2. Praktika: Spline bidezko eredu grafikoak

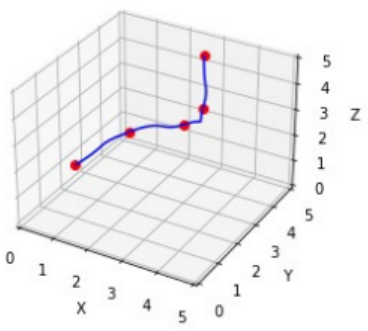
Praktika honetan Catmull-Rom interpolazio splineak eta B-Splineak erabiliko dira. Praktikak bi zati ditu eta zati bakoitzeko modulu bat erabiliko da. R_interp_splinea.py CR spline bat eraikitzeko beharrezko kodea du eta basis_spline.py SciPy liburutegiko funtzioak ditu.

Lehenengo Catmull-Rom-ekin hasi gara. Modulu hau, aurretik funtzio bat definituta zeukan: gamma_CR, funtzio honek CR interpolazio splinearen puntuak kalkulatzen ditu. Hainbat puntu sortzen duten splin baten emaitzak konparatuko ditugu hainbat deribatu-bektoreekin (r=1/10, r=1/3 eta r=1):

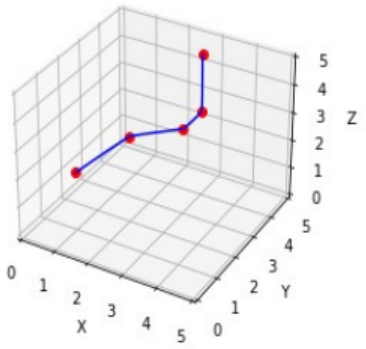
```
puntuak = np.array([[1, 1, 2], [2, 2, 3], [3, 3, 3], [3, 4, 3], [3, 4, 5]]).T
cr_kurba = np.zeros((3,100))
for k in range(100):
       cr_kurba[:,k] = gamma_CR(puntuak, 1/3, t_kurba[k])

fig = plt.figure()
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()
```

r-ren balorea aldatzen irudi hauek ateratzen dira:

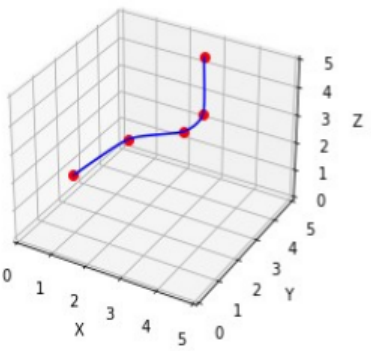
r= 1



r= 1/10



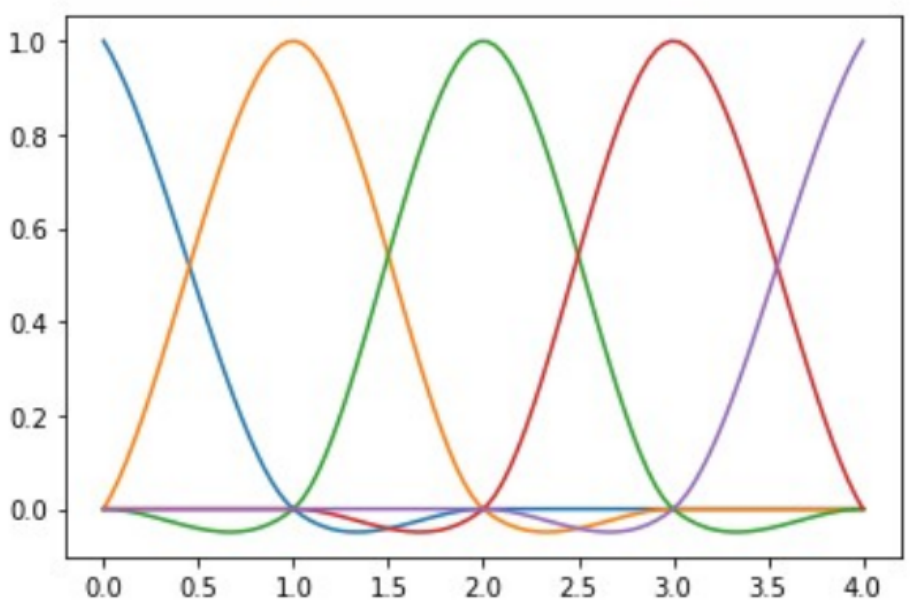
r= 1/3



Ikusten denez, r=1 denean splineak forma arraro batzuk hartzen ditu, r=1/10 denean zuzenegia da,
r=1/3 denean, aurreko bien nahasketa bat da eta hoberena, beraz, hemendik aurrera honekin lan
egingo dugu.
Splinea hau kontrol puntuen konbinazioa dela frogatuko dugu, horretarako dimentsio bateko kontrol
puntuak erabili ditugu eta bakoitzerako dagokion f i (t) bistaratzeko:

```
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])
```

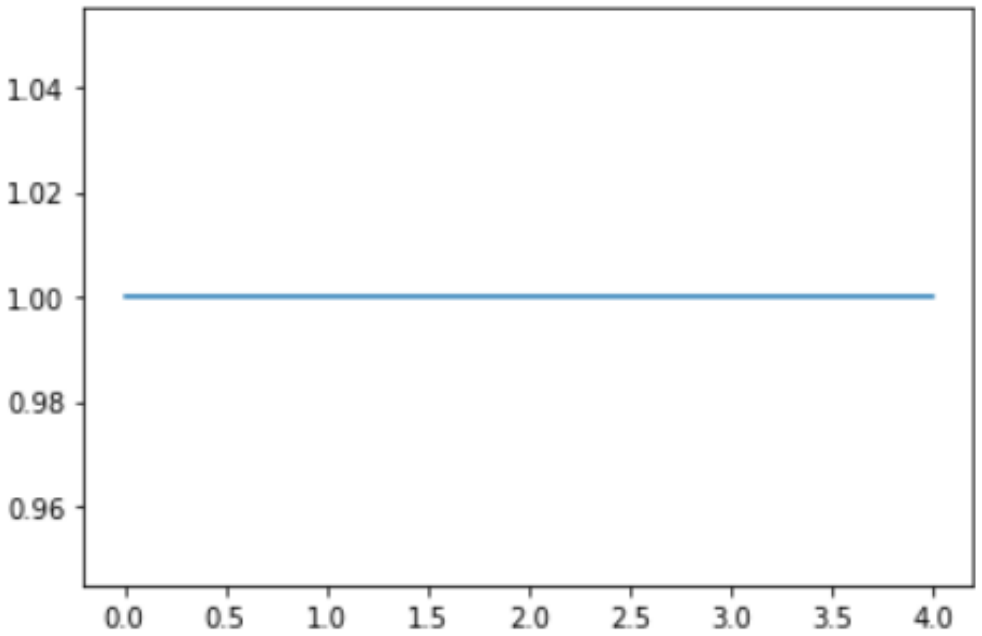
Hau exekutatzen dugunean, hau da bueltatzen duen emaitza:



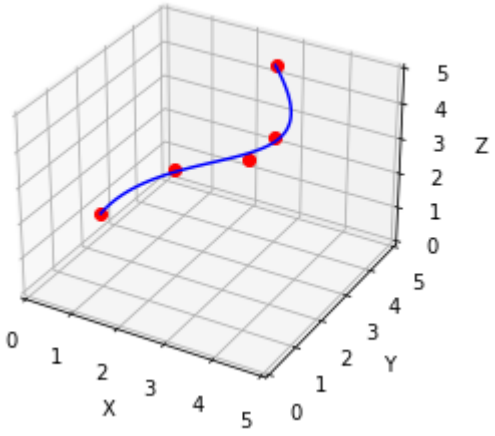
Irudian ikusi ahal den bezala, edozein puntu hartzen badugu, funtzio guztien batuketa da. Ikusten da kolore bakoitzaren goiko puntua hartzen dugunean, beste kolore guztiak 0 dira. Beste aldetik, osagai negatiboak sortzen direnez, formulako konbinazioa konbexua ez dela esan dezakegu. Puntuen batuketa hobeto ikusteko hurrengo kodea egin dugu:

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

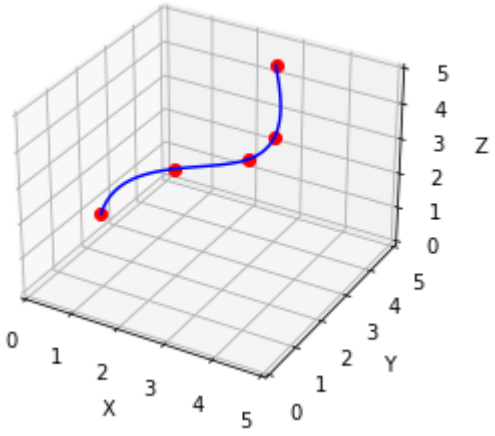
Eta kode honek irudi hau bueltatzen du:



Ondoren B-Spline kubikoaren sortzerakoan lehuntasunak duen garrantzia aztertu dugu. Horretarako *splprep()* funtzioaren lehuntasun-parametroaren *s=0* eta *s=1* balioekin egin ditugu frogak eta hemengo bi kurba hauek lortu ditugu:



B-Spline kubikoa s=1 denean



B-Spline kubikoa s=0 denean

Lehen esan bezala, kontrol puntuak guk definitun puntuak dira, *(1, 1, 2), (2, 2, 3), (3, 3, 3), (3, 4, 3)* eta *(3, 4, 5)* hain zuzen ere. Ikusten denez, *s=1* lehuntasun balioarekin kurba ez da erdiko kontrol puntuengaitik asko deformatzen, lehuntasuna dela eta, haien artean gelditzen da nahiko gutxi desbideratuz. Baina *s=0* denean kurba kontrol puntetatik oso gertu pasatzen da, hauek honen gainean duten “erakarpen indarra” handiagoa baita.

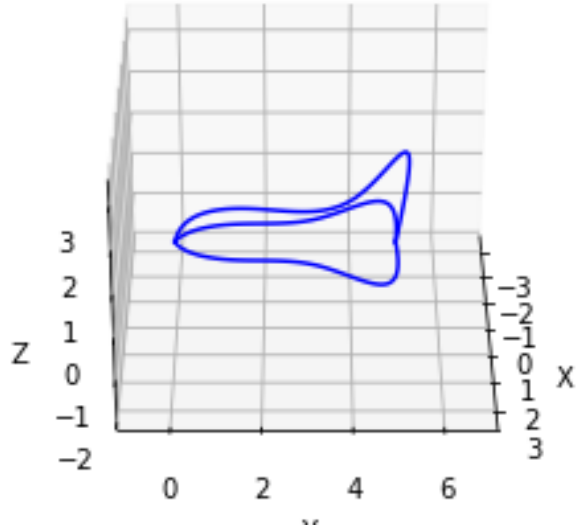
Praktika honekin bukatzeko, espazio-anezka edo transbordadore baten eredu grafikoa sortu dugu sei kontrol-puntuz osatutako hiru spline erabiliz. Puntu hauek honela definitu ditugu *python*-en:

```
puntuakGoikoak = 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
puntuakBehekoak1 = 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
puntuakBehekoak2 = 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
```

Puntu-bilduma bakoitzarekin B-Spline bat sortu dugu, *splev()* funtzioa erabiliz. Aurreko ataletik atera ditugun lehuntasunaren ondoriak kontuan hartuta, kurbak definitzerakoan aplikatu dugun lehuntasuna *s=0* izan da, guri interesatzen zaiguna transbordadorearen profila guk definitutako puntuek ematen dioten formatik ahalik eta gertuen egotea delako. Hau da kodea:

```
basis_splineaGoikoa, t_puntuakGoikoa = splprep(puntuakGoikoak, k=3, s=0)
basis_splineaBehekoa1, t_puntuakBehekoa1 = splprep(puntuakBehekoak1, k=3, s=0)
basis_splineaBehekoa2, t_puntuakBehekoa2 = splprep(puntuakBehekoak2, k=3, s=0)
```

Eta hau guztia modu estatikoan pantailaratuz lortzen duguna hau da:



Bukatzeko, transbordadore honekin animazio txiki bat egin dugu, zeinetan Z ardatzaren inguruan birarazten dugun gure ikuspegia, transbordadorea bueltaka dabilenaren sentzazioa emanez. Animazioa bukatzen da ikuspegiak 360º-ko bira bat ematen duenean kurben inguruan.

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