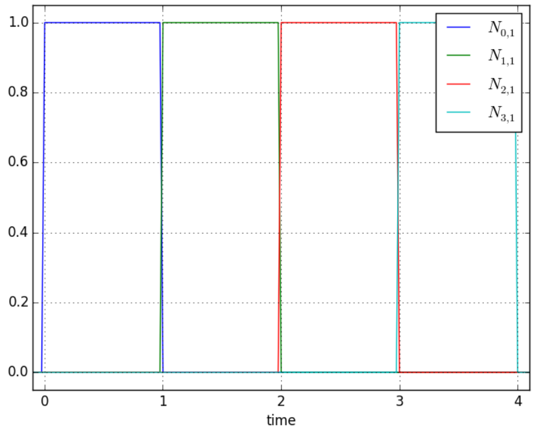
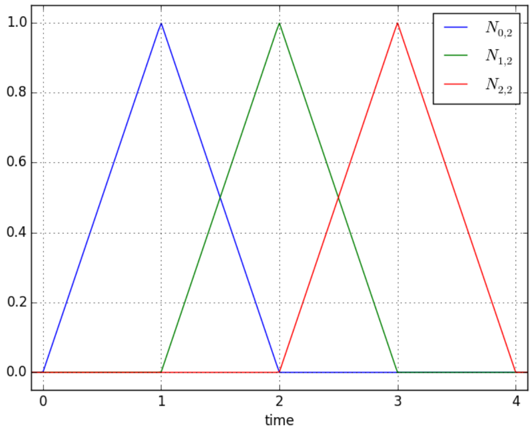
Robotics Assignment #05

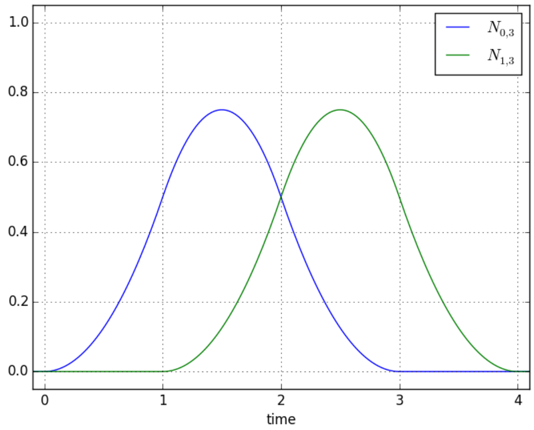
Paul Bienkowski, Konstantin Kobs

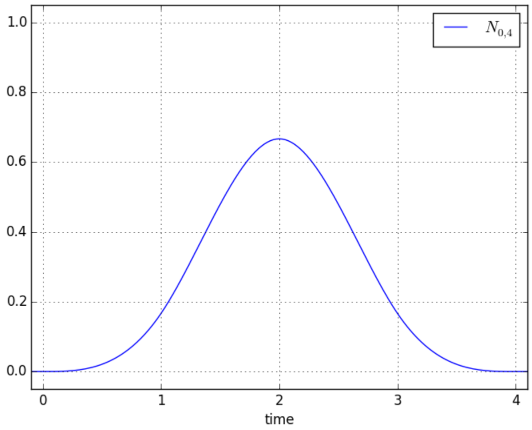
23. Juni 2015

Task 5.1. We wrote a script in Python to calculate the basis splines. These are the plots we got from our code:









We wrote the following code:

```
from matplotlib import pyplot as plt
import numpy as np

# time series
t = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

def N(i, k, time):
    """This function calculates the normalized basis spline as shown
       on slide 220"""
```

```
    if k == 1 and time >= i and time < i+1:
        return 1
    elif k == 1:
        return 0
    elif k > 1:
        return (time  t[i])/(t[i + k  1]  t[i]) * N(i, k 1, time)
          + (t[i+k]  time)/(t[i+k]  t[i+1]) * N(i+1, k 1, time)


# With this the function can be applied to numpy arrays element wise
func = np.vectorize(N)

# Create an array with 200 evenly spaced numbers between 0 and 5
x = np.linspace( 0.1, 5.0, num=200)

# Use every k with k in [1, 2, 3, 4]
for k in range(1, 5):
    for i in range(0, 5 k):
        # Calculate all the numbers
        y = func(i, k, x)
        # Plot the function
        plt.plot(x, y, label="$N_{{{i},␣{k}}}$".format(i=i, k=k))

    # Set the x axis label
    plt.xlabel("time")
    # Set ranges for axes
    plt.xlim([ 0.1, 4.1])
    plt.ylim([ 0.05, 1.05])
    # Grid
    plt.grid()
    # Legend
    plt.legend()
    # Save the plot as png
    plt.savefig('5 1 k=' + str(k) + '.png')
    # Clear the figure
    plt.clf()
```

Task 5.2. For $k = 1$, we can just use the given formula on slide 220:

$$N_{i,1}(t) = \begin{cases} 1 & \text{for } t_i \leq t < t_{i+1} \\ 0 & \text{else} \end{cases}$$

For $k = 2$, we need to insert this definition into the recursive definition

$$N_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t)$$

Since t can't be satisfying both summands, we can split this up into the desired direct definition.

Therefore, for $k = 2$:

$$N_{i,2}(t) = \begin{cases} \frac{t-t_i}{t_{i+k-1}-t_i} & \text{for } t_i \leq t < t_{i+1} \\ \frac{t_{i+k}-t}{t_{i+k}-t_{i+1}} & \text{for } t_{i+1} \leq t < t_{i+2} \\ 0 & \text{else} \end{cases}$$

For the last case, we need to input this definition again into the recursive definition and think about, what values t can have. Then we can define:

$$N_{i,3}(t) = \begin{cases} \left(\frac{t-t_i}{t_{i+k-1}-t_i}\right)^2 & \text{for } t_i \leq t < t_{i+1} \\ \frac{t-t_i}{t_{i+k-1}-t_i} \cdot \frac{t_{i+k}-t}{t_{i+k}-t_{i+1}} + \frac{t_{i+k}-t}{t_{i+k}-t_{i+1}} \cdot \frac{t-t_{i+1}}{t_{i+k}-t_{i+1}} & \text{for } t_{i+1} \leq t < t_{i+2} \\ \frac{t_{i+k}-t}{t_{i+k}-t_{i+1}} \cdot \frac{t_{i+k+1}-t}{t_{i+k+1}-t_{i+2}} & \text{for } t_{i+2} \leq t < t_{i+3} \\ 0 & \text{else} \end{cases}$$

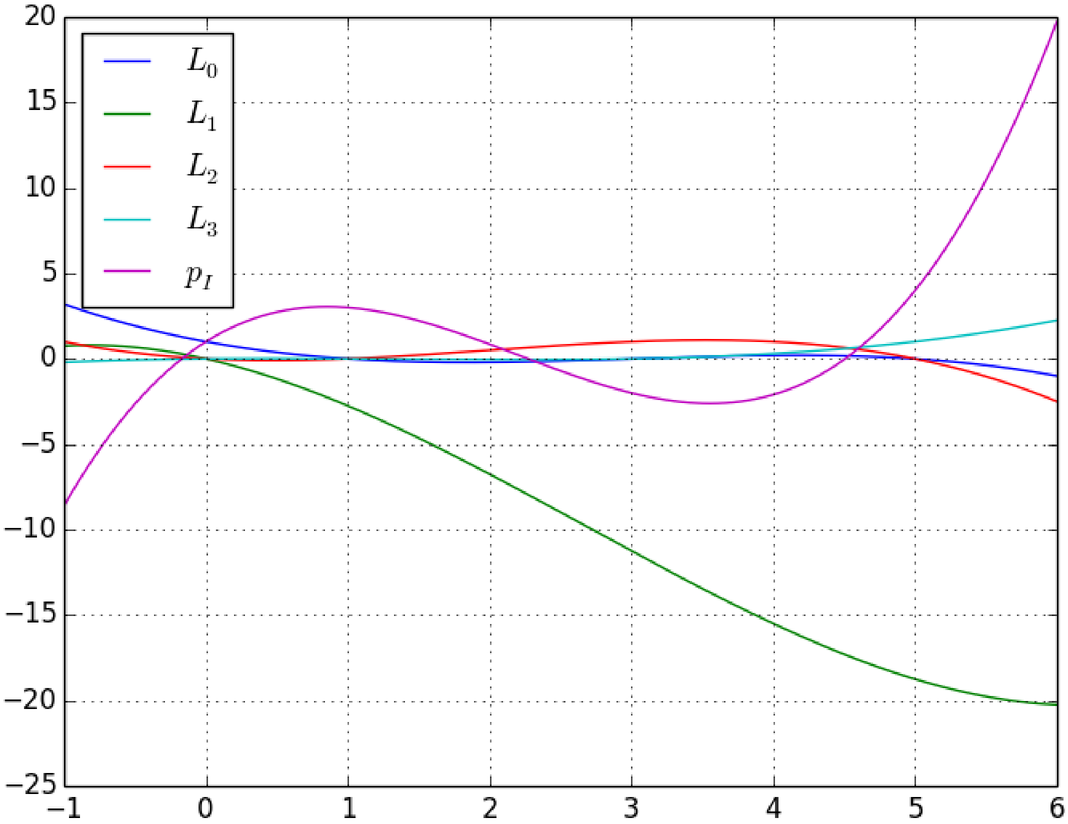
It is important to see, that whenever the second case of the $k = 2$ definition is inserted, i has to be increased by one, because it is passed so to the definition.

Task 5.3.

$$\begin{aligned}
L_0(x) &= \frac{(x-1)(x-3)(x-5)}{(0-1)(0-3)(0-5)} \\
&= \frac{(x^2-4x+3)(x-5)}{-15} \\
&= \frac{x^3-9x^2+23x-15}{-15} \\
&= -\frac{1}{15}x^3 + \frac{9}{15}x^2 - \frac{23}{15}x + 1 \\
L_1(x) &= \frac{(x-0)(x-3)(x-5)}{(1-0)(1-3)(1-5)} \\
&= \frac{x^3-8x^2+15x}{8} \\
&= \frac{1}{8}x^3 - x^2 + \frac{15}{8}x \\
L_2(x) &= \frac{(x-0)(x-1)(x-5)}{(3-0)(3-1)(3-5)} \\
&= \frac{x^3-6x^2+5x}{-12} \\
&= -\frac{1}{12}x^3 + \frac{1}{2}x^2 - \frac{5}{12}x \\
L_3(x) &= \frac{(x-0)(x-1)(x-3)}{(5-0)(5-1)(5-3)} \\
&= \frac{x^3-4x^2+3x}{40} \\
&= \frac{1}{40}x^3 - \frac{1}{10}x^2 + \frac{3}{40}x \\
p_I(x) &= y_0L_0(x) + y_1L_1(x) + y_2L_2(x) + y_3L_3(x) \\
&= 1 \cdot L_0(x) + 3 \cdot L_1(x) - 2 \cdot L_2(x) + 4 \cdot L_3(x) \\
&= \left(-\frac{1}{15}x^3 + \frac{9}{15}x^2 - \frac{23}{15}x + 1\right) + 3 \cdot \left(\frac{1}{8}x^3 - x^2 + \frac{15}{8}x\right)
\end{aligned}$$

$$
\begin{aligned}
&- 2 \cdot \left(-\frac{1}{12}x^3 + \frac{1}{2}x^2 - \frac{5}{12}x\right) + 4 \cdot \left(\frac{1}{40}x^3 - \frac{1}{10}x^2 + \frac{3}{40}x\right) \\
= &-\frac{1}{15}x^3 + \frac{9}{15}x^2 - \frac{23}{15}x + 1 + \frac{3}{8}x^3 - 3x^2 + \frac{45}{8}x \\
&+ \frac{1}{6}x^3 - x^2 + \frac{5}{6}x + \frac{1}{10}x^3 - \frac{2}{5}x^2 + \frac{3}{10}x \\
= &\left(-\frac{1}{15} + \frac{3}{8} + \frac{1}{6} + \frac{1}{10}\right)x^3 + \left(\frac{9}{15} - 3 - 1 - \frac{2}{5}\right)x^2 + \left(-\frac{23}{15} + \frac{45}{8} + \frac{5}{6} + \frac{3}{10}\right)x + 1 \\
= &\frac{69}{120}x^3 - \frac{19}{5}x^2 + \frac{627}{120}x + 1 \\
= &\frac{23}{40}x^3 - \frac{19}{5}x^2 + \frac{209}{40}x + 1
\end{aligned}
$$

We then used Python to visualize the data. This is what we've got:



This is the code we used:

```
from matplotlib import pyplot as plt
import numpy as np

# Define the functions as calculated
def L0(x):
    return  1.0/15.0 * x ** 3 + 9.0/15.0 * x ** 2   23.0/15.0 * x + 1
```

```
l0 = np.vectorize(L0)

# Define the functions as calculated
def L1(x):
    return 1.0/8.0 * x ** 3   x ** 2   15.0/8.0 * x
l1 = np.vectorize(L1)

# Define the functions as calculated
def L2(x):
    return  1.0/12.0 * x ** 3 + 1.0/2.0 * x ** 2   5.0/12.0 * x
l2 = np.vectorize(L2)

# Define the functions as calculated
def L3(x):
    return 1.0/40.0 * x ** 3   1.0/10.0 * x ** 2 + 3.0/40.0 * x
l3 = np.vectorize(L3)

# Define the functions as calculated
def pI(x):
    return 23.0/40.0 * x ** 3   19.0/5.0 * x ** 2 + 209.0/40.0 * x +
       1
pi = np.vectorize(pI)


# Create the x axis range
x = np.linspace( 1.0, 6.0, num=250)

# Calculate the data points
l0a = l0(x)
l1a = l1(x)
l2a = l2(x)
l3a = l3(x)
pia = pi(x)

# Plot the data
plt.plot(x, l0a, label="$L_0$")
plt.plot(x, l1a, label="$L_1$")
plt.plot(x, l2a, label="$L_2$")
plt.plot(x, l3a, label="$L_3$")
plt.plot(x, pia, label="$p_I$")

# Enable grid
plt.grid()
# Position the legend
plt.legend(bbox_to_anchor=(0.185, 1))
# Save the graphic
plt.savefig("5 3.png")
```

Task 5.4. (a) We can calculate the necessary parts of the equation and insert them:

$$\begin{aligned}\tau(t) &= k_p \cdot 1 + k_v \cdot 0 + k_i \cdot t \\ &= k_p + k_i \cdot t\end{aligned}$$

for $t \geq 0, 0$ else. So if we then use 1 for k_p, k_v, k_i we have

$$\tau(t) = 1 + t$$

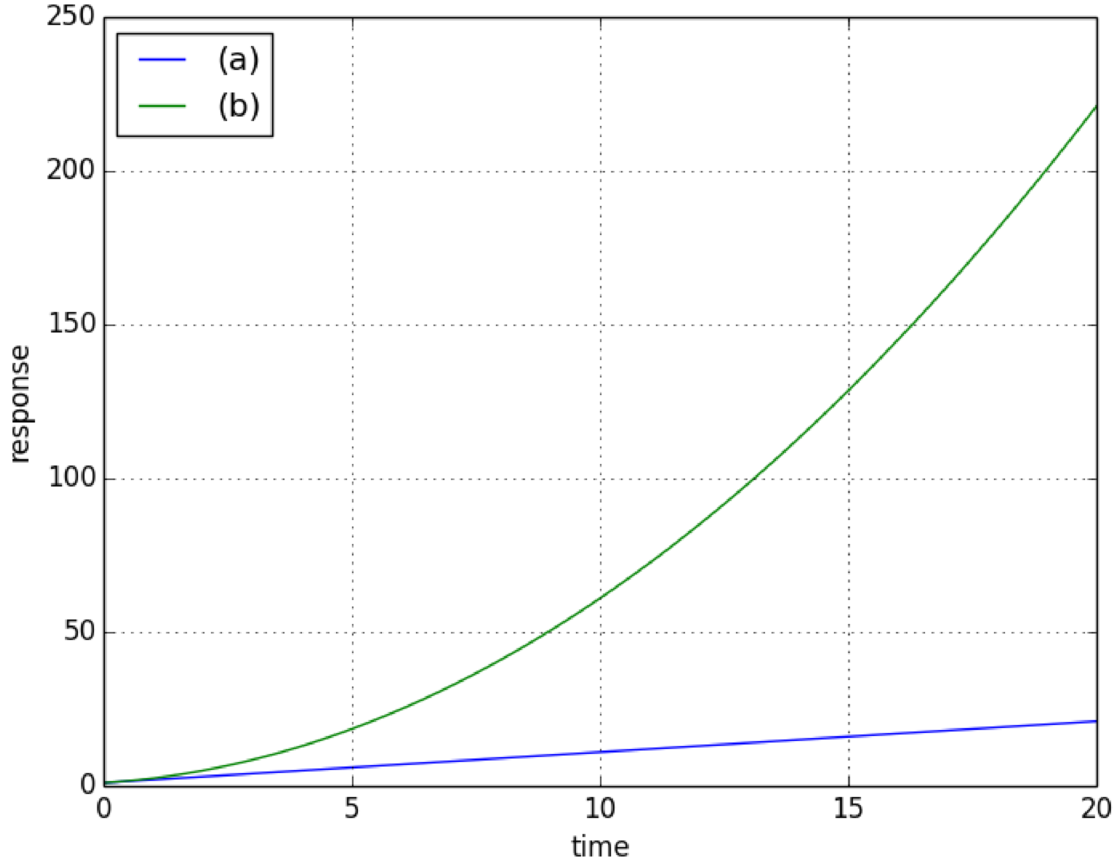
(b) As in (a), we can calculate the function as follows:

$$\begin{aligned}\tau(t) &= k_p \cdot t + k_v \cdot 1 + k_i \cdot \frac{t^2}{2} \\ &= k_p \cdot t + k_v + k_i \cdot \frac{t^2}{2}\end{aligned}$$

for $t \geq 0$, 0 else. For $k_p = k_v = k_i = 1$, we get

$$\tau(t) = t + 1 + \frac{t^2}{2}$$

We then plotted the functions with Python



with the following Code:

```python
from matplotlib import pyplot as plt
import numpy as np

# (a)
def a(t):
    return 1.0 + t
a = np.vectorize(a)

# (b)
def b(t):
    return t + 1 + t**2 * 1.0/2.0
b = np.vectorize(b)


# Create the x axis range
t = np.linspace(0.0, 20.0, num=250)

# Calculate the data points
a_array = a(t)
b_array = b(t)

# Plot the data
plt.plot(t, a_array, label="(a)")
plt.plot(t, b_array, label="(b)")

# Enable grid
plt.grid()
# Labels
plt.xlabel("time")
plt.ylabel("response")
# Position the legend
plt.legend(bbox_to_anchor=(0.185, 1))
# Save the graphic
plt.savefig("5 4.png")
```