

Signali i sistemi

Domaci 1

Branislav Đumić
Br. indeksa: 0260/2020

Parametri

$N = 0, P = 0, Q = 2, R = 0$

Sadržaj

1	Zadatak 1.	2
1.1	Prvi deo	2
1.2	Drugi deo	5
1.3	Treći deo	9
2	Zadatak 2.	13
2.1	Prvi deo	13
2.2	Drugi deo	19
3	Zadatak 3.	23
3.1	Prvi deo	23
3.2	Drugi deo	25
4	Zadatak 4.	27
4.1	Prvi deo	27
4.2	Drugi deo	30
4.3	Treći deo	34
4.4	Četvrti deo	36
4.5	Peti deo	38

1 Zadatak 1.

Osnovne osobine i vremenske transformacije signala

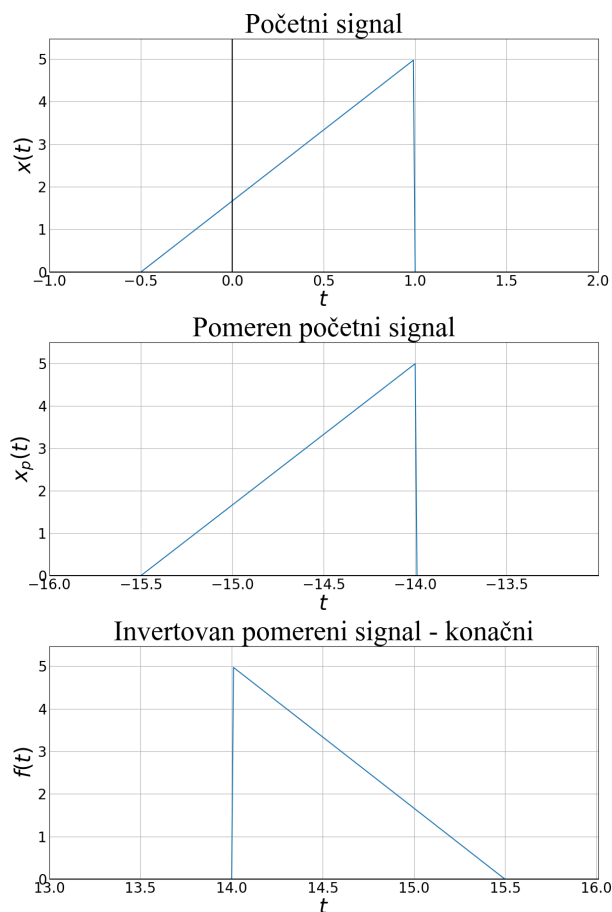
1.1 Odrediti analitički izraz za signal $f(t)$ i nacrtati grafike signala $x(t)$ i $f(t)$.

$$f(t) = x(3(Q+3) - (R+1)t) \quad (1)$$

Zamenom parametara zadatka u izraz (1) dobija se relacija koja povezuje signale $f(t)$ i $x(t)$.

$$f(t) = x(15 - t) \quad (2)$$

Na ovoj slici može se videti postupak transformacije signala $x(t)$ u $f(t)$



Slika 1: Transformacije $x(t)$ u $f(t)$

Za skiciranje grafika koristi se sledeći kod u programskom jezuku Python:

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 def u(t):
5     return 0 if t < 0 else 1
6
7 def array_u(t):
8     return np.where(t < 0, 0, 1)
9
10 def x(t):
11     return 5/3*(2*t+1)*(array_u(t+0.5)-array_u(t-1))
12
13 array_x = [
14     x,
15     lambda t: x(t+15),
16     lambda t: x(15-t)
17 ]
18
19 dt = 0.01
20
21 data = [
22     np.arange(-1, 2 + dt, dt),
23     np.arange(-16, -13 + dt, dt),
24     np.arange(13, 16 + dt, dt)
25 ]
26
27 values = list(map(
28     lambda param: array_x[param[0]](param[1]),
29     enumerate(data)))
30
31 y_label_list = ["$x(t)$", "$x_p(t)$", "$f(t)$"]
32 plot_names = [
33     "Pocetni signal",
34     "Pomeren pocetni signal",
35     "Invertovan pomereni signal - konacni"]
36
37 plt.style.use('_mpl-gallery')
38
39 resolution = 3
40 plot_count = 3
41 csfont = {'fontname' : 'Times New Roman', 'fontsize' : 40}
42 fig, ax = plt.subplots(plot_count,
43     figsize=(resolution*4, 2*resolution*plot_count),
44     constrained_layout=True)
45 for i in range(plot_count):
46     lx = min(data[i])
47     hx = max(data[i])
48     ly = min(values[i])
```

```

49 hy = max(values[i])
50 ax[i].set_title(plot_names[i], fontdict=csfont)
51 ax[i].xaxis.set_tick_params(labelsize=20)
52 ax[i].yaxis.set_tick_params(labelsize=20)
53 ax[i].set(xlim=(lx, hx),
54           xticks=np.arange(lx, hx, 0.5),
55           ylim=(ly, hy+0.5),
56           yticks=np.arange(ly, hy+0.5))
57 ax[i].plot(data[i], values[i], linewidth=1.5)
58 ax[i].axhline(0, color="black")
59 ax[i].axvline(0, color="black")
60 ax[i].set_xlabel("$t$", fontdict=csfont, fontsize=30)
61 ax[i].set_ylabel(
62     y_label_list[i], fontdict=csfont, fontsize=30)
63
64 plt.show()

```

Kod 1: Kod za generisanje grafika sa Slike 1

Orderdivanje analitičkog oblika

Sa slike se vidi da je signal $f(t)$ oblika:

$$f(t) = (at + b)(u(t - 14) - u(t - 15.5)) \quad (3)$$

Zamenom poznatih parova tačaka $(t, f(t))$, $A = (14, 5)$ i $B = (15.5, 0)$ u (3) dobijamo :

$$f(14) = 14a + b = 5 \quad (4)$$

$$f(15.5) = 15.5a + b = 0 \quad (5)$$

Oduzimanjem jednačina (4) i (5), dobija se $a = -\frac{10}{3}$ i $b = \frac{155}{3}$, tako da je konačan izraz za $f(t)$:

$$f(t) = \frac{5}{3}(-2t + 31)(u(t - 14) - u(t - 15.5)) \quad (6)$$

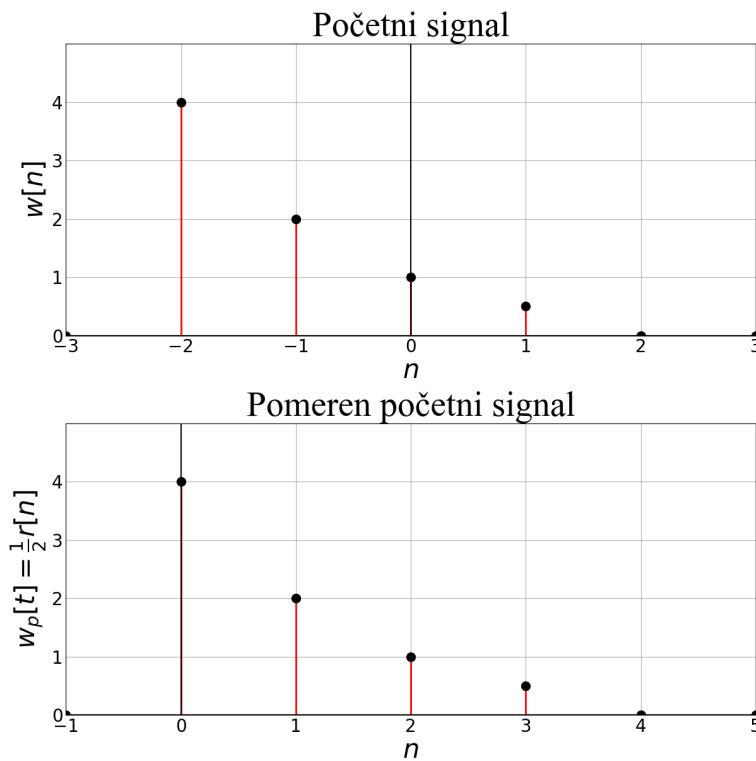
1.2 Predložiti metodu interpolacije, pa na osnovu nje prikazati grafik i napisati analitički oblik signala:

$$v[n] = 2 \cdot w\left[\frac{n}{3} - 2\right] \quad (7)$$

Na osnovu parametara zadatka dobija se da je oblik signala $w[n]$:

$$w[n] = 0.5^n \left(u[n+2] - u[n-2] \right) \quad (8)$$

Predlaže se metoda linearne interpolacije. Na ovoj slici može se videti postupak translacije signala $w[n]$ u signal $w[n-2]$. Uvode se novi signali $w_p[n] = w[n-2]$ i $r[n] = 2w_p[n]$. Konačan signal dobijamo skapiranjem i inercpolacijom signala iz $r[n]$, odnosno $v[n] = r[\frac{n}{3}]$



Slika 2: Transformacija $w[n]$ u $w[n-2]$

Za skiciranje grafika korišćen je sledeći kod u programskom jeziku Python:

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 def u(n):
5     return 0 if n < 0 else 1
6
7 def array_u(n):
8     return np.where(n < 0, 0, 1)
9
10 def w(n):
11     return 0.5**n * (array_u(n + 2) - array_u(n - 2))
12
13 array_x = [
14     w,
15     lambda n: w(n - 2),
16 ]
17
18 dt = 1
19
20 data = [
21     np.arange(-3, 3 + dt, dt),
22     np.arange(-1, 5 + dt, dt),
23 ]
24
25 values = list(map(
26     lambda param: array_x[param[0]](param[1]),
27     enumerate(data)))
28
29 y_label_list = ["$w[n]$", "$w_p[t]=\\frac{1}{2}r[n]$"]
30 plot_names = [
31     "Pocetni signal",
32     "Pomeren pocetni signal"]
33
34 plt.style.use('_mpl-gallery')
35
36 resolution = 3
37 plot_count = 2
38 csfont = {'fontname': 'Times New Roman', 'fontsize': 40}
39 fig, ax = plt.subplots(plot_count,
40                         figsize=(resolution * 4, 2 *
41                                resolution * plot_count),
42                         constrained_layout=True)
43 for i in range(plot_count):
44     lx = min(data[i])
45     hx = max(data[i])
46     ly = min(values[i])
47     hy = max(values[i])
48     ax[i].xaxis.set_tick_params(labelsize=20)
```

```

48 ax[i].yaxis.set_tick_params(labels=20)
49 ax[i].set_title(plot_names[i], fontdict=csfont)
50 ax[i].set(xlim=(lx, hx),
51           xticks=np.arange(lx, hx+1),
52           ylim=(ly, hy+1),
53           yticks=np.arange(ly, hy+1))
54 markerline, stemline, baseline, = ax[i].stem(data[i],
55 values[i])
56 ax[i].axhline(0, color="black")
57 ax[i].axvline(0, color="black")
58 ax[i].set_xlabel("$n$", fontdict=csfont, fontsize=30)
59 ax[i].set_ylabel(
60     y_label_list[i],
61     fontdict=csfont,
62     fontsize=30)
63 plt.setp(stemline, linewidth=2, color="red")
64 plt.setp(markerline, markersize=10, color="black")
65 plt.show()

```

Kod 2: Kod za generisanje grafika sa Slike 2

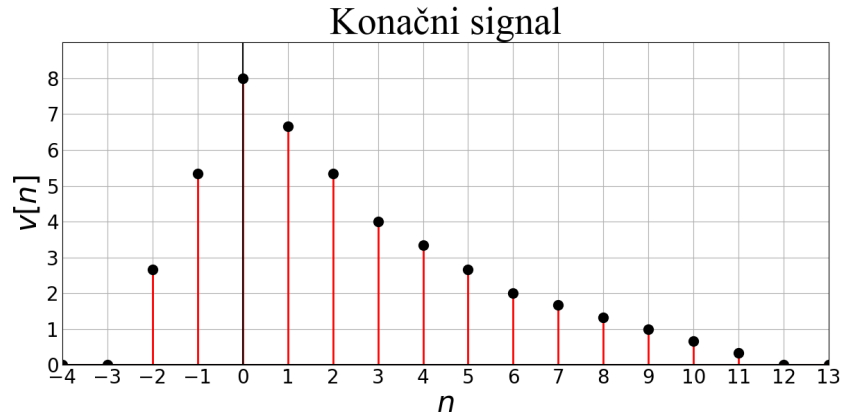
Nakon skapiranja, uz linearnu interpolaciju, rezultujući oblik signala $v[n]$ je:

$$v[n] = \begin{cases} r[k] & , \quad n = 3k \\ \frac{2 \cdot r[k] + r[k+1]}{3} & , \quad n = 3k + 1 \\ \frac{r[k] + 2 \cdot r[k+1]}{3} & , \quad n = 3k + 2 \end{cases} \quad (9)$$

Kako bi skicirali konačan grafik(signal $g[n]$), potrebno je na osnovu izraza (9) izračunati vrednosti signala $g[n]$. Vidimo da će vrednost signala biti nula za $n \leq -3$ i $n \geq 12$, dok ostale vrednosti dobijamo direktnom zamenom.

$$\begin{array}{lll}
 v[-3] = r[-1] = 0 & v[-2] = \frac{2 \cdot r[-1] + r[0]}{3} = \frac{4}{3} & v[-1] = \frac{r[-1] + 2 \cdot r[0]}{3} = \frac{8}{3} \\
 v[0] = r[0] = 8 & v[1] = \frac{2 \cdot r[0] + r[1]}{3} = \frac{20}{3} & v[2] = \frac{r[0] + 2 \cdot r[1]}{3} = \frac{16}{3} \\
 v[3] = r[1] = 4 & v[4] = \frac{2 \cdot r[1] + r[2]}{3} = \frac{10}{3} & v[5] = \frac{r[1] + 2 \cdot r[2]}{3} = \frac{8}{3} \\
 v[6] = r[2] = 2 & v[7] = \frac{2 \cdot r[2] + r[3]}{3} = \frac{5}{3} & v[8] = \frac{r[2] + 2 \cdot r[3]}{3} = \frac{4}{3} \\
 v[9] = r[3] = 1 & v[10] = \frac{2 \cdot r[3] + r[4]}{3} = \frac{2}{3} & v[11] = \frac{r[3] + 2 \cdot r[4]}{3} = \frac{1}{3}
 \end{array}$$

Kada unesemo ove vrednosti u grafik, dobijemo sledeću sliku:



Slika 3: Konačan oblik signala $v[n] = 2w[\frac{n}{3} - 2]$

Za skiciranje grafika korišćen je sledeći kod u programskom jeziku Python:

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 def u(n):
5     return 0 if n < 0 else 1
6
7 array_u = np.vectorize(u)
8
9 def w(n):
10    return 0.5 ** n * (array_u(n + 2) - array_u(n - 2))
11
12 def r(n):
13    return 2 * w(n - 2)
14
15 def v(n):
16    if n % 3 == 0:
17        return r(n // 3)
18    elif n % 3 == 1:
19        return (2 * r(n // 3) + r(n // 3 + 1)) / 3
20    else:
21        return (r(n // 3) + 2 * r(n // 3 + 1)) / 3
22
23 dt = 1
24
25 data = np.arange(-4, 13 + dt, dt)
26
27 values = np.vectorize(v)(data)
28
29 plt.style.use('_mpl-gallery')
```

```

30
31 resolution = 3
32 plot_count = 1
33 csfont = {'fontname': 'Times New Roman', 'fontsize': 40}
34 fig, ax = plt.subplots(plot_count,
35                         figsize=(resolution * 4, 2 *
36                                resolution * plot_count),
37                                constrained_layout=True)
38 lx = min(data)
39 hx = max(data)
40 ly = min(values)
41 hy = max(values)
42 ax.set_title("Konacni signal", fontdict=csfont)
43 ax.xaxis.set_tick_params(labelsize=20)
44 ax.yaxis.set_tick_params(labelsize=20)
45 ax.set(xlim=(lx, hx), xticks=np.arange(lx, hx + 1),
46        ylim=(ly, hy + 1), yticks=np.arange(ly, hy + 1))
47 markerline, stemline, baseline, = ax.stem(data, values)
48 ax.axhline(0, color="black")
49 ax.axvline(0, color="black")
50 ax.set_xlabel("$n$", fontdict=csfont, fontsize=30)
51 ax.set_ylabel("$v[n]$", fontdict=csfont, fontsize=30)
52 plt.setp(stemline, linewidth=2, color="red")
53 plt.setp(markerline, markersize=10, color="black")
54 plt.show()

```

Kod 3: Kod za generisanje grafika sa Slike 3.

1.3 Definisati paran i neparan deo signala $w[n]$. Prikazati grafike $w[n]$, $Ev\{w[n]\}$, $Od\{w[n]\}$:

Potrebno je odrediti vezu između signala $Ev\{w[n]\}$ i $Od\{w[n]\}$ sa signalom $w[n]$. Znamo da se svaki signal može predstaviti kao zbir parnog i neparnog signala, tako da imamo:

$$w[n] = Ev\{w[n]\} + Od\{w[n]\} \quad (10)$$

Ako zamenimo $n = -n$ u jednačinu (10) dobijamo:

$$w[-n] = Ev\{w[n]\} - Od\{w[n]\} \quad (11)$$

Kombinovanjem jednačina (10) i (11)

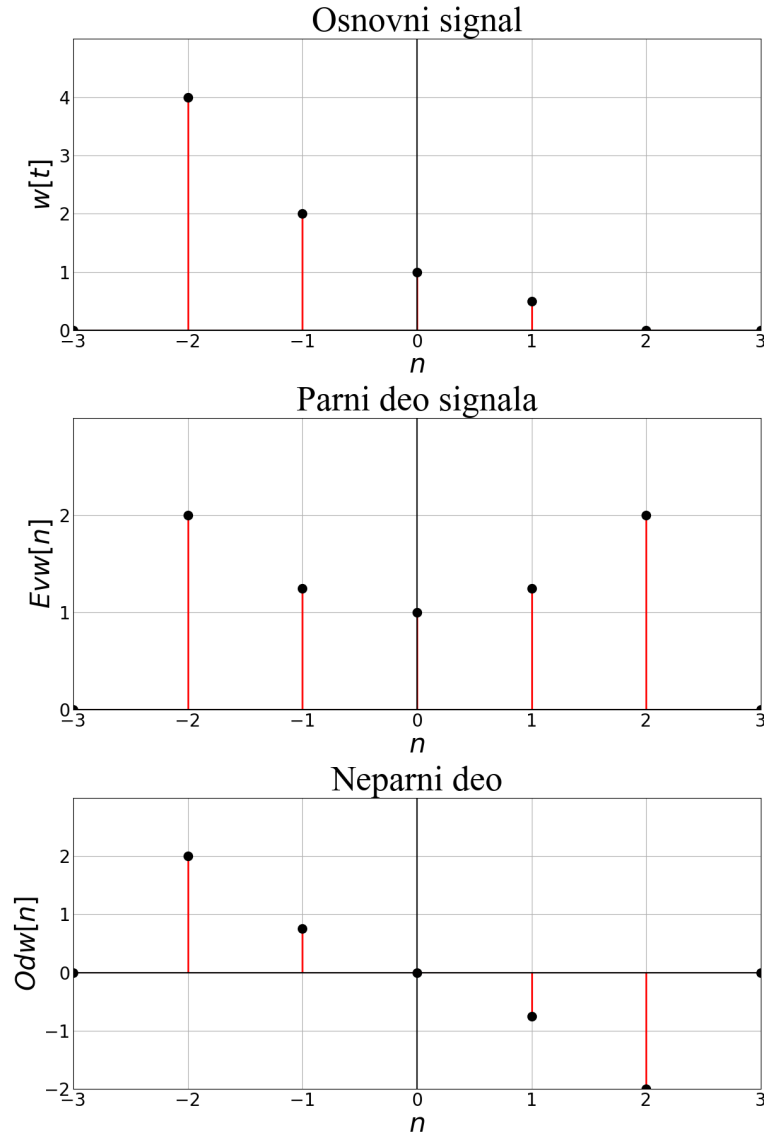
$$w[n] + w[-n] = 2 \cdot Ev\{w[n]\} \quad (10) + (11)$$

$$w[n] - w[-n] = 2 \cdot Od\{w[n]\} \quad (10) - (11)$$

Iz ovih jednačina dobijaju se konačni oblici za signale $Ev\{w[n]\}$ i $Od\{w[n]\}$:

$$Ev\{w[n]\} = \frac{w[n] + w[-n]}{2} \quad (12) \quad Od\{w[n]\} = \frac{w[n] - w[-n]}{2} \quad (13)$$

Na sledećoj slici mogu se videti grafici signala $w[n]$, $Ev\{w[n]\}$, $Od\{w[n]\}$:



Slika 4: Signali $w[n]$, $Ev\{w[n]\}$ i $Od\{w[n]\}$

Za skiciranje grafika korišćen je sledeći kod u programskom jeziku Python:

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 def u(n):
5     return 0 if n < 0 else 1
6
7 def w(n):
8     return 0.5**n*(u(n+2)-u(n-2))
9
10 def Evw(n):
11     return (w(n)+w(-n))/2
12
13 def Odw(n):
14     return (w(n)-w(-n))/2
15
16 array_x = [
17     np.vectorize(w),
18     np.vectorize(Evw),
19     np.vectorize(Odw)
20 ]
21
22 dt = 1
23
24 data = [
25     np.arange(-3, 3 + dt, dt),
26     np.arange(-3, 3 + dt, dt),
27     np.arange(-3, 3 + dt, dt)
28 ]
29
30 values = list(map(
31     lambda param: array_x[param[0]](param[1]),
32     enumerate(data)))
33
34 y_label_list = ["$w[t]$", "$Ev{w[n]}$", "$Od{w[n]}$"]
35 plot_names = [
36     "Osnovni signal",
37     "Parni deo signala",
38     "Neparni deo"]
39
40 plt.style.use('_mpl-gallery')
41
42 resolution = 3
43 plot_count = 3
44 csfont = {'fontname' : 'Times New Roman', 'fontsize' : 40}
45 fig, ax = plt.subplots(plot_count,
46     figsize=(resolution*4, 2*resolution*plot_count),
47     constrained_layout=True)
48 for i in range(plot_count):
```

```

49     lx = min(data[i])
50     hx = max(data[i])
51     ly = min(values[i])
52     hy = max(values[i])
53     ax[i].xaxis.set_tick_params(labelsize=20)
54     ax[i].yaxis.set_tick_params(labelsize=20)
55     ax[i].set_title(plot_names[i], fontdict=csfont)
56     ax[i].set(xlim=(lx, hx),
57               xticks=np.arange(lx, hx+1),
58               ylim=(ly, hy+1),
59               yticks=np.arange(ly, hy+1))
60     markerline, stemline, baseline, = ax[i].stem(data[i],
61 values[i])
62     ax[i].axhline(0, color="black")
63     ax[i].axvline(0, color="black")
64     ax[i].set_xlabel("$n$", fontdict=csfont, fontsize=30)
65     ax[i].set_ylabel(
66         y_label_list[i],
67         fontdict=csfont,
68         fontsize=30)
69     plt.setp(stemline, linewidth=2,color="red")
70     plt.setp(markerline, markersize=10,color="black")
71 plt.show()

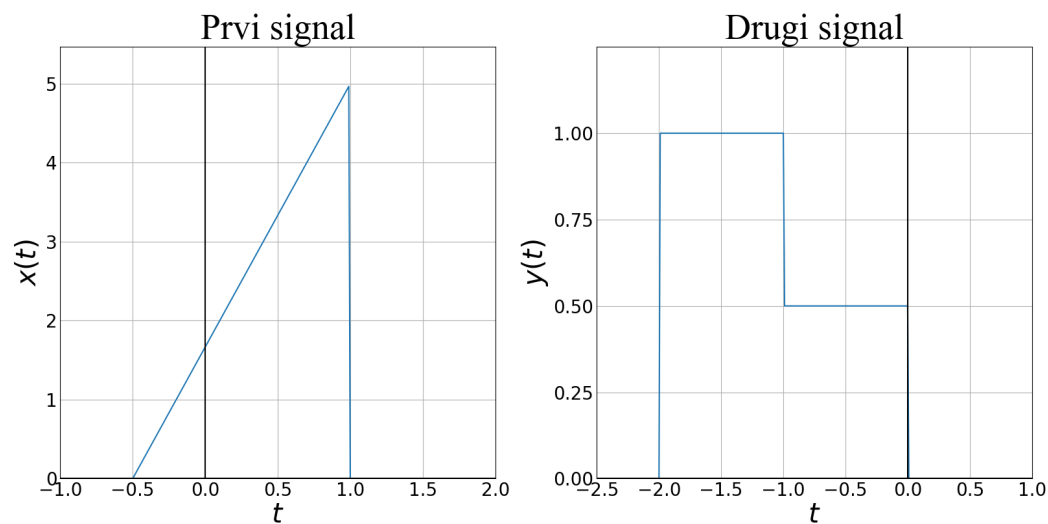
```

Kod 4: Kod za generisanje grafika sa Slike 4.

2 Zadatak 2.

Konvolucija

2.1 Analitički odrediti i skicirati konvoluciju signala $x(t)$ i $y(t)$



Slika 5: Signali $x(t)$ i $y(t)$

Za skiciranje grafika korišćen je sledeći kod u programskom jeziku Python:

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 def u(t):
5     return 0 if t < 0 else 1
6
7 def x(t):
8     return 5 / 3 * (2 * t + 1) * (u(t + 0.5) - u(t - 1))
9
10 def y(t):
11     return u(t + 2) - 0.5 * u(t + 1) - 0.5 * u(t)
12
13 array_x = [np.vectorize(x), np.vectorize(y)]
14
15 dt = 0.01
16
17 data = [
18     np.arange(-1, 2 + dt, dt),
19     np.arange(-2.5, 1 + dt, dt),
20 ]
```

```

21
22 values = list(map(
23     lambda param: array_x[param[0]](param[1]),
24     enumerate(data)))
25
26 y_label_list = ["$x(t)$", "$y(t)$"]
27 plot_names = [
28     "Prvi signal",
29     "Drugi signal"
30 ]
31
32 plt.style.use('_mpl-gallery')
33
34 resolution = 4
35 plot_count = 2
36 csfont = {'fontname': 'Times New Roman', 'fontsize': 40}
37 fig, ax = plt.subplots(1, plot_count,
38                         figsize=(2 * resolution * plot_count,
39                                 resolution * 2),
39                         constrained_layout=True)
40 for i in range(plot_count):
41     lx = min(data[i])
42     hx = max(data[i])
43     ly = min(values[i])
44     hy = max(values[i])
45     ax[i].set_title(plot_names[i], fontdict=csfont)
46     ax[i].xaxis.set_tick_params(labelsize=20)
47     ax[i].yaxis.set_tick_params(labelsize=20)
48     ax[i].set(xlim=(lx, hx),
49               xticks=np.arange(lx, hx+dt, 0.5),
50               ylim=(ly, hy + 0.5),
51               yticks=np.arange(ly, hy+0.5))
52     ax[i].plot(data[i], values[i], linewidth=1.5)
53     ax[i].axhline(0, color="black")
54     ax[i].axvline(0, color="black")
55     ax[i].set_xlabel("$t$", fontdict=csfont, fontsize=30)
56     ax[i].set_ylabel(
57         y_label_list[i],
58         fontdict=csfont,
59         fontsize=30)
60 ax[1].set(ylim=(ly, hy+0.25), yticks=np.arange(ly, hy
61         +0.25, 0.25))
62
63 plt.show()

```

Kod 5: Kod za generisanje grafika sa Slike 5.

Postupkom kao u zadatku 1.1 možemo odrediti analitički oblik signala $x(t)$, dok je oblik signala $y(t)$ jednostavno očitati sa slike. Primenom ovog postupka dobijamo:

$$x(t) = \frac{5}{3}(2t+1)(u(t+0.5) - u(t-1)) \quad (14)$$

$$y(t) = u(t+2) - 0.5u(t+1) - 0.5u(t) \quad (15)$$

Radi lakšeg nalaženja konvolucije signala $x(t)$ i $y(t)$, $z(t) = x(t) * y(t)$, razdvojimo signal $y(t)$ na signale $y_1(t) = u(t+2) - u(t+1)$ i $y_2(t) = \frac{1}{2}(u(t+1) - u(t))$, tako da je $y(t) = y_1(t) + y_2(t)$. Konvolucija je komutativna operacija, pa se konvolucija signala $x(t)$ i $y(t)$, nazovimo je $z(t)$, može definisati se kao:

$$z(t) = y(t) * x(t) = \int_{-\infty}^{+\infty} y(\tau) \cdot x(t - \tau) d\tau \quad (16)$$

Zamenom (14) i (15) u (16) dobijamo:

$$\begin{aligned} z(t) &= \int_{-\infty}^{+\infty} (y_1(\tau) + y_2(\tau)) \cdot x(t - \tau) d\tau \\ &= \int_{-\infty}^{+\infty} y_1(\tau) \cdot x(t - \tau) d\tau + \int_{-\infty}^{+\infty} y_2(\tau) \cdot x(t - \tau) d\tau \\ z(t) &= x(t) * y_1(t) + x(t) * y_2(t) \end{aligned} \quad (17)$$

Neka je $z_1(t) = x(t) * y_1(t)$ i $z_2(t) = x(t) * y_2(t)$. Zbog distributivnosti operacije konvolucije u odnosu na operaciju sabiranja znamo da je $z(t) = z_1(t) + z_2(t)$. Prvo računamo signal $z_1(t)$:

$$\begin{aligned} z_1(t) &= \int_{-\infty}^{+\infty} [u(\tau+2) - u(\tau+1)] \cdot x(t - \tau) d\tau \\ z_1(t) &= \int_{-2}^{-1} x(t - \tau) d\tau = \left\{ \begin{array}{l} t - \tau = \lambda \\ -d\tau = d\lambda \end{array} \right\} = \int_{t+1}^{t+2} x(\lambda) d\lambda \\ z_1(t) &= \frac{5}{3} \cdot \int_{t+1}^{t+2} (2\lambda + 1) (u(\lambda + 0.5) - u(\lambda + 1)) d\lambda \\ z_1(t) &= \begin{cases} 0 & , \quad t+2 < -0.5 \\ \frac{5}{3} \int_{-0.5}^{t+2} (2\lambda + 1) d\tau & , \quad t+2 \geq -0.5 \wedge t+2 < 1 \wedge t+1 < -0.5 \\ \frac{5}{3} \int_{t+1}^{t+2} (2\lambda + 1) d\tau & , \quad t+1 \geq -0.5 \wedge t+2 < 1 \\ \frac{5}{3} \int_{t+1}^1 (2\lambda + 1) d\tau & , \quad t+2 \geq 1 \wedge t+1 < 1 \\ 0 & , \quad t+1 \geq 1 \end{cases} \end{aligned}$$

Nastavljamo sa računom:

$$z_1(t) = \begin{cases} 0 & , \quad t < -2.5 \\ \frac{5}{3}[\lambda^2 + \lambda] \Big|_{-0.5}^{t+2} & , \quad t \in [-2.5, -1.5) \\ \frac{5}{3}[\lambda^2 + \lambda] \Big|_{t+1}^{t+2} & , \quad t \in [-1.5, -1) \\ \frac{5}{3}[\lambda^2 + \lambda] \Big|_{t+1}^1 & , \quad t \in [-1, 0) \\ 0 & , \quad t \geq 0 \end{cases}$$

Konačno za $z_1(t)$ dobijamo:

$$z_1(t) = \begin{cases} 0 & , \quad t < -2.5 \\ \frac{5}{3}(t^2 + 5t + 6.25) & , \quad t \in [-2.5, -1.5) \\ \frac{5}{3}(2t + 4) & , \quad t \in [-1.5, -1) \\ -\frac{5}{3}(t^2 + 3t) & , \quad t \in [-1, 0) \\ 0 & , \quad t \geq 0 \end{cases} \quad (18)$$

Vidimo da dobijeni signal nema jedinstvenu relaciju za ceo vremenski interval, već ima 5 podintervala. Ovakav postupak sada primenimo i za signal $z_2(t)$:

$$\begin{aligned} z_2(t) &= \frac{1}{2} \int_{-\infty}^{+\infty} [u(\tau + 1) - u(\tau)] \cdot x(t - \tau) d\tau \\ z_2(t) &= \frac{1}{2} \int_{-1}^0 x(t - \tau) d\tau = \left\{ \begin{array}{l} t - \tau = \lambda \\ -d\tau = d\lambda \end{array} \right\} = \int_t^{t+1} x(\lambda) d\lambda \\ z_2(t) &= \frac{1}{2} \cdot \frac{5}{3} \cdot \int_t^{t+1} (2\lambda + 1) (u(\lambda + 0.5) - u(\lambda + 1)) d\lambda \\ z_2(t) &= \begin{cases} 0 & , \quad t + 1 < -0.5 \\ \frac{5}{6} \int_{-0.5}^{t+1} (2\lambda + 1) d\tau & , \quad t + 1 \geq -0.5 \wedge t + 1 < 1 \wedge t < -0.5 \\ \frac{5}{6} \int_t^{t+1} (2\lambda + 1) d\tau & , \quad t \geq -0.5 \wedge t + 1 < 1 \\ \frac{5}{6} \int_t^1 (2\lambda + 1) d\tau & , \quad t + 1 \geq 1 \wedge t < 1 \\ 0 & , \quad t \geq 1 \end{cases} \end{aligned}$$

Nastavljamo sa računom:

$$z_2(t) = \begin{cases} 0 & , \quad t < -1.5 \\ \frac{5}{6}[\lambda^2 + \lambda]_{-0.5}^{t+1} & , \quad t \in [-1.5, -0.5) \\ \frac{5}{6}[\lambda^2 + \lambda]_t^{t+1} & , \quad t \in [-0.5, 0) \\ \frac{5}{6}[\lambda^2 + \lambda]_t^1 & , \quad t \in [0, 1) \\ 0 & , \quad t \geq 1 \end{cases}$$

Konačno za $z_1(t)$ dobijamo:

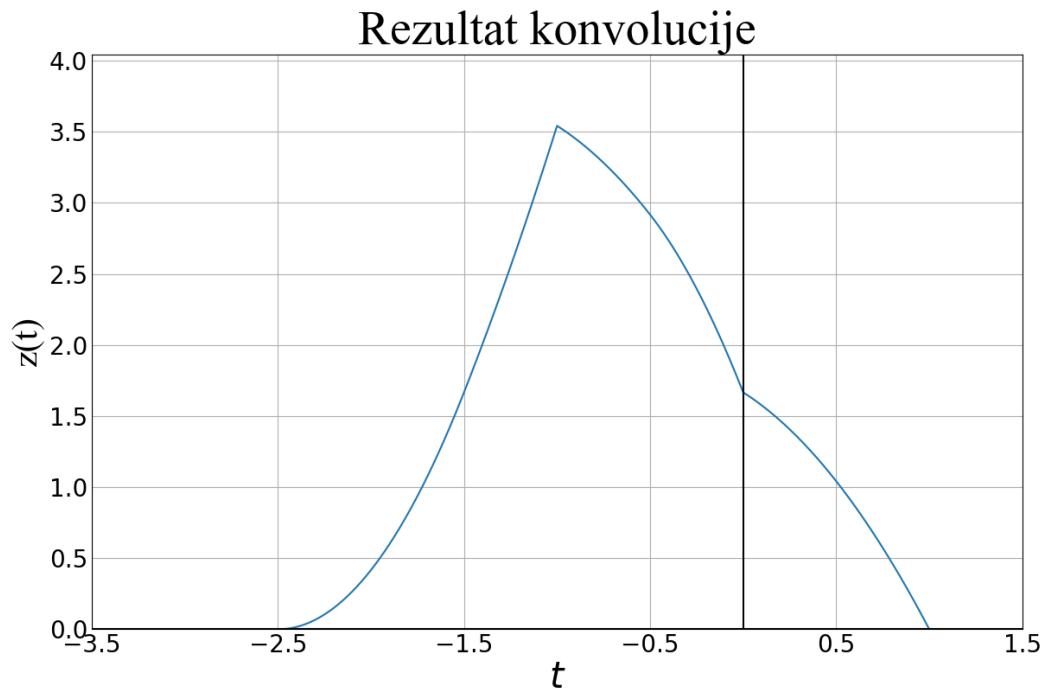
$$z_2(t) = \begin{cases} 0 & , \quad t < -1.5 \\ \frac{5}{6}(t^2 + 3t + 2.25) & , \quad t \in [-1.5, -0.5) \\ \frac{5}{3}(t + 1) & , \quad t \in [-0.5, 0) \\ -\frac{5}{6}(2 - t^2 - t) & , \quad t \in [0, 1) \\ 0 & , \quad t \geq 1 \end{cases} \quad (19)$$

Signal $z_2(t)$ takođe smo dobili 5 različitih izraza za 5 podintervala.

Kako je konvolucija signala $z(t) = x(t) * y(t) = z_1(t) + z_2(t)$ Potrebno je samo da saberemo jednačine (18) i (19), i dobićemo konačan izraz za signal $t(t)$:

$$z(t) = \begin{cases} 0 & , \quad t < -2.5 \\ \frac{5}{3}(t^2 + 5t + 6.25) & , \quad t \in [-2.5, -1.5) \\ \frac{5}{6}(t^2 + 7t + 10.25) & , \quad t \in [-1.5, -1) \\ -\frac{5}{6}(-t^2 - 3t + 2.25) & , \quad t \in [-1, 0.5) \\ -\frac{5}{3}(-t^2 - 2t + 1) & , \quad t \in [-0.5, 0) \\ -\frac{5}{6}(2 - t^2 - t) & , \quad t \in [0, 1) \\ 0 & , \quad t \geq 1 \end{cases} \quad (20)$$

Na sledećoj slici možete videti skicu signala $z(t)$:



Slika 6: Signal $z(t) = x(t) * y(t)$

Za skiciranje grafika korišćen je sledeći kod u programskom jeziku Python:

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 def z(t):
5     if t < -2.5 or t >= 1:
6         return 0
7     elif -2.5 <= t < -1.5:
8         return 5 / 3 * (t ** 2 + 5 * t + 6.25)
9     elif -1.5 <= t < -1:
10        return 5 / 6 * (t ** 2 + 7 * t + 10.25)
11    elif -1 <= t < -0.5:
12        return 5 / 6 * (-t ** 2 - 3 * t + 2.25)
13    elif -0.5 <= t < 0:
14        return 5 / 3 * (-t ** 2 - 2 * t + 1)
15    elif 0 <= t < 1:
16        return 5 / 6 * (2 - t ** 2 - t)
17    else:
18        return 0
19
20 dt = 0.005
```

```

21
22 data = np.arange(-3.5, 1.5 + dt, dt)
23 values = np.vectorize(z, otypes=[float])(data)
24
25 plt.style.use('_mpl-gallery')
26
27 resolution = 4
28 plot_count = 1
29 csfont = {'fontname': 'Times New Roman', 'fontsize': 40}
30 fig, ax = plt.subplots(plot_count,
31                        figsize=(resolution * 3, 2 * resolution
32                                * plot_count),
33                                constrained_layout=True)
34
35 lx = min(data)
36 hx = max(data)
37 ly = min(values)
38 hy = max(values)
39 ax.set_title("Rezultat konvolucije", fontdict=csfont)
40 ax.xaxis.set_tick_params(labelsize=20)
41 ax.yaxis.set_tick_params(labelsize=20)
42 ax.set(xlim=(lx, hx),
43        xticks=np.arange(lx, hx + dt),
44        ylim=(ly, hy + 0.5),
45        yticks=np.arange(ly, hy + 0.5, 0.5))
46 ax.plot(data, values, linewidth=1.5)
47 ax.axhline(0, color="black")
48 ax.axvline(0, color="black")
49 ax.set_xlabel("$t$", fontdict=csfont, fontsize=30)
50 ax.set_ylabel("$z(t)$", fontdict=csfont, fontsize=30)
51 plt.show()

```

Kod 6: Kod za generisanje grafika sa Slike 6.

2.2 Analitički odrediti i skicirati konvoluciju signala $w[n]$ i $z[n]$

$$w[n] = 0.5^n (u[n+2] - u[n-2]) \quad (21)$$

$$z[n] = 2u[n] \quad (22)$$

Neka je rezultat konvolucije signala $w[n]$ i $z[n]$ signal $\gamma[n]$. Signal $\gamma[n]$ je sada definisan sa:

$$\gamma[n] = w[n] * z[n] = \sum_{k=-\infty}^{+\infty} z[k] \cdot w[n-k] \quad (23)$$

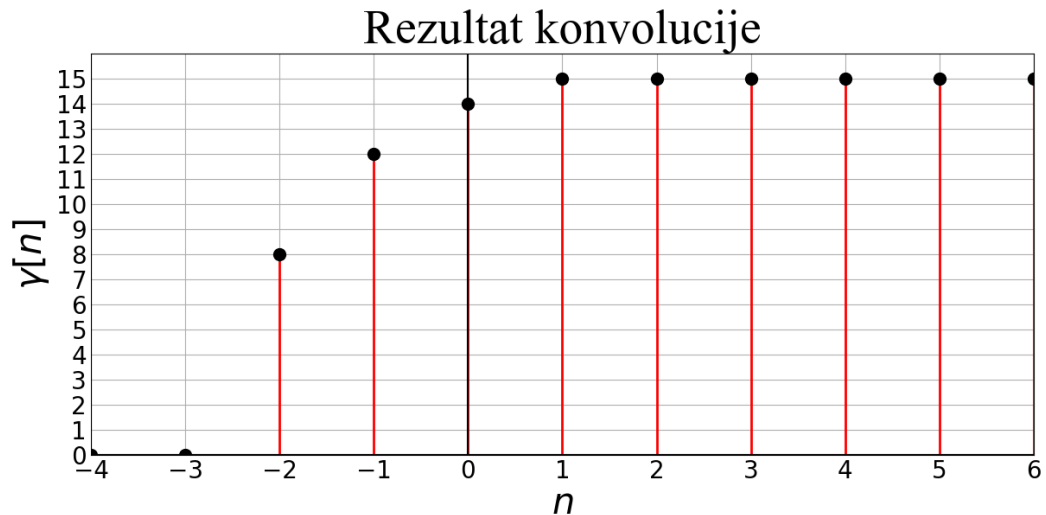
Zamenom definicija jednačina (21) i (22) u definiciju (23) možemo izračunati signal $\phi[n]$:

$$\begin{aligned}
\gamma[n] &= w[n] * z[n] = \sum_{k=-\infty}^{+\infty} z[k] \cdot w[n-k] \\
&= \sum_{k=-\infty}^{+\infty} 2u[k] \cdot w[n-k] \\
&= 2 \sum_{k=0}^{+\infty} w[n-k] = \left\{ \begin{array}{l} m = n-k \end{array} \right\} \\
&= 2 \sum_{m=n}^{-\infty} w[m] = 2 \sum_{m=-\infty}^n 0.5^m (u[m+2] - u[m-2]) \\
\\
\gamma[n] &= \begin{cases} 0 & , \quad n < -2 \\ 2 \sum_{m=-2}^n 0.5^m & , \quad n \geq -2 \wedge n < 1 \\ 2 \sum_{m=-2}^1 0.5^m & , \quad n \geq 1 \end{cases} \\
\gamma[n] &= \begin{cases} 0 & , \quad n < -2 \\ 2 \sum_{m=0}^{n+2} 0.5^{m-2} & , \quad n \geq -2 \wedge n < 1 \\ 2 \sum_{m=0}^3 0.5^{m-2} & , \quad n \geq 1 \end{cases} \\
\gamma[n] &= \begin{cases} 0 & , \quad n < -2 \\ 2 \cdot 4 \sum_{m=0}^n 0.5^m & , \quad n \geq -2 \wedge n < 1 \\ 2 \cdot 4 \sum_{m=0}^3 0.5^m & , \quad n \geq 1 \end{cases} \\
\gamma[n] &= \begin{cases} 0 & , \quad n < -2 \\ 8 \cdot \frac{1-0.5^{n+3}}{1-0.5} & , \quad n \geq -2 \wedge n < 1 \\ 8 \cdot \frac{1-0.5^4}{1-0.5} & , \quad n \geq 1 \end{cases}
\end{aligned}$$

Za konačan analitički oblik konvolucije signala dobijamo:

$$\gamma[n] = \begin{cases} 0 & , \quad n < -2 \\ 16(1 - 0.5^{n+3}) & , \quad n \geq -2 \wedge n < 1 \\ 15 & , \quad n \geq 1 \end{cases} \quad (24)$$

Na sledećoj slici prikazan je grafik rezultata konvolucije



Slika 7: Signal $\gamma[n] = w[n] * w[n]$

Za skiciranje grafika korišćen je sledeći kod u programskom jeziku Python:

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 def u(n):
5     return 0 if n < 0 else 1
6
7 array_u = np.vectorize(u)
8
9 def w(n):
10    return 0.5 ** n * (array_u(n + 2) - array_u(n - 2))
11
12 def r(n):
13    return 2 * w(n - 2)
14
15 def v(n):
16    if n < -2:
17        return 0
18    elif -2 <= n < 1:
19        return 16*(1-0.5**(n+3))
20    else:
21        return 15
22
23 dt = 1
24
25 data = np.arange(-4, 6 + dt, dt)
26

```

```

27 values = np.vectorize(v)(data)
28
29 plt.style.use('_mpl-gallery')
30
31 resolution = 3
32 plot_count = 1
33 csfont = {'fontname': 'Times New Roman', 'fontsize': 40}
34 fig, ax = plt.subplots(plot_count,
35                         figsize=(resolution * 4, 2 *
36                                resolution * plot_count),
37                                constrained_layout=True)
38 lx = min(data)
39 hx = max(data)
40 ly = min(values)
41 hy = max(values)
42 ax.set_title("Rezultat konvolucije", fontdict=csfont)
43 ax.xaxis.set_tick_params(labelsize=20)
44 ax.yaxis.set_tick_params(labelsize=20)
45 ax.set(xlim=(lx, hx), xticks=np.arange(lx, hx + 1),
46        ylim=(ly, hy + 1), yticks=np.arange(ly, hy + 1))
47 markerline, stemline, baseline, = ax.stem(data, values)
48 ax.axhline(0, color="black")
49 ax.axvline(0, color="black")
50 ax.set_xlabel("$n$", fontdict=csfont, fontsize=30)
51 ax.set_ylabel("$\gamma[n]$", fontdict=csfont, fontsize=30)
52 plt.setp(stemline, linewidth=2, color="red")
53 plt.setp(markerline, markersize=10, color="black")
54 plt.show()

```

Kod 7: Kod za generisanje grafika sa Slike 7.

3 Zadatak 3.

Osnovne osobine signala

3.1 Analitičkim postupkom odrediti da li je sistem S:

- linearan
- stacionaran
- sa memorijom
- kauzalan
- stabilan

Sistem S je definisan sa: $y(t) = x(2t) \cdot u(t)$

Linearnost

Definišemo sledeće signale:

$$y_1(t) = x_1(2t)u(t) \quad y_2(t) = x_2(2t)u(t) \quad x_3(t) = ax_1(t) + bx_2(t)$$

Dalje imamo

$$\begin{aligned} y_3(t) = x_3(t)u(t) &\Rightarrow y_3(t) = (ax_1(2t) + bx_2(2t))u(t) \\ &\Rightarrow y_3(t) = ax_1(2t)u(t) + bx_2(2t)u(t) \\ &\Rightarrow y_3(t) = ay_1(t) + by_2(t) \end{aligned}$$

Vidimo da je sistem linearan.

Stacionarnost

$$\begin{aligned} y(t - t_0) &= x(2(t - t_0)) \cdot u(t - t_0) = x(2t - 2t_0)u(t - t_0) \\ x_p(t) &= x(t - t_0) \\ y_p(t) &= x_p(2t) \cdot u(t) = x(2t - t_0) \cdot u(t) \neq y(t - t_0) \end{aligned}$$

Vidimo da sistem nije stacionaran.

Memorija

Signal $y(t)$ ne zavisi od prethodnih vrednosti drugih signala, ali zavisi od budućih vrednosti signala x_t , tako da sistem ima memoriju.

Kauzalnost

Signal $y(t)$ u trenutku t zavisi od signala x u trenutku $2t$, tako da sistem nije kauzalan.

Stabilnost

$$(\exists B_1)(\forall t)|x(t)| < B_1 \Rightarrow (\exists B_2)(\forall t)|y(t)| < B_2$$

Neka postoji B_1 , takvo da $|x(t)| < B_1$ za svako t .

$$\begin{aligned} |x(t)| \leq B_1 &\Rightarrow |y(t)| = |x(2t) \cdot u(t)| = |x(t)| \cdot |u(t)| \\ &\Rightarrow |y(t)| \leq B_1 |u(t)| \leq B_1 = B_2 \end{aligned}$$

Oдавде vidimo da je sistem BIBO stabilan.

3.2 Analitičkim postupkom utvrditi da li je sistem L:

- linearan
- stacionaran
- sa memorijom
- kauzalan
- stabilan

Sistem L je definisan sa: $y[n] = \sum_{k=n-2}^n (2x[k+1])$

Prvo sređujemo izraz za $y[n]$ kako bi lakše odredili osobine sistema

$$y[n] = \sum_{k=n-2}^n (2x[k+1]) = \left\{ \begin{matrix} m = k+1 \end{matrix} \right\}$$
$$y[n] = \sum_{k=n-1}^{n+1} 2x[k]$$

Linearnost

Definišemo sledeće signale:

$$y_1[n] = \sum_{k=n-1}^{n+1} (2x_1[k]) \quad y_2[n] = \sum_{k=n-1}^{n+1} (2x_2[k]) \quad x_3[n] = ax_1[n] + bx_2[n]$$

Dalje imamo

$$\begin{aligned} y_3[n] &= \sum_{k=n-1}^{n+1} (2x_3[k]) \Rightarrow y_3[n] = \sum_{k=n-1}^{n+1} 2(ax_1[k] + bx_2[k]) \\ &\Rightarrow y_3[n] = \sum_{k=n-1}^{n+1} (a(2x_1[k]) + b(2x_2[k])) \\ &\Rightarrow y_3[n] = a \sum_{k=n-1}^{n+1} 2x_1[k] + b \sum_{k=n-1}^{n+1} 2x_2[k] \\ &\Rightarrow y_3[n] = ay_1[n] + by_2[n] \end{aligned}$$

Vidimo da je sistem linearan.

Stacionarnost

$$y[n - n_0] = \sum_{k=n-n_0-1}^{n-n_0+1} 2x[k]$$

$$x_p[n] = x[n - n_0]$$

$$y_p[n] = \sum_{k=n-1}^{n+1} 2x_p[k] = \sum_{k=n-1}^{n+1} 2x[k - n_0] = \left\{ k - n_0 = m \right\}$$

$$y_p[n] = \sum_{k=n-n_0-1}^{n-n_0+1} 2x[k] = y[n - n_0]$$

Vidimo da je sistem stacionaran.

Memorija

$$y[n] = 2 \cdot (x[n - 1] + x[n] + x[n + 1])$$

Vrednost signala $y[n]$ u trenutku n zavisi od vrednosti ulaznog signala u trenutku $n - 1$, tako da sistem ima memoriju.

Kauzalnost

Vrednost signala $y[n]$ u trenutku n zavisi od vrednosti ulaznog signala u trenutku $n + 1$, pa sistem nije kauzalan.

Stabilnost

$$(\exists B_1)(\forall n)|x[n]| < B_1 \Rightarrow (\exists B_2)(\forall n)|y[n]| < B_2$$

Neka postoji B_1 , takvo da $|x[n]| < B_1$ za svako n .

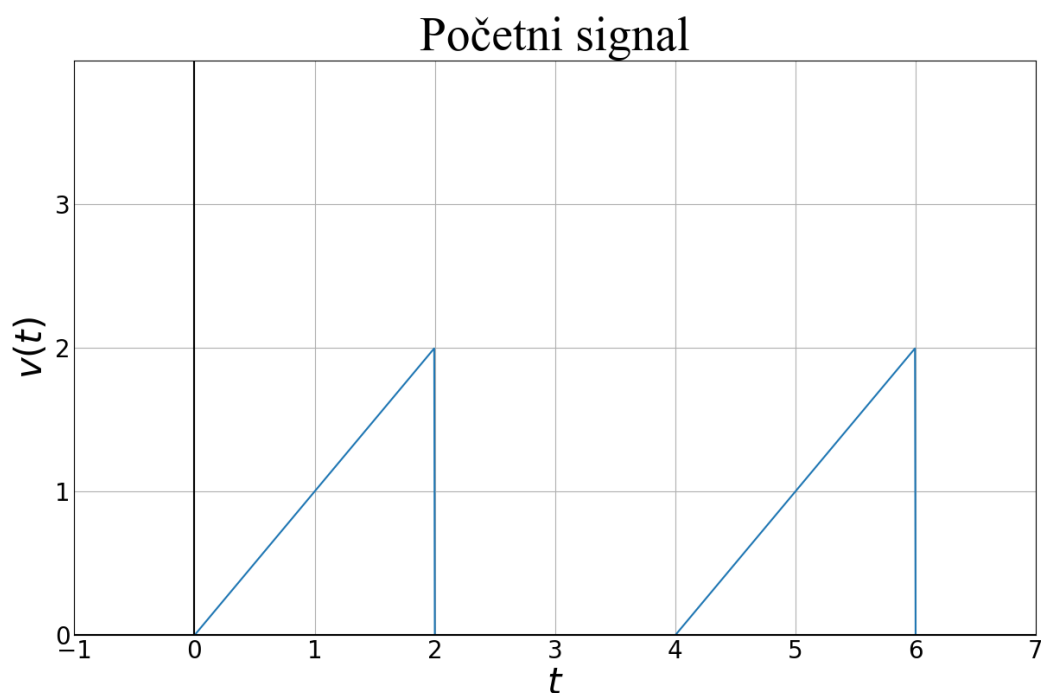
$$\begin{aligned} |x[n]| \leq B_1 &\Rightarrow |y[n]| = \left| \sum_{k=n-1}^{n+1} 2x[k] \right| \leq 2 \sum_{k=n-1}^{n+1} |x[k]| \\ &\Rightarrow |y[n]| \leq 6B_1 = B_2 \end{aligned}$$

Oдавде vidimo da je sistem BIBO stabilan.

4 Zadatak 4.

Furiheovi redovi

- 4.1** Napisati analitički oblik signala $v(t)$, odrediti njegovu osnovnu periodu T i učestanost ω_0 , i izvesti izraz za koeficijente Furijeovog reda a_k



Slika 8: Signal $v(t)$

Sa slike se vidi da je period signala $T = 4s$. Odavde sledi da je kružna učestanost $\omega_0 = \frac{2\pi}{T} = \frac{\pi}{2} rad/s$. Sa slike se može očitati i analitički oblik signala $v(t)$:

$$v(t) = \sum_{k=-\infty}^{\infty} (t - 4k) (u(t - 4k) - u(t - 4k - 2)) \quad (25)$$

Za skiciranje grafika korišćen je sledeći kod u programskom jeziku Python:

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 def u(t):
5     return 0 if t < 0 else 1
6
7 def v(L):
8     def vk(t):
9         s = 0
10        for k in range(-L, L + 1):
11            s += (t - 4 * k) * (u(t - 4 * k) - u(t - 4 * k -
12            2))
13        return s
14    return vk
15
16 dt = 0.005
17
18 data = np.arange(-1, 7 + dt, dt)
19 values = np.vectorize(v(9), otypes=[float])(data)
20
21 plt.style.use('_mpl-gallery')
22 resolution, plot_count = 4, 1
23 csfont = {'fontname': 'Times New Roman', 'fontsize': 40}
24 fig, ax = plt.subplots(plot_count,
25                        figsize=(resolution * 3, 2 *
26                        resolution * plot_count),
27                        constrained_layout=True)
28
29 lx = min(data)
30 hx = max(data)
31 ly = min(values)
32 hy = max(values)
33 ax.set_title("Pocetni signal", fontdict=csfont)
34 ax.xaxis.set_tick_params(labelsize=20)
35 ax.yaxis.set_tick_params(labelsize=20)
36 ax.set(xlim=(lx, hx),
37        xticks=np.arange(lx, hx + dt),
38        ylim=(ly, hy + 2),
39        yticks=np.arange(ly, hy + 2))
40 ax.plot(data, values, linewidth=1.5)
41 ax.axhline(0, color="black")
42 ax.axvline(0, color="black")
43 ax.set_xlabel("$t$", fontdict=csfont, fontsize=30)
44 ax.set_ylabel("$v(t)$", fontdict=csfont, fontsize=30)
45
46 plt.show()
```

Kod 8: Kod za generisanje grafika sa Slike 8

Izvođenje koeficijenata Furijeovog reda

Furijeov red funkcije $v(t)$ dat je sumom:

$$v(t) = \sum_{k=-\infty}^{+\infty} a_k \cdot e^{j\omega_0 k t} \quad (26)$$

gde su koeficijenti a_k definisani sa:

$$a_k = \frac{1}{T} \int_T v(t) e^{-j\omega_0 k t} dt \quad (27)$$

Odredimo sada koeficijente Furijeovog reda:

$$\begin{aligned} a_k &= \frac{1}{T} \int_T v(t) e^{-j\omega_0 k t} dt = \frac{1}{4} \int_0^2 t e^{-j\omega_0 k t} dt = \left\{ \begin{array}{l} u = t \\ du = dt \end{array} \quad , \quad \begin{array}{l} dv = e^{-j\omega_0 k t} \\ v = -\frac{1}{j\omega_0 k} e^{-j\omega_0 k t} \end{array} \right\} \\ a_k &= \frac{1}{4} \left(-\frac{1}{j\omega_0 k} t e^{-j\omega_0 k t} \Big|_0^2 + \frac{1}{j\omega_0 k} \int_0^2 e^{-j\omega_0 k t} dt \right) \\ a_k &= \frac{1}{4} \left(-\frac{1}{j\omega_0 k} \cdot 2e^{-jk\pi} + \frac{1}{j\omega_0 k} \cdot \left(-\frac{1}{j\omega_0 k} \right) e^{-j\omega_0 k t} \Big|_0^2 \right) \\ a_k &= \frac{1}{4} \left(-\frac{2}{j\frac{\pi}{2}k} \cdot e^{-jk\pi} + \frac{1}{(\frac{\pi}{2})^2 k^2} (e^{-jk\pi} - 1) \right) \\ a_k &= \frac{1}{4} \left(-\frac{4}{j\pi k} \cdot (e^{-j\pi})^k + \frac{4}{\pi^2 k^2} ((e^{-j\pi})^k - 1) \right) \\ a_k &= \left(\frac{j}{\pi k} \cdot (-1)^k - \frac{1}{\pi^2 k^2} (1 - (-1)^k) \right) \end{aligned} \quad (28)$$

Vidimo da koeficijent a_k nije definisan relacijom (28) za $k = 0$, tako da koeficijent a_0 moramo računati posebno, koristeći se relacijom (27) i uzimajući $k = 0$.

$$a_0 = \frac{1}{T} \int_T v(t) dt = \frac{1}{4} \int_0^4 v(t) dt = \frac{1}{4} \int_0^2 t dt = \frac{1}{4} \cdot \frac{1}{2} t^2 \Big|_0^2 = \frac{1}{8} \cdot 4 = \frac{1}{2} \quad (29)$$

Konačno, koeficijenti a_k Furijeovog reda funkcije $v(t)$ definisani su sa

$$a_k = \begin{cases} \frac{1}{2} & , \quad k = 0 \\ j\frac{1}{4\pi}(-1)^k - \frac{1}{\pi^2 k^2}(1 - (-1)^k) & , \quad k \neq 0 \end{cases} \quad (30)$$

4.2 k-ti harmonik $v_k(t)$ signala $v(t)$ može se odrediti kao:

$$v_k(t) = a_{-k}e^{-jk\omega_0 t} + a_k e^{jk\omega_0 t} = 2\operatorname{Re}\{a_k e^{jk\omega_0 t}\} = A_k \cos(k\omega_0 t + \phi_k), k \geq 1$$

Odrediti amplitudu A_k i faze ϕ_k za prva dva harmonika ($k = 1, 2$), i skicirati na istom grafiku signale $v(t)$, $v_1(t)$ i $v_2(t)$

Koeficijente A_k i ϕ_k ćemo najlakše nalaziti iz relacije:

$$v_k(t) = 2\operatorname{Re}\{a_k e^{jk\omega_0 t}\} = A_k \cos(k\omega_0 t + \phi_k) \quad (31)$$

Odredimo prvo prvi harmonik $v_1(t)$.

Prvo ćemo odrediti koeficijent a_1

$$\begin{aligned} a_1 &= j\frac{1}{\pi}(-1)^1 - \frac{1}{\pi^2}(1 - (-1)^1) \\ a_1 &= -j\frac{1}{\pi} - \frac{1}{\pi^2}(1 + 1) \\ a_1 &= -\frac{2}{\pi^2} - j\frac{1}{\pi} \end{aligned} \quad (32)$$

Za nalaženje koeficijenata A_1 i ϕ_1 najpogodnija je relacija $v_1(t) = 2\operatorname{Re}\{a_1 e^{j\omega_0 t}\}$, tako da je potrebno izraziti koeficijent a_1 u obliku $|a_1|e^{j\theta_k}$.

$$|a_1| = \sqrt{\left(\frac{2}{\pi^2}\right)^2 + \left(\frac{1}{\pi}\right)^2} = \sqrt{\frac{4}{\pi^4} + \frac{1}{\pi^2}} = \frac{1}{\pi^2} \sqrt{4 + \pi^2} \quad (33)$$

$$\theta_1 = \arg(a_1) = \arctan\left(\frac{-\frac{1}{\pi}}{-\frac{2}{\pi^2}}\right) - \pi = \arctan\frac{\pi}{2} - \pi \quad (34)$$

Ubacivanjem (33) i (34) u izraz (31) dobijamo:

$$\begin{aligned} v_1(t) &= 2\operatorname{Re}\left\{\frac{1}{\pi^2} \sqrt{4 + \pi^2} e^{j\left(\arctan\frac{\pi}{2} - \pi\right)} e^{j\frac{\pi}{2}t}\right\} \\ A_1 \cos\left(k\frac{\pi}{2} + \phi_k\right) &= \frac{2}{\pi^2} \sqrt{4 + \pi^2} \cos\left(\frac{\pi}{2}t + \arctan\frac{\pi}{2} - \pi\right) \end{aligned} \quad (35)$$

Izjednačavanjem koeficijenata konačno imamo:

$$A_1 = \frac{2}{\pi^2} \sqrt{4 + \pi^2} \quad (36)$$

$$\phi_1 = \arctan\frac{\pi}{2} - \pi \quad (37)$$

Istim postupkom sada određujemo drugi harmonik $v_2(t)$. Određujemo koeficijent a_2 :

$$a_2 = j \frac{1}{2\pi} (-1)^2 - \frac{1}{4\pi^2} (1 - (-1)^2) = j \frac{1}{2\pi} \quad (38)$$

Odredimo sada $|a_2|$ i θ_2 :

$$|a_2| = \left| j \frac{1}{2\pi} \right| = \frac{1}{2\pi} \quad (39) \quad \theta_2 = \frac{\pi}{2} \quad (40)$$

Izjednačavanjem koeficijenata, kao u (35), dobijamo:

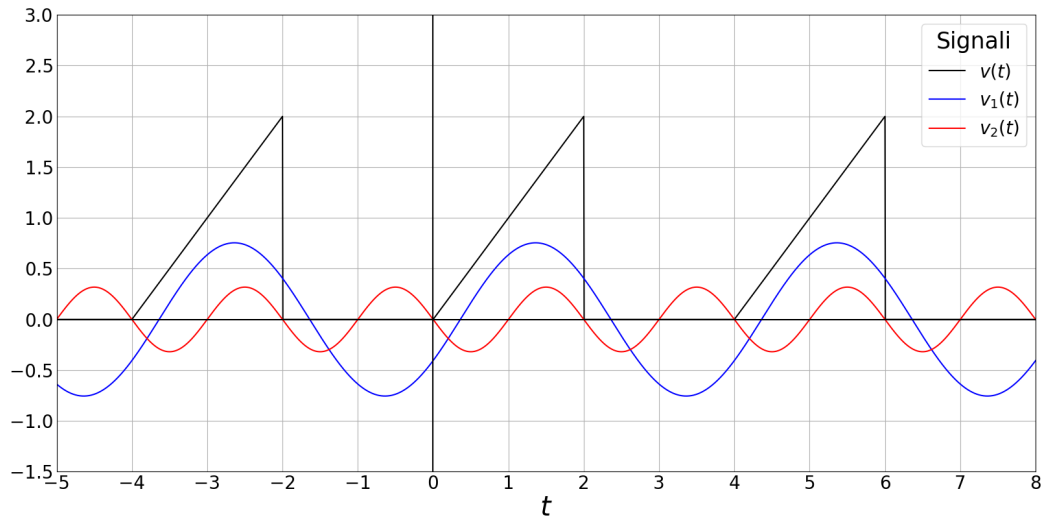
$$A_2 = \frac{1}{\pi} \quad (41) \quad \phi_2 = \frac{\pi}{2} \quad (42)$$

Zamenom dobijenih koeficijenata harmonika u definiciju k -tog harmonika dobija se:

$$v_1(t) = \frac{2}{\pi^2} \sqrt{4 + \pi^2} \cos \left(\frac{\pi}{2} t + \arctan \frac{\pi}{2} - \pi \right) \quad (43)$$

$$v_2(t) = \frac{1}{\pi} \cos \left(\pi t + \frac{\pi}{2} \right) \quad (44)$$

Na sledećoj slici može se videti grafik sa signalima $v(t)$, $v_1(t)$ i $v_2(t)$:



Slika 9: Signali $v(t)$, $v_1(t)$ i $v_2(t)$

Za skiciranje grafika korišćen je sledeći kod u programskom jeziku Python:

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4
5 def u(t):
6     return 0 if t < 0 else 1
7
8 def v(L):
9     def vk(t):
10         s = 0
11         for k in range(-L, L + 1):
12             s += (t - 4 * k) * (u(t - 4 * k) - u(t - 4 * k -
13             2))
14         return s
15     return vk
16
17 def v1(t):
18     pi2 = math.pi ** 2
19     pih = math.pi / 2
20     return 2 / pi2 * math.sqrt(4 + pi2) * math.cos(pih * t +
21     math.atan(pih) - math.pi)
22
23 def v2(t):
24     pi = math.pi
25     return 1/pi*math.cos(pi*t+pi/2)
26
27 dt = 0.005
28
29 signals = [
30     np.vectorize(v(9), otypes=[float]),
31     np.vectorize(v1, otypes=[float]),
32     np.vectorize(v2, otypes=[float])
33 ]
34 data = np.arange(-5, 8 + dt, dt)
35 values = np.array(list(map(lambda signal: signal(data),
36     signals)))
37
38 plt.style.use('_mpl-gallery')
39 signal_names = ["$v(t)$", "$v_1(t)$", "$v_2(t)$"]
40 signal_colors = ["black", "blue", "red"]
41 resolution = 4
42 plot_count = 1
43 csfont = {'fontname': 'Times New Roman', 'fontsize': 40}
44 fig, ax = plt.subplots(plot_count,
45     figsize=(resolution * 4, 2 *
46     resolution * plot_count),
47     constrained_layout=True)
```

```

45 lx = min(data)
46 hx = max(data)
47 ly = math.floor(min(map(min, values)))
48 hy = math.ceil(max(map(max, values)))
49 ax.set(xlim=(lx, hx),
50        xticks=np.arange(lx, hx + dt),
51        ylim=(ly - 0.5, hy + 0.5),
52        yticks=np.arange(ly - 0.5, hy + 1.5, 0.5))
53 for i in range(len(signals)):
54     ax.plot(data, values[i], linewidth=1.5,
55            label=signal_names[i], color=signal_colors[i])
56
57 ax.xaxis.set_tick_params(labelsize=20)
58 ax.yaxis.set_tick_params(labelsize=20)
59 ax.axhline(0, color="black")
60 ax.axvline(0, color="black")
61 ax.set_xlabel("$t$", fontdict=csfont, fontsize=30)
62 plt.legend(title="Signali", fontsize=20, title_fontsize=25)
63 # ax.set_ylabel("$v(t)$", fontdict=csfont, fontsize=30)
64
65 plt.show()

```

Kod 9: Kod za generisanje grafika sa Slike 9

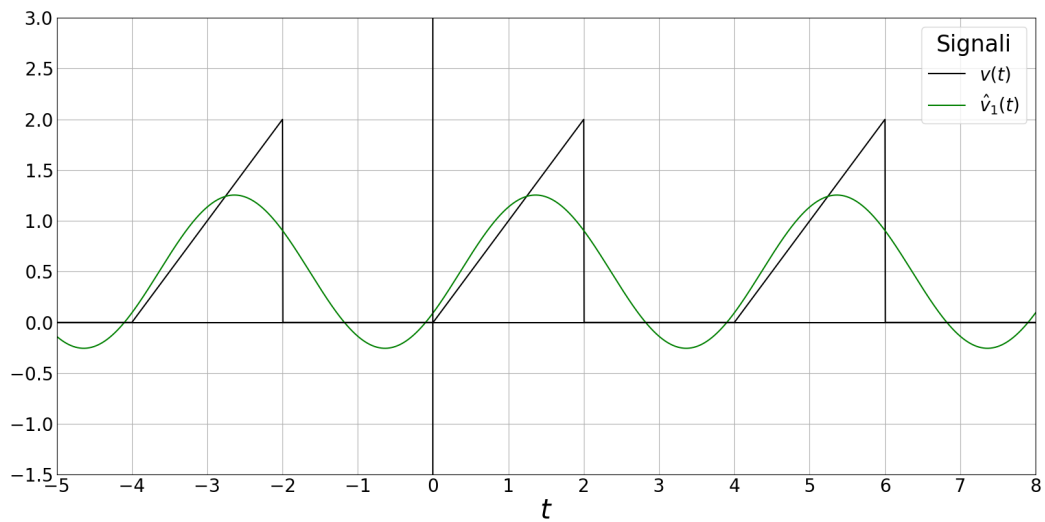
4.3 Skicirati na istom grafiku originalni signal $v(t)$ i aproksimaciju:

$$\hat{v}_1(t) = \sum_{k=-1}^1 a_k e^{jk\omega_0 t} = a_0 + v_1(t)$$

Zamenom izračunatih vrednosti koeficijenata dobijamo da je:

$$\hat{v}_1(t) = \frac{1}{2} + \frac{2}{\pi^2} \sqrt{4 + \pi^2} \cdot \cos\left(\frac{\pi}{2}t + \arctan \frac{\pi}{2} - \pi\right) \quad (45)$$

Na sledećoj slici može se videti grafik sa signalima $v(t)$ i $\hat{v}_1(t)$:



Slika 10: Signali $v(t)$ i $\hat{v}_1(t)$

Za skiciranje grafika korišćen je sledeći kod u programskom jeziku Python:

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4
5 def u(t):
6     return 0 if t < 0 else 1
7
8 def v(L):
9     def vk(t):
10         s = 0
11         for k in range(-L, L + 1):
12             s += (t - 4 * k) * (u(t - 4 * k) - u(t - 4 * k -
13             2))
14         return s
15     return vk
16
17 def v1(t):
18     pi2 = math.pi ** 2
19     pih = math.pi / 2
20     return 2 / pi2 * math.sqrt(4 + pi2) * math.cos(pih * t +
21     math.atan(pih) - math.pi)
22
23 def v2(t):
24     pi = math.pi
25     return 1/pi*math.cos(pi*t+pi/2)
26
27 dt = 0.005
28
29 signals = [
30     np.vectorize(v(9), otypes=[float]),
31     np.vectorize(lambda t: 1/2 + v1(t))
32 ]
33 data = np.arange(-5, 8 + dt, dt)
34 values = np.array(list(map(lambda signal: signal(data),
35     signals)))
36
37 plt.style.use('_mpl-gallery')
38 signal_names = ["$v(t)$", "$\\hat{v}_1(t)$"]
39 signal_colors = ["black", "green"]
40 resolution = 4
41 plot_count = 1
42 csfont = {'fontname': 'Times New Roman', 'fontsize': 40}
43 fig, ax = plt.subplots(plot_count,
44     figsize=(resolution * 4, 2 *
45     resolution * plot_count),
46     constrained_layout=True)
47 lx = min(data)
```

```

45 hx = max(data)
46 ly = math.floor(min(map(min, values)))
47 hy = math.ceil(max(map(max, values)))
48 ax.set(xlim=(lx, hx),
49        xticks=np.arange(lx, hx + dt),
50        ylim=(ly - 0.5, hy + 0.5),
51        yticks=np.arange(ly - 0.5, hy + 1.5, 0.5))
52 for i in range(len(signals)):
53     ax.plot(data, values[i], linewidth=1.5,
54            label=signal_names[i], color=signal_colors[i])
55
56 ax.xaxis.set_tick_params(labelsize=20)
57 ax.yaxis.set_tick_params(labelsize=20)
58 ax.axhline(0, color="black")
59 ax.axvline(0, color="black")
60 ax.set_xlabel("$t$", fontdict=csfont, fontsize=30)
61 plt.legend(title="Signali", fontsize=20, title_fontsize=25)
62 # ax.set_ylabel("$v(t)$", fontdict=csfont, fontsize=30)
63
64 plt.show()

```

Kod 10: Kod za generisanje grafika sa Slike 10

4.4 Grafički prikazati zavisnost modula koeficijenta od indeksa k , za $0 \leq k \leq 3$

Koeficijenti Furijeovog reda a_k signala $v(t)$ dati sa (30) mogu se napisati i u obliku:

$$a_k = \begin{cases} \frac{1}{2}j, & k = 0 \\ \frac{j}{k\pi}, & k = 2p \\ -\frac{2}{\pi^2 k^2} - j\frac{1}{k\pi}, & k = 2p + 1 \end{cases} \quad (46)$$

Odavde lako možemo naći moduo koeficijenta a_k :

$$|a_k| = \begin{cases} \frac{1}{2}, & k = 0 \\ \frac{1}{k\pi}, & k = 2p \\ \frac{1}{k^2\pi^2} \sqrt{4 + k^2\pi^2}, & k = 2p + 1 \end{cases} \quad (47)$$

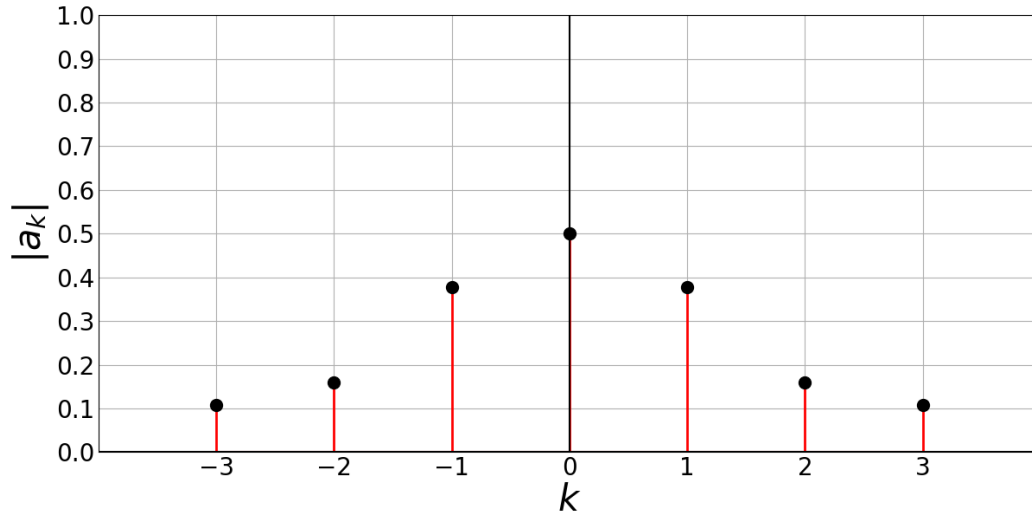
Kako je $a_k = a_{-k}^* \Rightarrow |a_k| = |a_{-k}|$ pomoću (47) možemo lako izračunati module koeficijenata za $|k| \leq 3$

$$|a_1| = |a_{-1}| = \frac{1}{\pi^2} \sqrt{4 + \pi^2} \approx 0.377 \quad (48)$$

$$|a_2| = |a_{-2}| = \frac{1}{2\pi} \approx 0.159 \quad (49)$$

$$|a_3| = |a_{-3}| = \frac{1}{9\pi^2} \sqrt{4 + 9\pi^2} \approx 0.109 \quad (50)$$

Na sledećoj slici može se videti grafik sa modulima koeficijenata a_k za $|k| \leq 3$:



Slika 11: Moduli koeficijenata a_k

Za skiciranje grafika korišćen je sledeći kod u programskom jeziku Python:

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import math
4
5 def ak(k):
6     k = abs(k)
7     if k == 0:
8         return 0.5
9     elif k % 2 == 0:
10        return 1/(k*math.pi)
11    else:
12        kpi = k*math.pi
13        return 1/(kpi**2)*math.sqrt(4+kpi**2)
14
15 dt = 1
16 data = np.arange(-3, 3 + dt, dt)
17 values = np.vectorize(ak)(data)
18
19 plt.style.use('_mpl-gallery')
20 resolution, plot_count = 3, 1
21 csfont = {'fontname': 'Times New Roman', 'fontsize': 40}
22
23 fig, ax = plt.subplots(plot_count,
24                        figsize=(resolution * 4, 2 *
25                                resolution * plot_count),
26                        constrained_layout=True)

```

```

26 lx,hx = min(data),max(data)
27 ax.xaxis.set_tick_params(labelsize=20)
28 ax.yaxis.set_tick_params(labelsize=20)
29 ax.set(xlim=(lx-1, hx+1), xticks=np.arange(lx, hx + 1),
30        ylim=(0, 0.5), yticks=np.arange(0, 1.1,0.1))
31 markerline, stemline, baseline, = ax.stem(data, values)
32 ax.axhline(0, color="black")
33 ax.axvline(0, color="black")
34 ax.set_xlabel("$k$", fontdict=csfont, fontsize=30)
35 ax.set_ylabel("$|a_k|$", fontdict=csfont, fontsize=30)
36 plt.setp(stemline, linewidth=2, color="red")
37 plt.setp(markerline, markersize=10, color="black")
38
39 plt.show()

```

Kod 11: Kod za generisanje grafika sa Slike 11

4.5 Ispitati konvergentnost Furijeovog reda signala u srednje-kvadratnom smislu

Furijski red signala $v(t)$ je konvergentan u srednje-kvadratnom smislu ako je integral

$$\int_T |v(t)|^2 dt \quad (51)$$

konačan, tj. konvergentan. Računajući integral

$$I = \int_T |v(t)|^2 dt = \int_0^4 |v(t)|^2 dt = \int_0^2 |t|^2 dt = \left. \frac{t^3}{3} \right|_0^2 = \frac{8}{3} < +\infty$$

vidimo da on konvergira. Možemo zaključiti da Furijski red konvergira u srednje-kvadratnom smislu.

Lista Kodova

1	Kod za generisanje grafika sa Slike 1	3
2	Kod za generisanje grafika sa Slike 2	6
3	Kod za generisanje grafika sa Slike 3.	8
4	Kod za generisanje grafika sa Slike 4.	11
5	Kod za generisanje grafika sa Slike 5.	13
6	Kod za generisanje grafika sa Slike 6.	18
7	Kod za generisanje grafika sa Slike 7.	21
8	Kod za generisanje grafika sa Slike 8	28
9	Kod za generisanje grafika sa Slike 9	32
10	Kod za generisanje grafika sa Slike 10	35
11	Kod za generisanje grafika sa Slike 11	37