error by writing to EPP");
    }
    CloseEPP();
}


#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```