2.Praktika: Spline bidezko eredu grafikoak

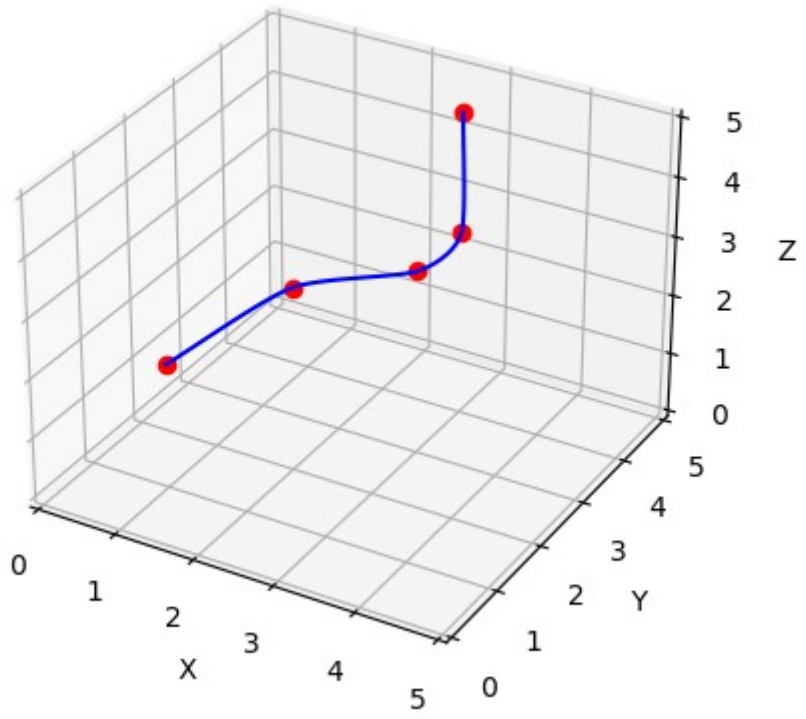
Bigarren praktika honetan Catmull-Rom interpolazio splineak eta B-Splineak erabiliko dira.

Lehenik eta behin Catmull-Rom zatiarekin hasiko gara. Jadanik eginda daukagun gamma_CR funtzioa dugu, zeinek Catmull-Rom interpolazio splinearen puntuak kalkulatzen dituen.

Hurrengo kodean ikusi dezakegun moduan, funtzioari deitzerako momentua ⅓ sartzen diogu. Zenbaki hau aldatuz, splinea itxura desberdin bat hartuko du, handiegia bada, kurba arraroak izango ditu eta aldiz, txikiagoa egiten badugu, orduan zuzenegia aterako da.

```
puntuak = np.array([[1, 1, 2], [2, 2, 3], [3, 3, 3], [3, 4, 3], [3, 4, 5]]).T
puntuak_0 = np.array([[1],[0],[0],[0],[0]]).T
puntuak_1 = np.array([[0],[1],[0],[0],[0]]).T
puntuak_2 = np.array([[0],[0],[1],[0],[0]]).T
puntuak_3 = np.array([[0],[0],[0],[1],[0]]).T
puntuak_4 = np.array([[0],[0],[0],[0],[1]]).T
 (variable) t_kurba: NDArray[floating]
t_kurba = np.linspace(0, 4, 100)

cr_kurba = np.zeros((3,100))
cr_0 = np.zeros((1,100))
cr_1 = np.zeros((1,100))
cr_2 = np.zeros((1,100))
cr_3 = np.zeros((1,100))
cr_4 = np.zeros((1,100))


for k in range(100):
    cr_kurba[:,k] = gamma_CR(puntuak, 1/3, t_kurba[k])

fig = plt.figure()
#ax = fig.gca(projection='3d')
ax = fig.add_subplot(projection='3d')

ax.plot(puntuak[0,:], puntuak[1,:], puntuak[2,:], 'ro')
ax.plot(cr_kurba[0], cr_kurba[1], cr_kurba[2], 'b-')

ax.set_xlim(0, 5)
ax.set_ylim(0, 5)
ax.set_zlim(0, 5)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

plt.show()
```

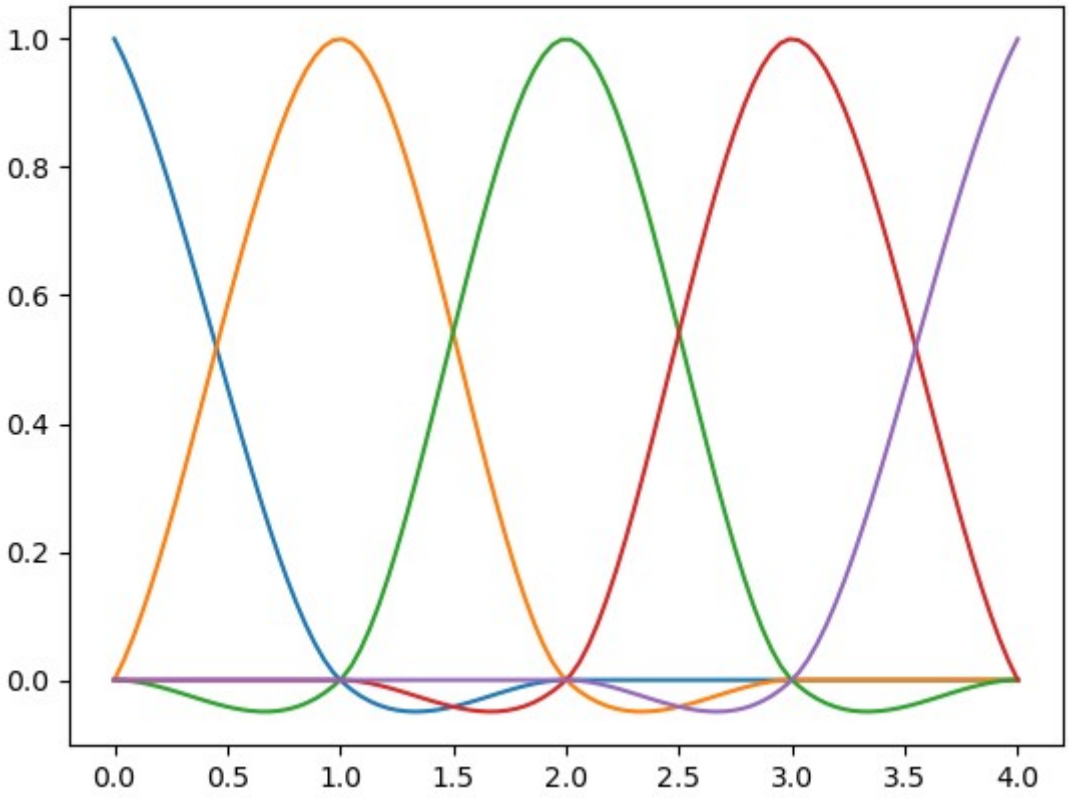
Honekin lortzen dugun splinea hurrengoa da:



Spline hau kontrol puntuen konbinazio bat dela frogatzeko, dimentsio bateko kontrol puntuak erabili ditugu, bakoitzerako dagokion $f_i(t)$ bistaratuz.

```
for k in range(100):
    cr_kurba[:,k] = gamma_CR(puntuak, 1/3, t_kurba[k])

for k in range(100):
    cr_0[:,k] = gamma_CR(puntuak_0, 1/3, t_kurba[k])

for k in range(100):
    cr_1[:,k] = gamma_CR(puntuak_1, 1/3, t_kurba[k])

for k in range(100):
    cr_2[:,k] = gamma_CR(puntuak_2, 1/3, t_kurba[k])

for k in range(100):
    cr_3[:,k] = gamma_CR(puntuak_3, 1/3, t_kurba[k])

for k in range(100):
    cr_4[:,k] = gamma_CR(puntuak_4, 1/3, t_kurba[k])


plt.plot(t_kurba, cr_0[0])
plt.plot(t_kurba, cr_1[0])
plt.plot(t_kurba, cr_2[0])
plt.plot(t_kurba, cr_3[0])
plt.plot(t_kurba, cr_4[0])

plt.show()
```

Honekin lortzen dugun irudua hurrengoa da:

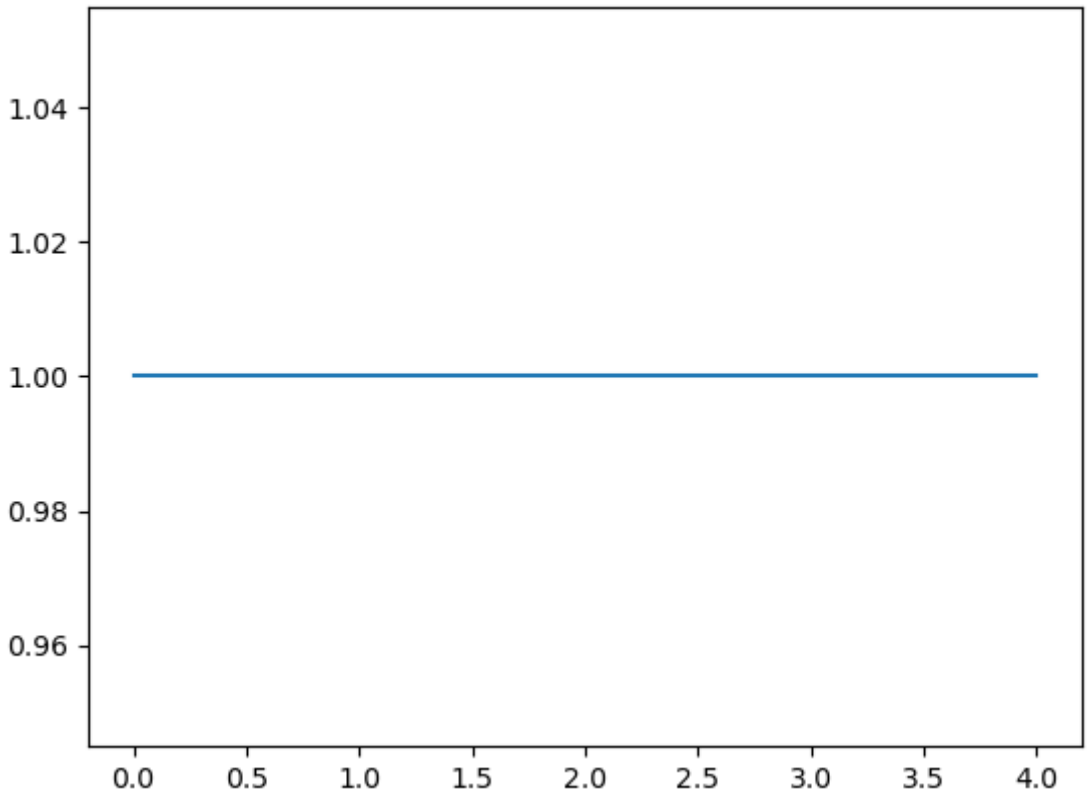


Irudian ikus daitekeen moduan, edozein puntu hartuta, funtzio guztien batuketa dela ikus daiteke. Adibidez, edozein koloreko puntu maximoa hartzen badugu, ikus daiteke nola beste funtzio guztien balioak 0 direla.

Puntuen batuketa ikusteko, hurrengo kodea dugu:

```
plt.plot(t_kurba, (cr_0 + cr_1 + cr_2 + cr_3 + cr_4)[0])
plt.show()
```

Honek hurrengo irudia sortzen du:



Ondoren, B-Spline-aren zatiarekin hasiko gara. Hasteko, splprep() funtzioaren lehuntasun-parametroa aztertu beharko dugu, hau da, s=1 edo s=0 erabiltzea. S = 0 erabiliz, sortzen den kurba hobeto ikusten da, beraz hori da erabiliko dugun parametroa, honek sortzen duen kuerba hurrengoa izanda:

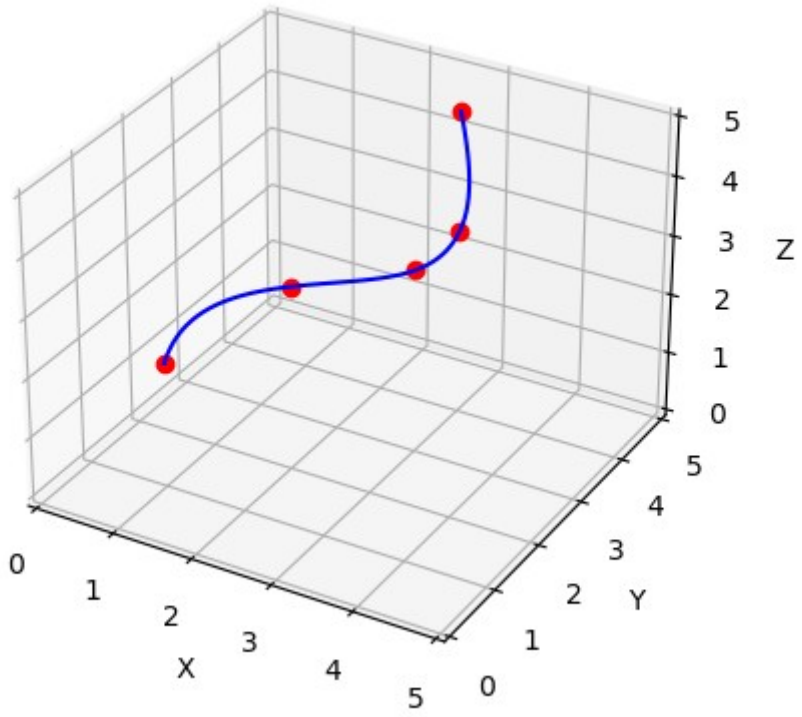


Figura 1: s=0 denean

Praktika hau guztiz burutzeko, transbordadore baten eredu grafikoa sortu behar dugu, sei kontrol puntu egindako hiru spline erabiliz. Puntu hauek horrela definitu ditugu:

```
#Transbordadorea sortu
goikoP = np.array([[0, 0, 0],[0, 0.2, 0.4], [0, 1.5, 0.8], [0, 4, 1], [0, 5, 2], [0, 5, 0]]).T
behekoP1 = np.array([[0, 0, 0],[0.1, 0.05, 0], [0.6, 0.8, 0], [0.9, 3.2, 0], [1.6, 4.5, 0],[0, 5, 0]]).T
behekoP2 = np.array([[0, 0, 0],[-0.1, 0.05, 0], [-0.6, 0.8, 0], [-0.9, 3.2, 0], [-1.6, 4.5, 0],[0, 5, 0]]).T
```

Ondoren, spline hauek izanda, B-Splineak egin ditugu, lehen aipatutako lehuntasun-parametroa erabiliz:

```
#Splineak definitu
basis_splineaG, t_goikoP = splprep(goikoP, k=3, s=0)   # spline kubikoa, leuntasuna s=0 eta s=1 hartu
basis_splineaB1, t_behekoP1 = splprep(behekoP1, k=3, s=0)   # spline kubikoa, leuntasuna s=0 eta s=1 hartu
basis_splineaB2, t_behekoP2 = splprep(behekoP2, k=3, s=0)   # spline kubikoa, leuntasuna s=0 eta s=1 hartu
```

Azkenik, transbordadorea lortzeko kurbak definitu ditugu, eta guztia pantailaratzeko funtzioa egin dugu, bi modu desberdinetan, bata estatikoki pantailaratzen du, eta bestea animazio txiki bat sortzen du, zeinetan ikuspegia Z ardatzaren inguru dugu, buelta oso bat eman arte.

```python
#Kurbak sortu
t_kurba = np.linspace(0, 1, 100)

bs_kurbaGoikoa = splev(t_kurba, basis_splineaG)
bs_kurbaBehekoa1 = splev(t_kurba, basis_splineaB1)
bs_kurbaBehekoa2 = splev(t_kurba, basis_splineaB2)

fig = plt.figure()
#ax = fig.gca(projection='3d')
ax = fig.add_subplot(projection='3d')

#Kurbak marraztu
ax.plot(bs_kurbaGoikoa[0], bs_kurbaGoikoa[1], bs_kurbaGoikoa[2], 'b-')

ax.plot(bs_kurbaBehekoa1[0], bs_kurbaBehekoa1[1], bs_kurbaBehekoa1[2], 'b-')

ax.plot(bs_kurbaBehekoa2[0], bs_kurbaBehekoa2[1], bs_kurbaBehekoa2[2], 'b-')

# Grafikoaren tamaina definitu
ax.set_xlim(-5, 5)
ax.set_ylim(-1, 10)
ax.set_zlim(-2.5, 5)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')

#IKUSPEGI ESTATIKOA
#plt.show()

#IKUSPEGI DINAMIKOA
for angle in range(0, 360):
    ax.view_init(30, angle)
    plt.draw()
    plt.pause(.01)
```

Transbordadoreak hurrengo itxura du, estatikoki pantailaratzen dugunean: