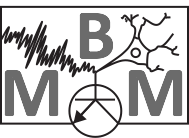
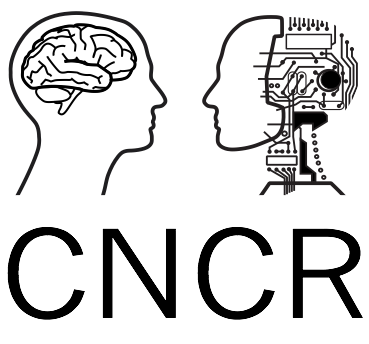
**M**ind, **B**rain, and **M**odels 22/23



Control Theory

In this lab, we will analyse and control an inverted pendulum system with the dynamic equation of motion:

invpendulum

where *x* is the cart displacement, *α* is the inclination angle from the vertical position, *mpendulum* and *lpendulum* are the mass and length of the pendulum. The system inputs are the acceleration applied to the cart *acart* *= d2/dt2 x* and a distortion torque *τdist* perpendicular to the pendulum orientation. Measurable outputs are all entries of the state vector: The angle *α* and its derivative *d/dt* *α* as well as the cart position *x* and velocity *d/dt x.*

Download and open the file “invpendulum.m”. The relevant variables required for the lab are

tau\_dist: a vector of the same size as *t* (time), containing disturbance torques (rotational forces, clockwise positive) for every time step

state: a matrix of state variables:

[*alpha, d/dt alpha, cart position, cart velocity*]  
- the matrix grows one row with every simulation time step.

- the latest state is always found in the last row

- if you wanted to play with delayed feedback, you could read earlier values, rather than the latest.

a\_cart: the acceleration to be applied to the cart (i.e. a force pushing the cart left or right).

The workshop has 6 tasks. Solutions to tasks 1-4 will be provided at the end of the workshop, and Tasks 5 and 6 should be done independently and submitted as a short report.

*1) Initialization and Uncontrolled pendulum dynamics.* Initialise tau\_dist with zero values and set the initial state state to a small alpha angle (e.g., 0.1) and zero values for d/dt alpha, cart position, cart velocity. Execute invpendulum and observe the pendulum motion. Design an impulse input vector tau\_dist with an impulse torque of magnitude 30 at t=1 (and zeros otherwise) and set the state variable to zeros(1,4). Execute invpendulum and observe the pendulum motion.

*2) Design a proportional controller to prevent the pendulum from falling.* Update the system input a\_cart every time step with the result of a proportional control law using the state variable. Test your code for *KPα = 200*. Is the controlled system stable with this controller? Now vary the control gain in the range *KPα* between *0* and *500*.

*3) Design a PD controller to prevent the pendulum from falling*. Set the proportional control gain *KPα = 200* and extend the control law to a PD controller using the rotational velocity value of d/dt(alpha) in the state variable. Vary the derivative gain *KDα* between 0 and 150 and observe the system behaviour. Find a value of *KDα* for which the pendulum stays upright.   
**Hint**: *d/dt e = d/dt αdes – d/dt αmeas*

4) *Design and analyse a dual controller of angle and position*Fix the controller gains of the angle PD controller to *KPα = 200* and *KDα = 50.* Formulate a second PD control law with gains *KPx* and *KDx* to achieve the positional goal *xdes = 0* by calculating a second control variable a\_cart2. Add the control values of both controllers together to compute the active control signal *a\_*cart.

*5) Describe the system behaviour in your report.* There is no fixed format for this description, but it should have a brief text section and an appropriately-labelled figure. For example, the state variable can be returned by the Matlab function and plotted against time. Alternatively, a screenshot of the simulation may be informative. Or use one of the analysis and concept covered in the lecture. Submit also the code - and good coding would typically include explanatory comments.

*6) Grid search.* Vary *KPx* and *KDx* in the range *KPx = 0…100* and *KDx = 0…50* to find one set of parameters that gives a stable solution. Search the parameter space systematically to find where the solutions are stable: this should be done by putting the main section of the program in a for loop, and defining a criteria for whether the controller is successful. You can define a ”metric” for stability or an acceptable range of position and angular velocities for a final state of the simulation that is considered stable.

NB Once you are ready to run the code of the parameter search, it would be helpful to turn off the real-time display, so that the simulations can run as fast as possible. For this, set plotting=false;

Plot a graph showing the sensitivity of the solutions (e.g., the final state versus the parameters *KPx* and *KDx )*. Describe the graph in your report.