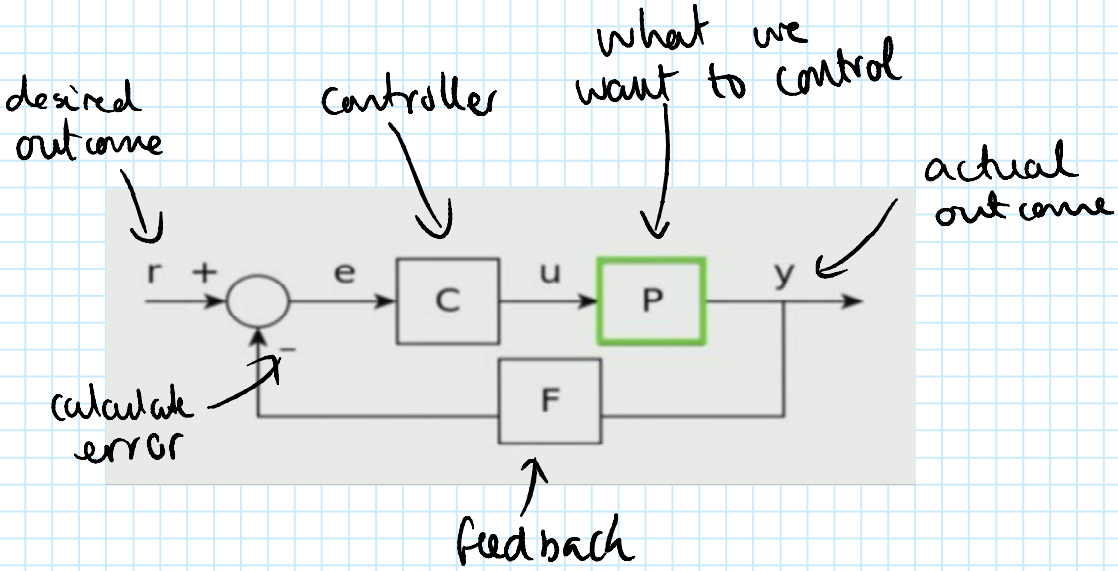
Bio-mimetic Robot Control

21 May 2017 17:23

- Nature is good, but the aim isn't optimality, the aim is to be good enough.
- Typical "classical" robot controller:

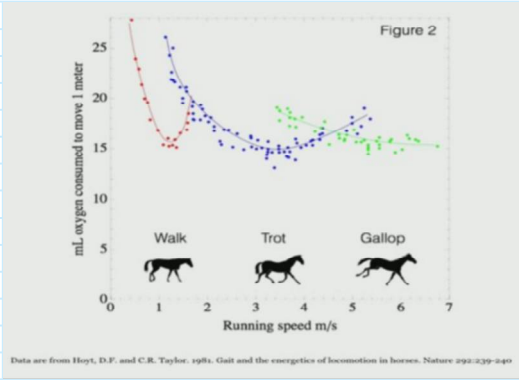


- This is very centralised, unlike natural systems
- Classically engineered versus natural controllers,

Engineered	Natural
- Easier to analyse range of behaviours. - Rigid bodies to increase predictability. - Difficult to hand engineer - Slower / less robust	- Adaptive over time - Non-rigid, which introduces many more degrees of freedom - Many more moving parts. - distributed control

Central Pattern Generators (CPGs):

- CPGs are biological neural networks that produce rythmic patterned outputs without sensory feedback.
- A typical example is gaits,



- A way to artificially implement CPGs is nonlinear oscillators,

Artficial CPGs using Nonlinear Oscillators

We can implement a controller that uses nonlinear dynamics to create a system that robustly tends towards a cyclic attractor in the state space. We will use the system of equations,

$$\tau\dot{v} = -\alpha\frac{x^2 + v^2 - E}{E}v - x$$
$$\tau\dot{x} = v$$

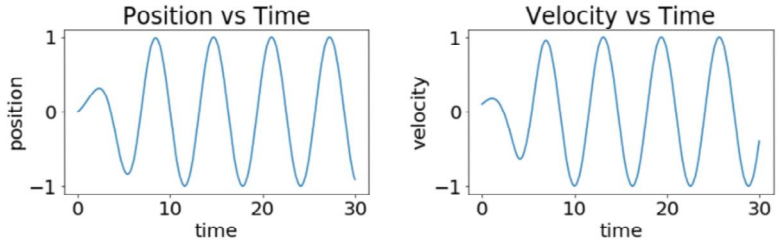
where x and v are the position and velocity, and α, τ and E are positive real parameters (note: the notation, $\dot{v}$, indicates the time-derivative of the variable v). This oscillator has the interesting property that its limit cycle behavior is a sinusoidal signal with amplitude and period $2\pi\tau$. The state variable $x(t)$ indeed converges to $\hat{x}(t) = \sqrt{E}\,sin(t/\tau + \varphi)$ where φ depends on the initial conditions.

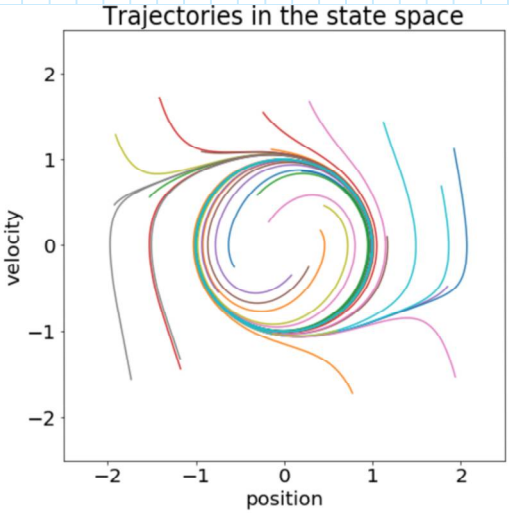
These differential equations can be used in conjunction with Euler's approximation method to update state (velocity and position). The general update rule is of the form,

$$f_{n+1} = f_n + \delta t\, F(f_n, t_n)$$

These are the equations derived in the paper, Simulation and Robotics Studies of Salamander Locomotion: Applying Neurobiological Principles to the Control of Locomotion in Robots

These notes came from a Jupyter notebook I wrote for exploring the topic.





As shown, both the position and velocity fall into cyclical patterns. The intial condition is at $(0, 0.1)$ in the position-velocity state space, note that the only position that doesn't converge on towards the perfect oscillator is $(0, 0)$ as from the equations above $\tau\dot{v} = 0$, so

$$f_{n+1} = f_n + \delta t\, F(f_n, t_n) = f_n + \delta t \times 0 = f_n$$

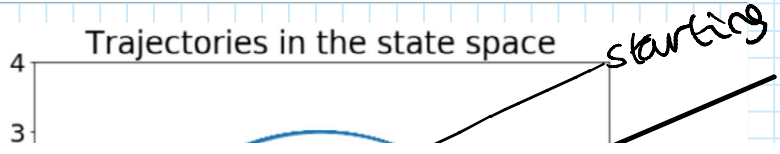
We can see other points converging on the same cycle in state space by observing the trajectories from random initial conditions,

Dynamic Attractors

The cycle that all the trajectories tend towards is called an attractor. Now to see how the system can be controlled (i.e. the attractor changed), let's varying the oscillating parameters as a function of time. As an example, we define,

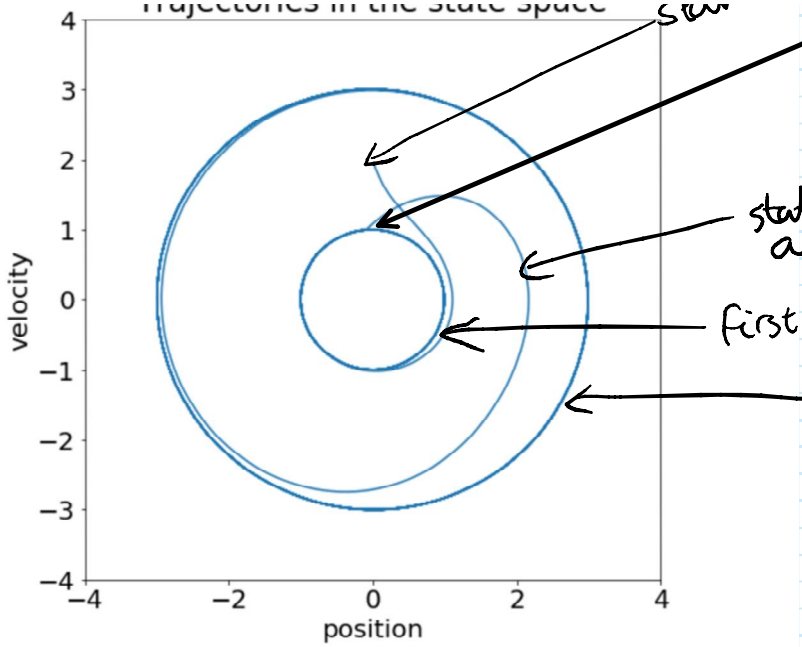
$$E(t) = \begin{cases} 1 & t < 50 \\ 9 & \text{otherwise} \end{cases}$$

Therefore at time $t = 50$, the system will start tending towards an attractor that corresponds to a greater oscillation amplitude.



starting pos.

$t=50$, i.e. the moment the

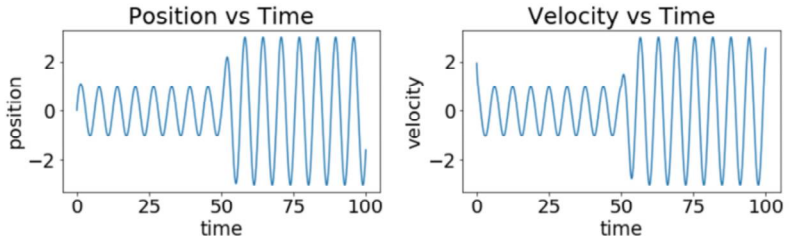


t=50, i.e. the moment the transition starts

state on transit to new attractor

first attractor

second attractor

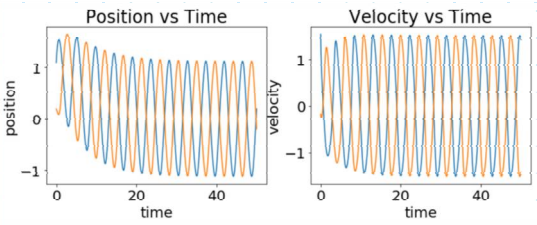


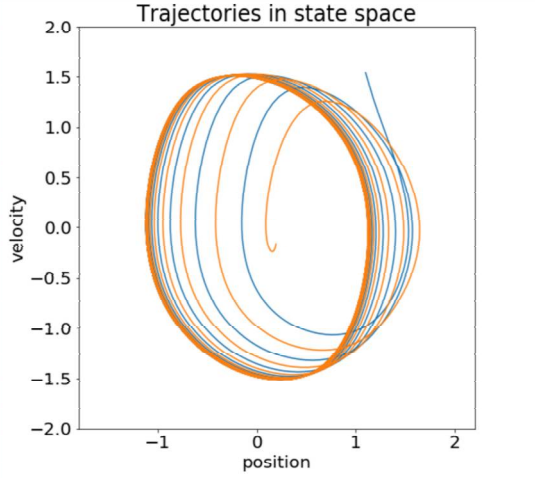
Coupling Oscillators

Here we will create a system of equations to update the internal states of multiple oscillators, whereby each affect each other. For each oscillator i, we define the dynamical equations,

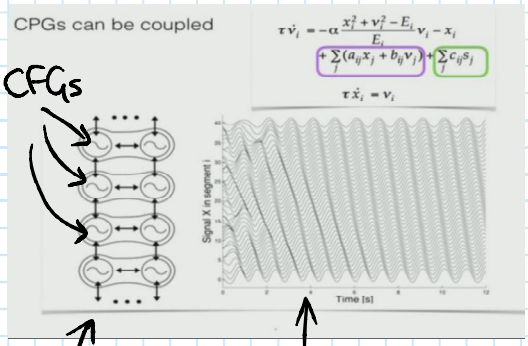
$$\tau \dot{v}_i = -\alpha \frac{x_i^2 + v_i^2 - E_i}{E_i} v_i - x_i + \underbrace{\sum_{j \neq i} (a_{ij} x_j + b_{ij} v_j)}_{\text{Influence of oscillators on one another}} + \underbrace{\sum_j (c_{ij} s_j)}_{\text{Influence from external factors}}$$

$$\tau \dot{x}_i = v_i$$





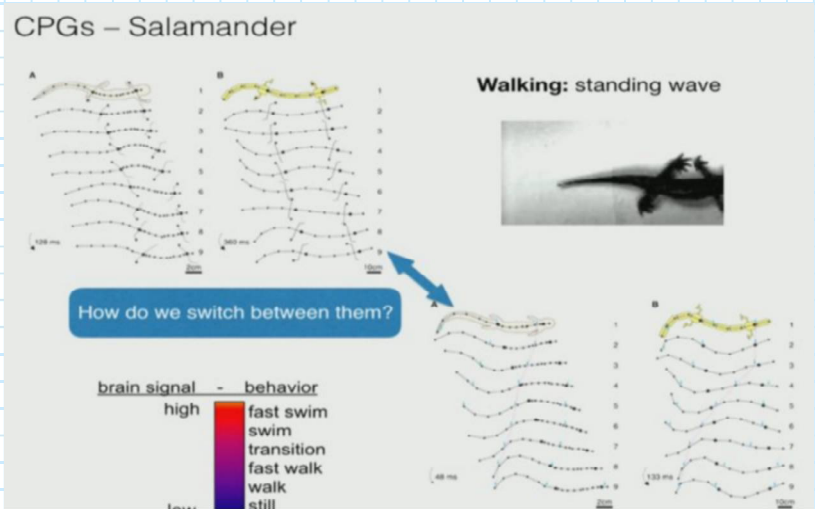
Here the influence matrices have been set up in such a way to push the oscillators into anti-phase behaviours. This is somewhat analgous to two legs running, with one foot reaching a maximum in



coupling diagram

CPGs eventually synchronise.

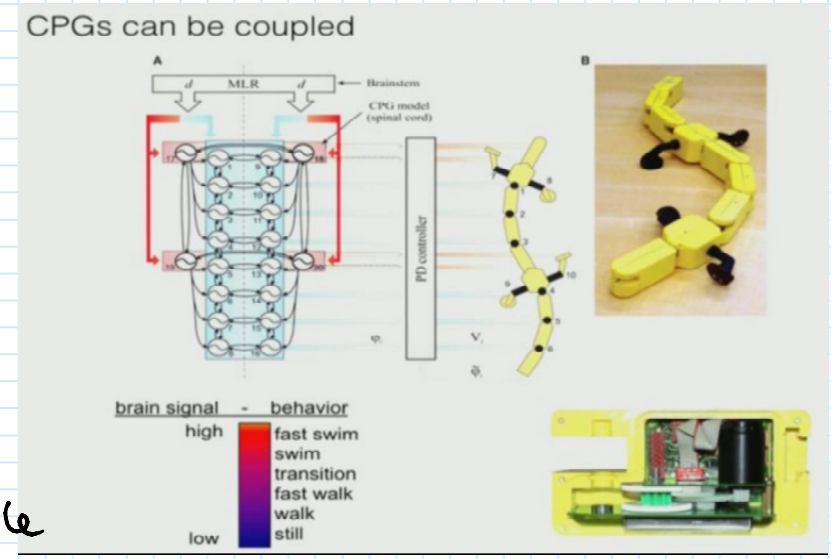
Example:



position

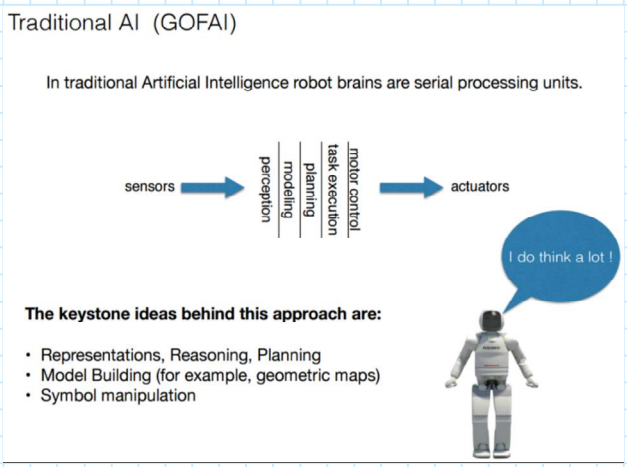
Here the influence matrices have been set up in such a way to push the oscillators into anti-phase behaviours. This is somewhat analgous to two legs running, with one foot reaching a maximum in while the other a minimum.

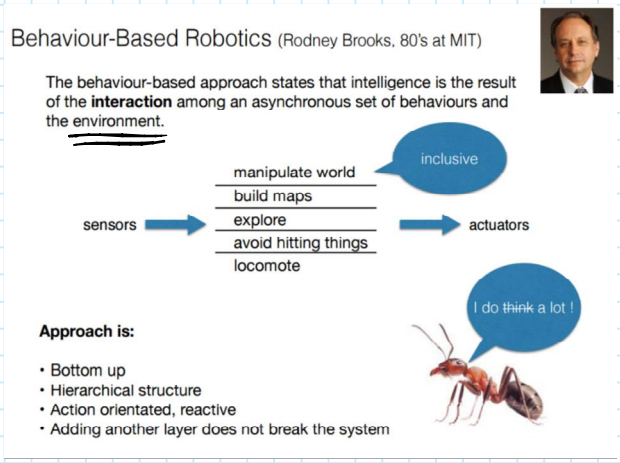
Therefore by using biomimicry a more optimal robot has been designed. High level functions are controlled centrally and CPGs handle low level tasks by keep state as cyclical, stable attractors.

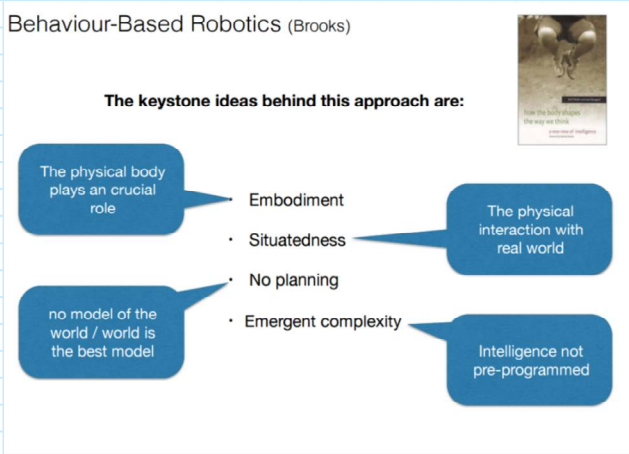


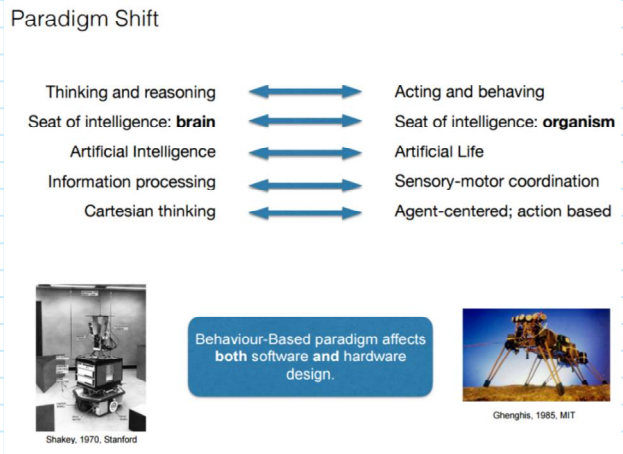
Behaviour Based Robotics (BBR):

Behaviour is defined as a reaction to stimulus.









- BBR is more energy efficient, but less optimal. In less noisy environment classical

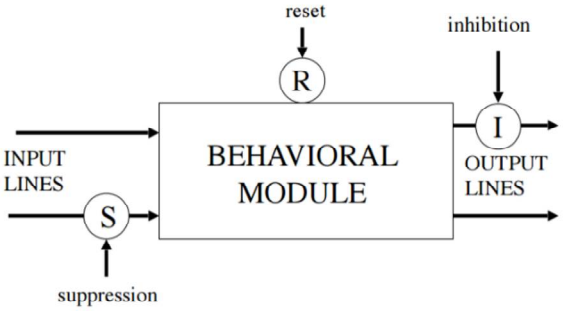
optimal. In less noisy environment classical techniques prevail.

Example of behaviours

- **Exploration/directional behaviours** (move in a general direction) – heading based, wandering
- **Goal-oriented appetitive behaviours** (move towards an attractor) – discrete object attractor, area attractor
- **Aversive/protective behaviours** (prevent collision) – avoid stationary objects, elude moving objects (escape), aggression
- **Path following behaviours** (move on a designated path) – road following, hallway navigation, stripe following
- **Postural behaviours –** balance, stability
- **Social/cooperative behaviours** – sharing, foraging, flocking

- **Perceptual behaviours** – visual search, ocular reflexes
- **Walking behaviours** (for legged robots) – gait control
- **Manipulator-specific behaviours** (for arm control) – reaching, moving
- **Gripper hand behaviours** (for object acquisition) grasping
- etc.

Modular design makes it easier to use in conjunction with other techniques and layer behaviours

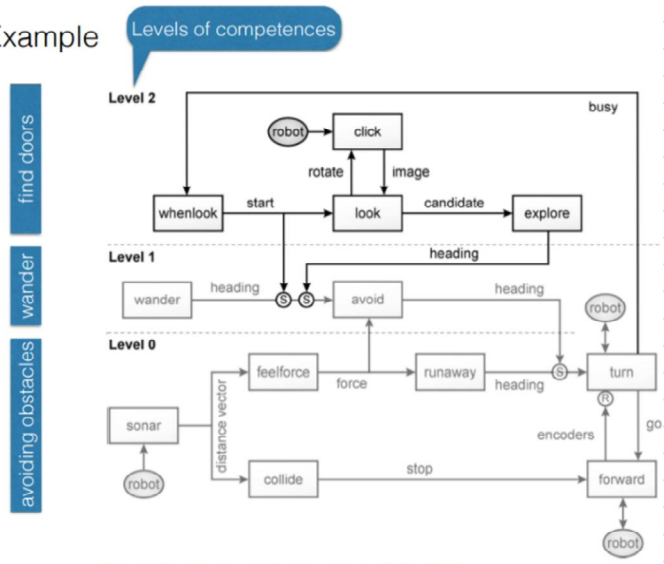


Basic building block

Augmented Finite State Machine

- local computation
- mappable into hardware
- no global clock, memory, bus
- no central models

Example



"A robust layered control system for a mobile robot"
R Brooks - IEEE journal on robotics and automation, 1986

Each level is a different behavioural module.
Upper layers trust lower
Earlier behaviours are not modified
The system is constructed by a design-test-debug cycle, there is no "correct" way to develop or analyse.