

Simulaciones en Tiempo Real

utilizando el paquete

RTW de Matlab

Simulación de Sistemas de Control (66.55) - UBA


ORGANIZACIÓN DE LA PRESENTACION

- SIMULINK
- S-FUNCTIONS
- RTW
- REAL TIME WINDOWS TARGET

SIMULINK

- Qué es y para qué sirve. Arranque. Directorio de trabajo. Librerías
- Modelos dinámicos continuos y discretos.
- Bloques discretos / continuos.
- Señales entre bloques. Escalares, Vectoriales
- Configuración de una simulación.

PARÁMETROS DE SIMULACIÓN

 **Simulation Parameters: prueba** [minimize] [maximize] [close]

Solver Workspace I/O Diagnostics Advanced Real-Time Workshop

Simulation time

Start time: Stop time:

Solver options

Type:

Max step size: Relative tolerance:

Min step size: Absolute tolerance:

Initial step size:

Output options

 Refine factor:

OK Cancel Help Apply

PARÁMETROS DE SIMULACIÓN

- Tiempo de simulación. Simulaciones infinitas
- Resolución por paso fijo y paso variable.
- Métodos de integración. Para qué usar diferentes métodos.
 - Paso Variable:
 - ode45: Es el algoritmo más apto en primera instancia. (paso simple)
 - ode23: Es más eficiente para tolerancias amplias. (paso simple)
 - ode113: Es más eficiente para tolerancias estrictas. (multi paso)
 - ode15s: Es el algoritmo más apto en primera instancia, para sistemas mal condicionados. (multi paso)
 - ode23s: Más eficiente para sistemas mal condicionados, con tolerancias amplias. (paso simple)
 - discrete: No es necesario resolver integración numérica

PARÁMETROS DE SIMULACIÓN

- Paso Fijo:

ode5: Versión de paso fijo del ode45.

ode4: Runge –Kutta de orden 4.

ode3: Versión de paso fijo del ode23.

ode2: Método de Jun, o Euler mejorado.

ode1: Método de Euler.

discrete: Método cuando no es necesario utilizar un integrador numérico.

PARÁMETROS DE SIMULACIÓN

- Control de errores en paso variable: Máximo paso, Mínimo paso, Paso inicial. Tolerancia relativa y absoluta. $e_i \leq \max(\text{rtol } |x_i|, \text{Atol}_i)$
- Opciones de paso fijo: Tamaño de paso y Modo.
- Opciones de Salida: Refinar salida, Generar salidas adicionales, Generar salidas específicas.

Parámetros Workspace I/O

Simulation Parameters: untitled

Solver | **Workspace I/O** | Diagnostics | Advanced | Real-Time Workshop

Load from workspace

☐ Input: [t, u]

☐ Initial state: xInitial

Save to workspace

☒ Time: tout

☐ States: xout

☒ Output: yout

☐ Final state: xFinal

Save options

☒ Limit data points to last: 1000

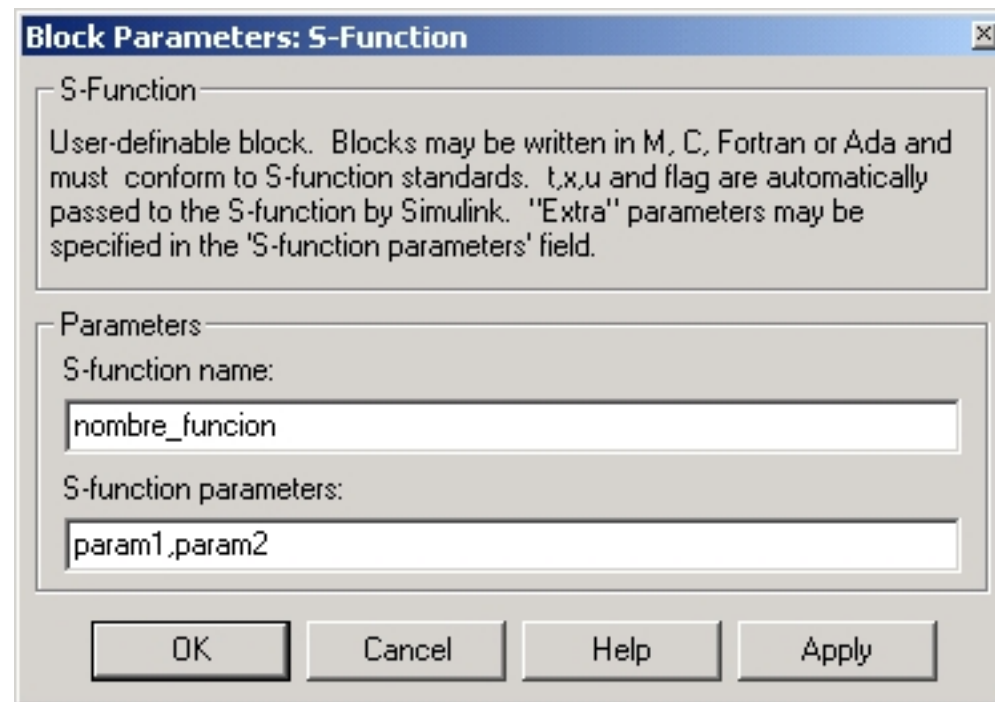
Decimation: 1

Format: Array

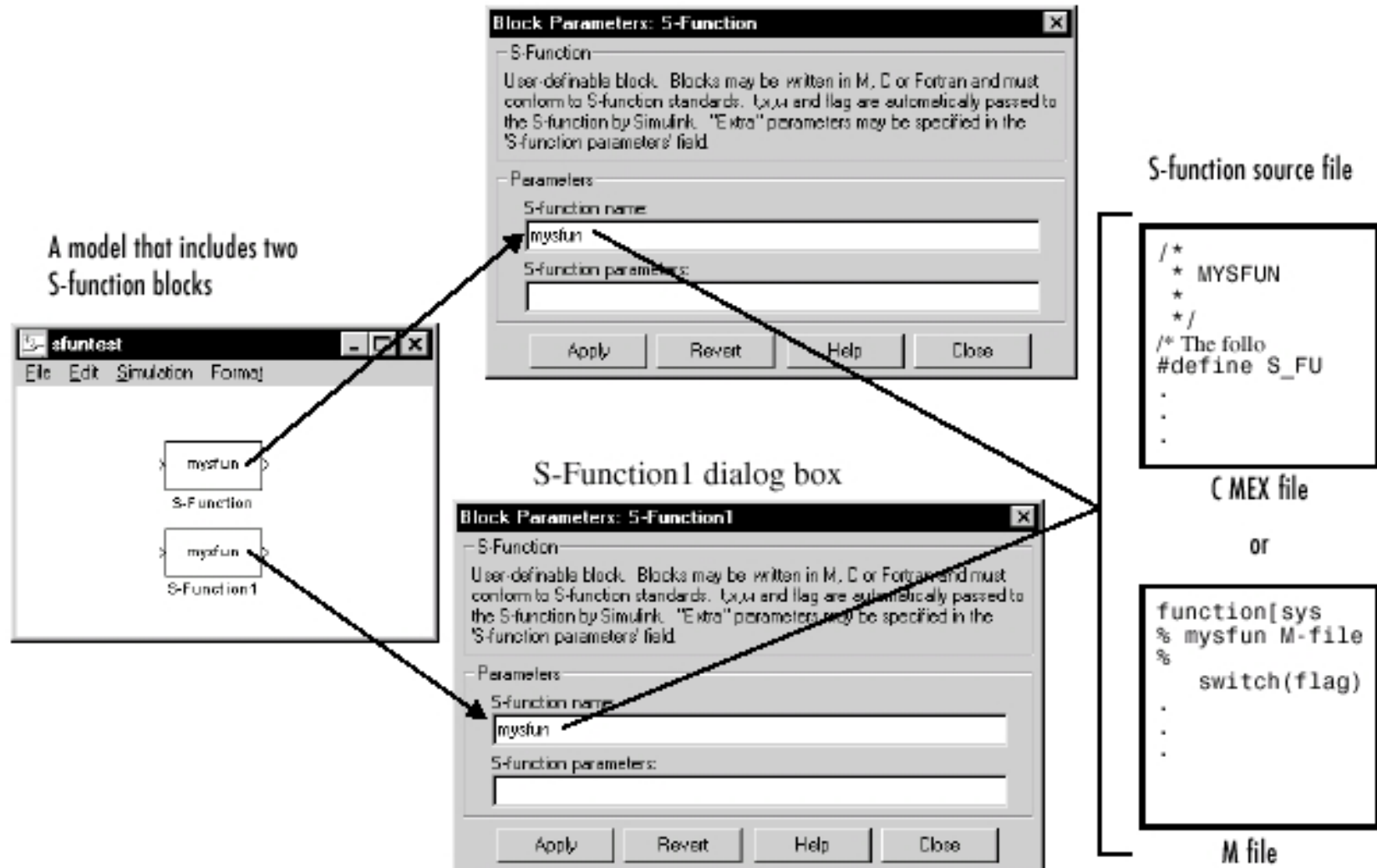
OK Cancel Help Apply

S-FUNCTIONS

- Para qué sirven?.
- Tipos de S-Functions: M-File (en lenguaje Matlab). C-MEX (en lenguaje “C”).
- Bloque de Simulink para las S-Functions. Librería User-Defined Functions.

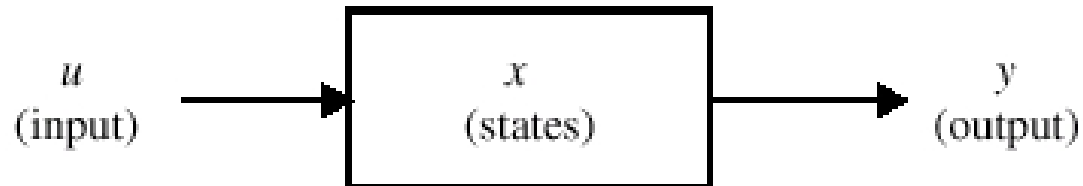


S-FUNCTIONS



S-FUNCTIONS

- Cómo interactúa simulink con cada bloque del modelo: Ganancia, Integrador, S-Functions.



The following equations express the mathematical relationships between the inputs, outputs, and the states.

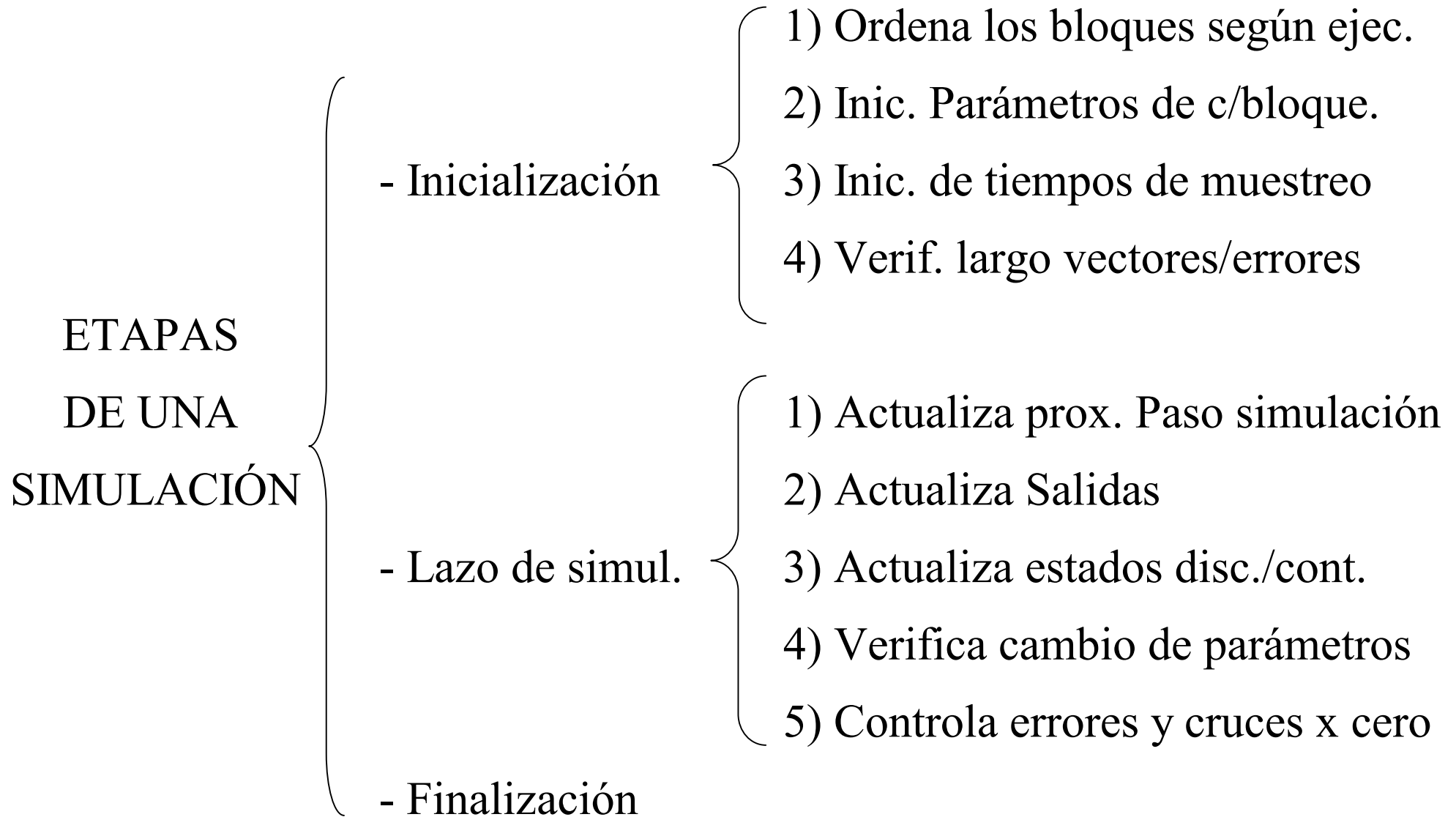
$$y = f_0(t, x, u) \quad (\text{Output})$$

$$\dot{x}_c = f_d(t, x, u) \quad (\text{Derivative})$$

$$x_{d_{k+1}} = f_u(t, x, u) \quad (\text{Update})$$

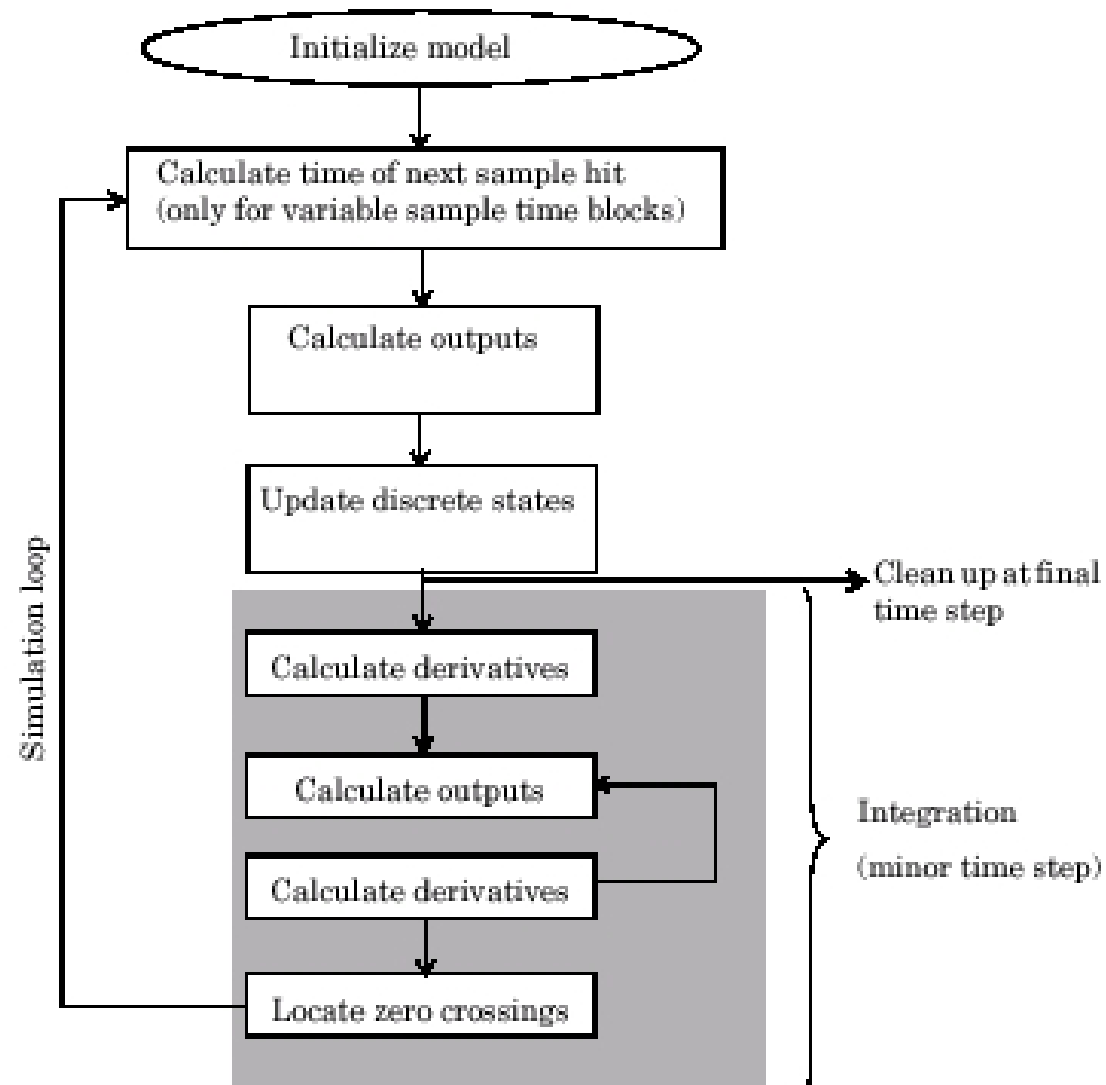
$$\text{where } x = x_c + x_d$$

S-FUNCTIONS



S-FUNCTIONS

The following figure illustrates the stages of a simulation.



S-Functions

- Se pueden escribir en lenguaje Matlab, C, C++, Fortran o ADA
- Simulink (en modo normal) las ejecuta a través de librerías de enlace dinámico o DLL's
- En lenguaje Matlab -> M-file S-Function. En "C" -> C-MEX S-Function.
- Para crear una C-MEX puedo usar LCC, Visual C, Borland C, Watcom.
- Se utiliza el comando "**mex *nombre.c***" desde MATLAB y genera "***nombre.dll***"
- Interacción de simulink y las S-Functions. Método de Callback.

Conceptos sobre S-Functions

- Bloque Direct Feedthrough (Paso directo de señal). Debido a las salida o debido al tiempo.
- Bloques: Ancho de entradas/salidas. Cantidad de entradas / salidas. Entradas con ancho dinámico.
- Tiempos de muestreo y offset.
 - Bloques de Tiempo Continuo
 - Bloques de Tiempo Discreto
 - Bloques de Tiempo Variable
 - Bloques de Tiempo Heredado
 - Bloques de Tiempo Múltiple o Multirate

Estructura M-File S-Functions

Function [sys,X0,str,ts]=NombreFunción(t,x,u,flag)

t : tiempo simulación

x : vector de estados $[x_c;x_d]^T$

u : vector de entradas

flag : etapa de la simulación

sys : parámetro genérico

X0 : estados iniciales

str : parámetro reservado

ts : matriz de muestreo y offset

- | | |
|--------------------------------------|--------|
| • Inicialización: | flag=0 |
| • Próximo Hit: | flag=4 |
| • Salidas | flag=3 |
| • Actualización de Estados Discretos | flag=2 |
| • Derivadas | flag=1 |
| • Finalización | flag=9 |

Tiempos de ejecución de las S-Functions

Los tiempos de muestreo pueden ser:

- CONTINUO (período = 0, offset = 0). Ejemplo ts=[0,0]
- HEREDADO (período = -1, offset = 0). Ejemplo ts=[-1,0]
- VARIABLE (período=-2, offset=0). Ejemplo ts=[-2,0]
- DISCRETO (período>0, $0 \leq \text{offset} < \text{período}$). Ejemplo ts=[1,0.1]

Ejemplo M-file

```
function [sys,x0,str,ts] = mulx2(t,x,u,flag)
% ejemplo de S-Function que dobla su entrada
switch flag,
    case 0, % llamada p/ inicialización de S-Functions (flag=0)
        sizes = simsizes;
        sizes.NumContStates = 0; % no posee estados continuos
        sizes.NumDiscStates = 0; % no posee estados discretos
        sizes.NumOutputs = 1; % una salida
        sizes.NumInputs = 1; % una entrada
        sizes.DirFeedthrough = 1; % es de paso directo
        sizes.NumSampleTimes = 1; % un tiempo de muestreo
        sys = simsizes(sizes); % crea una estructura con datos p/ inic. la S-functions
        x0 = [];
        str = [];
        ts = [0 0]; % de tiempo continuo
    case 1, % llamada p/ calculo de derivadas (flag=1)
        sys = [];
    case 2, % llamada p/ actualización de estados discretos (flag=2)
        sys = [];
    case 3, % llamada p/ calculo de salidas (flag=3)
        sys = 2*u;
    case 4, % llamada p/ calculo de prox. hit (flag=4)
        sys = [];
    case 9, % llamada p/ tareas de finalización (flag=9)
        sys = [];
    otherwise
        error(['flag no conocido= ',num2str(flag)]);
end
```

Estructura C-Mex S-Functions. Funciones CallBack

```
#define S_FUNCTION_NAME  doble
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"

static void mdlInitializeSizes(SimStruct *S)
{
    /* inicializacion. Nro. de estados continuos/discretos. Nro. de entradas/salidas */
}

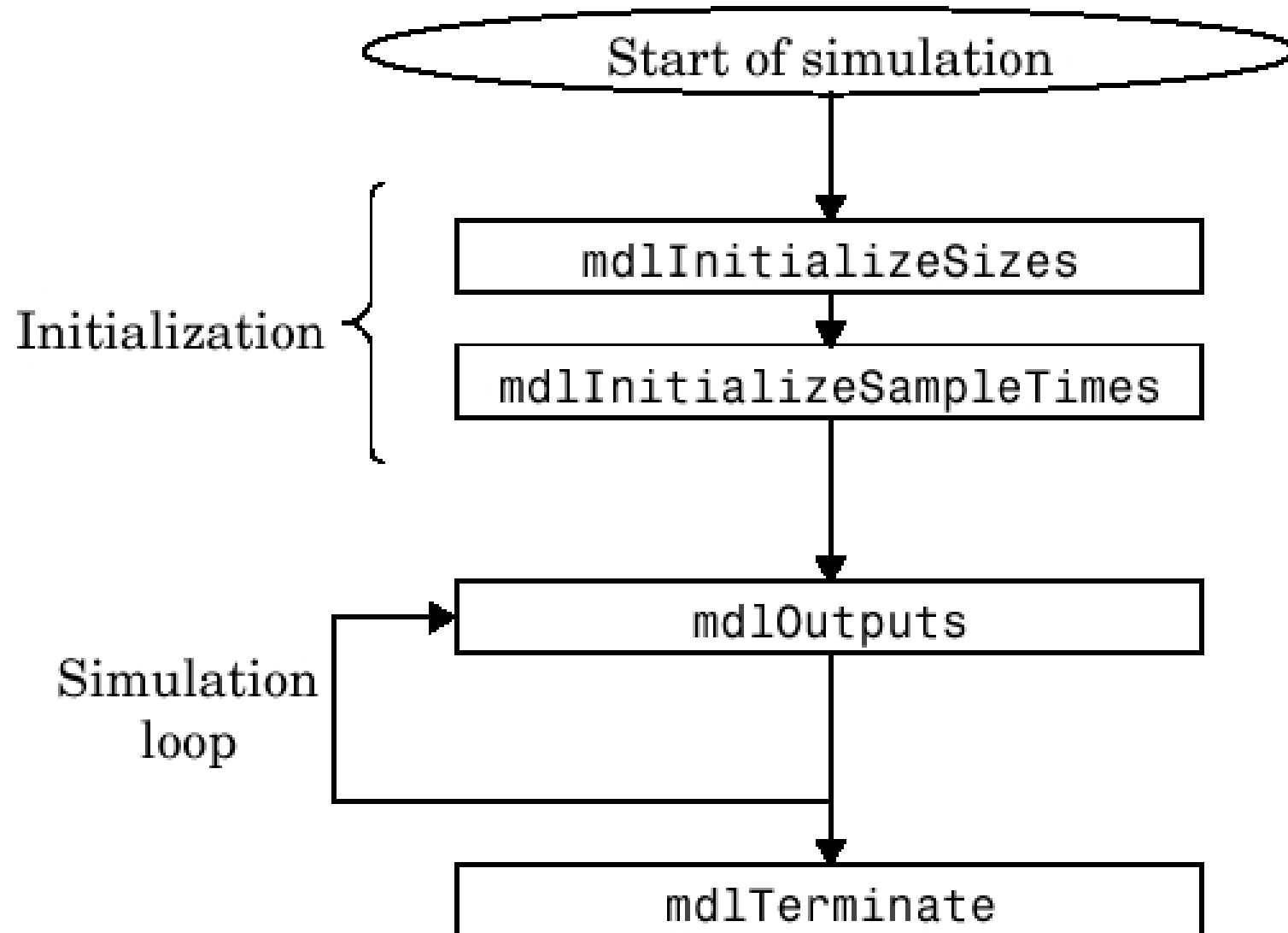
static void mdlInitializeSampleTimes(SimStruct *S)
{
    /* inicializacion de tiempos de muestreo */
}

static void mdlOutputs(SimStruct *S, int_T tid)
{
    /* actualización de salida */
}

static void mdlTerminate(SimStruct *S)
{
    /* tareas de finalización */
}

#ifdef MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cg_sfuns.h"
#endif
```

Ejemplo C-MEX S-Functions (doble entrada)



Ejemplo C-MEX S-Functions (dobla entrada)

```
#define S_FUNCTION_NAME  doble
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"

static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);          /* No recibe parámetros desde el bloque */

    ssSetNumInputPorts(S, 1);          /* Un puerto de entrada */
    ssSetInputPortWidth(S, 0, 1);      /* Ancho del único puerto de entrada es 1 */
    ssSetInputPortDirectFeedThrough(S, 0, 1); /* La función es de paso directo */

    ssSetNumOutputPorts(S, 1);          /* Un puerto de salida */
    ssSetOutputPortWidth(S, 0, 1);      /* Ancho del único puerto de salida es 1 */
    ssSetNumSampleTimes(S, 1);          /* tiene un sólo tiempo de muestreo */
    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE); /* S-Function libre excepciones */
}

static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME); /* único tiempo de muestreo es heredado */
    ssSetOffsetTime(S, 0, 0.0);              /* Offset NULO */
}
```

Ejemplo C-MEX S-Functions (dobla entrada)

```
static void mdlOutputs(SimStruct *S, int_T tid)
{
    real_T          *y = ssGetOutputPortRealSignal(S,0);    /* puntero a salida */
    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0); /* puntero a entrada */
    *y = *uPtrs[0] * 2;      /* actualiza salida. (salida = 2 x entrada) */
}

static void mdlTerminate(SimStruct *S)
{
    /* no hay código de finalización */
}

#ifdef MATLAB_MEX_FILE
#include "simulink.c"    /* caraga este archivo si es compilado como MEX-FILE */
#else
#include "cg_sfun.h"     /* caraga este archivo si no es compilada como MEX-FILE (x ej RTW) */
#endif
```

Ejemplo C-MEX S-Functions (doble entrada)

Macros definidas para diferentes tiempos de muestreo

- CONTINUOUS_SAMPLE_TIME (período=0,offset=0)
- INHERITED_SAMPLE_TIME (período=-1,offset=0)
- VARIABLE_SAMPLE_TIME (período=-2,offset=0)
- Si período>0 -> tiempo discreto ($0 \leq \text{offset} < \text{período}$)

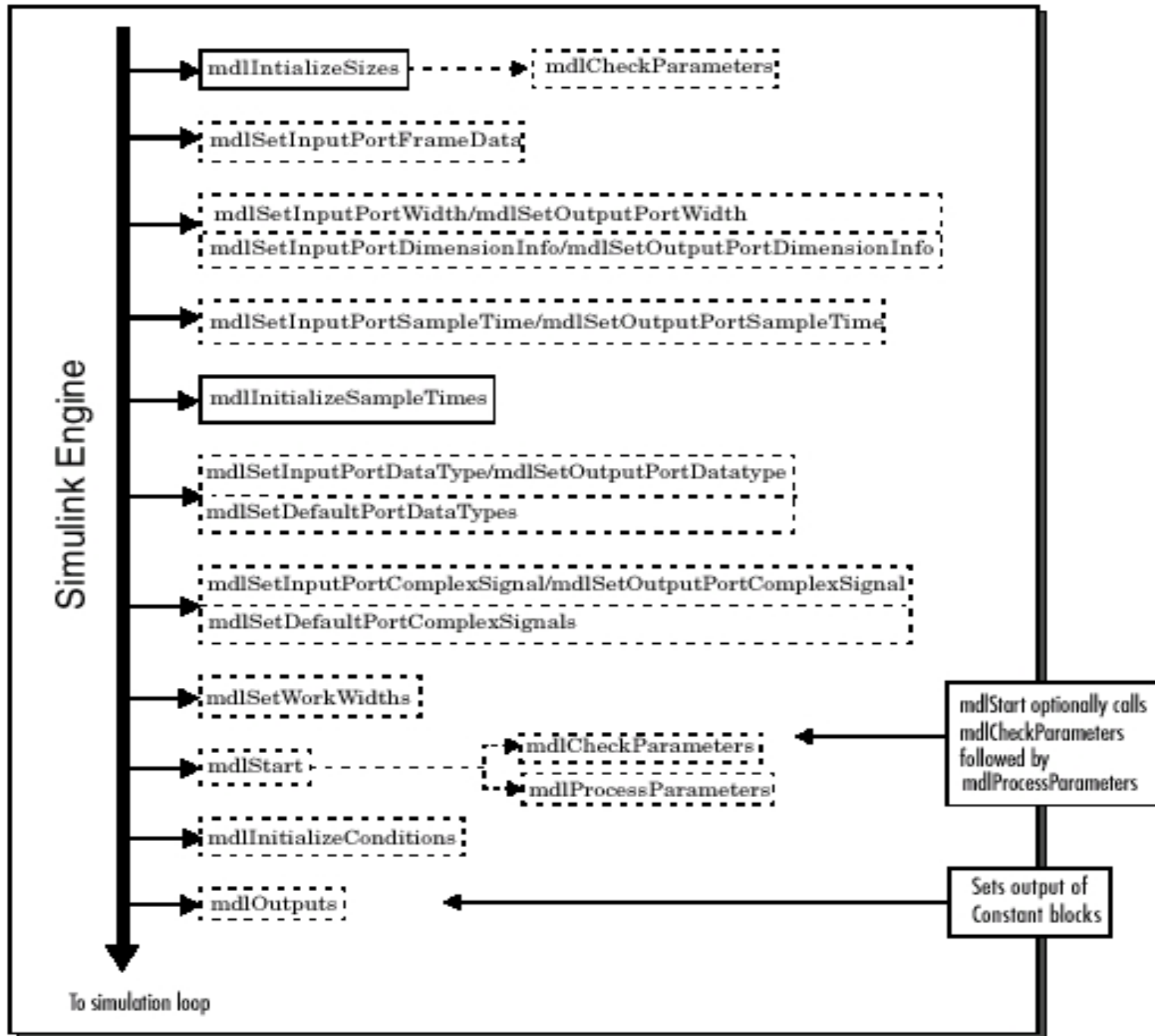
Plantilla de C-MEX S-Function

- Bajo el directorio simulink/src hay 2 plantillas: “*sfuntmpl_basic.c*” y “*sfuntmpl_doc.c*” así como varios ejemplos de C-MEX.

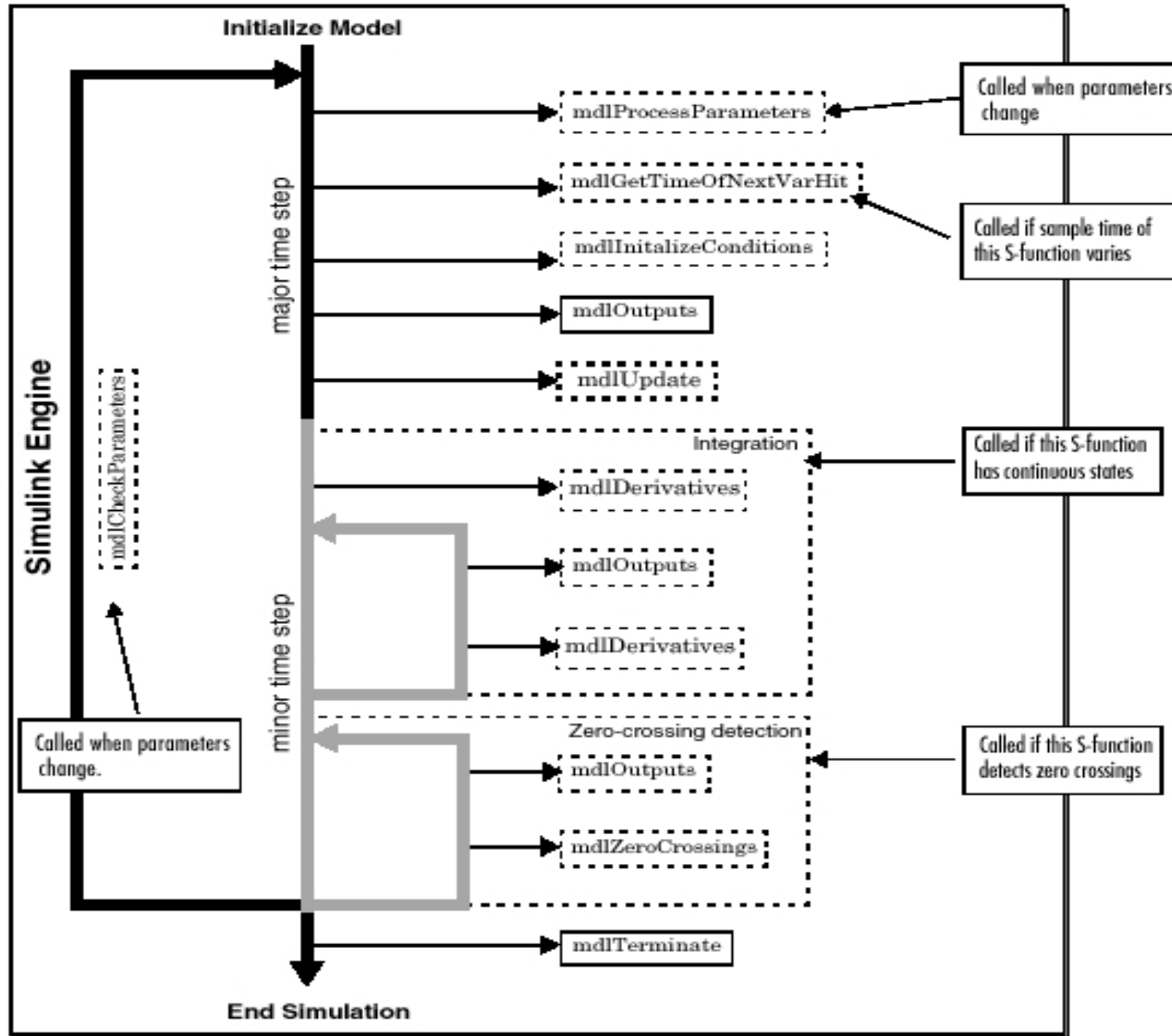
Tipos de datos pre-declarados por Simulink

- real_T, time_T, uint32_T, boolean_T, char_T, int_T, uint_T, byte_T

Tareas de Inicialización



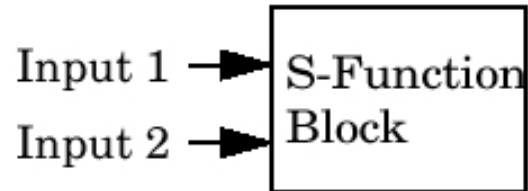
Lazo de simulación



Punteros a las entradas (doble direccionamiento)

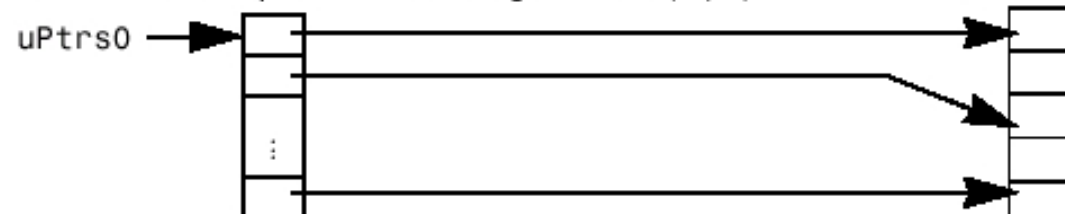
`*uPtrs[element]`

as described by this figure.



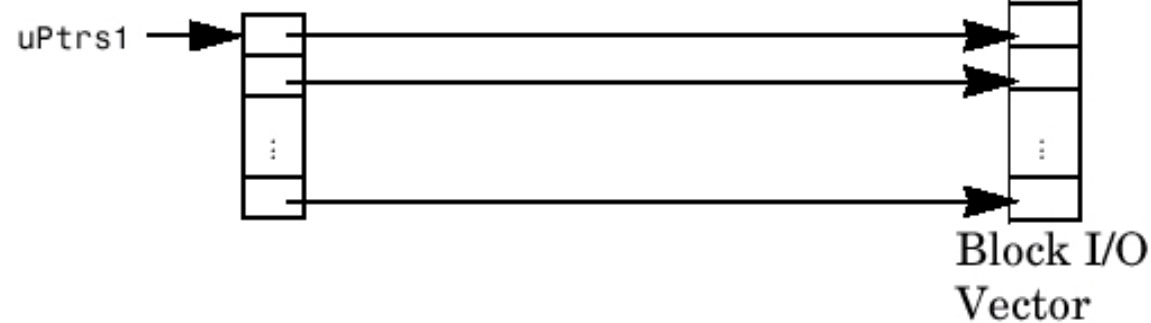
To access Input 1:

```
InputRealPtrsType uPtrs0 = ssGetInputPortRealSignalPtrs(S,0)
```



To access Input 2:

```
InputRealPtrsType uPtrs1 = ssGetInputPortRealSignalPtrs(S,1)
```



Pasando parámetros a una C-MEX S-Functions desde Simulink

```
#define S_FUNCTION_NAME param
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"
#define GAIN(S) ssGetSFcnParam(S, 0)    /* macro al primer parámetro */

static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 1);           /* cantidad de parametros */

    ssSetNumInputPorts(S, 1);           /* numero de puertos de entrada */
    ssSetInputPortWidth(S, 0, 1);       /* ancho del primer u unico puerto de entrada */
    ssSetInputPortDirectFeedThrough(S, 0, 1); /* el puerto es de paso directo */

    ssSetNumOutputPorts(S, 1);           /* numero de puertos de salida */
    ssSetOutputPortWidth(S, 0, 1);       /* ancho del puerto de salida */
    ssSetNumSampleTimes(S, 1);           /* un sólo tiempo de muestreo */
    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE); /* funcion libre de excepciones */
}

static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME); /* tiempo heredado */
    ssSetOffsetTime(S, 0, 0.0);                 /* offset 0 */
}
```

Pasando parámetros a una C-MEX S-Functions desde Simulink

```
static void mdlOutputs(SimStruct *S, int_T tid)
{
    real_T      *p;    /* declaro puntero */
    real_T      *y = ssGetOutputPortRealSignal(S,0);    /* puntero a la salida */
    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0); /* puntero a direcc. de entrada */
    p = mxGetPr(ssGetSFcnParam(S,0));    /* puntero a param. y luego obtiene el valor del param. */
    *y = *p *(*uPtrs[0]);    /* salida = paramerto * entrada */
}

static void mdlTerminate(SimStruct *S)
{
}

#ifdef MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cg_sfun.h"
#endif
```