Signali i sistemi
Domaci 1

Branislav Đumić
Br. indeksa: 0260/2020

Parametri

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

Sadžaj

1 Zadatak 1.

Osnovne osobine i vremenske transformacije signala

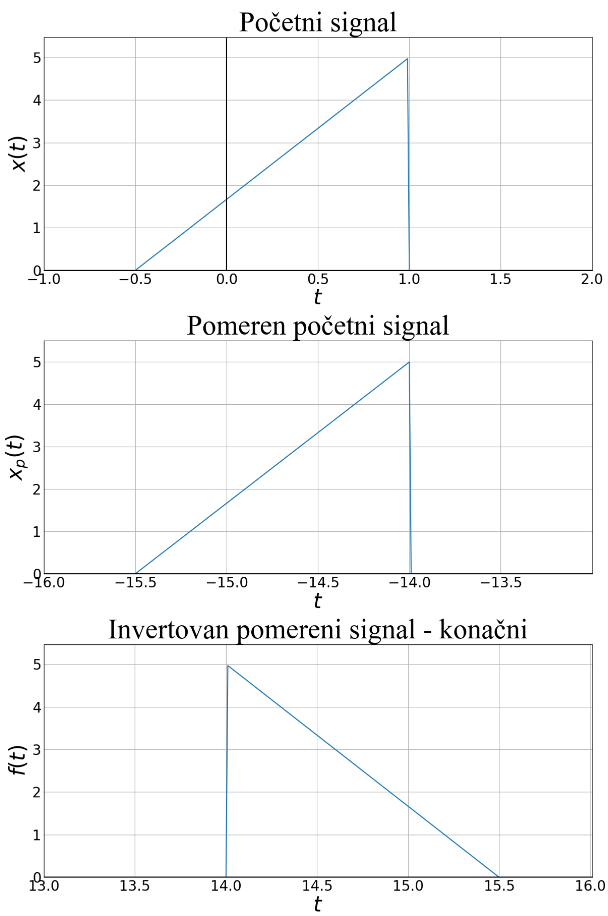
1.1 Odrediti analitički izraz za signal $f(t)$ i nacrtati grafike signala $x(t)$ i $f(t)$.

$$f(t) = x(3(Q+3) - (R+1)t) \tag{1}$$

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

$$f(t) = x(15 - t) \tag{2}$$

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:

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

def u(t):
    return 0 if t < 0 else 1

def array_u(t):
    return np.where(t < 0, 0, 1)

def x(t):
    return 5/3*(2*t+1)*(array_u(t+0.5)-array_u(t-1))

array_x = [
    x,
    lambda t: x(t+15),
    lambda t:x(15-t)
]

dt = 0.01

data = [
    np.arange(-1, 2 + dt,dt),
    np.arange(-16, -13 + dt, dt),
    np.arange(13, 16 + dt, dt)
]

values = list(map(
            lambda param: array_x[param[0]](param[1]),
            enumerate(data)))

y_label_list = ["$x(t)$", "$x_p(t)$", "$f(t)$"]
plot_names = [
    "Pocetni signal",
    "Pomeren pocetni signal",
    "Invertovan pomereni signal - konacni"]

plt.style.use('_mpl-gallery')

resolution = 3
plot_count = 3
csfont = {'fontname' : 'Times New Roman', 'fontsize' : 40}
fig, ax = plt.subplots(plot_count,
    figsize=(resolution*4, 2*resolution*plot_count),
    constrained_layout=True)
for i in range(plot_count):
    lx = min(data[i])
    hx = max(data[i])
    ly = min(values[i])
```

```
    hy = max(values[i])
    ax[i].set_title(plot_names[i], fontdict=csfont)
    ax[i].xaxis.set_tick_params(labelsize=20)
    ax[i].yaxis.set_tick_params(labelsize=20)
    ax[i].set(xlim=(lx, hx),
              xticks=np.arange(lx, hx, 0.5),
              ylim=(ly, hy+0.5),
              yticks=np.arange(ly, hy+0.5))
    ax[i].plot(data[i], values[i], linewidth=1.5)
    ax[i].axhline(0, color="black")
    ax[i].axvline(0, color="black")
    ax[i].set_xlabel("$t$", fontdict=csfont, fontsize=30)
    ax[i].set_ylabel(
        y_label_list[i], fontdict=csfont, fontsize=30)

plt.show()
```

Kod 1: Kod za generisanje grafika sa Slike 1

Orderđivanje analitičkog oblika

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

$$f(t) = (at+b)\big(u(t-14) - u(t-15.5)\big) \tag{3}$$

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

$$f(14) = 14a + b = 5 \tag{4}$$
$$f(15.5) = 15.5a + b = 0 \tag{5}$$

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

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

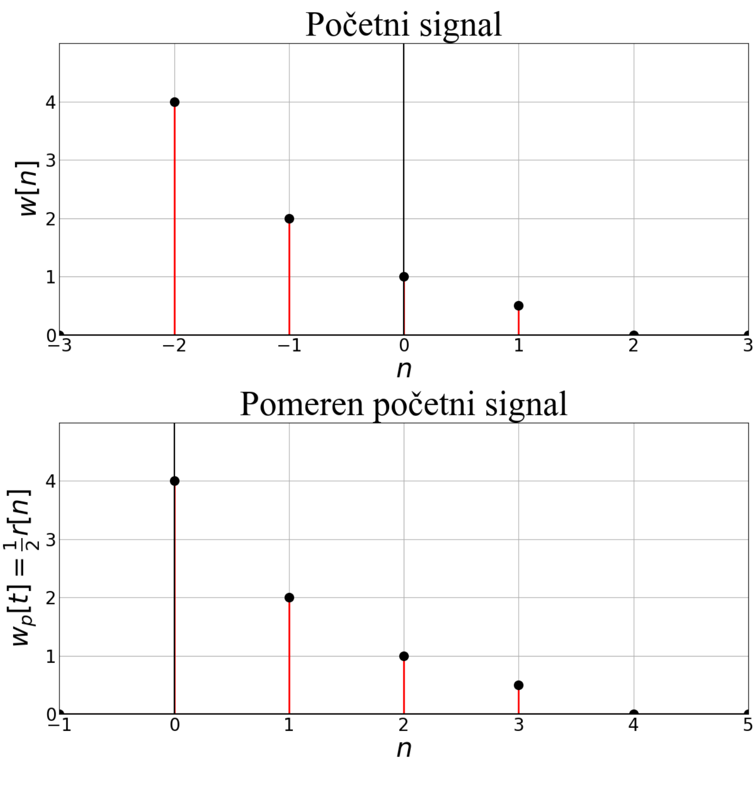
1.2 Predložiti metodu interpolacije, pa na osnovu nje prikazati grafik i napisati analitički oblik signala:

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

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

$$w[n] = 0.5^n \Big(u[n+2] - u[n-2]\Big) \tag{8}$$

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 inerpolacijom 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:

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

def u(n):
    return 0 if n < 0 else 1

def array_u(n):
    return np.where(n < 0, 0, 1)

def w(n):
    return 0.5**n * (array_u(n + 2) - array_u(n - 2))

array_x = [
    w,
    lambda n: w(n - 2),
]

dt = 1

data = [
    np.arange(-3, 3 + dt, dt),
    np.arange(-1, 5 + dt, dt),
]

values = list(map(
    lambda param: array_x[param[0]](param[1]),
    enumerate(data)))

y_label_list = ["$w[n]$", "$w_p[t]=\\frac{1}{2}r[n]$"]
plot_names = [
    "Pocetni signal",
    "Pomeren pocetni signal"]

plt.style.use('_mpl-gallery')

resolution = 3
plot_count = 2
csfont = {'fontname': 'Times New Roman', 'fontsize': 40}
fig, ax = plt.subplots(plot_count,
                       figsize=(resolution * 4, 2 *
    resolution * plot_count),
                       constrained_layout=True)
for i in range(plot_count):
    lx = min(data[i])
    hx = max(data[i])
    ly = min(values[i])
    hy = max(values[i])
    ax[i].xaxis.set_tick_params(labelsize=20)
```

```
    ax[i].yaxis.set_tick_params(labelsize=20)
    ax[i].set_title(plot_names[i], fontdict=csfont)
    ax[i].set(xlim=(lx, hx),
              xticks=np.arange(lx, hx+1),
              ylim=(ly, hy+1),
              yticks=np.arange(ly, hy+1))
    markerline, stemline, baseline,= ax[i].stem(data[i], values[i])
    ax[i].axhline(0, color="black")
    ax[i].axvline(0, color="black")
    ax[i].set_xlabel("$n$", fontdict=csfont, fontsize=30)
    ax[i].set_ylabel(
        y_label_list[i],
        fontdict=csfont,
        fontsize=30)
    plt.setp(stemline, linewidth=2,color="red")
    plt.setp(markerline, markersize=10,color="black")

plt.show()
```

Kod 2: Kod za generisanje grafika sa Slike 2

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

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

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.

$$v[-3] = r[-1] = 0 \qquad v[-2] = \frac{2\cdot r[-1]+r[0]}{3} = \frac{4}{3} \qquad v[-1] = \frac{r[-1]+2\cdot r[0]}{3} = \frac{8}{3}$$

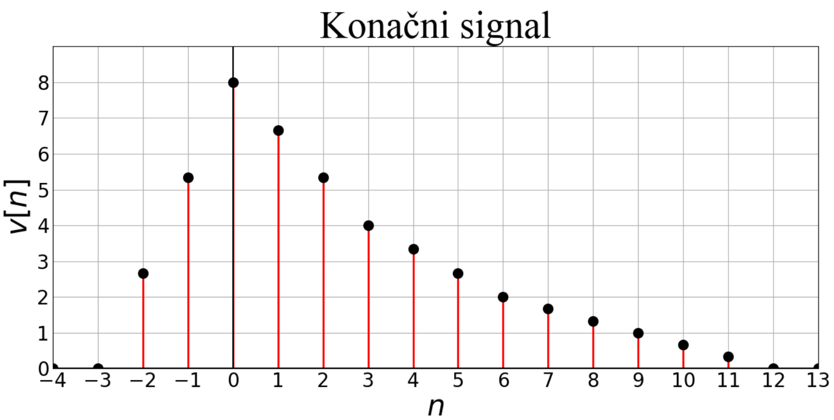
$$v[0] = r[0] = 8 \qquad v[1] = \frac{2\cdot r[0]+r[1]}{3} = \frac{20}{3} \qquad v[2] = \frac{r[0]+2\cdot r[1]}{3} = \frac{16}{3}$$

$$v[3] = r[1] = 4 \qquad v[4] = \frac{2\cdot r[1]+r[2]}{3} = \frac{10}{3} \qquad v[5] = \frac{r[1]+2\cdot r[2]}{3} = \frac{8}{3}$$

$$v[6] = r[2] = 2 \qquad v[7] = \frac{2\cdot r[2]+r[3]}{3} = \frac{5}{3} \qquad v[8] = \frac{r[2]+2\cdot r[3]}{3} = \frac{4}{3}$$

$$v[9] = r[3] = 1 \qquad v[10] = \frac{2\cdot r[3]+r[4]}{3} = \frac{2}{3} \qquad v[11] = \frac{r[3]+2\cdot r[4]}{3} = \frac{1}{3}$$

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:

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

def u(n):
    return 0 if n < 0 else 1

array_u = np.vectorize(u)

def w(n):
    return 0.5 ** n * (array_u(n + 2) - array_u(n - 2))

def r(n):
    return 2 * w(n - 2)

def v(n):
    if n % 3 == 0:
        return r(n // 3)
    elif n % 3 == 1:
        return (2 * r(n // 3) + r(n // 3 + 1)) / 3
    else:
        return (r(n // 3) + 2 * r(n // 3 + 1)) / 3

dt = 1

data = np.arange(-4, 13 + dt, dt)

values = np.vectorize(v)(data)

plt.style.use('_mpl-gallery')
```

```python

resolution = 3
plot_count = 1
csfont = {'fontname': 'Times New Roman', 'fontsize': 40}
fig, ax = plt.subplots(plot_count,
                       figsize=(resolution * 4, 2 * resolution * plot_count),
                       constrained_layout=True)
lx = min(data)
hx = max(data)
ly = min(values)
hy = max(values)
ax.set_title("Konacni signal", fontdict=csfont)
ax.xaxis.set_tick_params(labelsize=20)
ax.yaxis.set_tick_params(labelsize=20)
ax.set(xlim=(lx, hx), xticks=np.arange(lx, hx + 1),
       ylim=(ly, hy + 1), yticks=np.arange(ly, hy + 1))
markerline, stemline, baseline, = ax.stem(data, values)
ax.axhline(0, color="black")
ax.axvline(0, color="black")
ax.set_xlabel("$n$", fontdict=csfont, fontsize=30)
ax.set_ylabel("$v[n]$", fontdict=csfont, fontsize=30)
plt.setp(stemline, linewidth=2, color="red")
plt.setp(markerline, markersize=10, color="black")

plt.show()
```

Kod 3: Kod za generisanje grafika sa Slike 3.

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

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:

$$w[n] = Ev\{w[n]\} + Od\{w[n]\} \tag{10}$$

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

$$w[-n] = Ev\{w[n]\} - Od\{w[n]\} \tag{11}$$

Kombinovanjem jednačina (10) i (11)

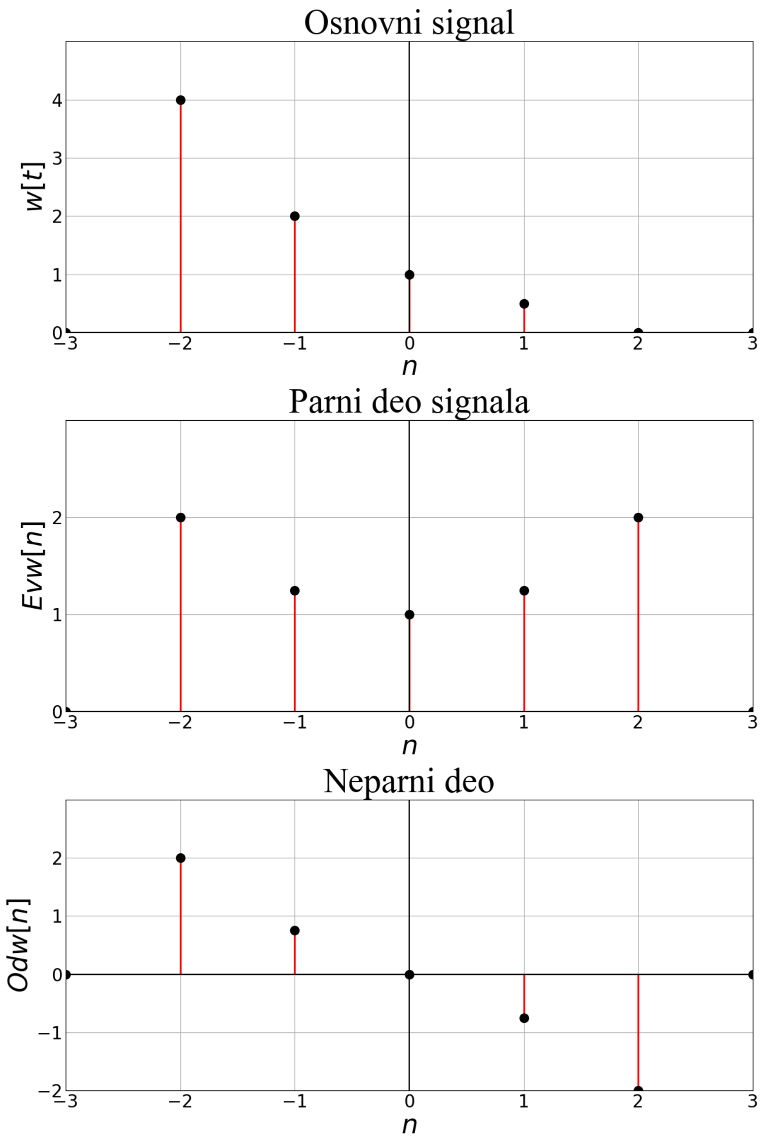
$$w[n] + w[-n] = 2 \cdot Ev\{w[n]\} \qquad (10) + (11)$$

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

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

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

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



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

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

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

def u(n):
    return 0 if n < 0 else 1

def w(n):
    return 0.5**n*(u(n+2)-u(n-2))

def Evw(n):
    return (w(n)+w(-n))/2

def Odw(n):
    return (w(n)-w(-n))/2

array_x = [
    np.vectorize(w),
    np.vectorize(Evw),
    np.vectorize(Odw)
]

dt = 1

data = [
    np.arange(-3, 3 + dt,dt),
    np.arange(-3, 3 + dt, dt),
    np.arange(-3, 3 + dt, dt)
]

values = list(map(
            lambda param: array_x[param[0]](param[1]),
            enumerate(data)))

y_label_list = ["$w[t]$", "$Ev{w[n]}$", "$Od{w[n]}$"]
plot_names = [
    "Osnovni signal",
    "Parni deo signala",
    "Neparni deo"]

plt.style.use('_mpl-gallery')

resolution = 3
plot_count = 3
csfont = {'fontname' : 'Times New Roman', 'fontsize' : 40}
fig, ax = plt.subplots(plot_count,
    figsize=(resolution*4, 2*resolution*plot_count),
    constrained_layout=True)
for i in range(plot_count):
```

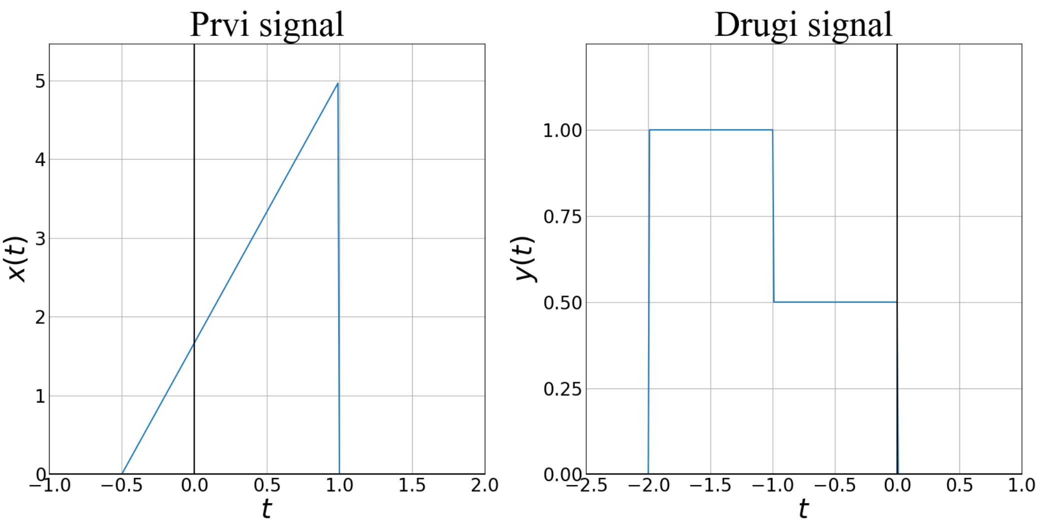
```python
    lx = min(data[i])
    hx = max(data[i])
    ly = min(values[i])
    hy = max(values[i])
    ax[i].xaxis.set_tick_params(labelsize=20)
    ax[i].yaxis.set_tick_params(labelsize=20)
    ax[i].set_title(plot_names[i], fontdict=csfont)
    ax[i].set(xlim=(lx, hx),
              xticks=np.arange(lx, hx+1),
              ylim=(ly, hy+1),
              yticks=np.arange(ly, hy+1))
    markerline, stemline, baseline,= ax[i].stem(data[i], values[i])
    ax[i].axhline(0, color="black")
    ax[i].axvline(0, color="black")
    ax[i].set_xlabel("$n$", fontdict=csfont, fontsize=30)
    ax[i].set_ylabel(
        y_label_list[i],
        fontdict=csfont,
        fontsize=30)
    plt.setp(stemline, linewidth=2,color="red")
    plt.setp(markerline, markersize=10,color="black")

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:

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

def u(t):
    return 0 if t < 0 else 1

def x(t):
    return 5 / 3 * (2 * t + 1) * (u(t + 0.5) - u(t - 1))

def y(t):
    return u(t + 2) - 0.5 * u(t + 1) - 0.5 * u(t)

array_x = [np.vectorize(x),np.vectorize(y)]

dt = 0.01

data = [
    np.arange(-1, 2 + dt, dt),
    np.arange(-2.5, 1 + dt, dt),
]
```

```python

values = list(map(
    lambda param: array_x[param[0]](param[1]),
    enumerate(data)))

y_label_list = ["$x(t)$", "$y(t)$"]
plot_names = [
    "Prvi signal",
    "Drugi signal"
]

plt.style.use('_mpl-gallery')

resolution = 4
plot_count = 2
csfont = {'fontname': 'Times New Roman', 'fontsize': 40}
fig, ax = plt.subplots(1,plot_count,
                       figsize=(2 * resolution * plot_count,
    resolution * 2),
                       constrained_layout=True)
for i in range(plot_count):
    lx = min(data[i])
    hx = max(data[i])
    ly = min(values[i])
    hy = max(values[i])
    ax[i].set_title(plot_names[i], fontdict=csfont)
    ax[i].xaxis.set_tick_params(labelsize=20)
    ax[i].yaxis.set_tick_params(labelsize=20)
    ax[i].set(xlim=(lx, hx),
              xticks=np.arange(lx, hx+dt, 0.5),
              ylim=(ly, hy + 0.5),
              yticks=np.arange(ly, hy+0.5))
    ax[i].plot(data[i], values[i], linewidth=1.5)
    ax[i].axhline(0, color="black")
    ax[i].axvline(0, color="black")
    ax[i].set_xlabel("$t$", fontdict=csfont, fontsize=30)
    ax[i].set_ylabel(
        y_label_list[i],
        fontdict=csfont,
        fontsize=30)
ax[1].set(ylim=(ly, hy+0.25),yticks=np.arange(ly, hy
    +0.25,0.25))


plt.show()
```

Kod 5: Kod za generisanje grafika sa Slike 5.

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

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

$$y(t) = u(t+2) - 0.5u(t+1) - 0.5u(t) \tag{15}$$

Radi lakšeg nalaženja konvolucije signala $x(t)$ i $y(t)$, $z(t) = x(t) * y(t)$, razdvojićemo signal $y(t)$ na signale $y_1(t) = u(t+2) - u(t+1)$ i $y_2(t) = \frac{1}{2}\big(u(t+1) - u(t)\big)$, 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:

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

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

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

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 $z_1(t)$:

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

Nastavljamo sa računom:

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

Konačno za $z_1(t)$ dobijamo:

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

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

$$z_2(t) \quad = \frac{1}{2}\int_{-\infty}^{+\infty}\Big[u(\tau+1)-u(\tau)\Big]\cdot x(t-\tau)d\tau$$

$$z_2(t) \quad = \frac{1}{2}\int_{-1}^{0} x(t-\tau)d\tau = \left\{ \begin{array}{ll} t-\tau & = \lambda \\ -d\tau & = d\lambda \end{array} \right\} = \int_{t}^{t+1} x(\lambda)d\lambda$$

$$z_2(t) \quad = \frac{1}{2}\cdot\frac{5}{3}\cdot\int_{t}^{t+1}\big(2\lambda+1\big)\Big(u(\lambda+0.5)-u(\lambda+1)\Big)d\lambda$$

$$z_2(t) \quad = \begin{cases} 0 & , \quad t+1 < -0.5 \\ \frac{5}{6}\int_{-0.5}^{t+1}\big(2\lambda+1\big)d\tau & , \quad t+1 \geq -0.5 \wedge t+1 < 1 \wedge t < -0.5 \\ \frac{5}{6}\int_{t}^{t+1}\big(2\lambda+1\big)d\tau & , \quad t \geq -0.5 \wedge t+1 < 1 \\ \frac{5}{6}\int_{t}^{1}\big(2\lambda+1\big)d\tau & , \quad t+1 \geq 1 \wedge t < 1 \\ 0 & , \quad t \geq 1 \end{cases}$$

Nastavljamo sa računom:

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

Konačno za $z_1(t)$ dobijamo:

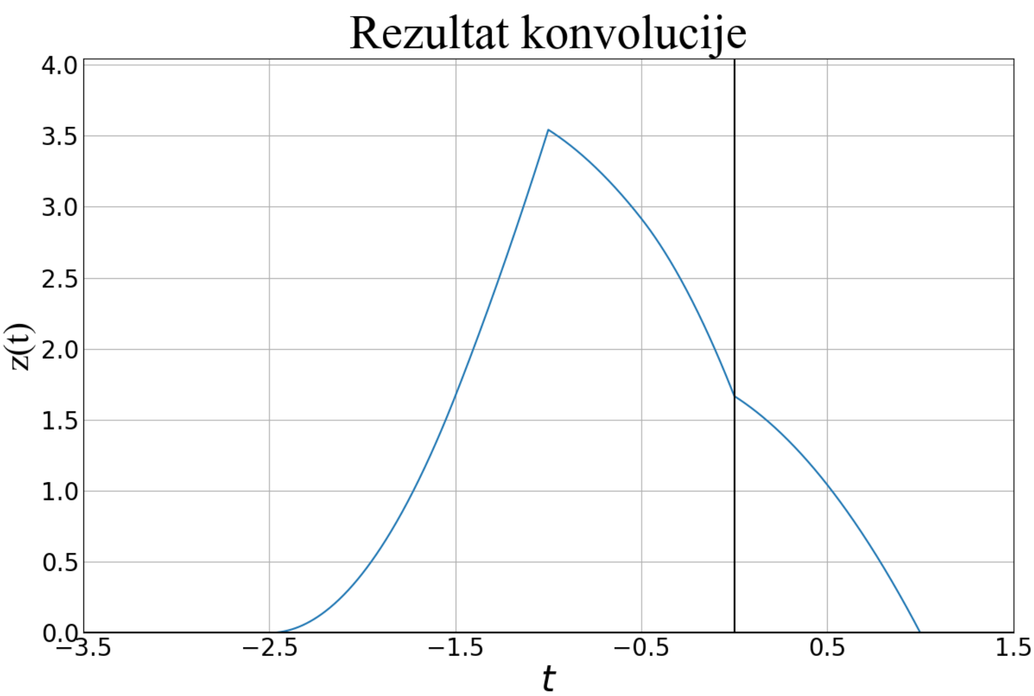
$$z_2(t) = \begin{cases} 0 & , \quad t < -1.5 \\ \frac{5}{6}(t^2+3t+2.25) & , \quad t \in [-1.5,-0.5) \\ \frac{5}{3}(t+1) & , \quad t \in [-0.5,0) \\ -\frac{5}{6}(2-t^2-t) & , \quad t \in [0,1) \\ 0 & , \quad t \geq 1 \end{cases} \tag{19}$$

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 $t(t)$:

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

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:

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

def z(t):
    if t < -2.5 or t >= 1:
        return 0
    elif -2.5 <= t < -1.5:
        return 5 / 3 * (t ** 2 + 5 * t + 6.25)
    elif -1.5 <= t < -1:
        return 5 / 6 * (t ** 2 + 7 * t + 10.25)
    elif -1 <= t < -0.5:
        return 5 / 6 * (-t ** 2 - 3 * t + 2.25)
    elif -0.5 <= t < 0:
        return 5 / 3 * (-t ** 2 - 2 * t + 1)
    elif 0 <= t < 1:
        return 5 / 6 * (2 - t ** 2 - t)
    else:
        return 0

dt = 0.005
```

```

data = np.arange(-3.5, 1.5 + dt, dt)
values = np.vectorize(z, otypes=[float])(data)

plt.style.use('_mpl-gallery')

resolution = 4
plot_count = 1
csfont = {'fontname': 'Times New Roman', 'fontsize': 40}
fig, ax = plt.subplots(plot_count,
                       figsize=(resolution * 3,2 * resolution * plot_count),
                       constrained_layout=True)
lx = min(data)
hx = max(data)
ly = min(values)
hy = max(values)
ax.set_title("Rezultat konvolucije", fontdict=csfont)
ax.xaxis.set_tick_params(labelsize=20)
ax.yaxis.set_tick_params(labelsize=20)
ax.set(xlim=(lx, hx),
       xticks=np.arange(lx, hx + dt),
       ylim=(ly, hy + 0.5),
       yticks=np.arange(ly, hy + 0.5,0.5))
ax.plot(data, values, linewidth=1.5)
ax.axhline(0, color="black")
ax.axvline(0, color="black")
ax.set_xlabel("$t$", fontdict=csfont, fontsize=30)
ax.set_ylabel("$z(t)$", fontdict=csfont, fontsize=30)

plt.show()
```

Kod 6: Kod za generisanje grafika sa Slike 6.

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

$$w[n] = 0.5^n \big(u[n+2] - u[n-2]\big) \tag{21}$$

$$z[n] = 2u[n] \tag{22}$$

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

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

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

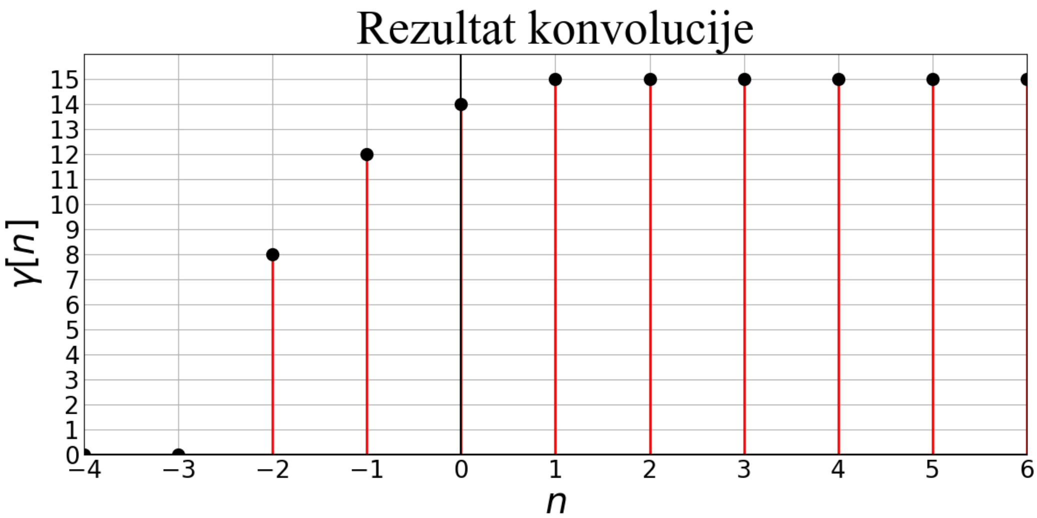
$$
\begin{aligned}
\gamma[n] \quad &= w[n] * z[n] = \sum_{k=-\infty}^{+\infty} z[k] \cdot w[n-k] \\
&= \sum_{k=-\infty}^{+\infty} 2u[k] \cdot w[n-k] \\
&= 2\sum_{k=0}^{+\infty} w[n-k] = \left\{ \begin{array}{c} m = n-k \\ \end{array} \right\} \\
&= 2\sum_{m=n}^{-\infty} w[m] = 2\sum_{m=-\infty}^{n} 0.5^m \big(u[m+2] - u[m-2]\big)
\end{aligned}
$$

$$
\begin{aligned}
\gamma[n] \quad &= \begin{cases} 0 & , \quad n < -2 \\ 2\sum_{m=-2}^{n} 0.5^m & , \quad n \geq -2 \wedge n < 1 \\ 2\sum_{m=-2}^{1} 0.5^m & , \quad n \geq 1 \end{cases} \\
\gamma[n] \quad &= \begin{cases} 0 & , \quad n < -2 \\ 2\sum_{m=0}^{n+2} 0.5^{m-2} & , \quad n \geq -2 \wedge n < 1 \\ 2\sum_{m=0}^{3} 0.5^{m-2} & , \quad n \geq 1 \end{cases} \\
\gamma[n] \quad &= \begin{cases} 0 & , \quad n < -2 \\ 2 \cdot 4\sum_{m=0}^{n} 0.5^m & , \quad n \geq -2 \wedge n < 1 \\ 2 \cdot 4\sum_{m=0}^{3} 0.5^m & , \quad n \geq 1 \end{cases} \\
\gamma[n] \quad &= \begin{cases} 0 & , \quad n < -2 \\ 8 \cdot \frac{1-0.5^{n+3}}{1-0.5} & , \quad n \geq -2 \wedge n < 1 \\ 8 \cdot \frac{1-0.5^4}{1-0.5} & , \quad n \geq 1 \end{cases}
\end{aligned}
$$

Za konačan analitički oblik konvolucije signala dobijamo:

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

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:

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

def u(n):
    return 0 if n < 0 else 1

array_u = np.vectorize(u)

def w(n):
    return 0.5 ** n * (array_u(n + 2) - array_u(n - 2))

def r(n):
    return 2 * w(n - 2)

def v(n):
    if n < -2:
        return 0
    elif -2 <= n < 1:
        return 16*(1-0.5**(n+3))
    else:
        return 15

dt = 1

data = np.arange(-4, 6 + dt, dt)

```

```
values = np.vectorize(v)(data)

plt.style.use('_mpl-gallery')

resolution = 3
plot_count = 1
csfont = {'fontname': 'Times New Roman', 'fontsize': 40}
fig, ax = plt.subplots(plot_count,
                       figsize=(resolution * 4, 2 * resolution * plot_count),
                       constrained_layout=True)
lx = min(data)
hx = max(data)
ly = min(values)
hy = max(values)
ax.set_title("Rezultat konvolucije", fontdict=csfont)
ax.xaxis.set_tick_params(labelsize=20)
ax.yaxis.set_tick_params(labelsize=20)
ax.set(xlim=(lx, hx), xticks=np.arange(lx, hx + 1),
       ylim=(ly, hy + 1), yticks=np.arange(ly, hy + 1))
markerline, stemline, baseline, = ax.stem(data, values)
ax.axhline(0, color="black")
ax.axvline(0, color="black")
ax.set_xlabel("$n$", fontdict=csfont, fontsize=30)
ax.set_ylabel("$\gamma[n]$", fontdict=csfont, fontsize=30)
plt.setp(stemline, linewidth=2, color="red")
plt.setp(markerline, markersize=10, color="black")

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:

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

Dalje imamo

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

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

$$(\exists B_1)(\forall t)|x(t)| < B1 \Rightarrow (\exists B_2)(\forall t)|y(t)| < B_2$$

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

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

Odavde 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: $y[n] = \sum_{k=n-2}^{n} \big(2x[k+1]\big)$

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

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

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

Linearnost

Definišemo sledeće signale:

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

Dalje imamo

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

Vidimo da je sistem linearan.

Stacionarnost

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

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

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

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

Vidimo da je <u>sistem stacionaran</u>.

Memorija

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

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

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

Stabilnost

$$(\exists B_1)(\forall n)|x[n]| < B1 \Rightarrow (\exists B_2)(\forall n)|y[n]| < B_2$$

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

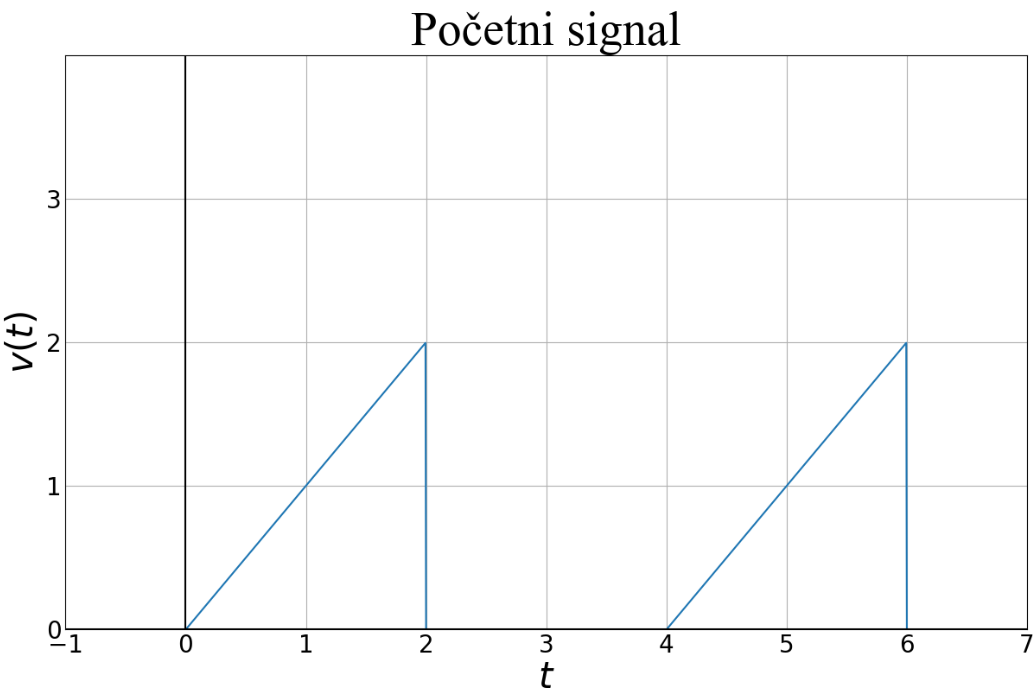
$$|x[n]| \le B_1 \Rightarrow |y[n]| = \Big| \sum_{n-1}^{n+1} 2x[k] \Big| \le 2 \sum_{n-1}^{n+1} \Big| x[k] \Big|$$
$$\Rightarrow |y[n]| \le 6B_1 = B_2$$

Odavde vidimo da je <u>sistem BIBO stabilan</u>.

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 analitčki oblik signala $v(t)$:

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

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

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

def u(t):
    return 0 if t < 0 else 1

def v(L):
    def vk(t):
        s = 0
        for k in range(-L, L + 1):
            s += (t - 4 * k) * (u(t - 4 * k) - u(t - 4 * k - 2))
        return s

    return vk

dt = 0.005

data = np.arange(-1, 7 + dt, dt)
values = np.vectorize(v(9), otypes=[float])(data)

plt.style.use('_mpl-gallery')
resolution,plot_count = 4,1
csfont = {'fontname': 'Times New Roman', 'fontsize': 40}
fig, ax = plt.subplots(plot_count,
                       figsize=(resolution * 3, 2 * resolution * plot_count),
                       constrained_layout=True)
lx = min(data)
hx = max(data)
ly = min(values)
hy = max(values)
ax.set_title("Pocetni signal", fontdict=csfont)
ax.xaxis.set_tick_params(labelsize=20)
ax.yaxis.set_tick_params(labelsize=20)
ax.set(xlim=(lx, hx),
       xticks=np.arange(lx, hx + dt),
       ylim=(ly, hy + 2),
       yticks=np.arange(ly, hy + 2))
ax.plot(data, values, linewidth=1.5)
ax.axhline(0, color="black")
ax.axvline(0, color="black")
ax.set_xlabel("$t$", fontdict=csfont, fontsize=30)
ax.set_ylabel("$v(t)$", fontdict=csfont, fontsize=30)

plt.show()
```

Kod 8: Kod za generisanje grafika sa Slike 8

Izvođenje koeficijenata Furijeovog reda
Furijeov red funkcije $v(t)$ dat je sumom:

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

gde su koeficijenti a_k definisani sa:

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

Odredimo sada koeficijente Furijeovog reda:

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

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

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

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

$$a_k = \left\{ \begin{array}{lll} \frac{1}{2} & , & k=0 \\ j\frac{1}{4\pi}(-1)^k - \frac{1}{\pi^2 k^2}\left(1-(-1)^k\right) & , & k \neq 0 \end{array} \right. \tag{30}$$

4.2 k-ti harmonik $v_k(t)$ signala $v(t)$ može se odrediti kao: $v_k(t) = a_{-k}e^{-jk\omega_0 t} + a_k e^{jk\omega_0}t = 2Re\{a_k e^{jk\omega_0 t}\} = A_k \cos(k\omega_0 t + \phi_k),\, k \geq 1$ Odrediti amplitude 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:

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

Odredimo prvo prvi harmonik $v_1(t)$.
Prvo ćemo odrediti koeficijent a_1

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

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

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

7

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

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

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

Izjednačavanjem koeficijenata konačno imamo:

$$A_1 = \frac{2}{\pi^2}\sqrt{4+\pi^2} \tag{36}$$

$$\phi_1 = \arctan\frac{\pi}{2} - \pi \tag{37}$$

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

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

Odredimo sada $|a_2|$ i θ_2:

$$|a_2| = \left|j\frac{1}{2\pi}\right| = \frac{1}{2\pi} \tag{39}$$

$$\theta_2 = \frac{\pi}{2} \tag{40}$$

Izjednačavanjem koeficijenata, kao u (35), dobijamo:

$$A_2 = \frac{1}{\pi} \tag{41}$$

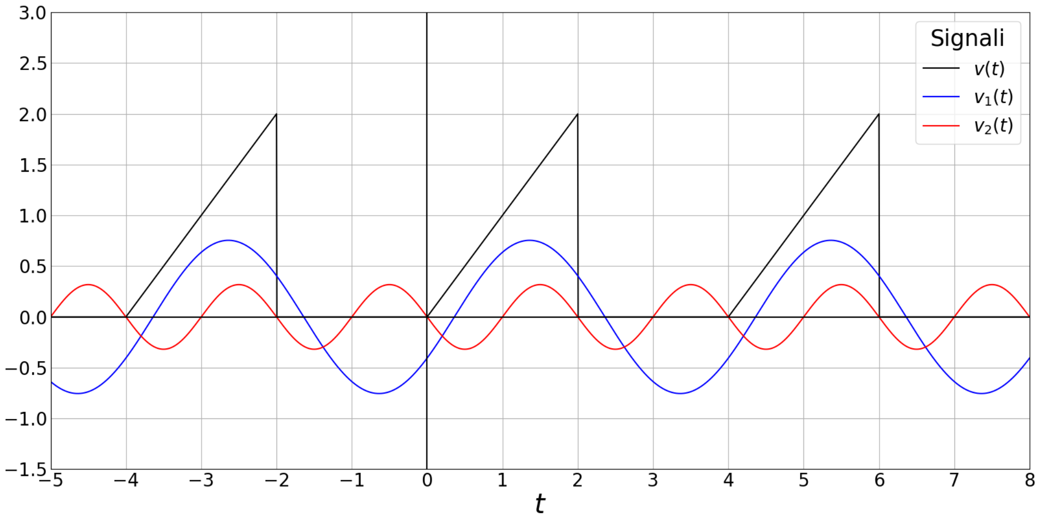
$$\phi_2 = \frac{\pi}{2} \tag{42}$$

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

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

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

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:

```python
import matplotlib.pyplot as plt
import numpy as np
import math

def u(t):
    return 0 if t < 0 else 1

def v(L):
    def vk(t):
        s = 0
        for k in range(-L, L + 1):
            s += (t - 4 * k) * (u(t - 4 * k) - u(t - 4 * k - 2))
        return s

    return vk

def v1(t):
    pi2 = math.pi ** 2
    pih = math.pi / 2
    return 2 / pi2 * math.sqrt(4 + pi2) * math.cos(pih * t + math.atan(pih) - math.pi)

def v2(t):
    pi = math.pi
    return 1/pi*math.cos(pi*t+pi/2)

dt = 0.005

signals = [
    np.vectorize(v(9), otypes=[float]),
    np.vectorize(v1, otypes=[float]),
    np.vectorize(v2, otypes=[float])
]
data = np.arange(-5, 8 + dt, dt)
values = np.array(list(map(lambda signal: signal(data), signals)))

plt.style.use('_mpl-gallery')
signal_names = ["$v(t)$", "$v_1(t)$", "$v_2(t)$"]
signal_colors = ["black","blue","red"]
resolution = 4
plot_count = 1
csfont = {'fontname': 'Times New Roman', 'fontsize': 40}
fig, ax = plt.subplots(plot_count,
                       figsize=(resolution * 4, 2 * resolution * plot_count),
                       constrained_layout=True)
```

```
lx = min(data)
hx = max(data)
ly = math.floor(min(map(min, values)))
hy = math.ceil(max(map(max, values)))
ax.set(xlim=(lx, hx),
       xticks=np.arange(lx, hx + dt),
       ylim=(ly - 0.5, hy + 0.5),
       yticks=np.arange(ly - 0.5, hy + 1.5, 0.5))
for i in range(len(signals)):
    ax.plot(data, values[i], linewidth=1.5,
            label=signal_names[i],color=signal_colors[i])

ax.xaxis.set_tick_params(labelsize=20)
ax.yaxis.set_tick_params(labelsize=20)
ax.axhline(0, color="black")
ax.axvline(0, color="black")
ax.set_xlabel("$t$", fontdict=csfont, fontsize=30)
plt.legend(title="Signali", fontsize=20, title_fontsize=25)
# ax.set_ylabel("$v(t)$", fontdict=csfont, fontsize=30)

plt.show()
```
Kod 9: Kod za generisanje grafika sa Slike 9

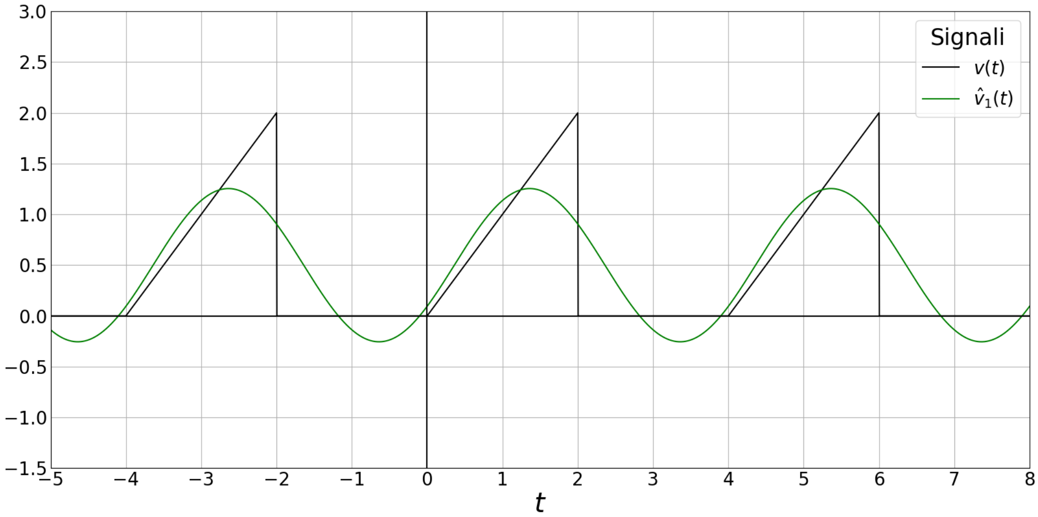
4.3 Skicirati na istom grafiku originalni signal $v(t)$ i aproksimaciju:

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

Zamenom izračunatih vrednosti koeficijenata dobijamo da je:

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

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:

```python
import matplotlib.pyplot as plt
import numpy as np
import math

def u(t):
    return 0 if t < 0 else 1

def v(L):
    def vk(t):
        s = 0
        for k in range(-L, L + 1):
            s += (t - 4 * k) * (u(t - 4 * k) - u(t - 4 * k - 2))
        return s

    return vk

def v1(t):
    pi2 = math.pi ** 2
    pih = math.pi / 2
    return 2 / pi2 * math.sqrt(4 + pi2) * math.cos(pih * t + math.atan(pih) - math.pi)

def v2(t):
    pi = math.pi
    return 1/pi*math.cos(pi*t+pi/2)

dt = 0.005

signals = [
    np.vectorize(v(9), otypes=[float]),
    np.vectorize(lambda t: 1/2 + v1(t))
]
data = np.arange(-5, 8 + dt, dt)
values = np.array(list(map(lambda signal: signal(data), signals)))

plt.style.use('_mpl-gallery')
signal_names = ["$v(t)$", "$\\hat{v}_1(t)$"]
signal_colors = ["black","green"]
resolution = 4
plot_count = 1
csfont = {'fontname': 'Times New Roman', 'fontsize': 40}
fig, ax = plt.subplots(plot_count,
                       figsize=(resolution * 4, 2 * resolution * plot_count),
                       constrained_layout=True)
lx = min(data)
```

```
hx = max(data)
ly = math.floor(min(map(min, values)))
hy = math.ceil(max(map(max, values)))
ax.set(xlim=(lx, hx),
       xticks=np.arange(lx, hx + dt),
       ylim=(ly - 0.5, hy + 0.5),
       yticks=np.arange(ly - 0.5, hy + 1.5, 0.5))
for i in range(len(signals)):
    ax.plot(data, values[i], linewidth=1.5,
            label=signal_names[i],color=signal_colors[i])

ax.xaxis.set_tick_params(labelsize=20)
ax.yaxis.set_tick_params(labelsize=20)
ax.axhline(0, color="black")
ax.axvline(0, color="black")
ax.set_xlabel("$t$", fontdict=csfont, fontsize=30)
plt.legend(title="Signali", fontsize=20, title_fontsize=25)
# ax.set_ylabel("$v(t)$", fontdict=csfont, fontsize=30)

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:

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

Odavde lako možemo naći moduo koeficijenta a_k:

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

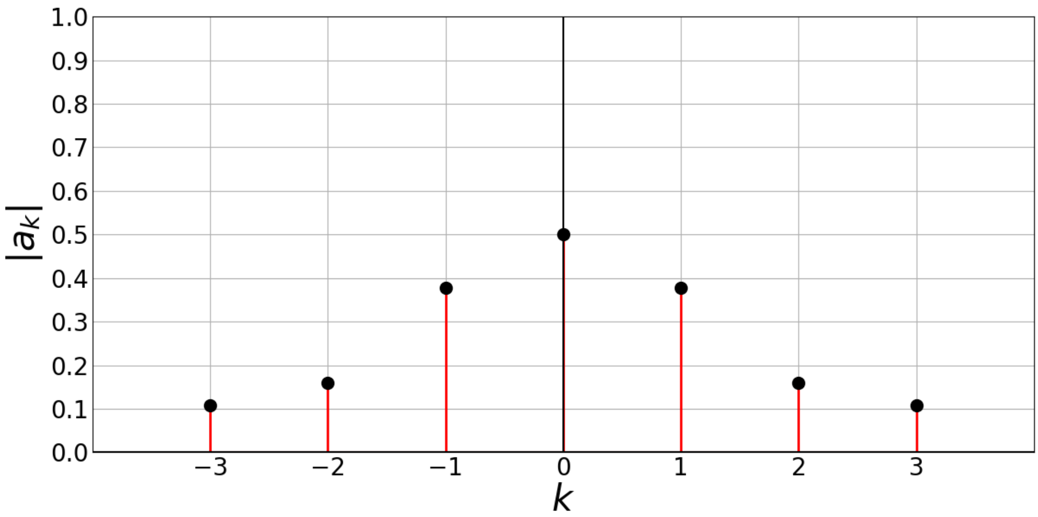
Kako je $a_k = a_{-k}^* \Rightarrow |a_k| = |a_{-k}|$ pomoću (47) možemo lako izračunati module koeficijenata za $|k| \leq 3$

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

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

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

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:

```
import matplotlib.pyplot as plt
import numpy as np
import math

def ak(k):
    k = abs(k)
    if k == 0:
        return 0.5
    elif k % 2 == 0:
        return 1/(k*math.pi)
    else:
        kpi = k*math.pi
        return 1/(kpi**2)*math.sqrt(4+kpi**2)

dt = 1
data = np.arange(-3, 3 + dt, dt)
values = np.vectorize(ak)(data)

plt.style.use('_mpl-gallery')
resolution, plot_count = 3, 1
csfont = {'fontname': 'Times New Roman', 'fontsize': 40}

fig, ax = plt.subplots(plot_count,
                       figsize=(resolution * 4, 2 * resolution * plot_count),
                       constrained_layout=True)
```

```
lx,hx = min(data),max(data)
ax.xaxis.set_tick_params(labelsize=20)
ax.yaxis.set_tick_params(labelsize=20)
ax.set(xlim=(lx-1, hx+1), xticks=np.arange(lx, hx + 1),
       ylim=(0, 0.5), yticks=np.arange(0, 1.1,0.1))
markerline, stemline, baseline, = ax.stem(data, values)
ax.axhline(0, color="black")
ax.axvline(0, color="black")
ax.set_xlabel("$k$", fontdict=csfont, fontsize=30)
ax.set_ylabel("$|a_k|$", fontdict=csfont, fontsize=30)
plt.setp(stemline, linewidth=2, color="red")
plt.setp(markerline, markersize=10, color="black")

plt.show()
```

Kod 11: Kod za generisanje grafika sa Slike 11

4.5 Ispitati konvergentnost Furijeovog reda signala u srednje-kvadratnom smislu

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

$$\int_T \left|v(t)\right|^2 dt \tag{51}$$

konačnan, tj. konvergentan. Računajući integral

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

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

Lista Kodova