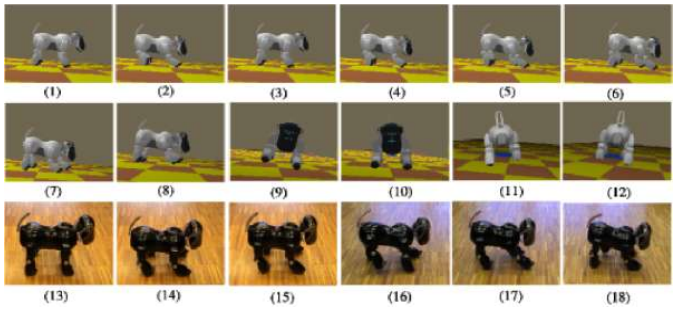
30/03/2006

How to simulate a spring on an Aibo knee

Giorgio Brambilla



I have simulated a spring behaviour on the Aibo knees, using a dynamical system.
A spring-damper rotational system produces in the knee torques proportional to two components: the angle (γ , i.e. the spring torque) and the the angular velocity ($\dot{\gamma}$, i.e. the viscous damping force): $T = k\gamma - d\dot{\gamma}$.

The level chosen to simulate the spring is the Open-R Aibo programming language.
To simulate a spring on Open-R we have the following instruments:

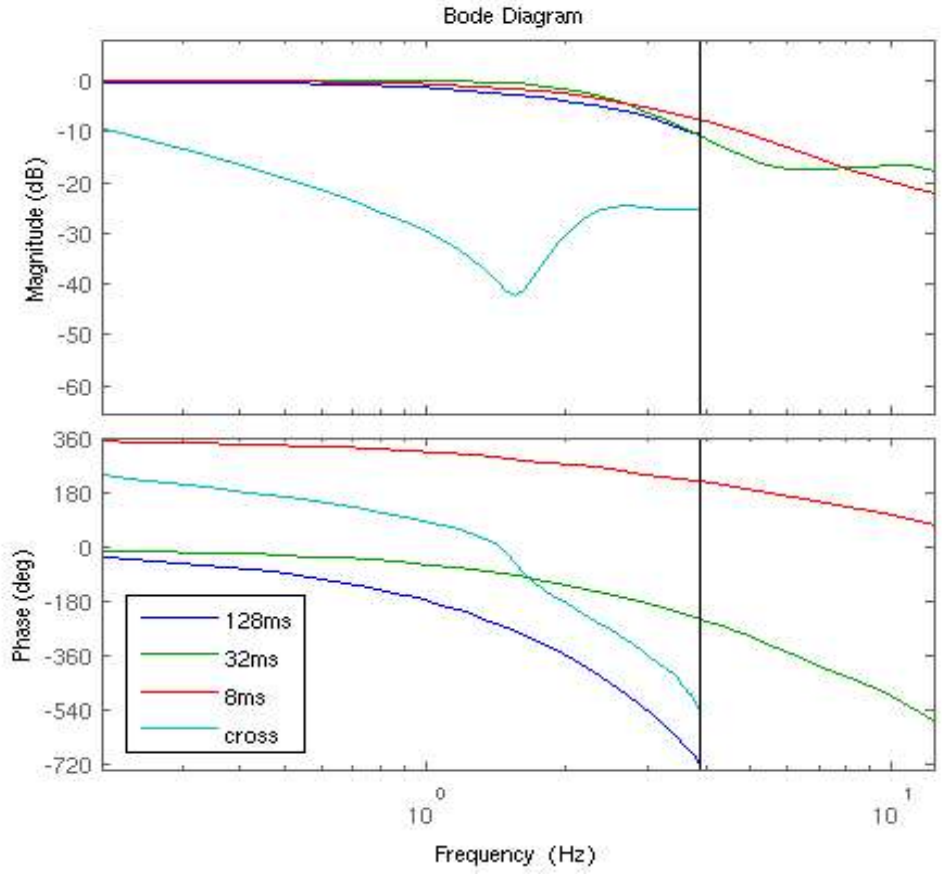
- Knee encoder information;
- An approximative model of the hardware controller;
- An Open-R built-in function to set the knee position;

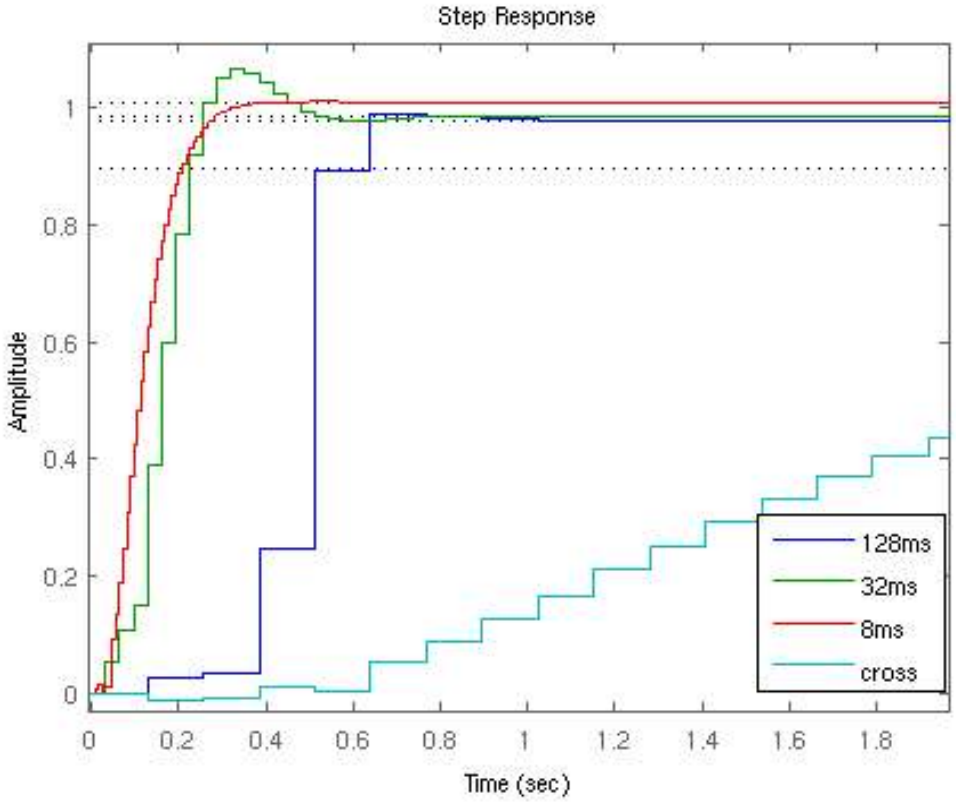
Aibo knees model:

In order to simulate a physics function - such as the spring-damp law - on Aibo, it is needed to know the controller model. In other words, we need to know (an approximation) of the servo transfer function. Sony does not give any information about it. Hence, we have applied a linear inversion method to identify the transfer function.

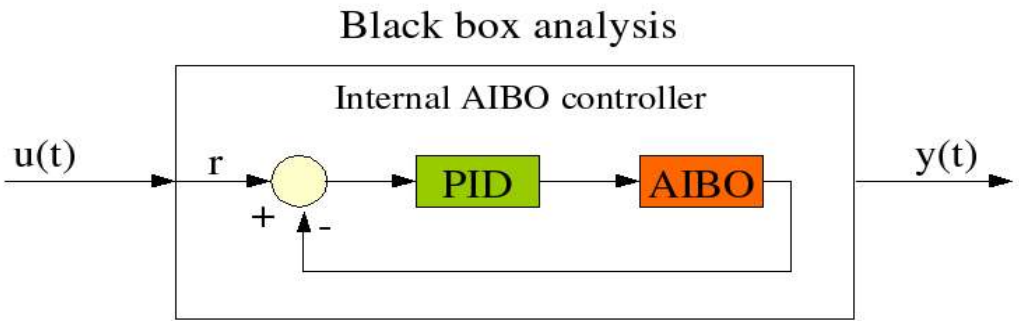
We have tested the frequency response function of the Aibo knees. Giving a series of random positions to reach, with a frequency of variation of the random signal, in the range of work interested. In this case between 0 and 3 Hz (range of walking frequency). These random positions are for the controller « step functions », good candidates to study the transfer function. Using a linear inversion method we have identified the transfer functions of the servos.

Of course, this strategy can not found a good approximation of a « strongly non linear » system as a « step in motor » servos and a really approximated encoder. But it was the fastest method to approximate the delay of response of the controller loop and to understand if it was possible simulate a spring-damper system on such platform.


Bode Diagram
Magnitude (dB)
0
-10
-20
-30
-40
-50
-60
Phase (deg)
360
180
0
-180
-360
-540
-720
128ms
32ms
8ms
cross
10^0
10^1
Frequency (Hz)


Step Response
Amplitude
0
0.2
0.4
0.6
0.8
1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
Time (sec)
128ms
32ms
8ms
cross

In figures, the experiments correspond to different time iteration steps of the controller program on aibo (8ms, 32ms, 128ms) and the last experiment (cross) is computed using the Cross-Compilation features of Webots.



Our choices:

Once identified the model we have decided to use:

1. 8 ms as time step, to have the best frequency response;
2. about 58ms of pure delay to a step input response.

Our solution:

A spring-damp dynamical system is a system of two differential equations. These equations are integrated on a delta step of 8ms, and they attempt to simulate a real spring-damp system. But such a system needs to know the value of the system variable (x) at time t to obtain the next value at time t+dt. $x(t+dt)=f(x(t))$

The robot has a pure delay of 58ms. But 58ms are too much to simulate a fluid spring motion and moreover our controller can make much better, integrating the equations each 8ms. Hence, we have decided to make a prediction of the value of the spring after 58 seconds, and use this prevision to control the servos. Of course, as our controller has 8ms as time step we can repeat this prediction each 8ms, making the system more precise.

```
Pseudo code:
while(true){
      next_knee_angle = Runge-Kutta (f, x(i-1), using 58ms);
      x(i) = Runge-Kutta (f, x(i-1), using 8ms);
      i++;
      delay of 8ms;
}
where x is the state variable that corresponds to the angle, and f is the
spring-damp law.
```

Spring-Damp dynamical system used:

$$\dot{x} = -(k/m)\,x - (d/m)\,\dot{x}\,;$$

where:

m = 1 [kg]	when the real mass change, the equation remains the same. But a higher (or a lower) mass (experimentally) make the oscillating behavior similar to a real spring-mass system, decreasing (or increasing) the oscillating frequency;
d	damping parameter;
k	oscillator constant.