# Artificial Life lecture 8

**(A) Braitenberg** vehicles as an introduction to …

**(B) ... Programming** simple basic robots in the Robot Lab.

Rationale:- everyone should have some very basic hands-on experience of the problems involved in building a robot that interacts sensibly with its world.
Difference between **reality** and **simulation**.
Then facilities available for those who want to carry on in their own time, or for course projects.



# Braitenberg Vehicles

Braitenberg (1984), Vehicles: Experiments in Synthetic Psychology, MIT Press.

R. Pfeifer & C. Scheier (1999),"Understanding intelligence" MIT Press

Arkin (1998), Behavior-Based Robotics, MIT Press.

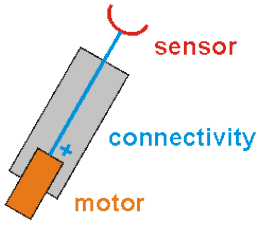


# BV

Simplest vehicle has just one sensor and one motor

It can only change its speed according to the stimulus

The +sign means positive excitation






# BV

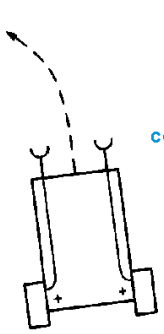

Crossed positive connections mean that it will turn towards the side with the greater stimulus

-- an **aggressor**

Contra-lateral excitation



# BV

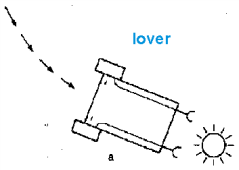



If the connections are not crossed, then it turns away from excitation – **coward**

Ipsilateral excitation



# BV

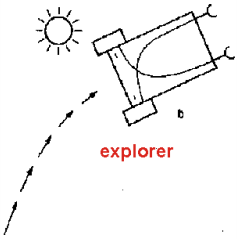



Uncrossed inhibitory connections (note the –sign) give ipsilateral inhibition:

Turns towards the light and slows down as it gets close

Note: there will need to be some constant positive signal to the motors as well, that gets inhibited by the effects of the stimulus

## BV




Contralateral inhibition:

Slows down on seeing light, but then turns away and speeds up again.

Again, some constant positive signal to motors needed also.



## Simulations of Agents

We generally reckon that simple 2-D simulations of agents should ideally be written by YOU in the language of your choice.

If you want 3-D with (fairly) realistic physics, then a Physics Engine like ODE is invaluable (http://www.ode.org/ )

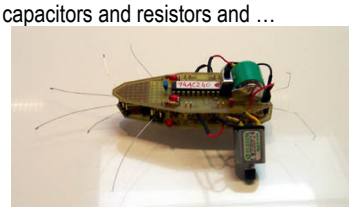
"ODE is an open source, high performance library for simulating rigid body dynamics " -- classes provided in January.

Or you could build robots for real – and a lot more interesting issues will crop up.



## Theory and Practice

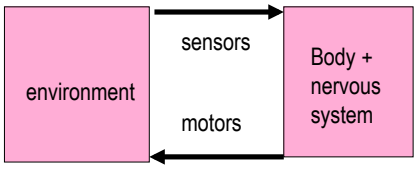

These sets of conceptual 'wires' could be implemented in real wires and capacitors and resistors and …



… or you can emulate all this with a computer



## Basics




Many choices, but in this case **body** = Lego and

**Nervous system** = C program running on a mini-computer, hooked up to various sensors and motors



## Program is a cheat

The computer program here is a bit of a cheat – it is **emulating** a nervous system of components passing **real values** around in **real time**.

Actually real values to/from motors/sensors are translated into **digital** signals, and real time is handled by going round a program loop indefinitely (… caution needed)

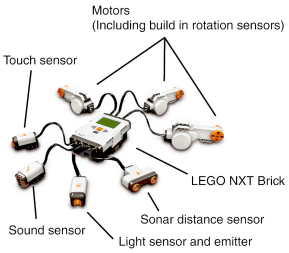
***SEE important FAQ !!*** Relates also to GA+CTRNN exercise *http://www.informatics.susx.ac.uk/users/inmanh/easy/alife10/TimeSteps.html*



## Lego NXT

We previously used our own EASyMind brick in a Lego structure ----- plug in sensors and motors appropriately --- and write a program to load onto it

But now Lego has improved, we are using the Lego NXT system

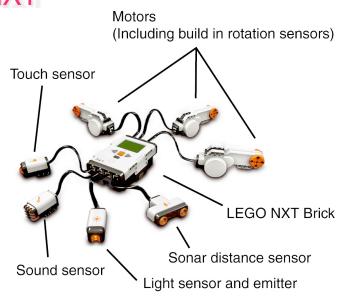


## NXT

Up to 3 motors (built-in rotary encoders, measure speed)

Up to 4 sensors:

- Light (emitter+sensor)
- Sonar (partly relates to distance)
- Touch
- Sound (microphone)






## Warnings about sensors/motors

In simulations, **magic** sensors deliver reliable information about eg distances. **Magic** motors move the vehicle precise distances.

In the real world, it is nothing like that.

Real sensors are noisy, and react to far more than you might guess.

Calibration of sensors

Wheels slip on the floor, get jammed on bumps.



## Programming NXT

USB port connects to a PC for downloading programs (also has Bluetooth)

Atmel 32-bit ARM processor, (AT91SAM7S256) 256 KB FLASH, 64 KB RAM and runs at 48 MHz.

There *is* a mode for (Labview-like) graphically programming it (simple, cannot do ANNs) – but we will be using RobotC (http://www.robotc.net/) which is a fairly basic C programming language with pre-written functions for controlling the NXT sensors motors and other peripherals.



## Useful links

Browse these before the Robot Lab class

RobotC **www.robotc.net**

Our local programming instructions **tinyurl.com/587zdx**

For more adventurous (build your own hardware) people

**Mindstorms.lego.com/Overview/nxtreme.aspx**



## What a simple program looks like

```
int LeftSensorValue;
int RightSensorValue;
task main()
{
        while(1)
        {
                LeftSensorValue = SensorValue(LeftSensor);
                RightSensorValue = SensorValue(RightSensor);
                motor[LeftMotor] = RightSensorValue;
                motor[RightMotor] = LeftSensorValue;
        }
}
```



## What does it do?

Every time round the loop, it reads in sensor values from where you told the program to look.

and sets the equivalent power levels (suitably scaled) to whatever you have plugged into the Motor Ports

All sorts of stuff is done in the background, hidden from you: A/D conversion, scaling factors, how fast round the loop?

This is an infinite loop *while(1)*

**Get this going first, then modify and extend**

## Worries about cheating

Sometimes people forget they are using a program to emulate a 'nervous system', and sometimes this breaks down.

**while ( 1) {**

**… … }**

How long does this loop take? Does the time vary according to circumstances? If you want something to happen at a particular rate, how do you ensure that?



## Robot lab classes

**Start at normal seminar times (check exact numbers of people_**

**TUESDAY** 2nd Nov 11:00 to 13:00 max

**THURSDAY** 4th Nov 11:00 to 13:00 max

**PLACE**: in Informatics Lab 4 – Chichester-2R226

2 people to a robot – the rule is that the *worst* programmer sits at the Keyboard to write the program(s), and the *clumsiest* person does the Robot assembly
See **http://tinyurl.com/587zdx**



## GA+CTRNN exercise

This is an optional exercise – but strongly recommended if you may beusing CTRNNs at some stage, for modelling Dynamical Systems – and particularly 'robot nervous systems'.

See via the course webpage

www.cogs.susx.ac.uk/users/inmanh/easy/alife10/ga_exercise2.html

Hand in your efforts, for feedback.



## Evolve CTRNN to display "Interesting Dynamics"

$$\dot{y}_i \equiv \frac{1}{\tau_i}\left(-y_i + \sum_{j=1}^{N} w_{ji}\sigma(y_j + \theta_j) + I_i\right) \quad i = 1, 2, ..., N$$

In simulation, node activations are calculated forward through time by straightforward time-slicing using Euler integration;

I.e. choose a **suitably small** time-slice delta_t, use the above equations to calculate for each node how much their values change in time delta_t, and update them all simultaneously

Then repeat many times over, like a slightly jerky movie, though if your delta_t is small enough, hopefully it will be fairly smooth!



## Simulating a Dynamical System using Update Steps

You will be doing this **both** for the GA+CTRNN exercise **AND** in the robot lab class.

So many people misunderstand what one is doing with Update Steps, that I have written a special FAQ

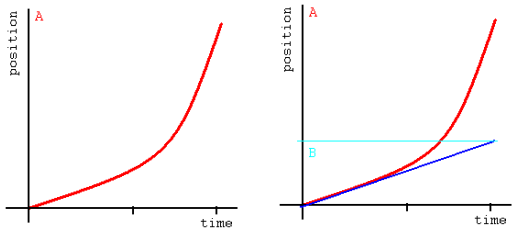
**www.cogs.susx.ac.uk/users/inmanh/easy/alife10/TimeSteps.html**

It is absolutely **ESSENTIAL** to make the update-steps **small-enough**,

So that your 'jerky approximation' is close enough to the smooth continuous dynamics of what you are simulating.

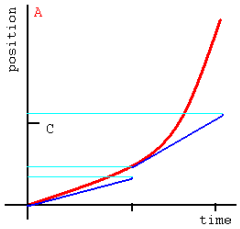


## Tangents to a Smooth curve




The Blue Line is close-enough to the curve for only a short distance

## Successive approximations …




… will only work if they are close enough

How close is that ?



## The ultimate test

The rigorous test is:

Pick a timestep that by Rule of Thumb (see below) you *hope* is small enough, run the code, see what result it gives.

Then *run the whole program again* with a different smaller timestep, 1/10 the size

If the end-result is (almost) the same, you were OK -- stick with it.

If the end-result is significantly different, your original guess was not small enough. Repeat steps 2-4 until you get there



## Rule of Thumb for DSs including CTRNNs

Look at the smallest relevant timescale of the real dynamics of what you are simulating.

WIth CTRNNs, that might be the smallest time-parameter value that any node can have.

Then pick a simulation-update timestep at least one order of magnitude (1/10) smaller than that

Then you should also go through the rigorous test above, to double-check..

**www.cogs.susx.ac.uk/users/inmanh/easy/alife10/TimeSteps.html**



## Some Public Service Announcements

1. Looking ahead to doing your Alife project
2. (Plagiarism)
3. Standards in Science
4. Bonfire Night



## Alife projects

Next week I shall give a lecture on 'how to plan an Alife project'….

… and the following week (deadline to be decided) you should be submitting your proposal for your own project, for feedback.

One ***strongly*** recommended approach is to take some existing piece of published work (paper, textbook, on web …) and replicate it – then adjust it, tweak it, play with it, extend it.



## Plagiarism

This of course (…like most other work in science…) requires you to distinguish carefully in your report between ideas/code/text that you learnt from others – and new (or new tweaks to) ideas/code/text that you brought to this yourself.

***Any*** piece of work contains both. The 'Academic Development Course' should have covered this issue, but ***y***ou will also see among my FAQs a FAQ page of my own on this.

Go to

http://www.informatics.susx.ac.uk/users/inmanh/FAQ/FAQ.html

And follow link to "Local access only" FAQs

## Science: Honesty, Integrity, Proof

You should be trying to demonstrate something that you **believe** to be true, that you can lay out clearly and with **integrity**, using proper standards of **proof**. If there are gaps in your argument, you will be scrupulous in searching them out, and bringing them to the attention of your readers.

Honesty and integrity should also include maintaining an open mind -- being able, indeed keen, to question one's own beliefs and revise them when evidence and argument point to new conclusions. Honesty includes being scrupulous to give credit to the origins of ideas, and *of course* no plagiarism; see my relevant FAQ via my homepage.



## Standards of Proof

Imagine that you are giving the prosecution case against a defendant in a murder trial, with a potential life prison sentence.

The defence lawyer is convinced his client is innocent, and will do everything possible to point out the flaws in your arguments. For a scientific paper, we expect **higher** standards than the sort of prosecution case that will convince a jury.



## Presenting Evidence

In a court case, every item that fits into the prosecution's argument will be cited, labelled, presented, and cross-examined by the other side. You must meet at least these standards in a scientific paper.

That is why assertions drawn from the literature must be sourced. That is why quotations from Wikipedia are not acceptable, however useful.

Wikipedia might be in tracking down the proper evidential support. In a court case it is not acceptable to say "I heard someone in the pub say he was the murderer" -- you have to bring that person to court and get them to give evidence under cross-examination..



## Reminder for Friday Nov 5th

**Fri 5th is Bonfire Night**
**Lewes Bonfire (6am-2am)**
**Main processions ~7pm on**
**6 Bonfires from ~10pm**
**www.lewesbonfirecouncil.org.uk/**

Semi-Compulsory

Artificial Life

Exercise !!!!