Introduction to Robotics

Assignment #5

Caus Danu 7014833 | Massimo Innocentini 7016313 | Daniel Bremer 6834136

Task 5.1 (4 points) Basis-Splines - Direct Computation:

For a **polynomial degree of 0** we will have the following:

$$N_{0,1}(t) = \begin{cases} 1 & \text{if } t_0 \leq t < t_1 \\ 0 & \text{otherwise} \end{cases}$$

For a **polynomial degree of 1** we will have the following basis functions:

$$N_{0,2}(t) = \frac{t - t_0}{t_1 - t_0} N_{0,1}(t) + \frac{t_2 - t}{t_2 - t_1} N_{1,1}(t)$$

For interval [t0, t1) only $N_{0,1}(t)$ is set. Therefore:

$$s(t) = \frac{t - t_0}{t_1 - t_0}, \text{ for } t_0 \neq t_1$$

For interval [t1, t2) only $N_{1,1}(t)$ is set. Therefore:

$$s(t) = \frac{t_2 - t}{t_2 - t_1}, \text{ for } t_1 \neq t_2$$

For a **polynomial degree of 2** we will have the following basis functions:

$$N_{0,2}(t) = \frac{t - t_0}{t_1 - t_0} N_{0,1}(t) + \frac{t_2 - t}{t_2 - t_1} N_{1,1}(t)$$

$$N_{1,2}(t) = \frac{t - t_1}{t_2 - t_1} N_{1,1}(t) + \frac{t_3 - t}{t_3 - t_2} N_{2,1}(t)$$

$$N_{0,3}(t) = \frac{t - t_0}{t_2 - t_0} N_{0,2}(t) + \frac{t_3 - t}{t_3 - t_1} N_{1,2}(t)$$

or, more complete formula:

$$N_{0,3}(t) = \frac{t - t_0}{t_2 - t_0} * \left(\frac{t - t_0}{t_1 - t_0} N_{0,1}(t) + \frac{t_2 - t}{t_2 - t_1} N_{1,1}(t) \right) + \frac{t_3 - t}{t_3 - t_1} * \left(\frac{t - t_1}{t_2 - t_1} N_{1,1}(t) + \frac{t_3 - t}{t_3 - t_2} N_{2,1}(t) \right)$$

For interval [t0, t1) only $N_{0,1}(t)$ is set. Therefore:

$$s(t) = \frac{t - t_0}{t_2 - t_0} \frac{t - t_0}{t_1 - t_0}, \text{ for } t_0 \neq t_1 \neq t_2$$

For interval [t1, t2) only $N_{1,1}(t)$ is set. Therefore:

$$s(t) = \frac{t - t_0}{t_2 - t_0} \frac{t_2 - t}{t_2 - t_1} + \frac{t_3 - t}{t_3 - t_1} \frac{t - t_1}{t_2 - t_1}, \text{ for } t_0 \neq t_1 \neq t_2 \neq t_3$$

For interval [t2, t3) only $N_{2,1}(t)$ is set. Therefore:

$$s(t) = \frac{t_3 - t}{t_3 - t_1} \frac{t_3 - t}{t_3 - t_2}, \text{ for } t_1 \neq t_2 \neq t_3$$

For a **polynomial degree of 3** we will have the following basis functions:

$$N_{0,1}(t) = \frac{t-t_0}{t_1-t_0}N_{0,0}(t) + \frac{t_2-t}{t_2-t_1}N_{1,0}(t)$$

$$N_{1,1}(t) = \frac{t-t_1}{t_2-t_1}N_{1,0}(t) + \frac{t_3-t}{t_3-t_2}N_{2,0}(t)$$

$$N_{2,1}(t) = \frac{t-t_2}{t_3-t_2}N_{2,0}(t) + \frac{t_4-t}{t_4-t_3}N_{3,0}(t)$$

$$N_{0,2}(t) = \frac{t-t_0}{t_2-t_0}N_{0,1}(t) + \frac{t_3-t}{t_3-t_1}N_{1,1}(t)$$

or, more complete formula:

$$N_{0,2}(t) = \frac{t-t_0}{t_2-t_0} * \left(\frac{t-t_0}{t_1-t_0}N_{0,0}(t) + \frac{t_2-t}{t_2-t_1}N_{1,0}(t)\right) + \frac{t_3-t}{t_3-t_1} * \left(\frac{t-t_1}{t_2-t_1}N_{1,0}(t) + \frac{t_3-t}{t_3-t_2}N_{2,0}(t)\right)$$

$$N_{1,2}(t) = \frac{t-t_1}{t_3-t_1}N_{1,1}(t) + \frac{t_4-t}{t_4-t_2}N_{2,1}(t)$$

or, more complete formula:

$$N_{1,2}(t) = \frac{t-t_1}{t_3-t_1}\left(\frac{t-t_1}{t_2-t_1}N_{1,0}(t) + \frac{t_3-t}{t_3-t_2}N_{2,0}(t)\right) + \frac{t_4-t}{t_4-t_2}\left(\frac{t-t_2}{t_3-t_2}N_{2,0}(t) + \frac{t_4-t}{t_4-t_3}N_{3,0}(t)\right)$$

$$N_{0,3}(t) = \frac{t-t_0}{t_3-t_0}N_{0,2}(t) + \frac{t_4-t}{t_4-t_1}N_{1,2}(t)$$

or, more complete formula:

$$N_{0,3}(t) = \frac{t-t_0}{t_3-t_0}\left[\frac{t-t_0}{t_2-t_0}*\left(\frac{t-t_0}{t_1-t_0}N_{0,0}(t)+\frac{t_2-t}{t_2-t_1}N_{1,0}(t)\right)+\frac{t_3-t}{t_3-t_1}*\left(\frac{t-t_1}{t_2-t_1}N_{1,0}(t)+\frac{t_3-t}{t_3-t_2}N_{2,0}(t)\right)\right]+$$
$$+\frac{t_4-t}{t_4-t_1}\left[\frac{t-t_1}{t_3-t_1}\left(\frac{t-t_1}{t_2-t_1}N_{1,0}(t) + \frac{t_3-t}{t_3-t_2}N_{2,0}(t)\right) + \frac{t_4-t}{t_4-t_2}\left(\frac{t-t_2}{t_3-t_2}N_{2,0}(t) + \frac{t_4-t}{t_4-t_3}N_{3,0}(t)\right)\right]$$

For interval [t0, t1) only $N_{0,0}(t)$ is set. Therefore:

$$s(t) = \frac{t-t_0}{t_3-t_0}\frac{t-t_0}{t_2-t_0}\frac{t-t_0}{t_1-t_0}, \text{ for } t_0 \neq t_1 \neq t_2 \neq t_3$$

For interval [t1, t2) only $N_{1,0}$ is set. Therefore:

$$s(t) = \frac{t-t_0}{t_3-t_0}\frac{t-t_0}{t_2-t_0}\frac{t_2-t}{t_2-t_1}+\frac{t-t_0}{t_3-t_0}\frac{t_3-t}{t_3-t_1}\frac{t-t_1}{t_2-t_1}+\frac{t_4-t}{t_4-t_1}\frac{t-t_1}{t_3-t_1}\frac{t-t_1}{t_2-t_1}, \text{ for } t_0 \neq t_1 \neq t_2 \neq t_3 \neq t_4$$

For interval [t2, t3) only $N_{2,0}$ is set. Therefore:

$$s(t) = \frac{t-t_0}{t_3-t_0}\frac{t_3-t}{t_3-t_1}\frac{t_3-t}{t_3-t_2}+\frac{t_4-t}{t_4-t_1}\frac{t-t_1}{t_3-t_1}\frac{t_3-t}{t_3-t_2}+\frac{t_4-t}{t_4-t_1}\frac{t_4-t}{t_4-t_2}\frac{t-t_2}{t_3-t_2}, \text{ for } t_0 \neq t_1 \neq t_2 \neq t_3 \neq t_4$$

For interval [t3, t4) only $N_{3,0}$ is set. Therefore:

$$s(t) = \frac{t_4-t}{t_4-t_1}\frac{t_4-t}{t_4-t_2}\frac{t_4-t}{t_4-t_3}, \text{ for } t_1 \neq t_2 \neq t_3 \neq t_4$$

Task 5.2 (2 points) Basis-Splines - Uniform Visualization:

For all four orders we have used exactly the functions derived in the previous exercise by coding them in a Matlab program (source code attached) and got the following plots:

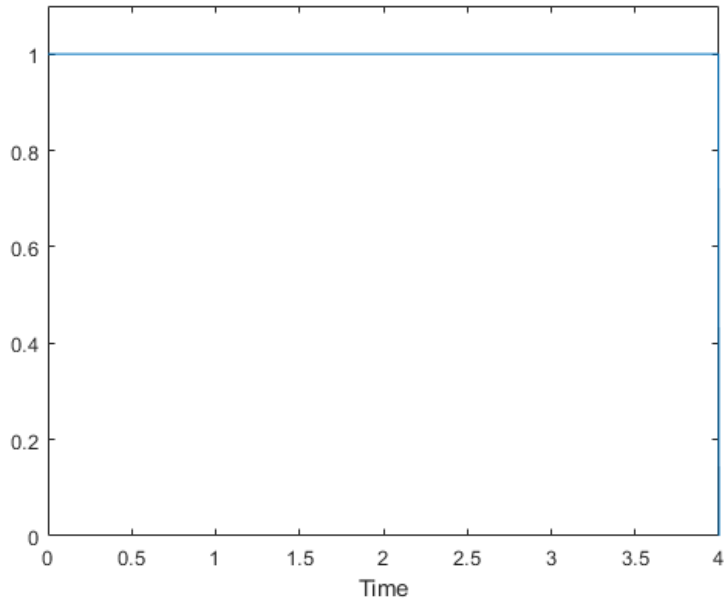

Figure 1: Order 1 Spline

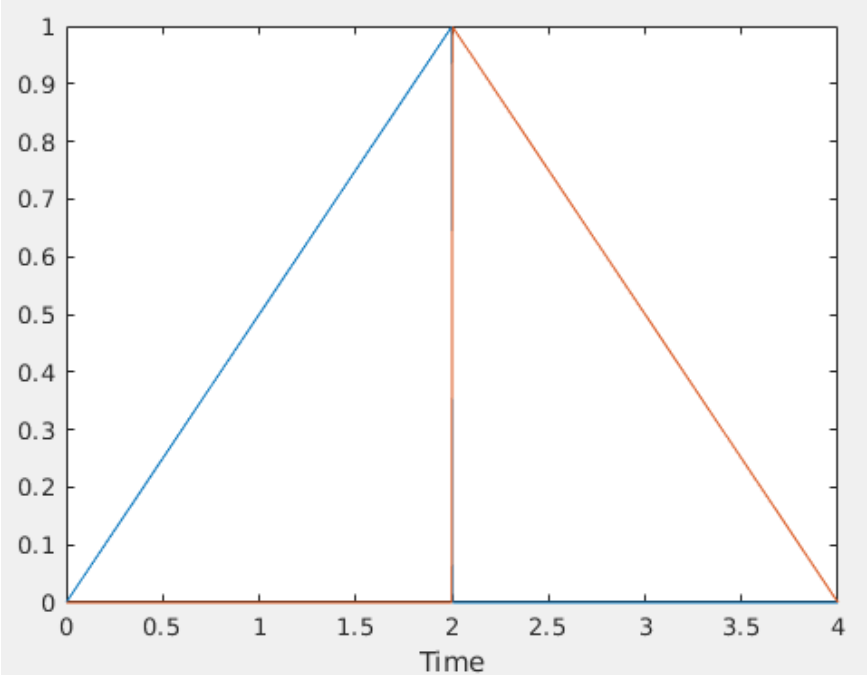

Figure 2: Order 2 Spline

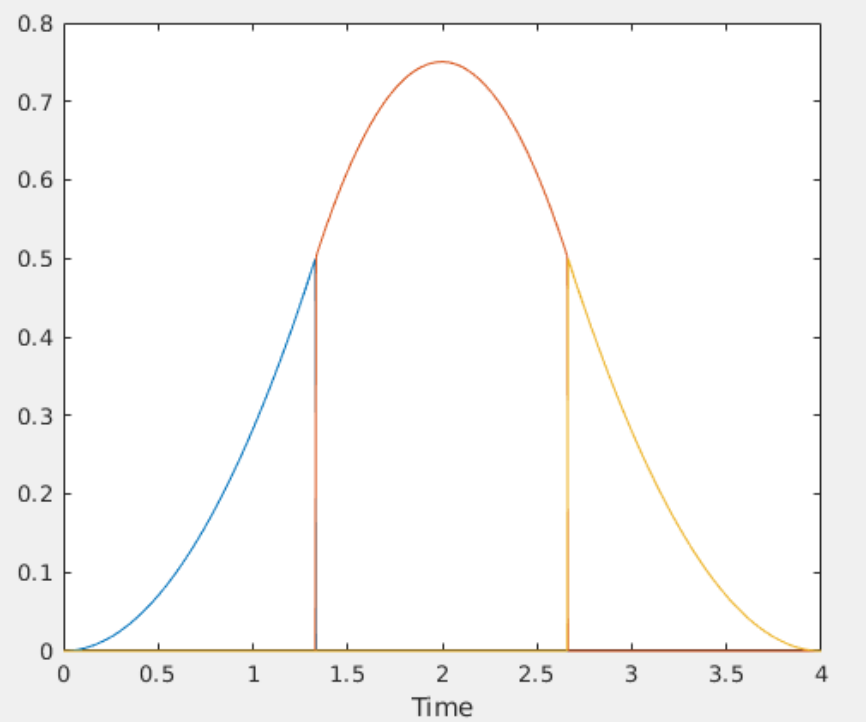

Figure 3: Order 3 Spline

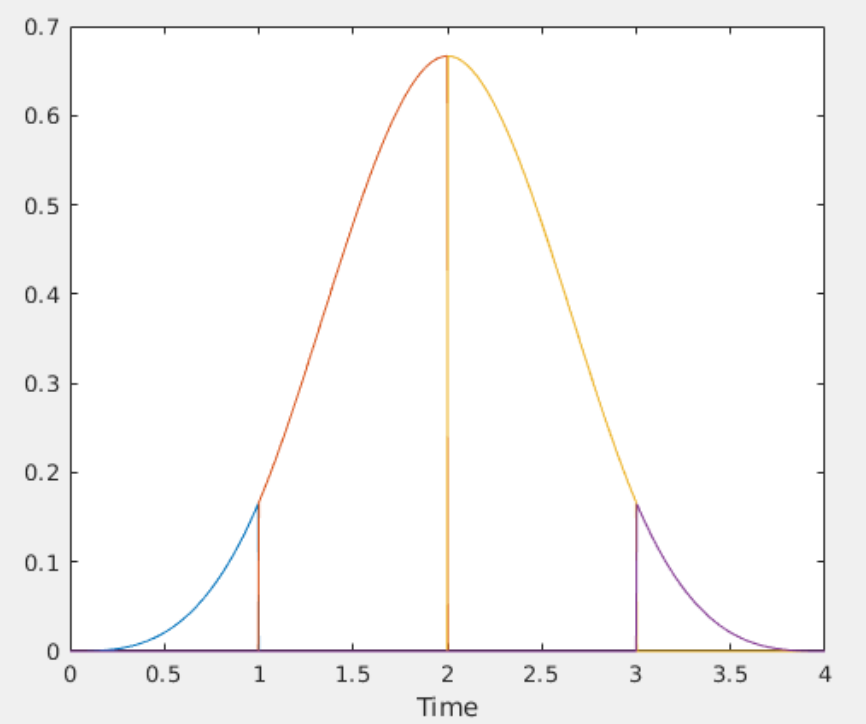

Figure 4: Order 4 Spline

Task 5.3 (4 points) Basis-Splines - Non-uniform Visualization:

For this exercise we have used initially the **bspligui** convenient Matlab tool, setting the 4 knots graph-

ically via the movable vertical axes (see figure 5). The tool also shows the 3 subsequent derivatives of the original bspline. Afterwards we have made use of our derived formula and got the result plotted in figure 6.

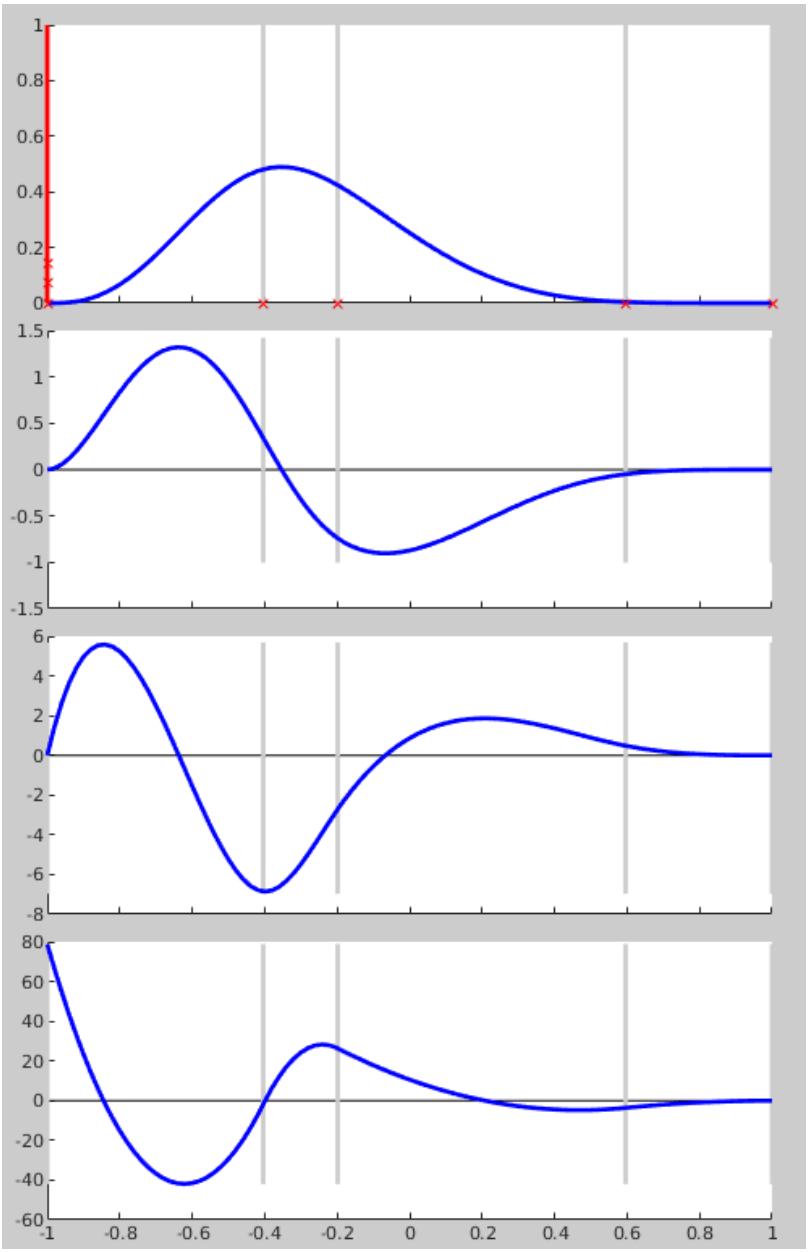

Figure 5: bispligui tool shows the non-uniform spline at the top and its 3 derivatives afterwards.

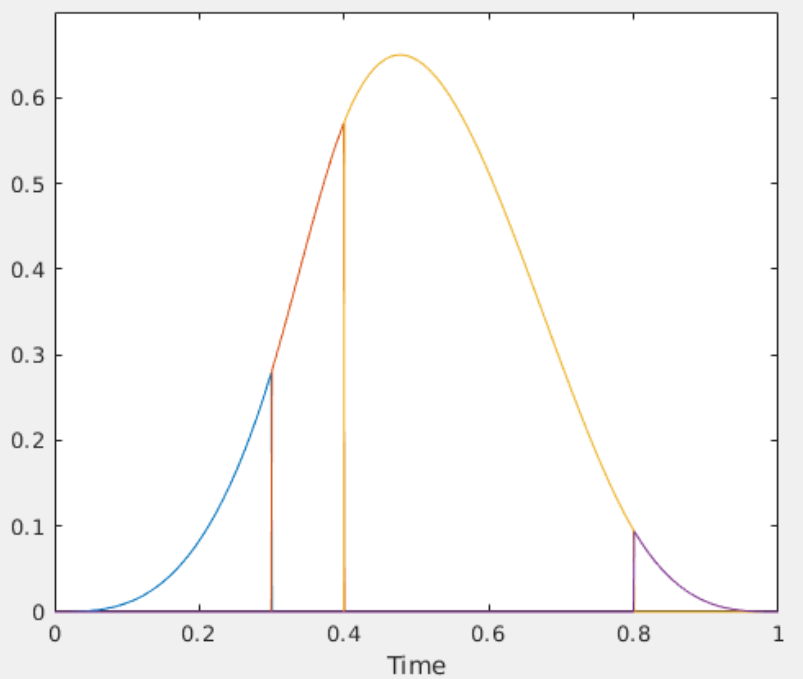

Figure 6: The non-uniform spline according to our formula

Task 5.4 (4 points) Lagrange polynomial:

$$L_1(x) = \frac{(x-5)(x-10)(x-11)}{(2-5)(2-10)(2-11)} = \frac{(x-5)(x-10)(x-11)}{-216}$$

$$L_2(x) = \frac{(x-2)(x-10)(x-11)}{(5-2)(5-10)(5-11)} = \frac{(x-2)(x-10)(x-11)}{90}$$

$$L_3(x) = \frac{(x-2)(x-5)(x-11)}{(10-2)(10-5)(10-11)} = \frac{(x-2)(x-5)(x-11)}{-40}$$

$$L_4(x) = \frac{(x-2)(x-5)(x-10)}{(11-2)(11-5)(11-10)} = \frac{(x-2)(x-5)(x-10)}{54}$$

Taking in consideration the Y coordinates of the 4 points we want to interpolate, as well as the 4 Lagrange basis functions, we can compute $P_3(x)$ as follows:

$$P_3(x) = 1 * L_1(x) - 4 * L_2(x) - 2 * L_3(x) - 3 * L_4(x) =$$
$$\frac{(x-5)(x-10)(x-11)}{-216} - \frac{2*(x-2)(x-10)(x-11)}{45} + \frac{(x-2)(x-5)(x-11)}{20} - \frac{(x-2)(x-5)(x-10)}{18}$$

or alternatively, by expanding the above result we get:

$$P_3(x) = \frac{-x^3+26x^2-215x+550}{216} - 2*\frac{x^3-23x^2+152x-220}{45} + \frac{x^3-18x^2+87x-110}{20} - \frac{x^3-17x^2+80x-100}{18}$$
$$= \frac{-59}{1080}x^3 + \frac{641}{540}x^2 - \frac{8473}{1080}x + \frac{1337}{108} \approx -0.0546x^3 + 1.187x^2 - 7.845x + 12.3796$$

In the following figure we can see how the polynomial interpolates the 4 star-labeled points. We can also inspect visually and see that all the basis polynomials behave properly: i.e. they have a value of 1 at the X location of the corresponding interpolated point and 3 values of 0 (zero) at the other 3 points (that have each their own basis function).

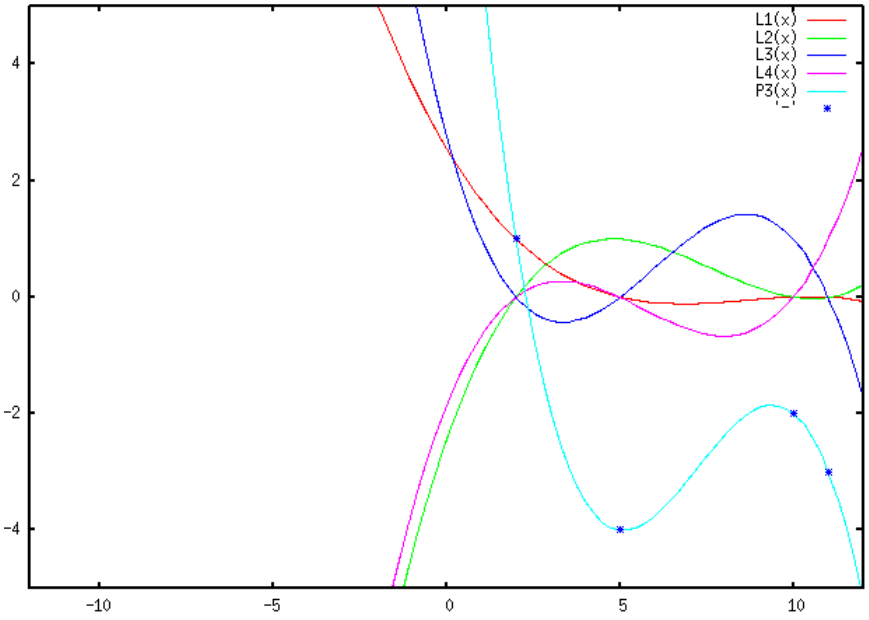

Figure 7: Interpolated points: (2,1), (5,-4), (10,-2) and (11,-3)

Task 5.5 (6 points) PID-controller:

5.5.1 (4 points):

The PID controller can be mathematically expressed as:

$$u(t) = K_p e(t) + K_i \int_0^t e(t')dt' + K_d \frac{d}{dt} e(t)$$

If we perform Laplace Transform on this we will model the response of this PID Controller as:

$$U(s) = K_p + K_i \frac{1}{s} + K_d * s$$

As we can see, integration in the time domain becomes a division by s in the frequency domain, while derivation in the time domain corresponds to multiplying by s in the frequency space.

The frequency response of our DC motor is given to be:

$$T(s) = \frac{K}{(Js + b)(Ls + R) + K^2}$$

Connecting these 2 systems together serially will result in a multiplication of the following form:

$$Sys(s) = U(s) * T(s)$$

We use the result above in a matlab program (that can be seen in the appendix below) to get the following results:

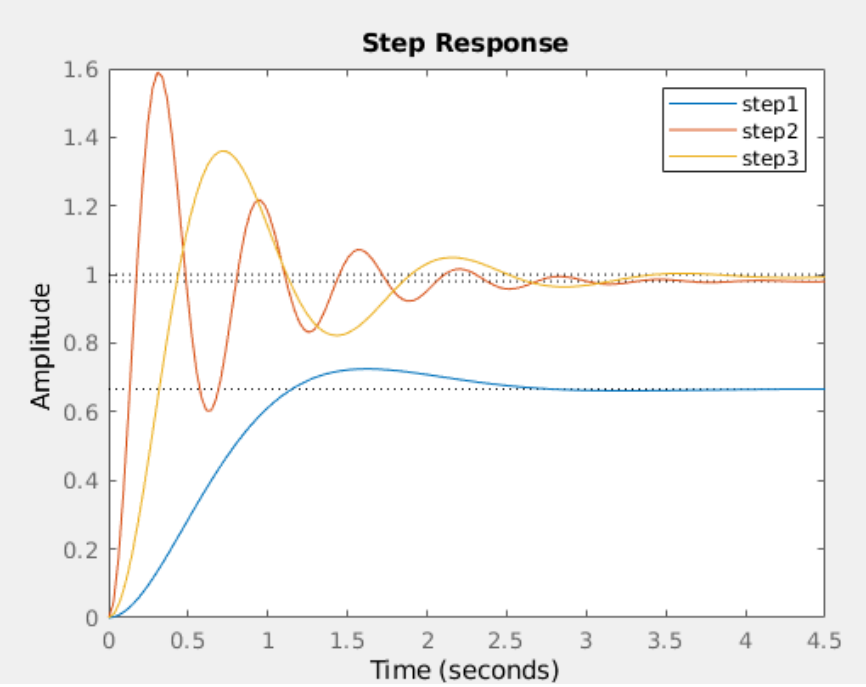

Figure 8: Step Response for the 3 cases: step1(Kp=20, Ki=0, Kd=0), step2(Kp=500, Ki=0, Kd=0) and step3(Kp=100, Ki=50, Kd=0)

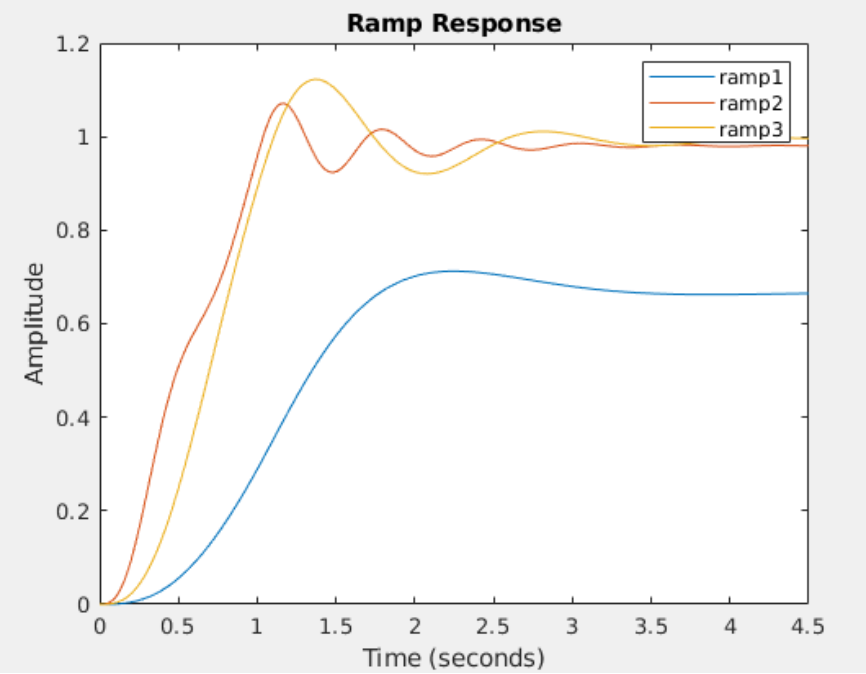

Figure 9: Ramp Response for the 3 cases: ramp1(Kp=20, Ki=0, Kd=0), ramp2(Kp=500, Ki=0, Kd=0) and ramp3(Kp=100, Ki=50, Kd=0)

5.5.2 (2 points): The values k_p, k_i and k_d weight the impact of the proportional, integral and derivative parts of the PID controller.
A P-Controller returns a change based on the error of the current signal value to the target value. By increasing k_p, the error is corrected "faster". Therefore, with a high value, overshooting/undershooting

is more likely and the oscillation around the target value is higher. If set to low, the value would never be reached and the response is asymptotic to it.

k_i is the factor for the integral part of the controller. The integral increases by the persistence of an error, integrating it over time. The longer an error persists, the higher the correction done by the integral component. The factor k_i manipulates the weight of the I term in the total controller, as the problem with the integral part is that it just slowly decreases it's impact as it approaches the target value.

By combining P and I, the controller can react fast and overshooting/undershooting is reduced.

A plain D-Controller reacts to changes of the input signal, but much faster than the P-Controller. It only uses the change of the input signal and therefore allows fast changes when the input signal varies, but reduces it's impact very fast. This allows "approaching" the target value fast and coarse. The "fine" tuning of the signal then is done with P and I.

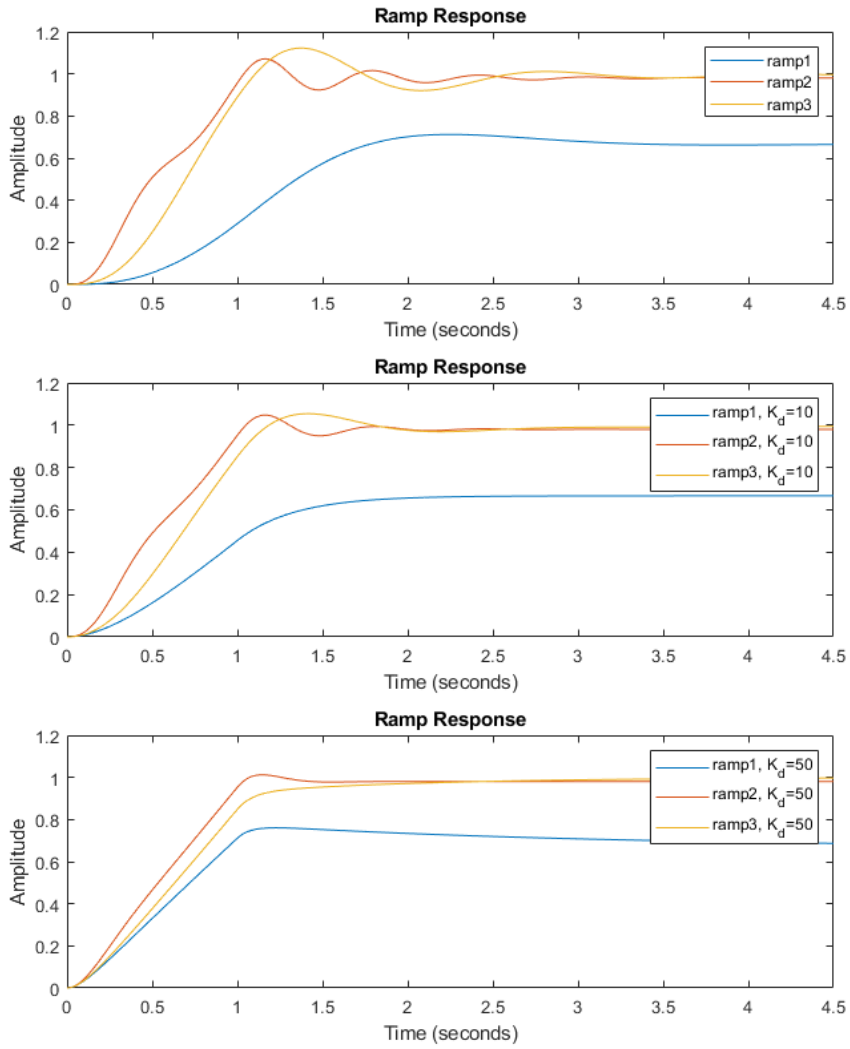

Figure 10: The ramp function with different values for k_d.

5.5.3 (4 points): *Bonus*

To solve this task we have used the "PID Tuner" provided in the "APPS" section of Matlab. As it can be seen in figure 11, we have set the "Transient Behavior" slider to "Robust", such that we have a stable response overall and low steady-state error. We have also set the "Response Time" slider to 1.1 seconds to reach the desired goal of less than 2 seconds settling time. The resulting controller parameters were:

$$Kp = 27.2727, Ki = 19.2291, Kd = 9.4022$$

These resulting parameters along with the other performance and robustness indicators can be seen in the dedicated Matlab window from the figure below.

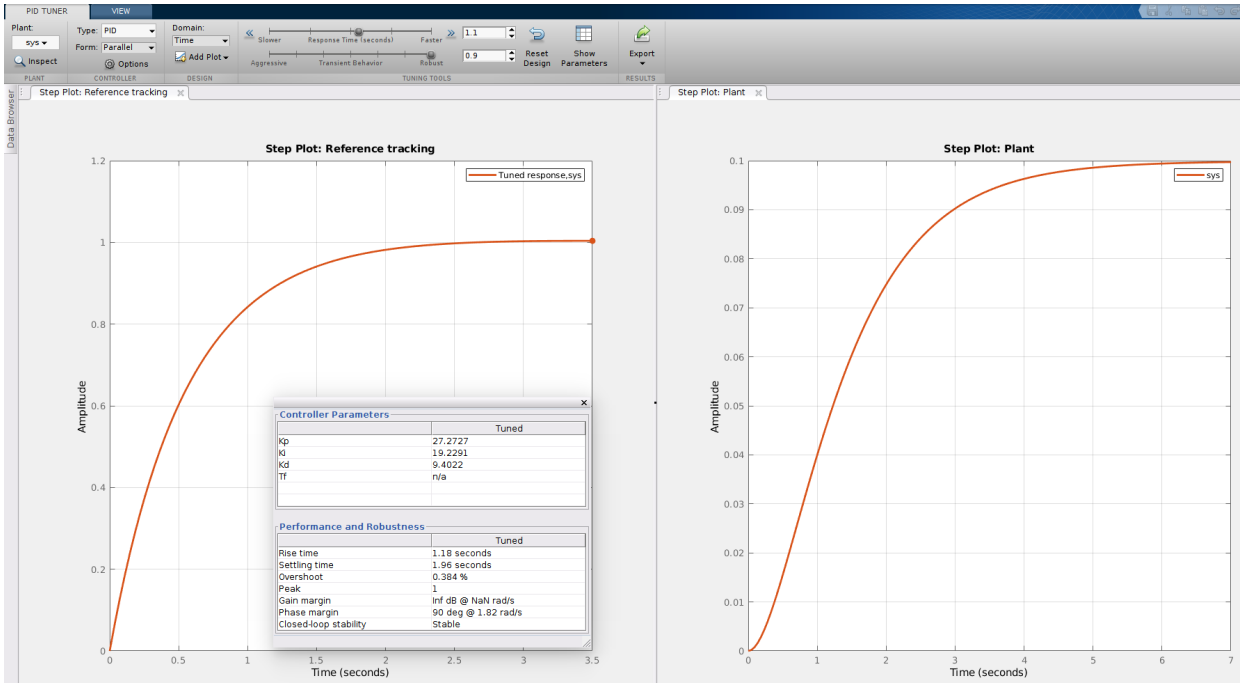

Figure 11: Settling time<2s (here 1.96s), overshoot<5% (here 0.384%) and steady state error<1%

Code Appendix

Task 5.5 PID-controller (all other tasks, including this one are sent electronically)

```
t = (0:0.01:4.5); % will simulate from time = 0 seconds to time = 4.5 seconds in small increments of 0.01 seconds

% Calculate the saturated ramp function:
unitstep = t>=0;
shiftedstep = t>=1; % will be used to saturate the part beyond t = 1
shiftedandinvertedstep = t < 1; % will be used to strip the part beyond t = 1

ramp = t.*unitstep; % this will continue growing beyond t = 1

% First strip out the amplitude beyond t = 1 and then add 1's (ones) instead to simulate the saturation:
ramp= ramp.*shiftedandinvertedstep + shiftedstep; % this will become saturated after t = 1

% Describe the motor:
J = 0.1; % Moment of inertia of the rotor: Kg*m*m
b = 0.1; % Motor viscous friction constant: N*m*s
K = 0.01; % Motor torque constant: N*m/Amp
R = 1; % Electric resistance: Ohm
L = 0.5; % electric inductance: H (Henry)

% Describe its transfer function:
s = tf('s'); % s is the transfer function frequency variable as it is typically denoted in Laplace theory
sys = K/((J*s+b)*(L*s+R)+K*K); % describe the transfer function of the system in terms of 's'

% Case 1 from the task:
% The PID control variables are:
Kp1=20;
Ki1=0;
Kd1=0;
contr1 = Kp1 + Ki1/s + Kd1*s; % PID model in the frequency domain: integrator representation is 1/S and derivative operator is times s : ' * S'
unitstep1 = feedback(contr1*sys,1); % feedback function with a gain of 1
step(unitstep1); % use the builtin functionality to get AND plot the step response

[rampresponse1,t]=lsim(unitstep1, ramp,t); % feed in the ramp function defined previously and get the response that we will plot later in a separate figure

% Case 2 from the task:
% The comments above apply to the section below as well
```

```
37 hold on;
38 Kp2=500;
39 Ki2=0;
40 Kd2=0;
41 contr2 = Kp2 + Ki2/s + Kd2*s;
42 unitstep2 = feedback(contr2*sys,1);
43 step(unitstep2);
44 [rampresponse2,t]=lsim(unitstep2, ramp,t);
45
46 % Case 3 from the task:
47 % The previous comments still apply as we only change the variable
       values
48 hold on;
49 Kp3=100;
50 Ki3=50;
51 Kd3=0;
52 contr3 = Kp3 + Ki3/s + Kd3*s;
53 unitstep3 = feedback(contr3*sys,1);
54 step(unitstep3);
55 [rampresponse3,t]=lsim(unitstep3, ramp,t);
56
57 legend('step1','step2','step3'); % legend for the first figure
58
59
60 % Plot the ramp response in a separate figure to avoid overclutter:
61 figure;
62 plot(t,[rampresponse1, rampresponse2, rampresponse3]);
63 legend('ramp1','ramp2','ramp3');
64 title('Ramp Response');
65 xlabel('Time (seconds)');
66 ylabel('Amplitude');
```