Teoría del Cerebro y Neuroinformática

Neural Simulation Language

Maestría en Ciencias en Computación

Agenda

- Introducción
- NSLM
- Ejemplo (Selector máximo)
- NSLS
- Descarga e instalación de NSL

Metodología

- La metodología general para hacer un modelo neural complejo de la función cerebral es combinar diferentes módulos correspondientes a diferentes regiones cerebrales.

- Para modelar una región particular del cerebro, la dividimos anatómica o fisiológicamente en diferentes conjuntos neuronales.

- Cada región cerebral se modela entonces como un conjunto de matrices neuronales, donde cada neurona se describe por ejemplo por el integrador con fugas, un modelo de un solo compartimiento de potencial de membrana y velocidad de disparo.

ITAM

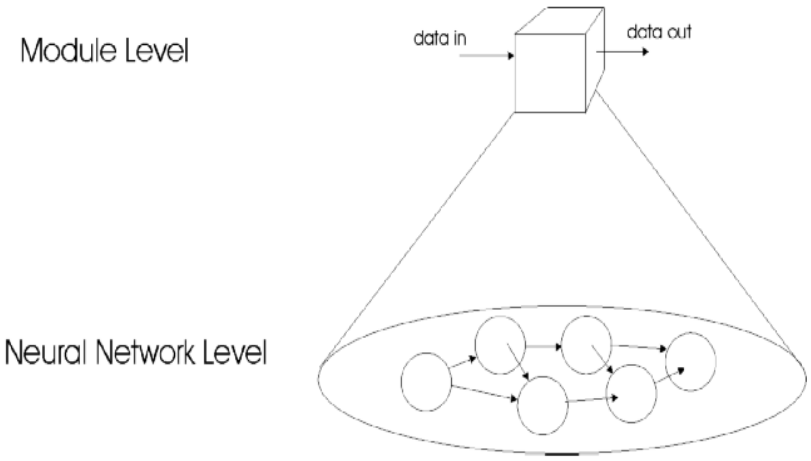
NSL

- El Lenguaje de Simulación Neuronal provee una plataforma para construir neuro-arquitecturas (modelado) y para su ejecución (simulación).
- Es un lenguaje orientado a objetos basado en el Abstract Schema Language (ASL).
- Tanto NSL como ASL fueron diseñados por Alfredo Weitzenfeld y Michael Arbib

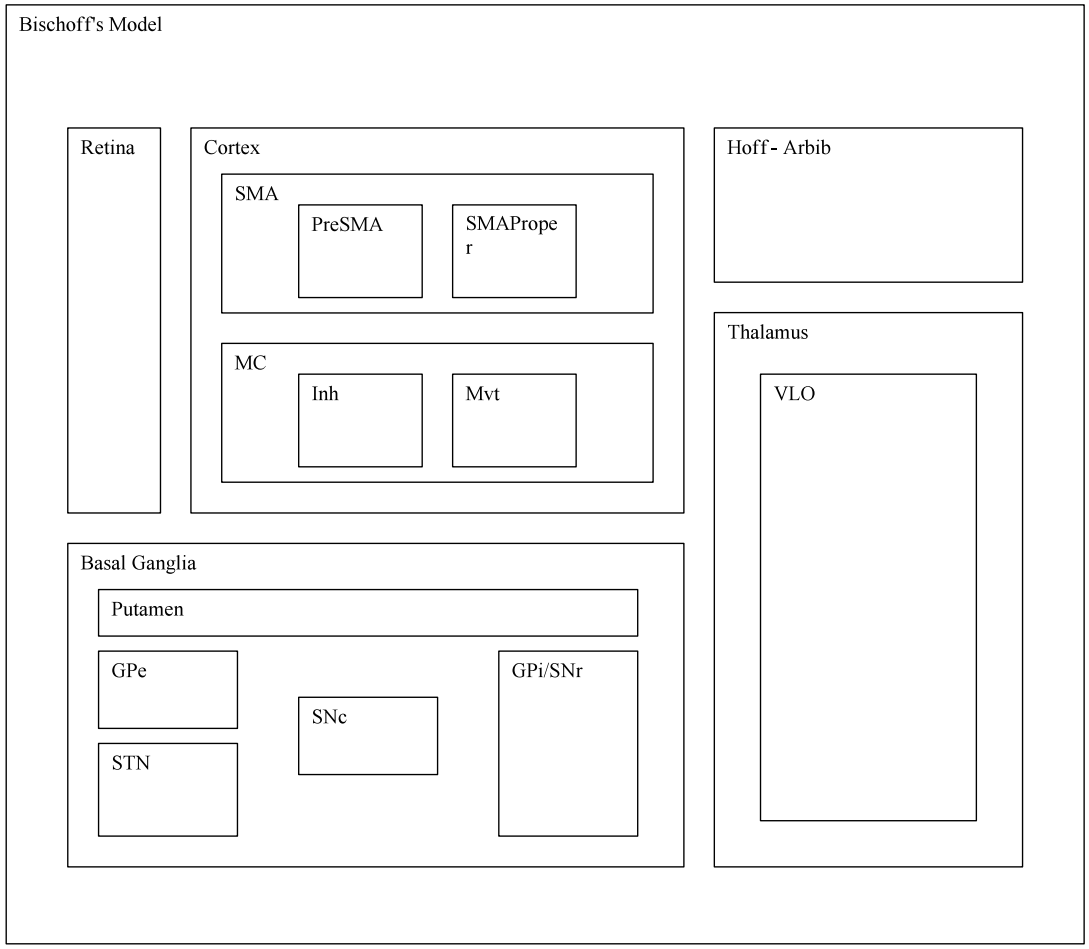
ITAM

Niveles de abstracción

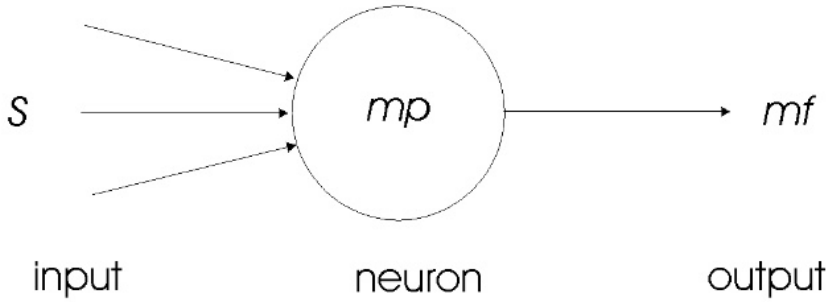
- Un modelo completo en NSL requiere los siguientes componentes:
  - un conjunto de módulos que definen el modelo completo
  - neuronas comprendidas en cada módulo neuronal
  - interconexiones neuronales
  - dinámica neuronal
  - métodos numéricos para resolver las ecuaciones diferenciales.

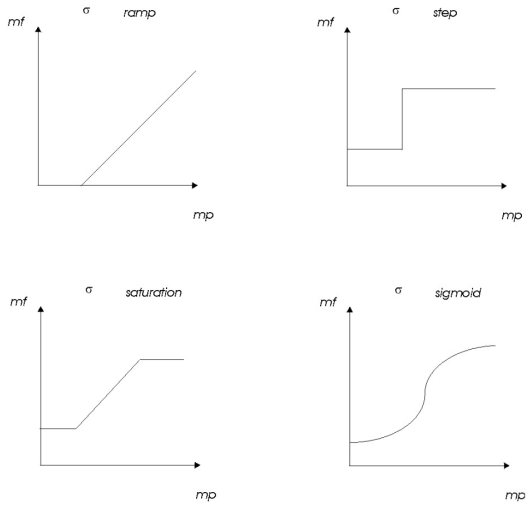


Ejemplos de módulos



Integrador con fugas





$$\tau \frac{dmp(t)}{dt} = f(S, mp, t)$$

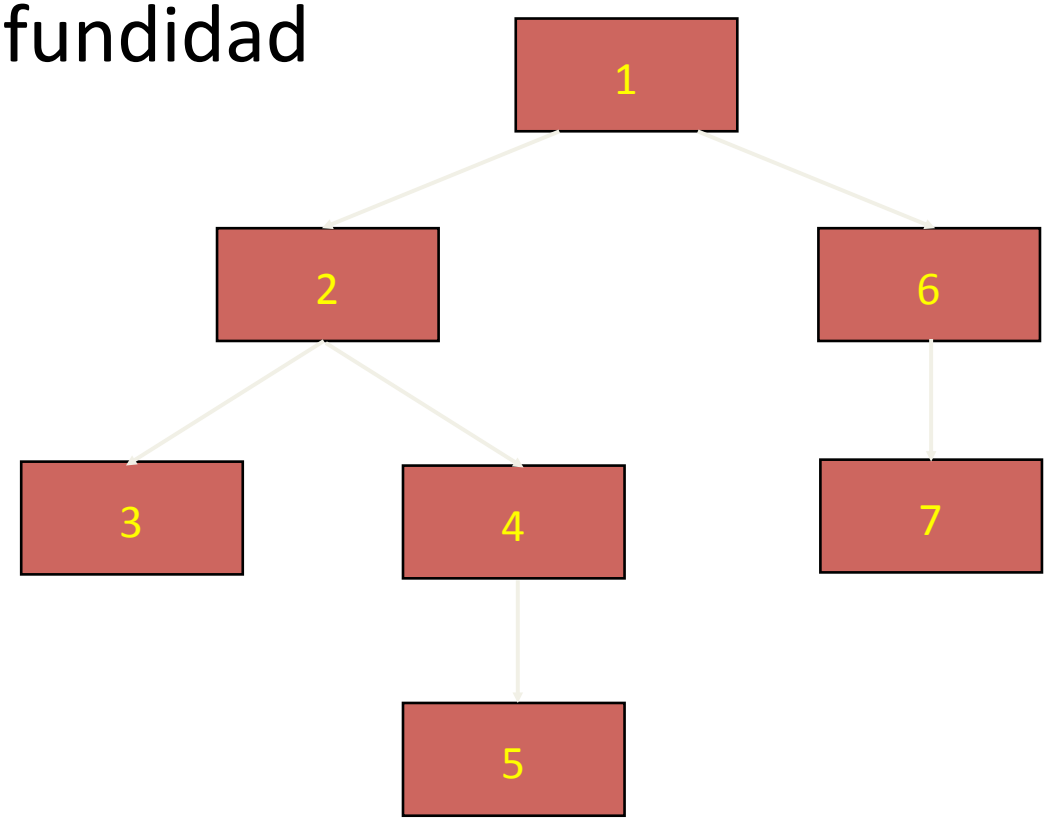
$$f(s, mp, t) = -mp(t) + s(t)$$

$$mf(t) = \sigma(mp(t))$$

$$sv_j = \sum_{i=0}^{n-1} w_i uf_i(t)$$

Simulación

- Ejecución de módulos por búsqueda en profundidad



Métodos de simulación

initSys

initModule

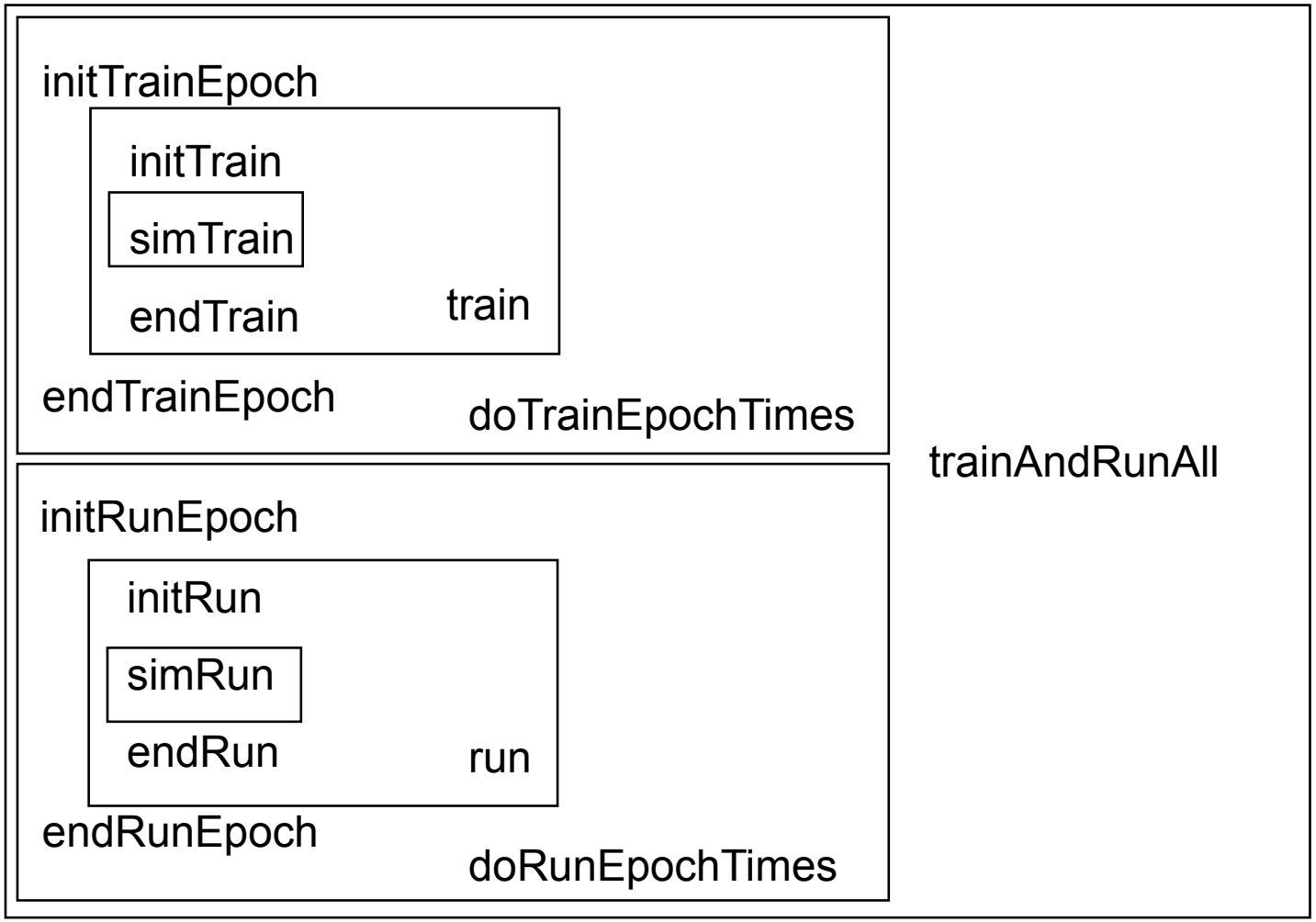
makeConn

… (pasos de simulación)

endModule

endSys

Pasos de simulación



Estructuras de modelado

- Model
  - Nivel más alto
- Modules
  - Redes Neuronales
- InModules
- OutModules
  - Interfaces Gráficas
- MotorModules
  - Robótica

- NslClass
  - Bibliotecas
  - Nuevos liensos
  - Nuevos comandos NSLS

Tipos NSLM

- Tipos primitivos
  - int
  - float
  - double
  - boolean
  - char

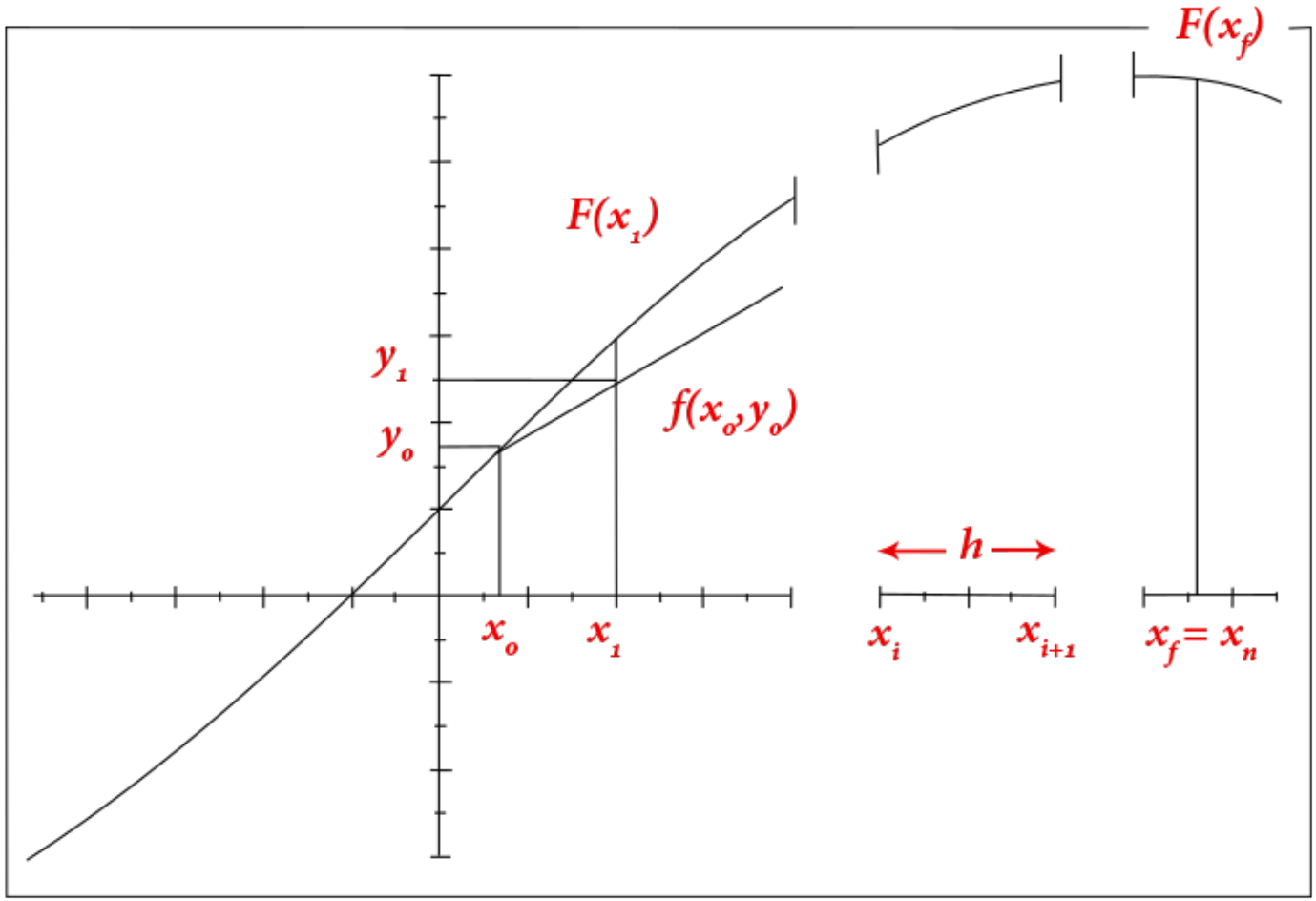
- Tipos NslData (0, 1, 2, 3, 4)
  - NslInt
  - NslFloat
  - NslDouble
  - NslBoolean
  - NslString (0)

- Pueden ser públicos, privados y protegidos

- Tipos NslPort (0, 1, 2, 3, 4)
  - NslDinInt
  - NslDinFloat
  - NslDinDouble
  - NslDinBoolean
  - NslDinString (0)
  - NslDoutInt
  - NslDoutFloat
  - NslDoutDouble
  - NslDoutBoolean
  - NslDoutString (0)

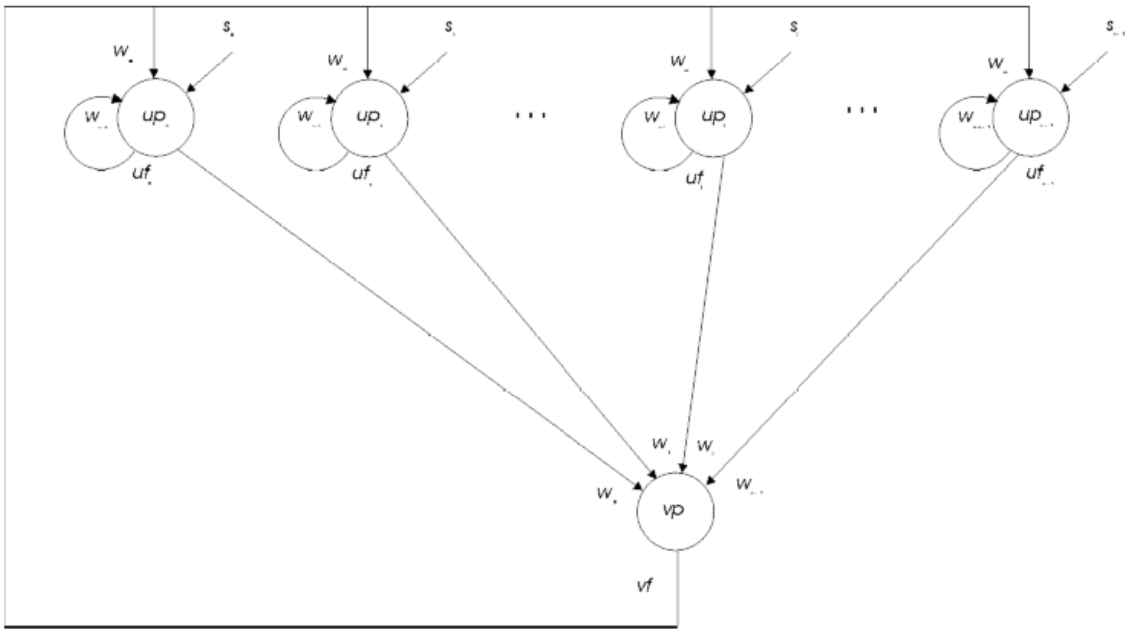
- Los puertos deben ser públicos

Método de integración numérica de Euler (EDO)

- 

Modelo Selector Máximo

- Los detalles pueden ser encontrado en la sección 4.4 de TMB2



Modelo Selector Máximo

- El modelo utiliza mecanismos de competencia para obtener, en muchos casos, un único ganador en la red donde la señal de entrada con mayor resistencia se propaga a lo largo de la salida de la red.

Modelo Selector Máximo

$$\tau_u \frac{du_i(t)}{dt} = -u_i + w_u f(u_i) - w_m g(v) - h_1 + s_i$$

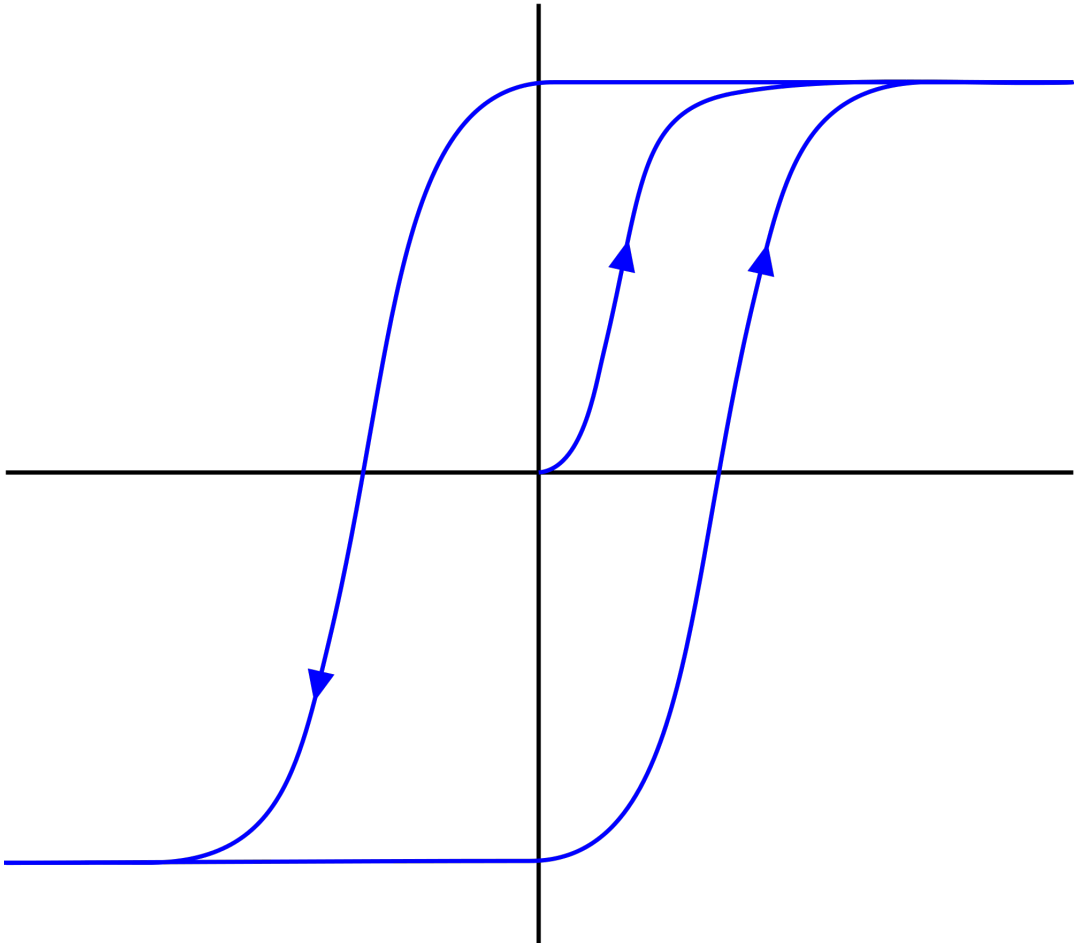
$$\tau_v \frac{dv}{dt} = -v + w_n \sum_{i=1}^{n} f(u_i) - h_2$$

$$f(u_i) = \begin{cases} 1 & u_i > 0 \\ 0 & u_i \leq 0 \end{cases}$$

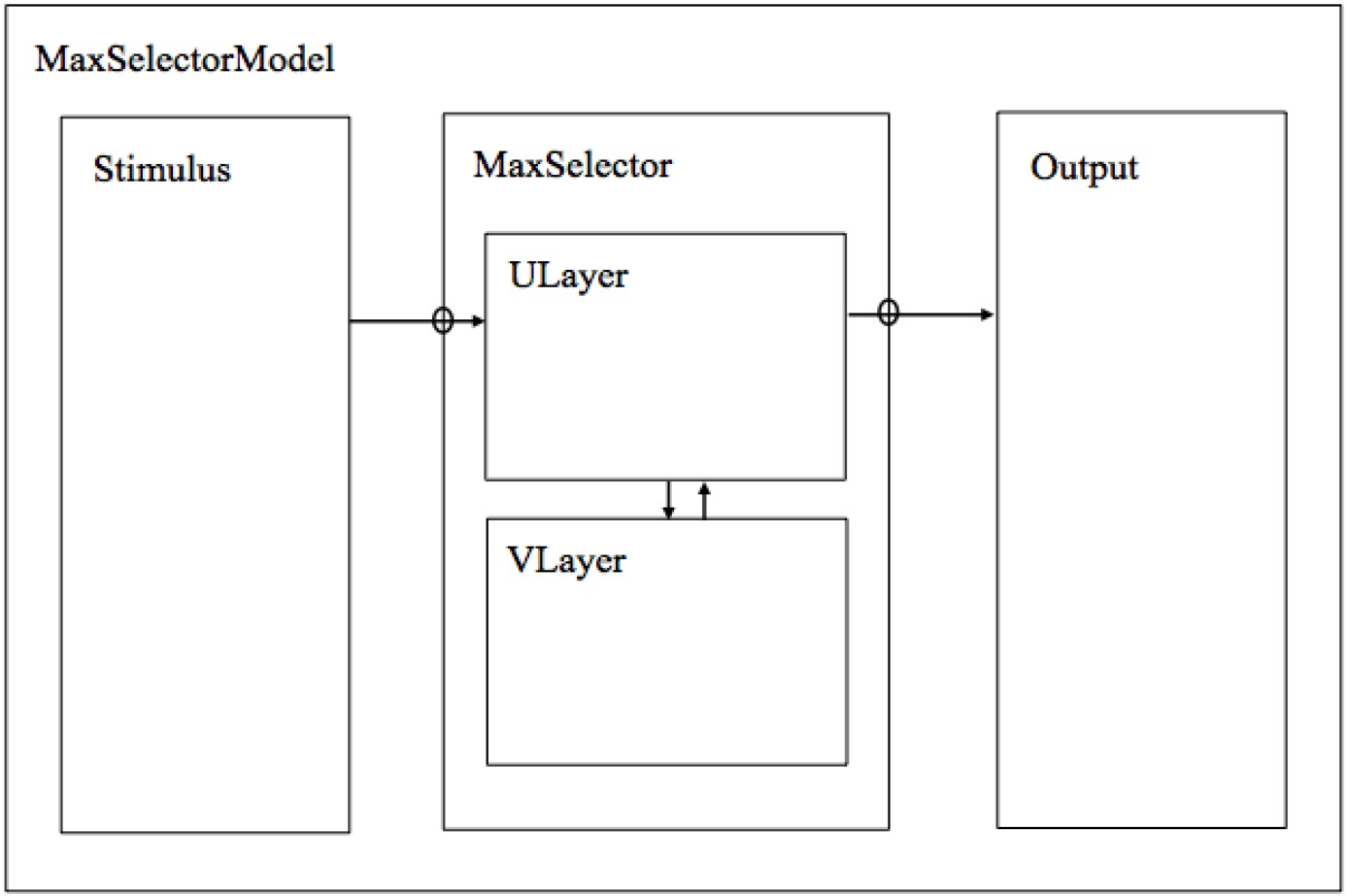
$$g(v) = \begin{cases} v & v > 0 \\ 0 & v \leq 0 \end{cases}$$

ITAM

Histéresis



Modelo Selector Máximo



MaxSelectorModel

```
nslImport nslAllImports;

nslImport MaxSelectorStimulus;
nslImport MaxSelector;
nslImport MaxSelectorOutput;

nslModel MaxSelectorModel() {

    nslConst int size = 10;

    private MaxSelectorStimulus stimulus(size);
    private MaxSelector maxselector(size);
    private MaxSelectorOutput output(size);

    public void initSys() {
         system.setRunEndTime(10.0);
         system.nslSetRunDelta(0.1);
    }

    public void makeConn() {
         nslConnect(stimulus.s_out, maxselector.in);
         nslConnect(stimulus.s_out, output.s_out);
         nslConnect(maxselector.out, output.uf);
    }
}
```

MaxSelectorStimulus

```
nslImport nslAllImports;

nslModule MaxSelectorStimulus(int size) {

    public NslDoutDouble1 s_out(size);

    public void initRun() {
        s_out=0;
        s_out[1]=0.5;
        s_out[3]=1.0;
    }
}
```

MaxSelectorOutput

```
nslImport nslAllImports;

nslOutModule MaxSelectorOutput(int size) {
    public NslDinDouble1 s_out(size);
    public NslDinDouble1 uf(size);
    private NslDouble1 up(size);
    private boolean worked= false;

    public void initModule() {
        up.nslSetAccess('W');
        nslAddAreaCanvas(s_out,0,1);
        nslAddTemporalCanvas(up,-2.5,2.5);
        nslAddAreaCanvas(uf,0,1);
    }

    public void simRun() {
        worked=nslSetValue(up,"maxSelectorModel.maxselector.u1.up");
    }
}
```

MaxSelector

```
nslImport nslAllImports;

nslImport Ulayer;
nslImport Vlayer;

nslModule MaxSelector(int size) {

    public NslDinDouble1 in(size);
    public NslDoutDouble1 out(size);

    private Ulayer u1(size);
    private Vlayer v1();

    public void makeConn() {
        nslRelabel(this.in, u1.s_in);
        nslConnect(u1.uf, v1.u_in);
        nslConnect(v1.vf, u1.v_in);
        nslRelabel(u1.uf, this.out);
    }
}
```

ULayer

```
nslImport nslAllImports;

nslModule Ulayer(int size) {
    //inports
    public NslDinDouble1 s_in();
    public NslDinDouble0 v_in();
    //outports
    public NslDoutDouble1 uf(size);
    //variables
    private NslDouble1 up(size);
    private NslDouble0 w1();
    private NslDouble0 w2();
    private NslDouble0 h1();
    private NslDouble0 k();
    private double tau;

      …
```

ULayer

```
    public void initRun(){
        uf = 0;
        up = 0;
        tau = 1.0;
        w1= 1.0;
        w2= 1.0;
        h1= 0.1;
        k= 0.1;
    }

    public void simRun(){
        //compute : up=up+((timestep/tu)*du/dt)
        up = nslDiff(up, tau, -up + w1*uf-w2*v_in - h1 + s_in);
        uf = nslStep(up,k.get(),0,1.0);
    }
}
```

VLayer

```
nslImport nslAllImports;

nslModule Vlayer() {

    // ports
    public NslDinDouble1 u_in();

    // output port
    public NslDoutDouble0 vf();

    // variables
    private NslDouble0 vp();  // neuron potential
    private NslDouble0 h2();
    private double tau;       // time constant

    …
```

VLayer

```
    public void initRun() {
        vf=0;
        vp=0;
        tau=1.0;
        h2 = 0.5;
    }

    public void simRun() {
        // vp=vp+((timestep/tv)*dv/dt)
        vp = nslDiff(vp, tau, -vp + nslSum(u_in) -h2);
        vf = nslRamp(vp);
    }
}
```

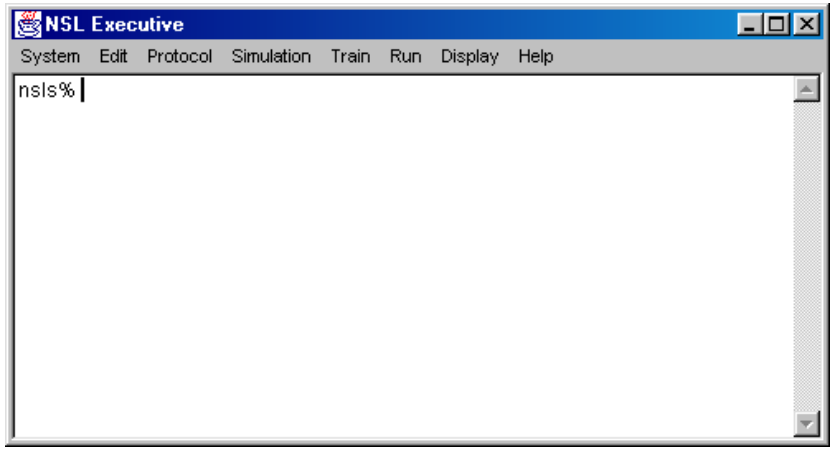
Compilación

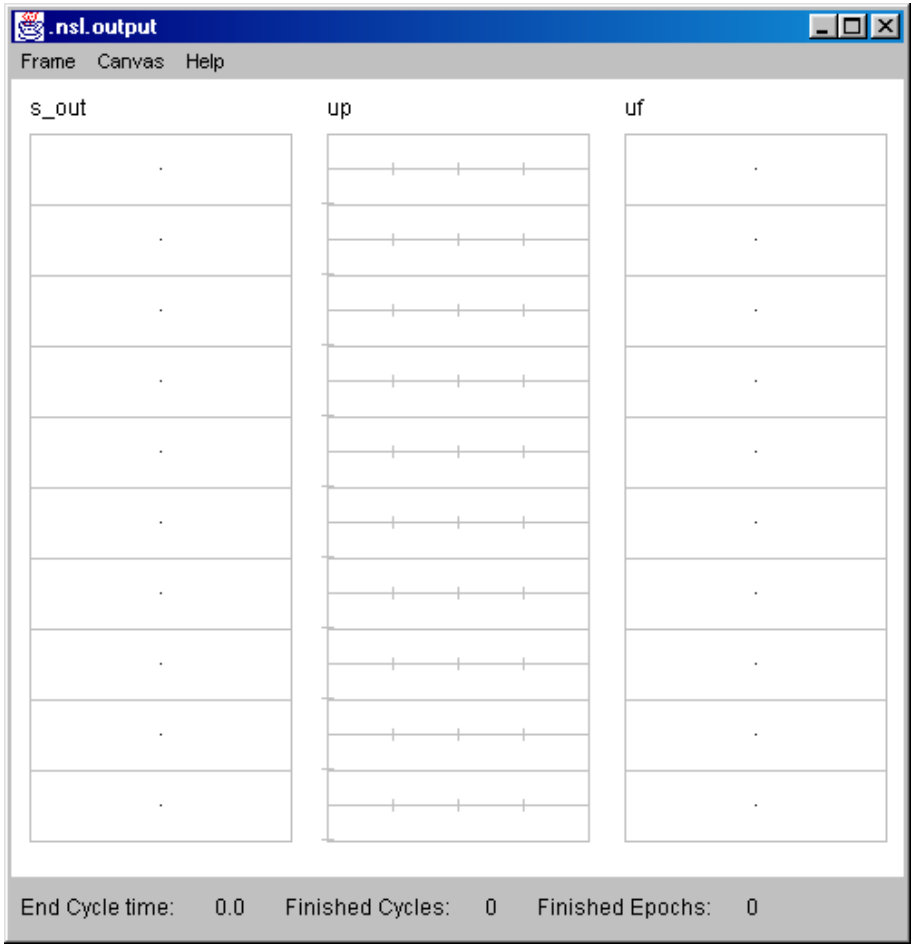
- Un modelo / módulo por archivo
  - La extensión de archivo debe ser .mod
- Recomendamos limpiar el directorio del modelo antes de compilar con el comando: *nslc -clean*
- Para compilar el modelo sólo tiene que ejecutar el siguiente comando: *nslc modelName*
  - Donde nombreModelo es el Nombre del archivo que contiene la estructura del modelo.
- Para este ejemplo debemos escribir: *nslc MaxSelectorModel*
- Tener en cuenta que no escribimos la extensión de archivo al final del nombre.

Ejecución

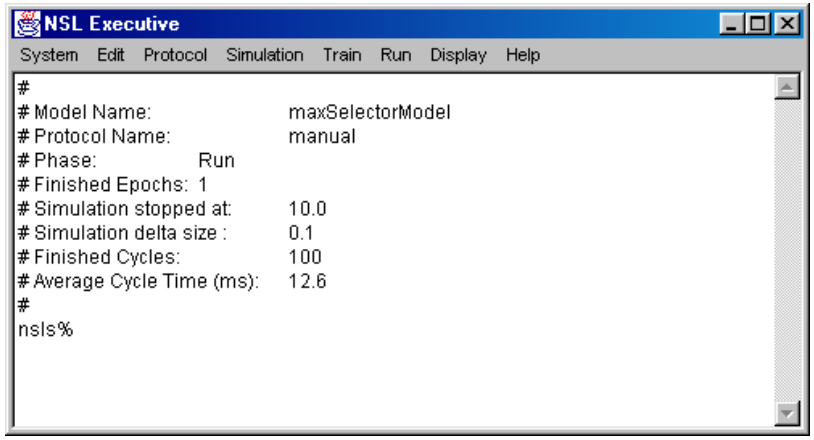
- Para simular tu modelo debes usar el comando nsl.
- Para este ejemplo debe escribir: nsl MaxSelectorModel
- Dos modos de funcionamiento
  - Texto (-noDisplay)
  - Interfaz gráfica (predeterminada)
- Para redirigir la salida estándar (-stdout console)
- Para redirigir el error estándar (-stderr console)

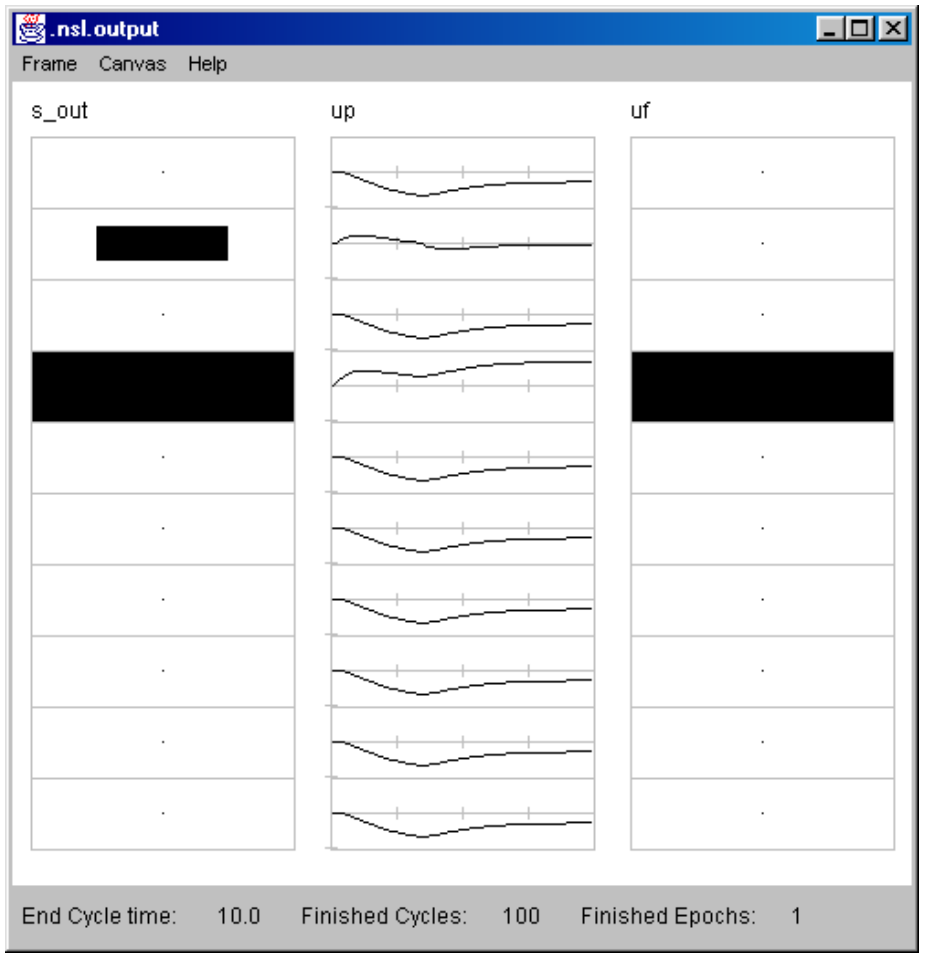
Interfaz


NSL Executive
System Edit Protocol Simulation Train Run Display Help
nsls%


.nsl.output
Frame Canvas Help
s_out
up
uf
End Cycle time: 0.0 Finished Cycles: 0 Finished Epochs: 0

Interfaz


NSL Executive
System Edit Protocol Simulation Train Run Display Help
#
Model Name: maxSelectorModel
Protocol Name: manual
Phase: Run
Finished Epochs: 1
Simulation stopped at: 10.0
Simulation delta size : 0.1
Finished Cycles: 100
Average Cycle Time (ms): 12.6
#
nsls%


.nsl.output
Frame Canvas Help
s_out
up
uf
End Cycle time: 10.0 Finished Cycles: 100 Finished Epochs: 1

NSLS

- Para evitar volver a compilar cada vez que modifiquen los parámetros del modelo, proporcionamos el lenguaje NSL script conocido como NSLS que también proporciona un entorno de control de usuario dinámico.

- NSLS proporciona la siguiente funcionalidad:
  - Asignación de parámetros del modelo
  - Especificación de entrada
  - Control de simulación
  - Control de archivos
  - Control de gráficos

- NSLS es una extensión del lenguaje de guiones TCL, proporcionando así funcionalidad tanto de NSL como de TCL.

NSLS

- Sintaxis del comando NSL: nsl subcomando [opciones]

- Comandos NSL importantes:
  - nsl source fileName
    - nsl source hopfield.nsls
  - nsl set variable value
    - nsl set maxSelectorModel.stimulus.s_out {1 0 0.5}
  - nsl get variable
    - nsl get maxSelectorModel.stimulus.s_out
  - nsl run
  - nsl train
  - …
  - nsl exit

Ejemplo de NSLS

```
#
# Hopfield Network
#

set A {}
set B {}
set C {}
set D {}

proc memorize { x } {

    puts "Memorizing $x"

    nsl set hopfield.inModule.input $x
    nsl train
}

proc test { x d } {

    nsl set hopfield.dis $d
    nsl set hopfield.inModule.input $x
    nsl run
}
```

Ejemplo de NSLS

```
proc initData {} {

    global A B C D

    set A {
      { -1 -1  1  1 -1 -1 }
      { -1  1 -1 -1  1 -1 }
      { -1  1  1  1  1 -1 }
      { -1  1  1  1  1 -1 }
      { -1  1 -1 -1  1 -1 }
      { -1  1 -1 -1  1 -1 }
    }

    set B {
      { 1  1 -1 -1 -1 -1 }
      { 1  1 -1 -1 -1 -1 }
      { 1  1  1  1 -1 -1 }
      { 1  1 -1 -1  1 -1 }
      { 1  1 -1 -1  1 -1 }
      { 1  1  1  1 -1 -1 }
    }
  …
}
```

Ejemplo de NSLS

```
proc trainNetwork {} {
    global A B C D

    memorize $A
    memorize $B
    memorize $C
    memorize $D
}

proc NslMain {} {
    global A B C D
    puts "Initializing"
    initData
    puts "Training"
    trainNetwork
    puts "Testing"
    for { set i 10 } { $i<20 } { incr i } {
     puts "Testing with distortion $i"
     test $A $i
    }
    nsl set hopfield.dis 0
}

NslMain
```

Instalación de NSL

- Instalar la versión 1.4.2_19 de Java development kit
- Descargar y descompactar la carpeta de NSL3.0
- Editar el archivo "NSL3_0_s\resume.bat" de forma que coincida con el entorno (se deberá especificar la ruta donde se instaló Java, donde se instaló NSL, etc.).
- Ejecutar el archivo de reanudación por lotes antes de iniciar una sesión NSL.

Referencias

- A Weitzenfeld, MA Arbib and A Alexander, 2002, NSL Neural Simulation Language, MIT Press