Sequence learning in a model of the basal ganglia

Thesis submitted for MSc in computer science

Stian Soiland

NTNU, Trondheim 2006-09-08

http://soiland.no/master

This presentation

Theory
- Control theory & Actor-Critic
- Basal Ganglia and pathways
- CTRNN

Previous work by Berns & Sejnowski

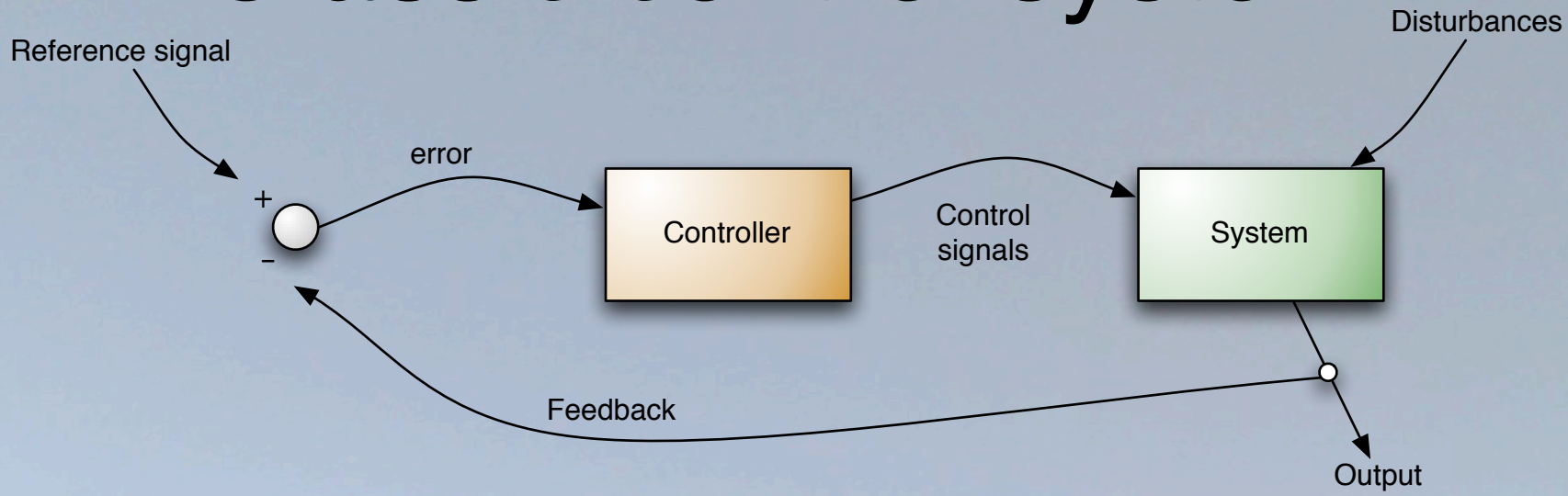
What was done?

Results
- Globus pallidus, Weight learning
- Error function

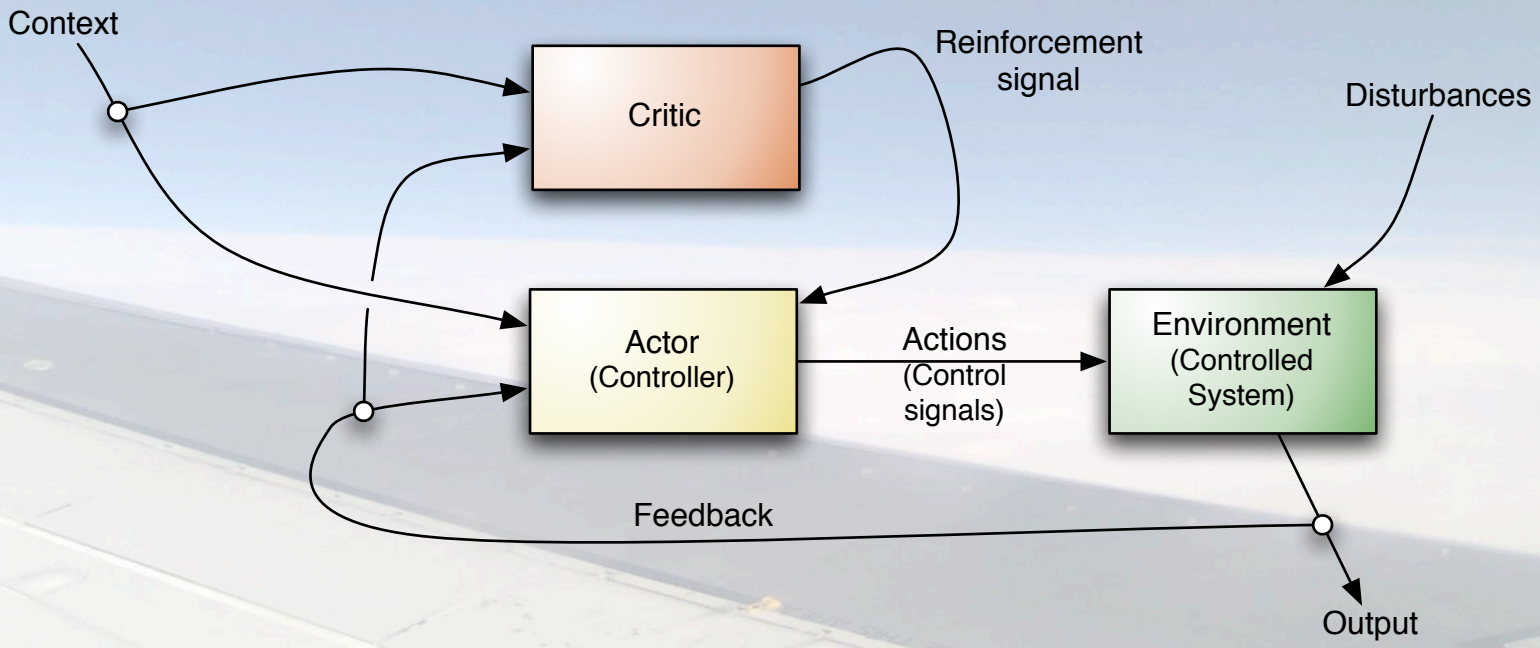
Experiments & Code
- Noise, Integrator update rule
- CTRNN library

Discussion
- Equation or code?

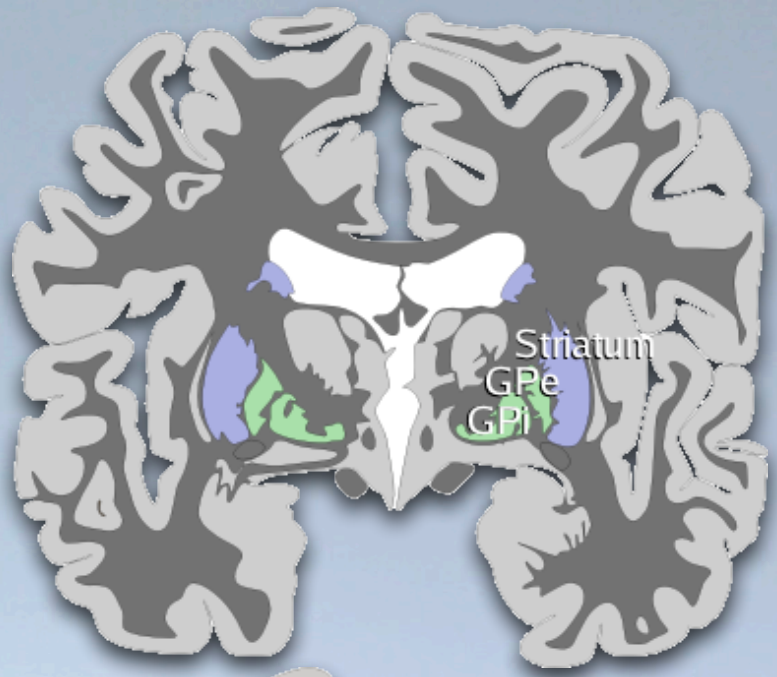
Classic control system



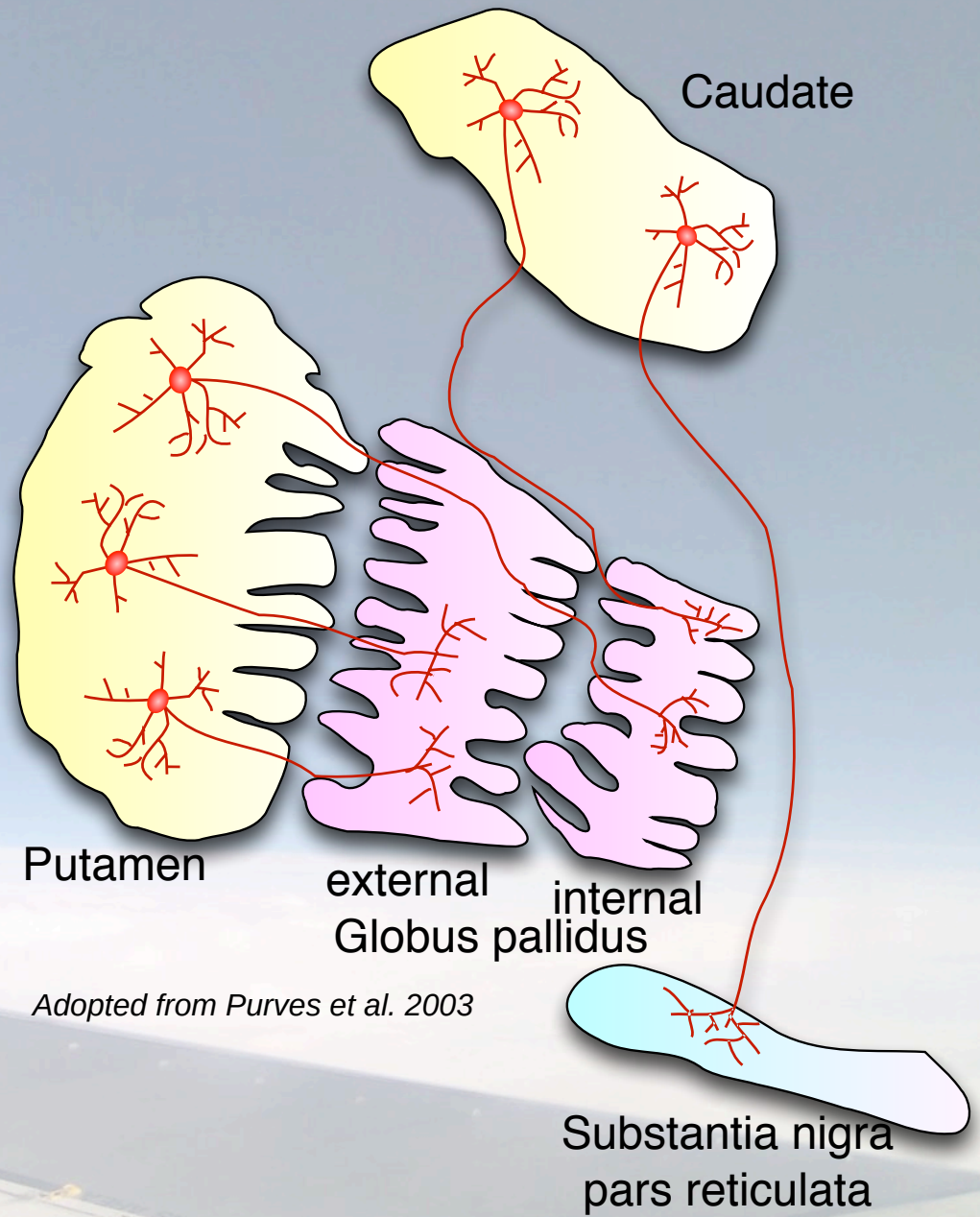
Actor-critic architecture



Basal ganglia

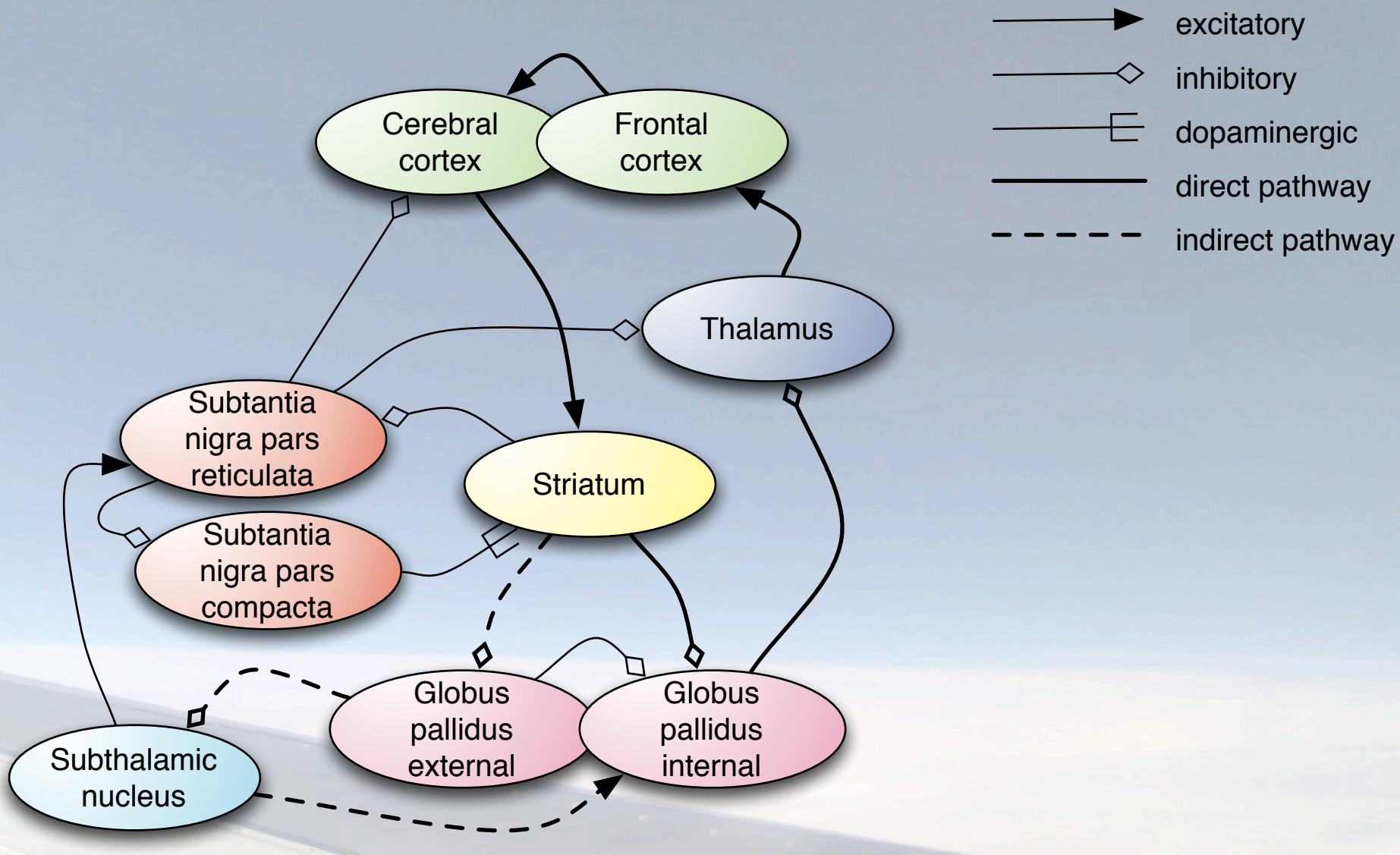


Wikipedia 2006

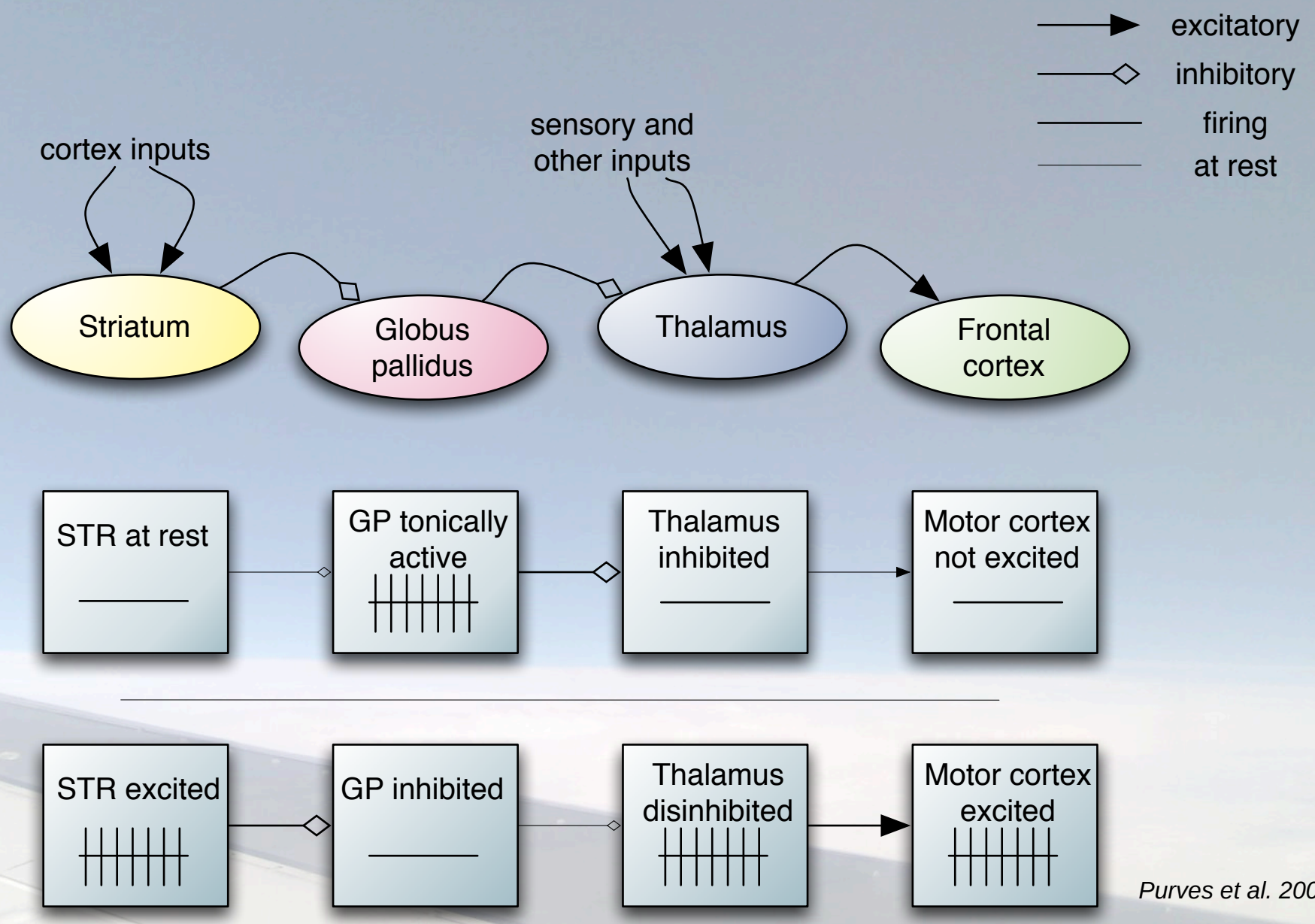


Adopted from Purves et al. 2003

Basal ganglia pathways

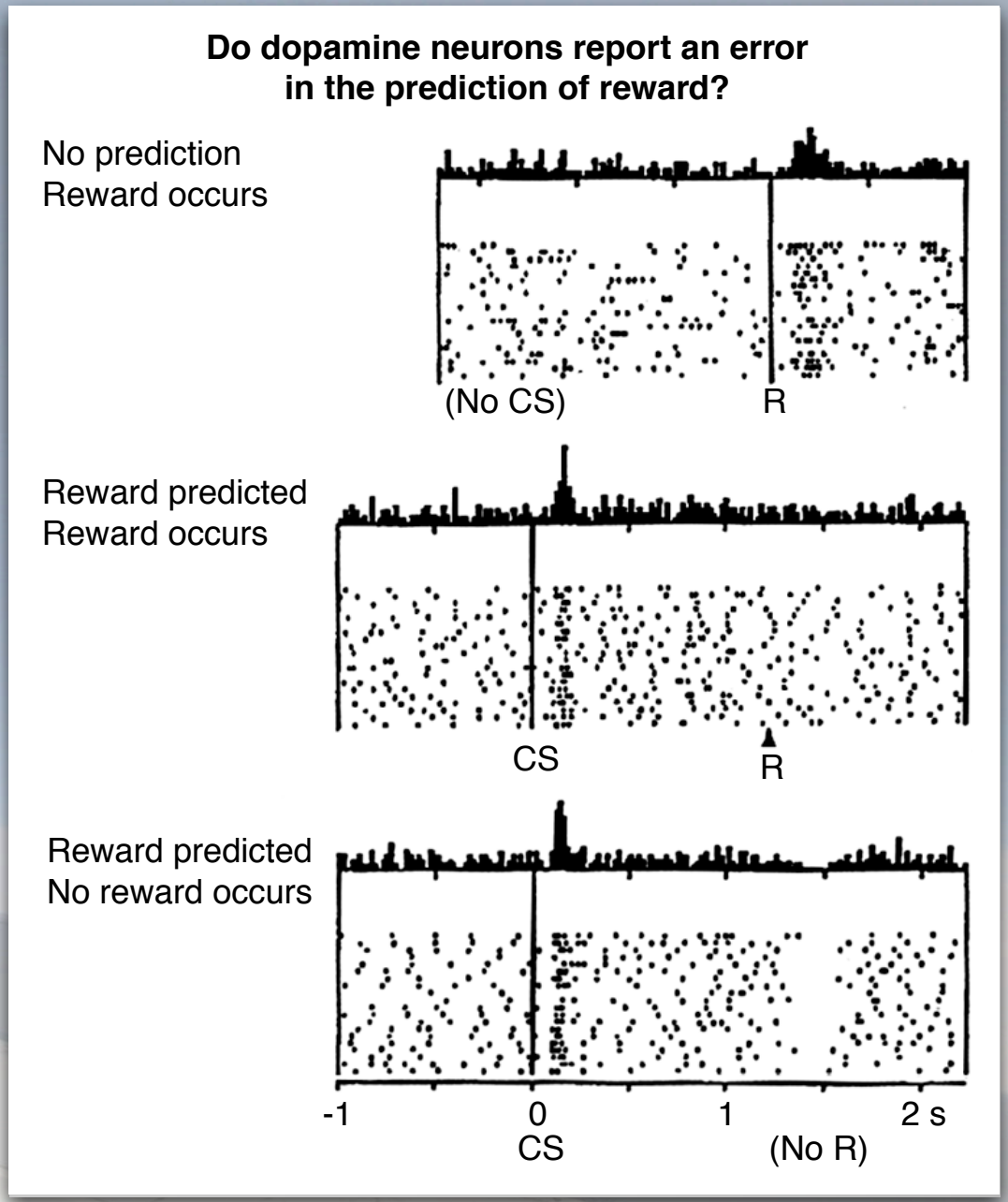


Inhibitory connections



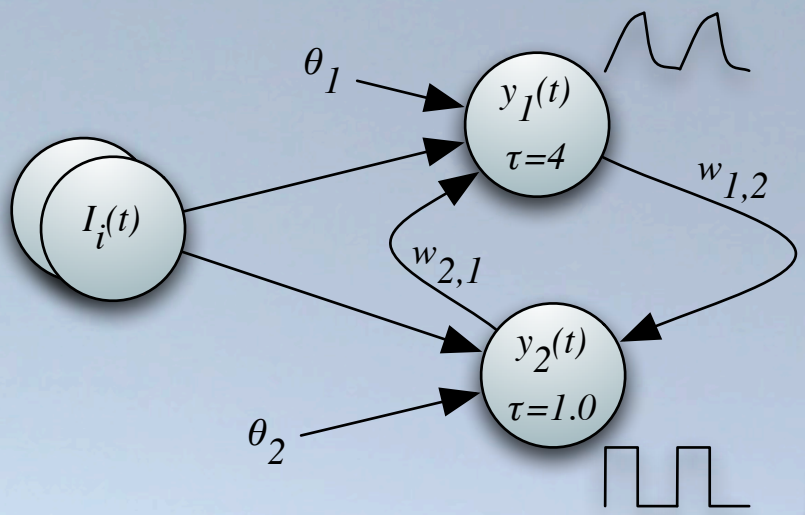
Purves et al. 2003

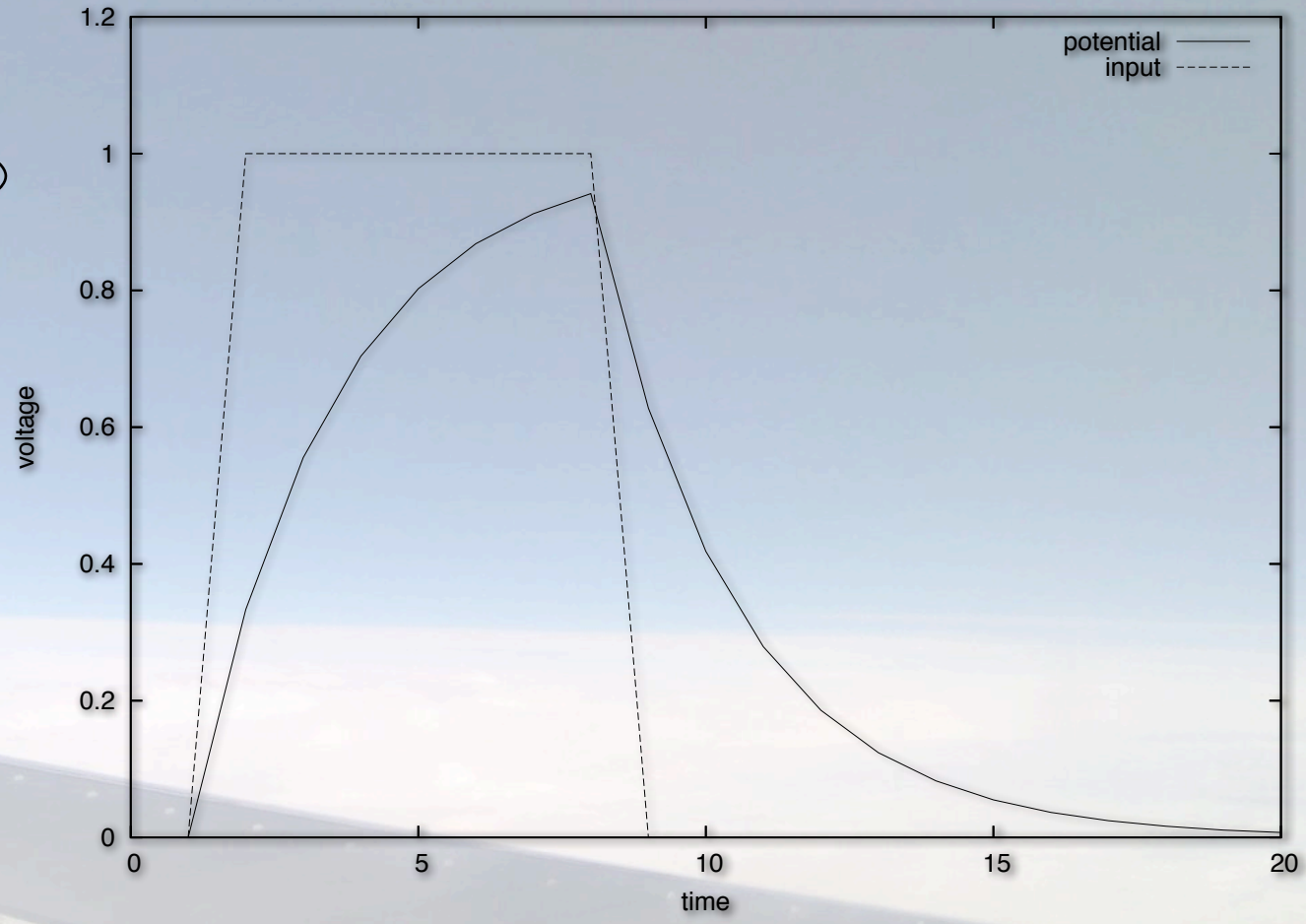
Basal ganglia in action



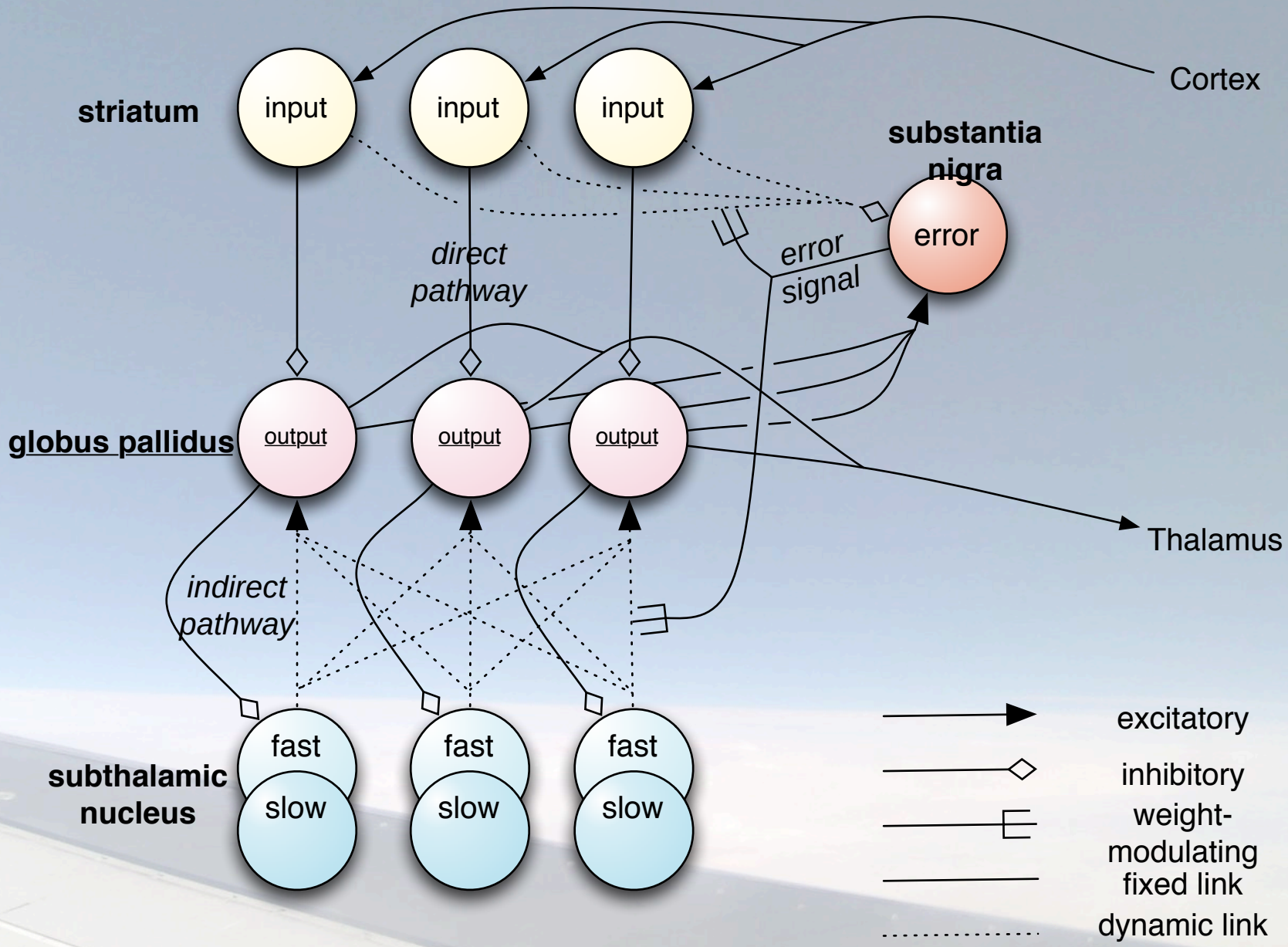
Scultz et al. 1997

Continious time recurrent neural network (CTRNN)





Berns & Sejnowski's model



What have I done?

Early attempts: C++ implementations of CTRNN & B&S

Direct reproduction of Berns & Sejnowski, equation to code

Experiments and tweaks on the direct reproduction

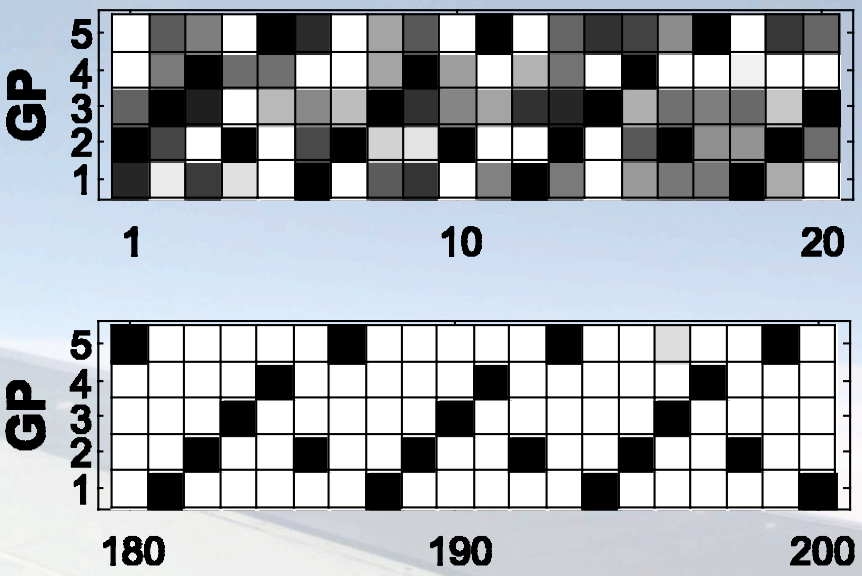
Implementation of CTRNN library

Reproducing B&S using CTRNN (failed)

Reproducing Prescott (2006) using CTRNN (worked)

Reproducing response of globus pallidus

Berns & Sejnowski

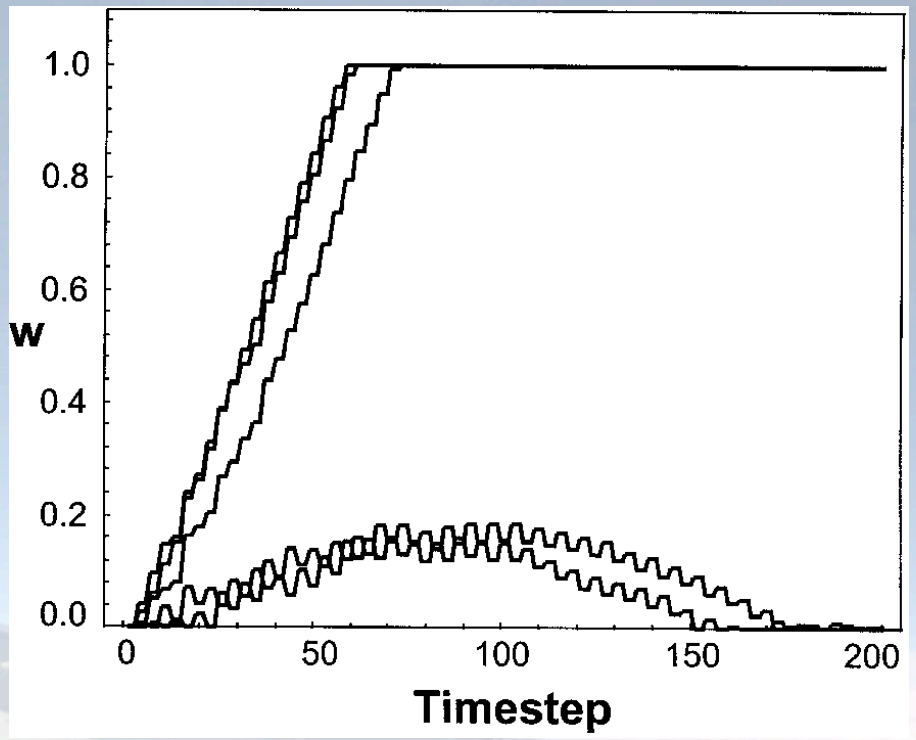


Berns & Sejnowski 1998

Soiland

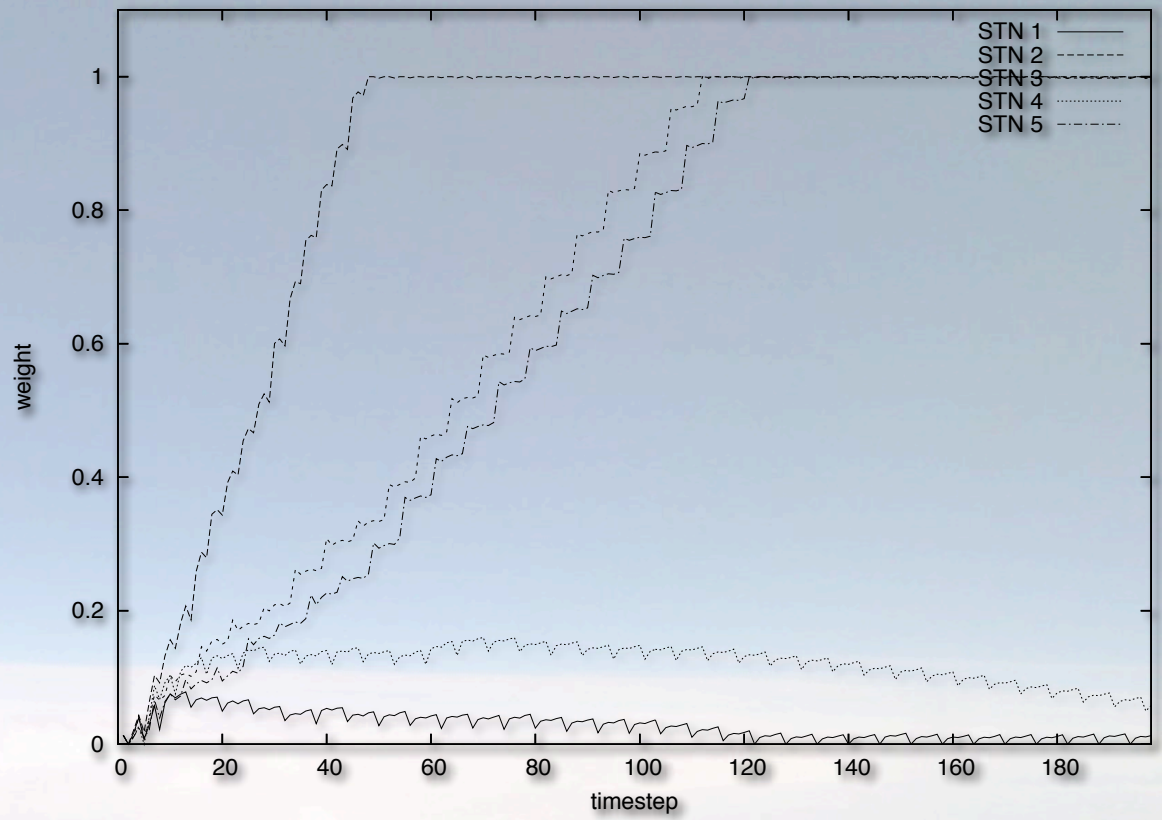
Weight learning

Berns & Sejnowski



Berns & Sejnowski 1998

Soiland

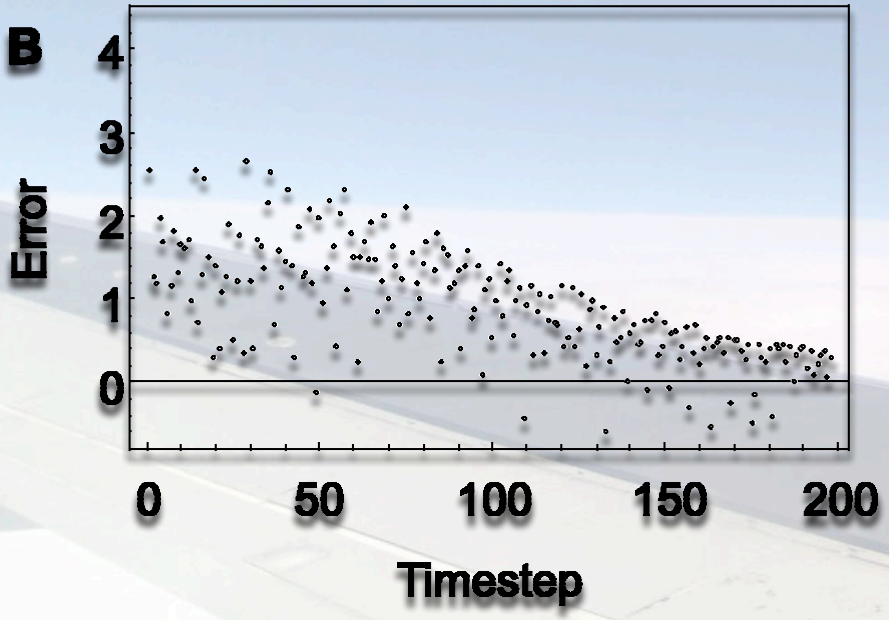


Error function

$$e(s) = \sum_i \left(G_i(s) - v_i(s) S_i(s) \right)$$

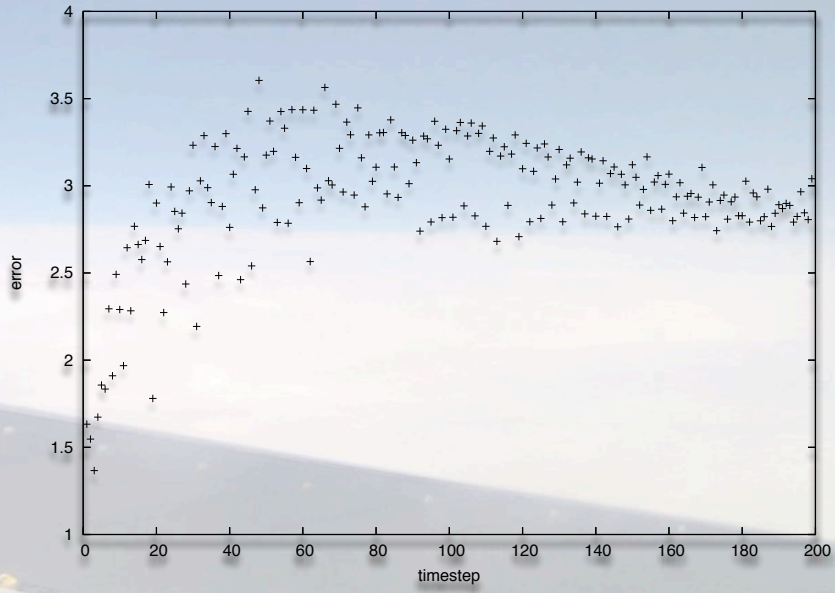
```
def calc_error(self):
    sum = 0.0
    for i in range(self.inputs):
        sum += self.GP[i]
        sum -= self.v[i]* self.STR[i]
    return sum
```

Berns & Sejnowski

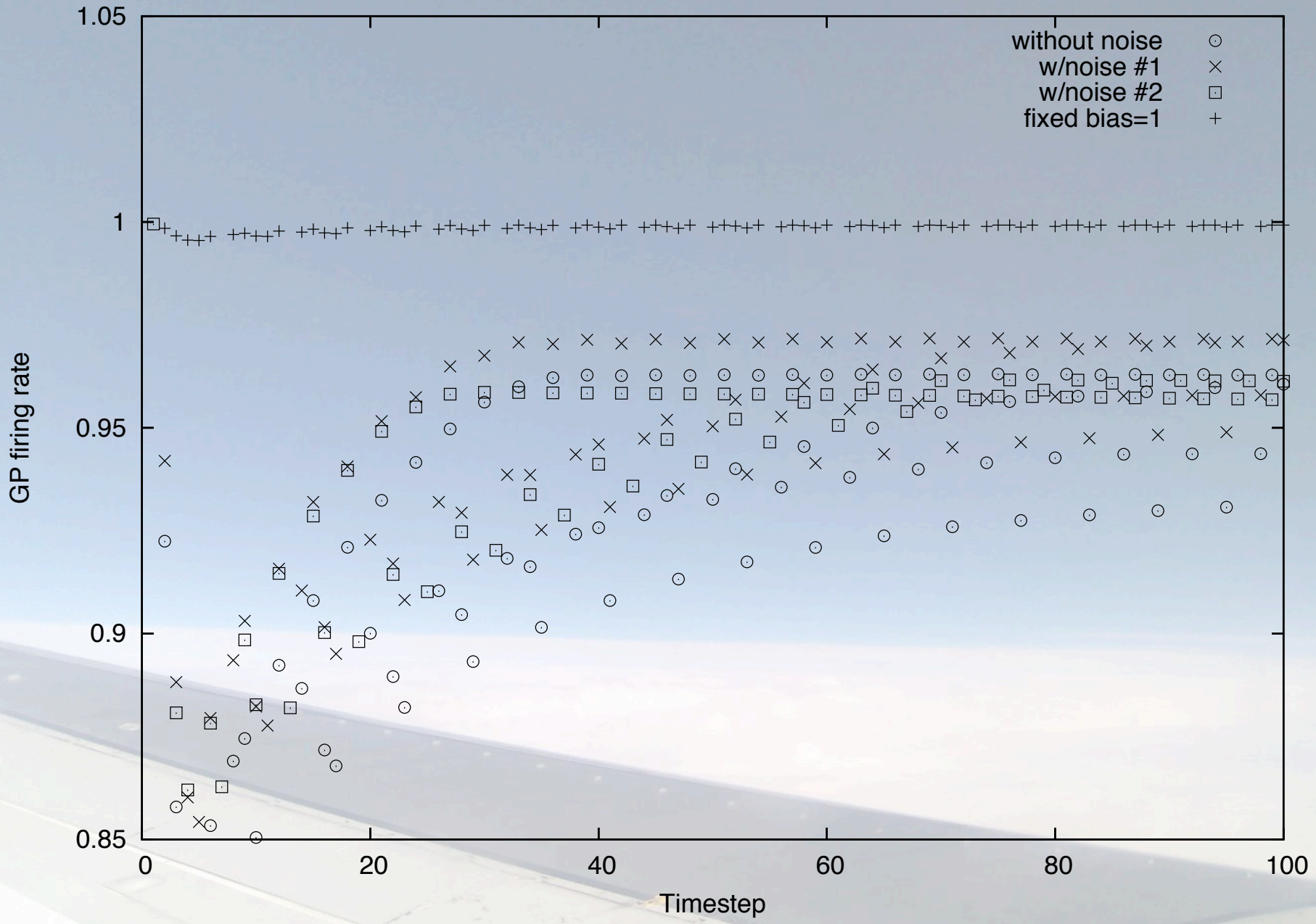


Berns & Sejnowski 1998

Soiland



Experiment: Noise



Experiment: Sigmoidal update rule

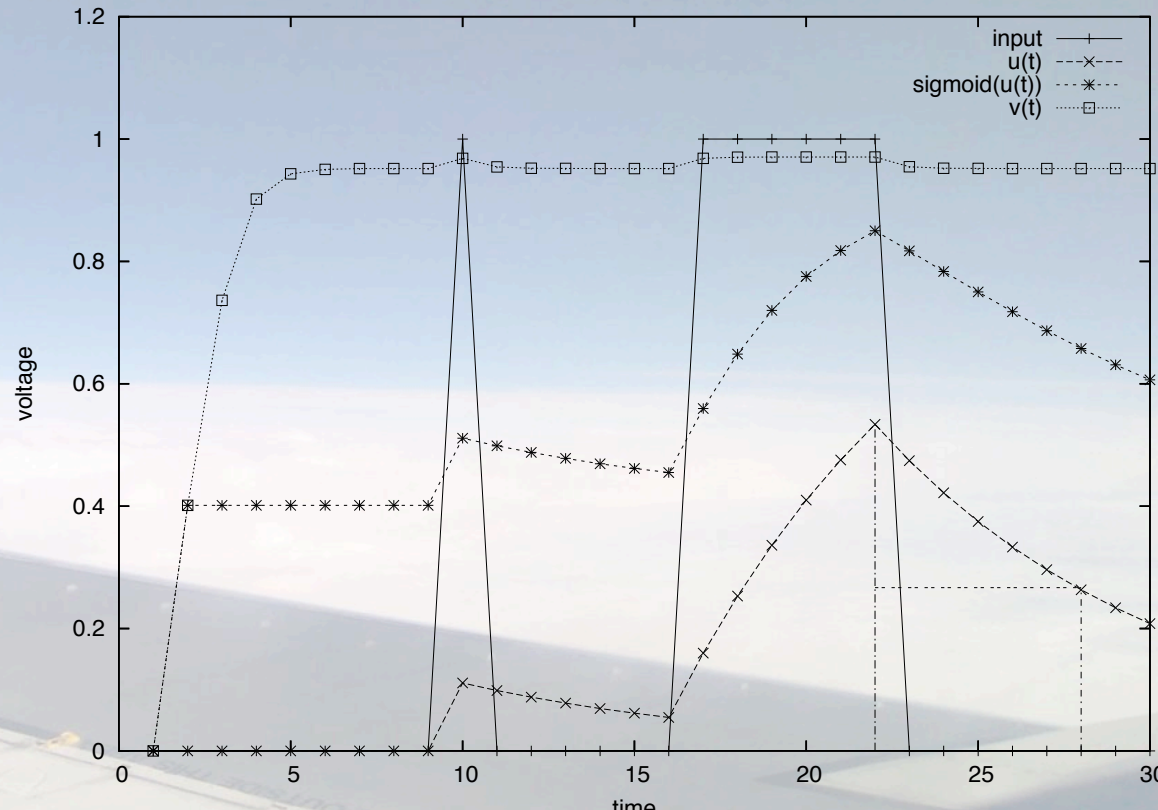
```
def calc_STN(self, i):
  return self.sigmoid(lambd * self.STN[i] -
         (1-lambd) * self.effect * self.GP[i/2])
(..)
STN[i] = calc_STN(i)

def sigmoid(self, x):
    return 1.0 / (1.0 +
      math.exp(-self.gain * (x-self.bias)))
```

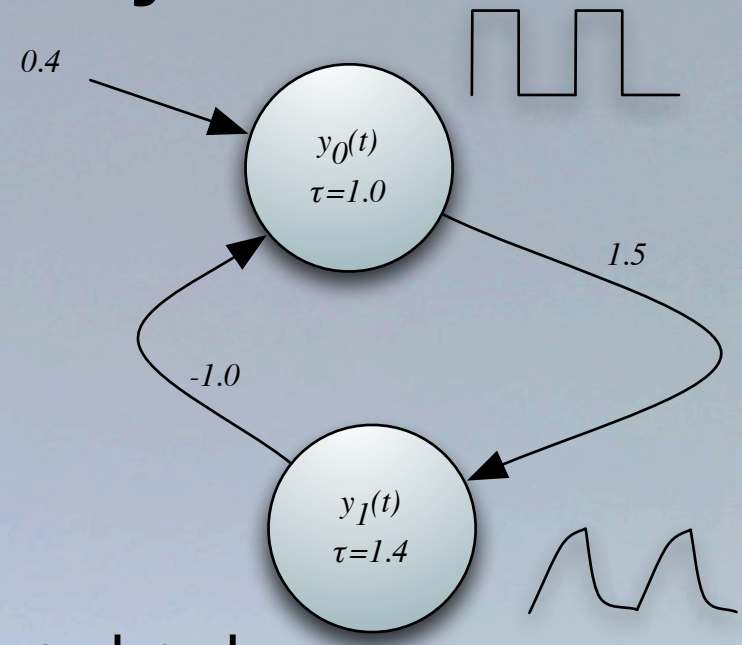
$$B(s) = \sigma\left(\alpha B(s-1) + \beta\right)$$

$$B(s+1) = \sigma\left(\alpha\sigma\left(\alpha B(s-1) + \beta\right) + \beta\right)$$

$$\sigma(x) = \frac{1}{1 + e^{-\gamma(x-\beta)}}$$



CTRNN library for Python




```
import ctrnn
net = ctrnn.CTRNN(2)
net.bias[0] = 0.4
net.timeconst[1] = 1.4
net.weight[0,1] = 1.5
net.weight[1,0] = -1.0
net.calc_timestep(); print net.output
net.calc_timestep(); print net.output

[ 0.59868766  0.5       ]
[ 0.47502081  0.6550814 ]
```

Equations or code?

$$\vec{y}(t+\Delta t) = \vec{y}(t) + \frac{\Delta t}{\vec{\tau}}\Big(-\vec{y}(t) + \vec{u}(t) \times W + \vec{\theta}\Big)$$

$$\vec{u}(t) = \sigma\Big(\vec{y}(t)\Big)$$

Concise, but can be difficult to understand

```
inputs = numpy.matrixmultiply(output, weight)
change = timestep/timeconst *
         (-potential + inputs + bias)
potential += change
output = map(transfer, potential)
```

More readable, but also more verbose

Mathematics can be general, but code is reproducible.
Maybe the best is a combination?

Questions?

All code, thesis and presentation at:

http://soiland.no/master