Custom Neurons Interface

by Quarta Jonny
Supervisor: Gewaltig Marc-Oliver

Content

- Introduction
- NEST Structure
- What is Cython ?
- Interface Structure
- Compiling the Neuron
- Performances and Optimizations
- Problems
- Conclusion

Introduction

- NEST (NEural Simulation Tool)
- Creating custom neurons not flexible

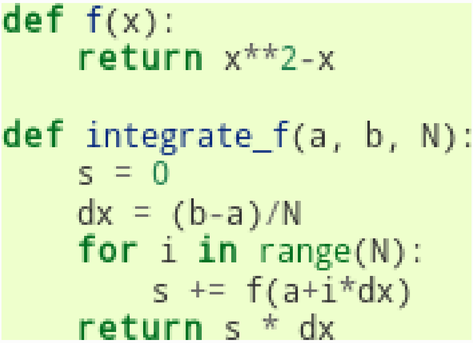
  => create more flexible system
- Neurons written in Python

NEST Structure

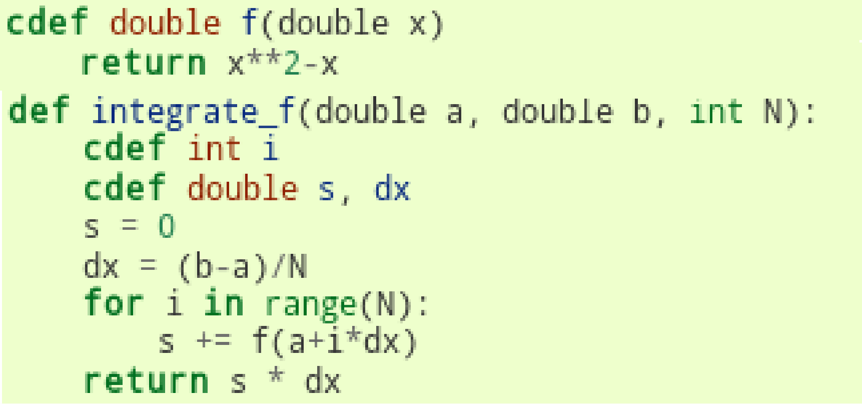
- Simulation Kernel
- SLI Interface
- CyNEST Interface

What is Cython ?

- Compiling Python code into C/C++
- Makes Python code faster



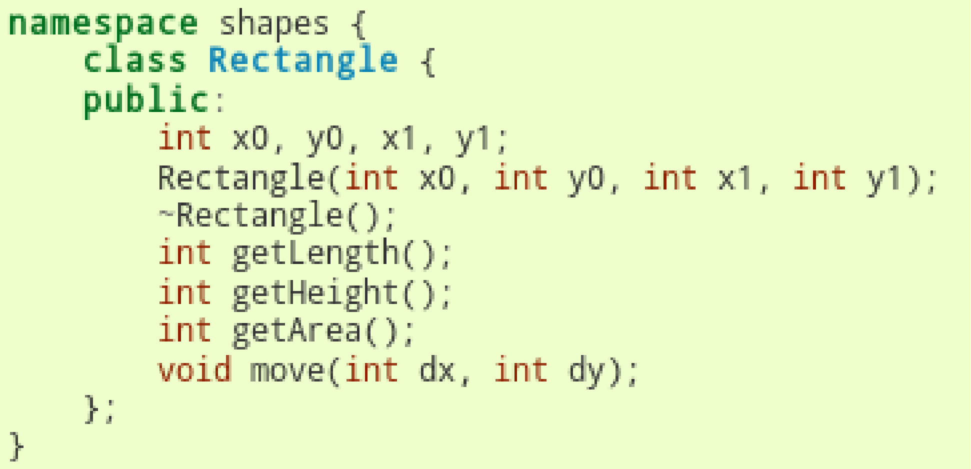
```
def f(x):
    return x**2-x

def integrate_f(a, b, N):
    s = 0
    dx = (b-a)/N
    for i in range(N):
        s += f(a+i*dx)
    return s * dx
```



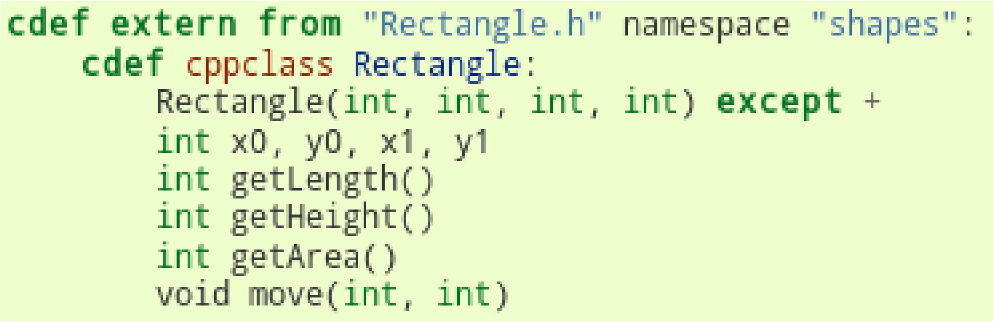
```
cdef double f(double x)
    return x**2-x
def integrate_f(double a, double b, int N):
    cdef int i
    cdef double s, dx
    s = 0
    dx = (b-a)/N
    for i in range(N):
        s += f(a+i*dx)
    return s * dx
```

What is Cython ? (2)

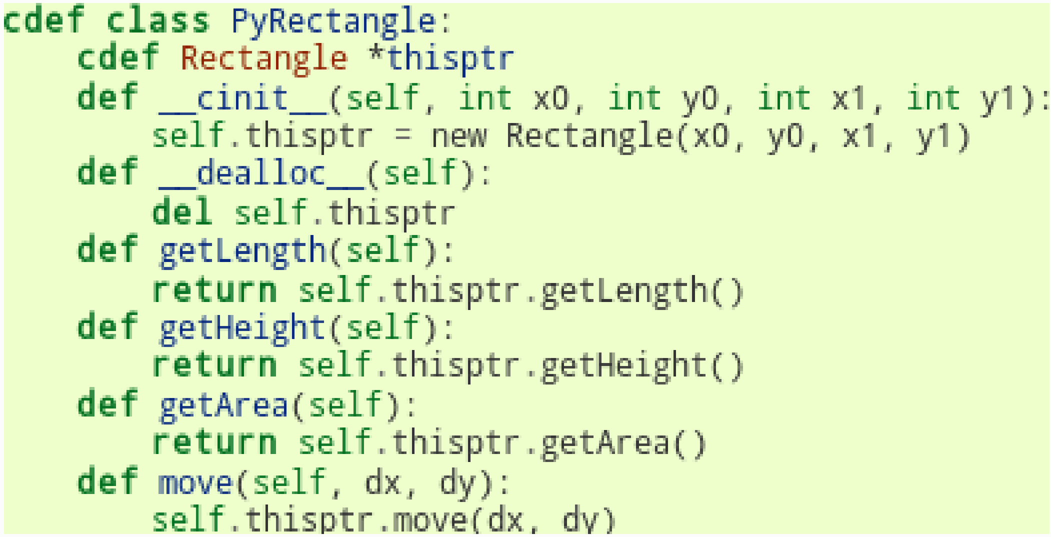
- Importing C/C++ libraries into Python
- Importing Python code into C/C++ !



```
namespace shapes {
    class Rectangle {
    public:
        int x0, y0, x1, y1;
        Rectangle(int x0, int y0, int x1, int y1);
        ~Rectangle();
        int getLength();
        int getHeight();
        int getArea();
        void move(int dx, int dy);
    };
}
```

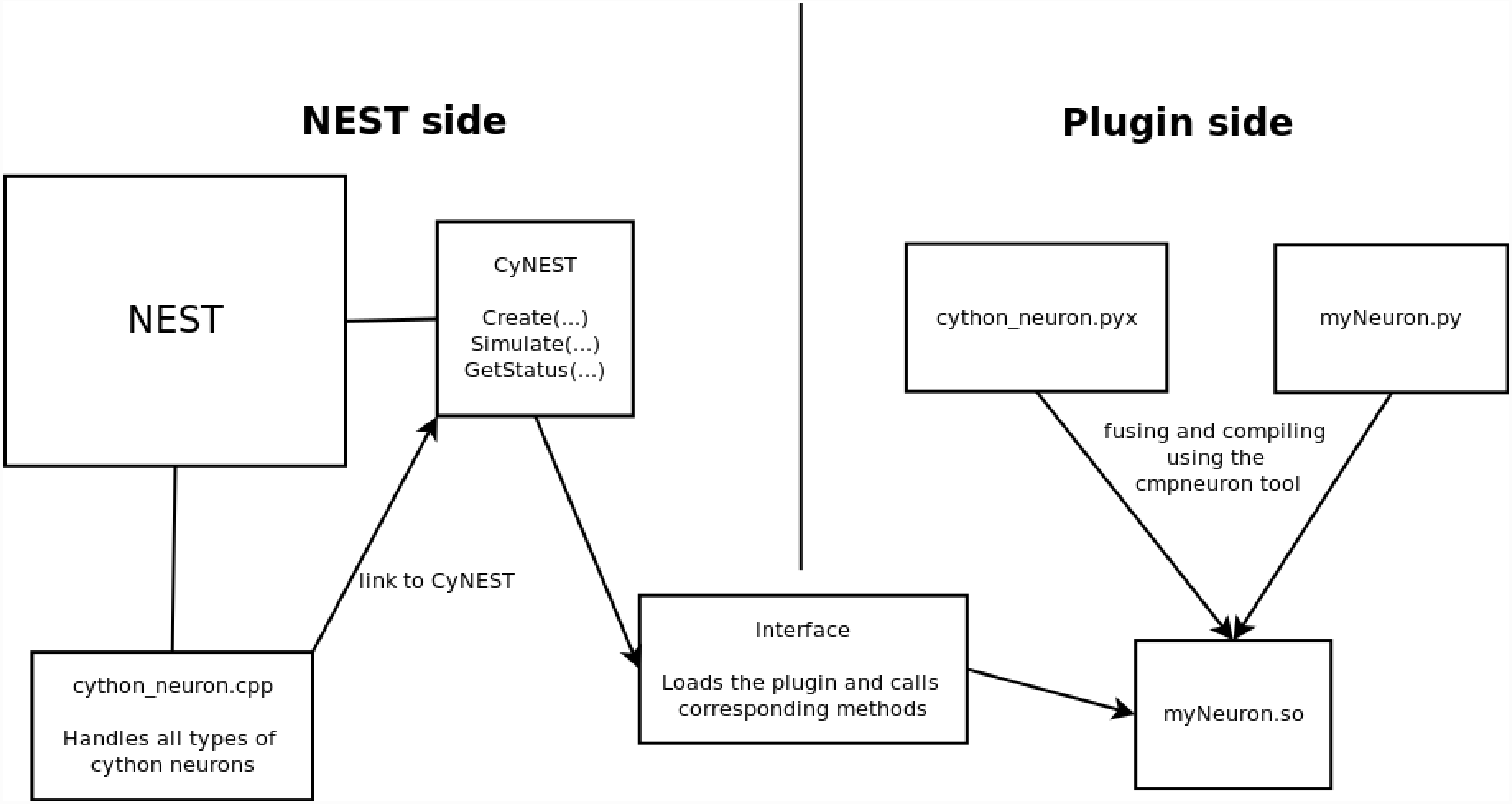


```
cdef extern from "Rectangle.h" namespace "shapes":
    cdef cppclass Rectangle:
        Rectangle(int, int, int, int) except +
        int x0, y0, x1, y1
        int getLength()
        int getHeight()
        int getArea()
        void move(int, int)
```



```
cdef class PyRectangle:
    cdef Rectangle *thisptr
    def __cinit__(self, int x0, int y0, int x1, int y1):
        self.thisptr = new Rectangle(x0, y0, x1, y1)
    def __dealloc__(self):
        del self.thisptr
    def getLength(self):
        return self.thisptr.getLength()
    def getHeight(self):
        return self.thisptr.getHeight()
    def getArea(self):
        return self.thisptr.getArea()
    def move(self, dx, dy):
        self.thisptr.move(dx, dy)
```

Interface Structure



NEST side

Plugin side

NEST

CyNEST

Create(...)
Simulate(...)
GetStatus(...)

cython_neuron.cpp

Handles all types of cython neurons

link to CyNEST

Interface

Loads the plugin and calls corresponding methods

cython_neuron.pyx

myNeuron.py

fusing and compiling using the cmpneuron tool

myNeuron.so

Basic Neuron Functions

- Calibrate()
- Update()
- GetStatus()
- SetStatus()

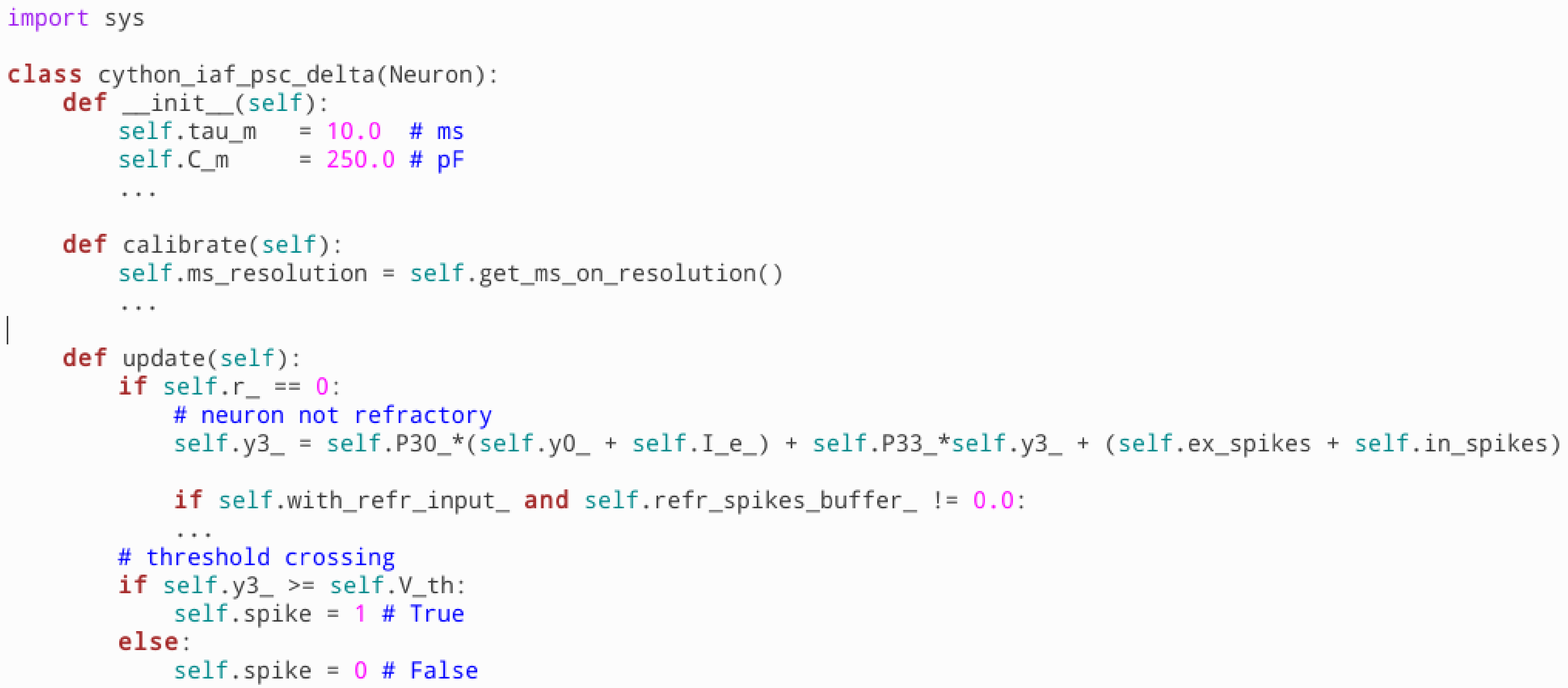
Interface Details

- Library loading for every neuron type (ctypes)
- Library contains a list of neurons

  => every C++ neuron has a python counterpart
- Status dict vs python fields

Compiling the Neuron

- Compiling with the *cmpneuron* Tool
- Put the result file in a special location



```
import sys

class cython_iaf_psc_delta(Neuron):
    def __init__(self):
        self.tau_m    = 10.0  # ms
        self.C_m      = 250.0 # pF
        ...

    def calibrate(self):
        self.ms_resolution = self.get_ms_on_resolution()
        ...

    def update(self):
        if self.r_ == 0:
            # neuron not refractory
            self.y3_ = self.P30_*(self.y0_ + self.I_e_) + self.P33_*self.y3_ + (self.ex_spikes + self.in_spikes)

            if self.with_refr_input_ and self.refr_spikes_buffer_ != 0.0:
            ...
        # threshold crossing
        if self.y3_ >= self.V_th:
            self.spike = 1 # True
        else:
            self.spike = 0 # False
```

Optimizations

- Update method critical
- Passing direct pointers
- Standard Parameters
  - currents
  - in_spikes
  - ex_spikes
  - t_lag
  - spike

Performances

Neuron Type	Duration	Nb neurons	Real-time factor
Native	40 ms	~1000	0.3254
SLI	40 ms	~1000	0.0046
Cython	40 ms	~1000	0.0049

=> Cython *neuron 66.38 slower than the native,*
but slightly faster than SLI

Problems

- Project not difficult, but little problems

- Cython syntax sometimes confusing
- Direct pointer passing not possible
- Global Interpreter Lock activated for persistent objects

Conclusion

- Useful feature for users
- Too slow
- Needs some improvements
  - Improve speed
  - Enable events handling
- Other approaches possible
  - Highly Configurable Neuron

Questions ?