Æfing 2

Merki og Kerfi

Georg Orlov og Rósa Elísabet

September 2023

1 Dæmi 2

Hér eru okkur gefin þrjú merki:

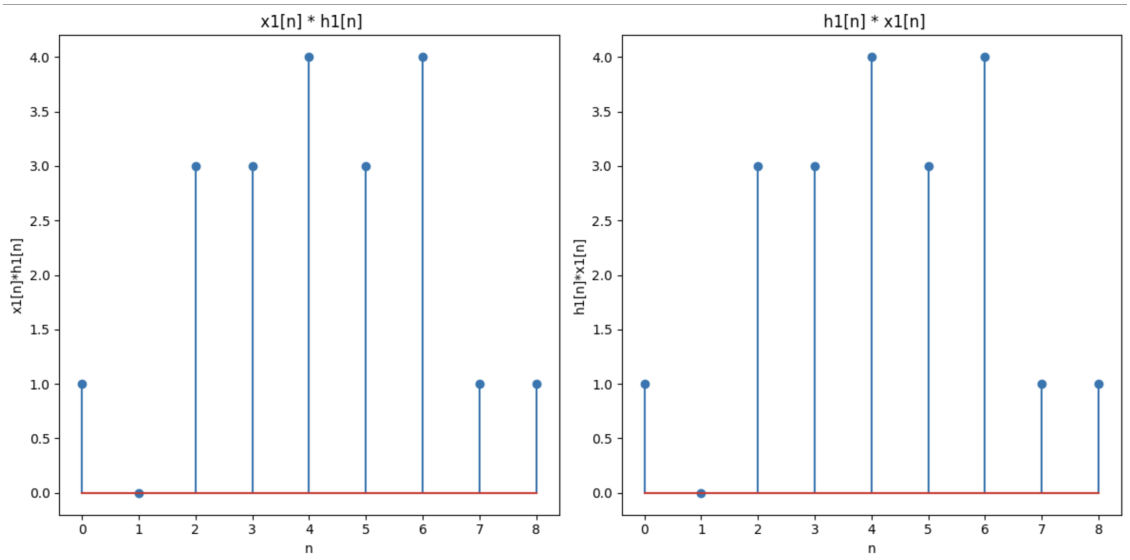
$$x_1[n] = \begin{cases} 1, & 0 \leq n \leq 4 \\ 0, & \text{annars,} \end{cases} \qquad h_1[n] = \begin{cases} 1, & n = 0 \\ -1, & n = 1 \\ 3, & n = 2 \\ 1, & n = 4 \\ 0, & \text{annars,} \end{cases} \qquad h_2[n] = \begin{cases} 2, & n = 1 \\ 5, & n = 2 \\ 4, & n = 3 \\ -1, & n = 4 \\ 0, & \text{annars} \end{cases}$$

Búa átti til merkin og nota þau til þess að sannreyna það að földun sé víxlin, dreifin og tengin. Hér að neðan má sjá kóða skrifaðan í python sem smíðar merkin $x_1[n]$, $x_2[n]$ og $x_3[n]$.

```
n_x1 = np.arange(0, 5)
n_h1 = np.arange(0, 5)
n_h2 = np.arange(0, 5)

x1 = np.array([1 if 0 <= n <= 4 else 0 for n in n_x1])
h1 = np.array([1 if n == 0 else -1 if n == 1 else 3 if n == 2 else 1 if n == 4 else 0 for n in n_h1])
h2 = np.array([2 if n == 1 else 5 if n == 2 else 4 if n == 3 else -1 if n == 4 else 0 for n in n_h2])
```

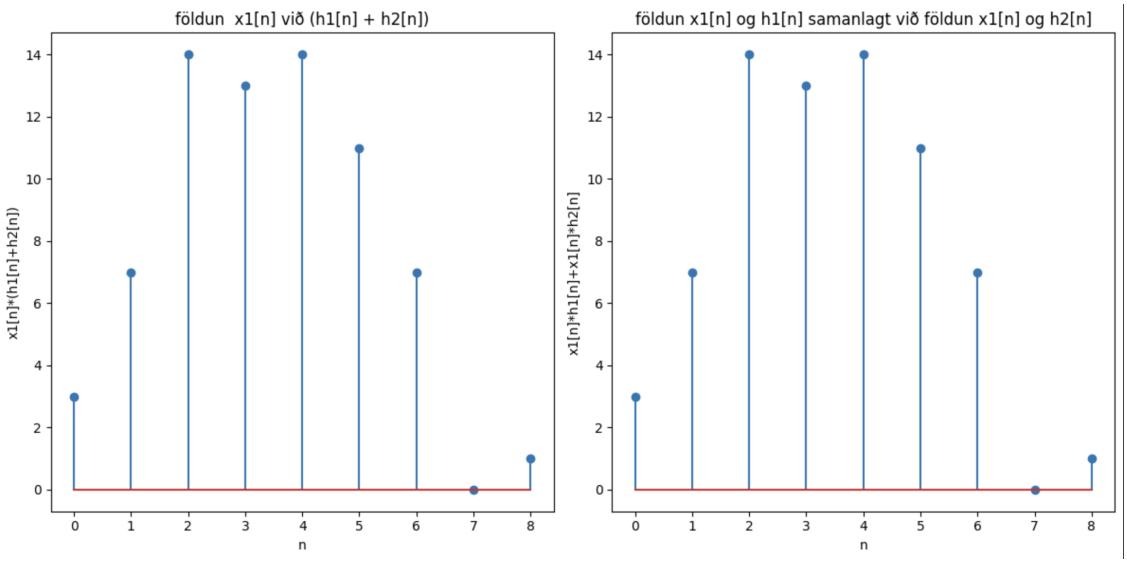
Það að földun sé víxlin aðgerð þýðir að $x * h = h * x$.Prófum þetta með því að nota fallið numpy.convolve á $x_1[n]$ og $h_1[n]$. Á mynd 1 má sjá gildi $x_1[n] * h_1[n]$ og $h_1[n] * x_1[n]$, þar sem $0 <= n <= 4$



Mynd 1: Sannreyning víxlunareiginleika földunar

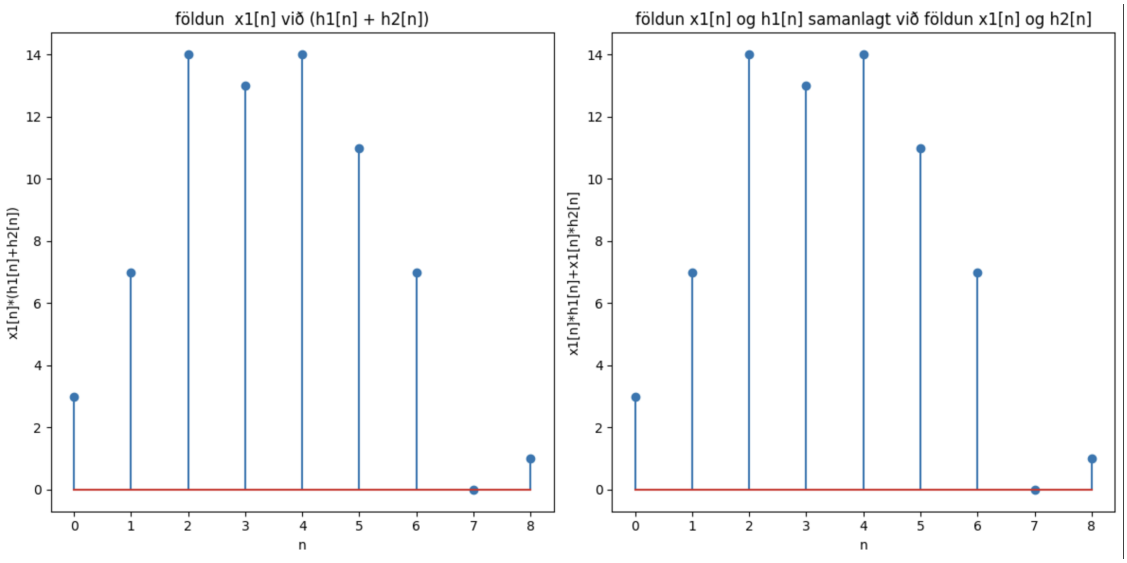
Eins og sjá má eru gildin þau sömu og þar með höfum við sannreynt það að földun sé víxlin aðgerð. Nú skulum við sannreyna það að földun sé dreifin, en það þýðir að $x * (h_1 + h_2) = x * h_1 + x * h_2$. Hér notum við aftur numpy.convolve til

þess að reikna gildin á $x_1[n] * (h_1[n] + h_2[n])$ og berum saman við $x_1[n] * h_1[n] + x_1[n] + h_2[n]$. Niðurstöðuna má sjá á mynd 2.



Mynd 2: Sannreyning dreifieiginleika földunar

Eins og sjá má á mynd tvö eru gildin þau sömu og þar með höfum við sannreynt að földun sé dreifin aðgerð. Að lokum skulum við sannreyna það að földun sé tengin, en það þýðir að $x * (h_1 * h_2) = (x * h_1) * h_2$. Sannreynum þetta með því að reikna $x_1[n] * (h_1[n] * h_2[n])$ með numpy.convolve og bera niðurstöðuna saman við $(x_1[n] * h_1[n]) * h_2[n]$. Á mynd 3 má sjá niðurstöðu þessa reikninga.



Mynd 3: Sannreyning tengingareiginleika földunar

2 Dæmi 3

Hér skoðuðum við þrjú kerfi:

- Kerfi 1: $w[n] = x[n] - x[n-1] - x[n-2]$
- Kerfi 2: $y[n] = \cos(x[n])$
- Kerfi 3: $z[n] = nx[n]$

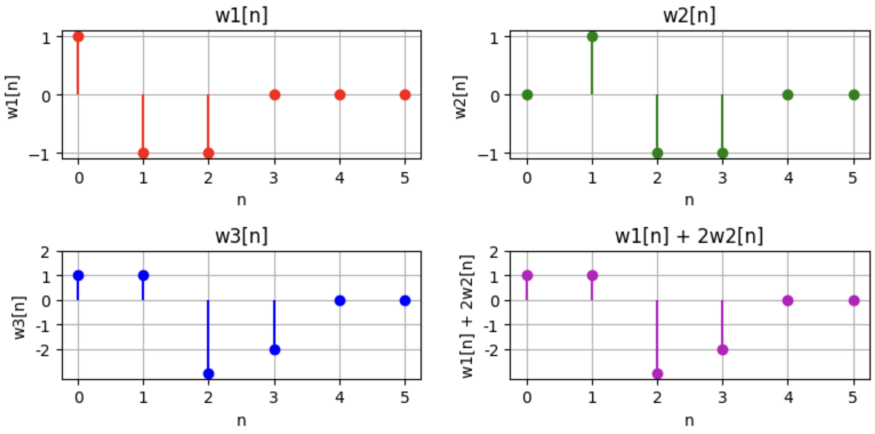
og skoðuðum útmerki þeirra fyrir þrjú ólík innmerki:

- $x_1[n]$: $\delta[n]$
- $x_2[n]$: $\delta[n-1])$
- $x_3[n]$ $\delta[n] + 2\delta[n-1]$

Hér köllum við svörun kerfis 1 við x_1 $w_1[n]$, svörun kerfis 2 við $x_2[n]$ $y_2[n]$ og svo framvegis. Nú viljum við komast að því hvort að kerfin hér að ofan séu línuleg og tímaóhað. Kerfi er tímaóhað ef hliðrun í innmerki veldur bara sömu hliðrun í innmerki, og línulegt ef það uppfyllir þessi tvö skilyrði:

- innmerkið $x_1 + x_2$ gefur útmerkið $y_1 + y_2$
- innmerkið ax_1 gefur útmerkið $ay_1 (a \in C$ fasti)

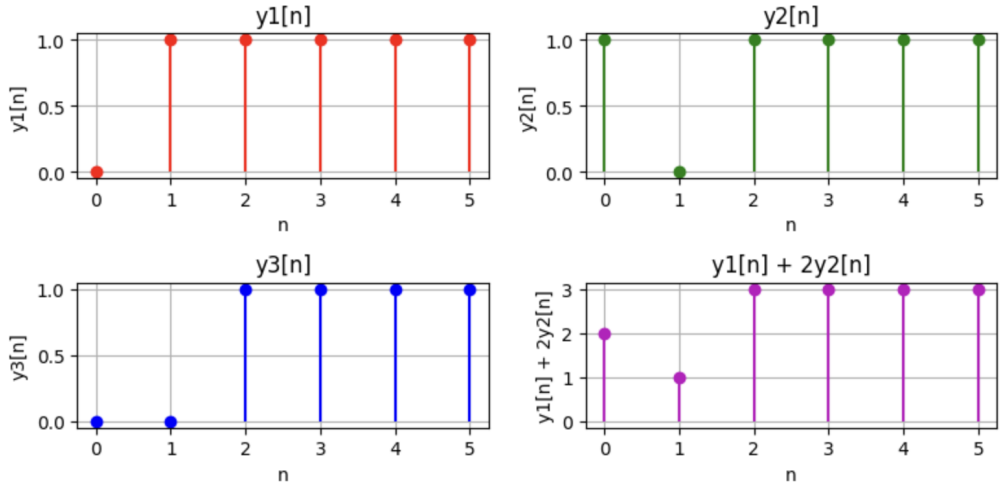
Athugum að $x_3[n] = x_1[n] + 2x_2[n]$ og innmerkin $x_1[n]$ og $x_2[n]$ eru þau sömu, þeim er bara hliðrað. Nýtum okkur þetta með því að bera saman útmerkin w_1 og w_2 til þess að kanna tímaóhæði, og útmerkið w_3 og $w_1 + 2w_2$ til þess að kanna línuleika.



Mynd 4: Svörun kerfis 1 við innmerkjunum x_1, x_2 og x_3

Á mynd 4 má sjá að svörun kerfis 1 er sú sama fyrir x_1 og x_2, nema bara hliðruð. Því er kerfið tímaóháð. Athugum einnig að svörun kerfis 1 við x_3 er sú sama og svörun kerfisins við x_1 samanlögð 2* svörun við x_2. Bæði skilyrði línuleika eru hér uppfyllt og þar með er kerfið línulegt.

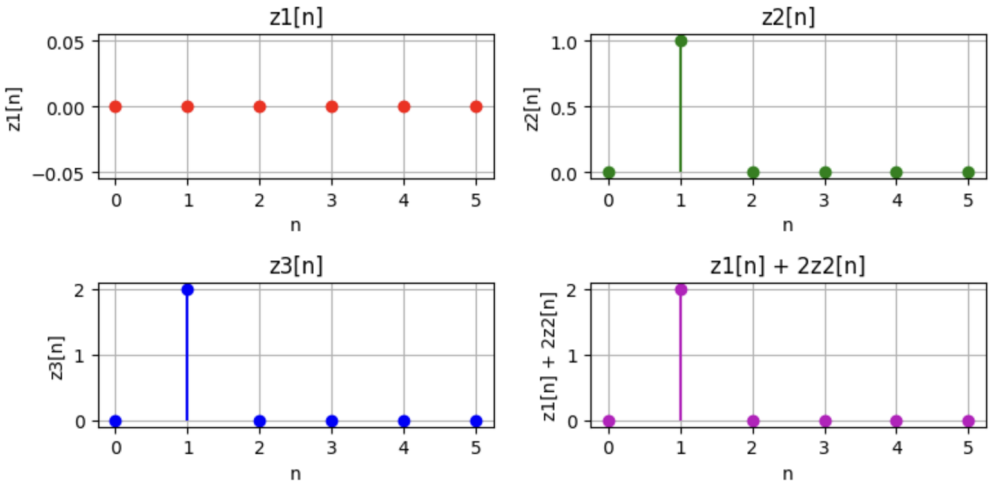
Skoðum nú kerfi 2 á sama hátt:



Mynd 5: Svörun kerfis 2 við innmerkjunum x_1, x_2 og x_3

Á mynd 5 má sjá að svörun kerfis 2 við x_3 er ekki sú sama og svörun kerfisins við x_1 samanlögð 2* svörun við x_2 og því er kerfið ekki línulegt. Það er aftur á móti ljóst að svörun kerfisins við hliðruðu innmerki er sama innmerki nema bara hliðrað, og hliðrunin hefur engin önnur áhrif. Þar með er kerfið tímaóháð.

Skoðum nú að lokum kerfi 3:



Mynd 6: Svörun kerfis 3 við innmerkjunum x_1, x_2 og x_3

Á mynd 6 má sjá að svörun kerfis 3 við x_3 er sú sama og svörun kerfisins við

x_1 samanlögð 2* svörun við x_2. Bæði skilyrði línuleika eru því uppfyllt, og kerfi 3 þar með línulegt. Athugum að aftur á móti hefur hliðrun í innmerki meiri áhrif heldur en bara hliðrun útmerkis, og því er kerfið tímaháð.

3 Dæmi 4

3.1 a-liður

Hér fáum við að $h[n] = \delta[n-a] + \delta[n-b]$ og $x[n] = \delta[n-c] + \delta[n-d]$.
Nú eigum við að finna hvar er $y[n]$ skilgreint sem fall af a, b, c og d ef að
$y[n] = h[n] * x[n]$

Til að byrja með er $h[n] * x[n] = \sum_{-\infty}^{\infty} h[k] \cdot x[n-k] \rightarrow$
$= \sum_{-\infty}^{\infty} (\delta[k-a] + \delta[k-b])(\delta[n-k-c] + \delta[n-k-d])$
Fáum þá→

$$\sum_{-\infty}^{\infty} \delta[k-a]\delta[n-k-c] + \delta[k-a]\delta[n-k-d] + \delta[k-b]\delta[n-k-c] + \delta[k-b]\delta[n-k-d]$$

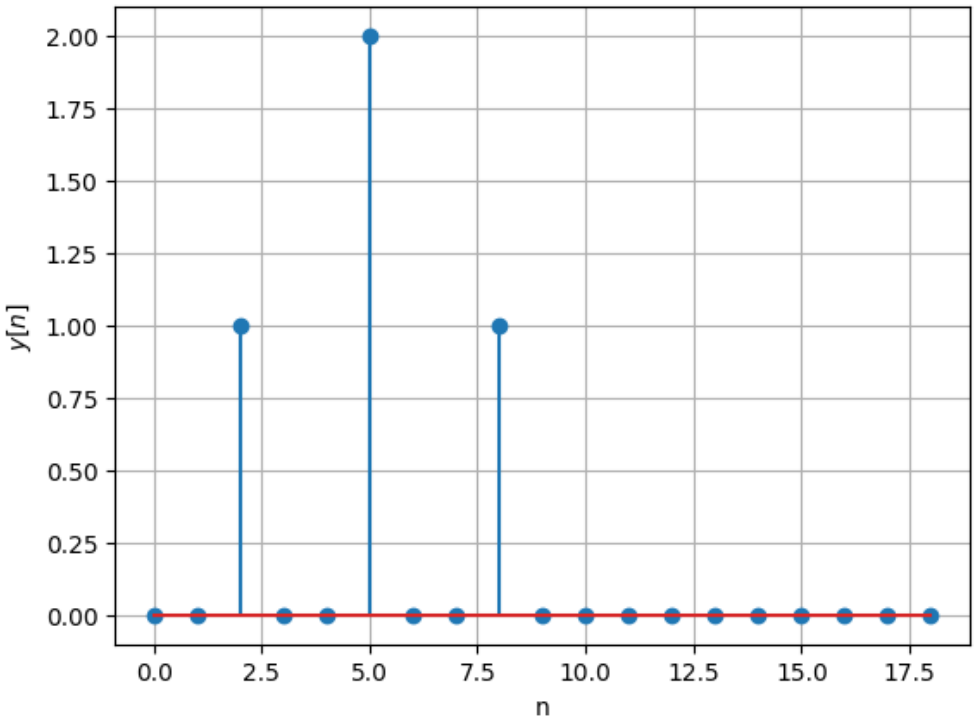
(Jafna [3.1])

Ef við skoðum td fyrsta liðin í jöfnu [3.1] ($\delta[k-a]\delta[n-k-c]$) þá sjáum við á fyrri hlutanum að hér þarf k=a til þess að y[n] taki gildi.
Seinni krafa á þessu merki er að $n-k-c = 0$ sem við getum breytt í $n-a-c = 0$ af því að það eru fyrstu kröfurnar okkar. umbreytum þeirri jöfnu í $n = a + c$
Ef við gerum þetta við alla liðina fáum við tilvikin þar sem $y[n]$ er skilgreint.

$$y[n] = \begin{cases} \delta, & n = a + c \\ \delta, & n = a + d \\ \delta, & n = b + c \\ \delta, & n = b + d \\ 0, & \text{annars} \end{cases}$$

þ.e.a.s. að ef við segjum að $h[n]$ byrjar í a og endar í b, og $x[n]$ byrjar í c og endar í d, þá er lægsta gildið sem kerfið getur tekið $a + c$ en hæsta þá $b + d$
Ef við skoðum Jöfnu [3.1] Þá sjáum við að hæsta fræðilega gildi sem að merkið getur tekið er þegar að 2 eða fleiri $\delta[n]$ virkjast á sama tíma. Ef við setjum $a = b = c = d$ þá ættu öll $\delta[n]$ að virkjast á sama tíma og þá mundum við fá $y[n] = 4\delta[n-r]$ þar sem er er td $a + d$. En fef við setjum þau gildi í numpy þá fáum við bara $1\delta[n-r]$. Hæsta gildi sem að fallið í numpy getur skilað af sér er þegar tvö $\delta[n]$ virkjast á sama tíma. Þetta getur gerst ef að $\delta[k-a]\delta[n-k-d] = \delta[k-b]\delta[n-k-c]$, semsagt þegar $a + d = b + c$.
Ef við setjum sem dæmi $a = 0$, $b = 3$, $c = 2$ og $d = 5$ þá fáum við merkin sem sjást á næstu síðu (Sjá Mynd [**??**])

Hér sjáum við þá að í $n = 5$ er $\delta[k-a]\delta[n-k-d] = \delta[k-b]\delta[n-k-c]$ bæði virkt, þá fáum við að $y[5] = \delta[5] + \delta[5] = 2\delta[5]$ sem er hæsta fræðilega

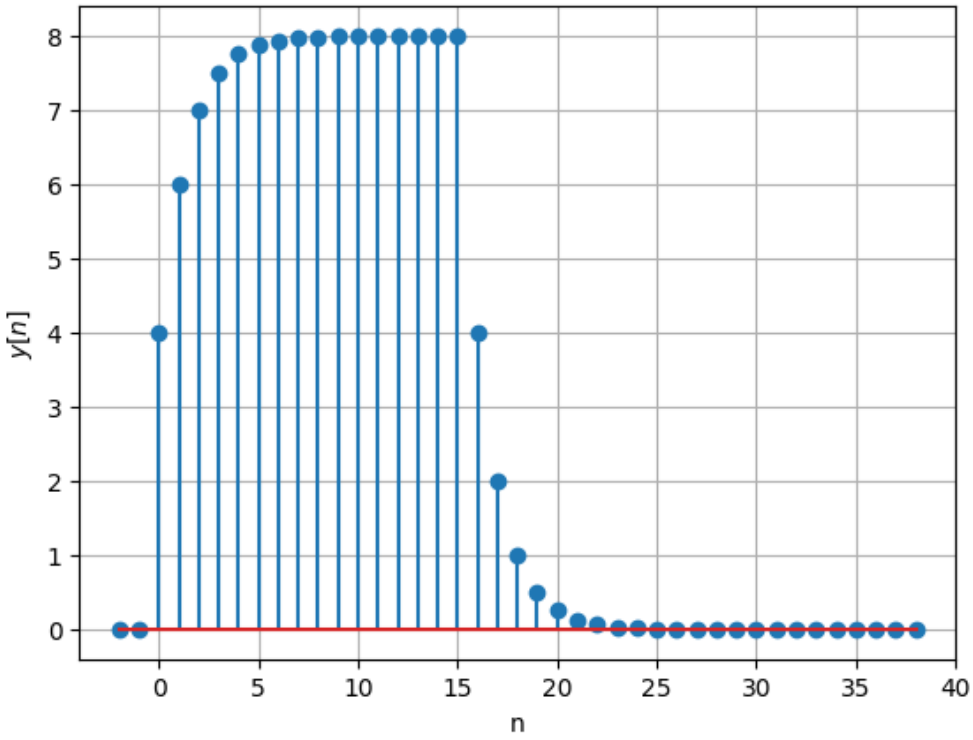


Mynd 7: Ath að á þessari mynd gildir að $\delta[n] = 1$ þegar það er virkt

gildið í numpy fyrir þetta innmerki og þessa impúlssvörun, en af því að $\delta[n]$ er óendanlega stórt þá er þetta bara fræðilegt (eins og td. að segja 2∞).

3.2 b-liður

Hér fáum við innmerki $x[n] = (\frac{1}{2})^{n-2}u[n-2]$ og impúlssvöruninna $h[n] = u[n+2]$ þar sem $x[n]$ er skilgreint á $0 < n < 24$ og $h[n]$ er skilgreint á $-2 < n < 14$.
Ef bið berum saman þetta kerfi við kerfið úr a-lið þá sjáum við að af því að fyrir $h[n]$ þá er $a = -2$ og $b = 14$, en fyrir $x[n]$ þá er $c = 0$ og $d = 24$. Ef við segjum eins og síðast, að $y[n]$ er skilgreint á bilinu $a + c = -2 + 0$ til $b + d = 14 + 24$ þá er $y[n]$ skilgreint frá -2 til 38.
Nú erum við með öll gögnin sem við þurfum til þess að setja upp mynd.
Eins og sést á Mynd [3.2] þá byrjar útmerkin frá kerfinu að hrynja eftir $n = 15$.



Mynd 8

Þetta gerist af því að $h[n]$ hlættir að vera skilgreint eftir $n = 14$ og þá hættir
Ef við skoðum hvað gerist þegar við fölldum þetta merki (án þess að taka tillit til takmarkanir á $h[n]$ eða $x[n]$) þá fáum við $y[n] = h[n] * x[n] \rightarrow$
$= (\frac{1}{2})^{n-2-k}u[n-2-k]u[k+2]$. Eini liðurinn hér sem hefur áhrif á kerfið þannig að $y[n] \neq 0$ eða $y[n] \neq 1$ er $(\frac{1}{2})^{n-2-k}$. Við sjáum þá að k hækkar og hækkar og loks þegar $h[n]$ hættir að vera skilgreint hættir k að vaxa og $y[n]$ hrynur niður þangað til að það er heldur ekki skilgreint.

4 Kóði

Mynd 1:

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


x1 = np.array([1, 1, 1, 1, 1])
h1 = np.array([1, -1, 3, 0, 1])


conv_x1_h1 = np.convolve(x1, h1, 'full')
conv_h1_x1 = np.convolve(h1, x1, 'full')


plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.stem(conv_x1_h1, use_line_collection=True)
plt.title(' x1[n] * h1[n]')
plt.xlabel('n')
plt.ylabel('x1[n]*h1[n]')

plt.subplot(1, 2, 2)
plt.stem(conv_h1_x1, use_line_collection=True)
plt.title(' h1[n] * x1[n]')
plt.xlabel('n')
plt.ylabel('h1[n]*x1[n]')

plt.tight_layout()
plt.show()
```

Mynd 2:

```
    h2 = np.array([2, 5, 4, -1])


conv_x1_h1 = np.convolve(x1, h1, 'full')
conv_x1_h2 = np.convolve(x1, h2, 'full')

#  x[n] * (h1[n] + h2[n])
h1_h2_sum = np.zeros(max(len(h1), len(h2)))
h1_h2_sum[:len(h1)] = h1
h1_h2_sum[:len(h2)] += h2

conv_x1_h1_h2_sum = np.convolve(x1, h1_h2_sum, 'full')

#  x[n] * h1[n] + x[n] * h2[n]
conv_x1_h1_h2_distributive = np.zeros(max(len(conv_x1_h1), len(conv_x1_h2)))
conv_x1_h1_h2_distributive[:len(conv_x1_h1)] = conv_x1_h1
conv_x1_h1_h2_distributive[:len(conv_x1_h2)] += conv_x1_h2



plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.stem(conv_x1_h1_h2_sum, use_line_collection=True)
```

```
plt.title('f ldun  x1[n] vi  (h1[n] + h2[n])')
plt.xlabel('n')
plt.ylabel('x1[n]*(h1[n]+h2[n])')

plt.subplot(1, 2, 2)
plt.stem(conv_x1_h1_h2_distributive, use_line_collection=True)
plt.title('f ldun x1[n] og h1[n] samanlagt vi  f ldun x1[n] og h2[n]')
plt.xlabel('n')
plt.ylabel('x1[n]*h1[n]+x1[n]*h2[n]')

plt.tight_layout()
plt.show()
```

Mynd 3:

```
    w = np.convolve(x1, h1, 'full')
yd1 = np.convolve(w, h2, 'full')
hh = np.convolve(h1,h2,'full')
yd2 = np.convolve(x1, hh, 'full')

n = np.linspace(0,len(w),len(w)-1)

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.stem(yd1, use_line_collection=True)
plt.title('(x1[n]*h1[n])*h2[n]')
plt.xlabel('n')
plt.ylabel('(x1[n]*h1[n])*h2[n]')

plt.subplot(1, 2, 2)
plt.stem(yd2, use_line_collection=True)
plt.title('x1[n]*(h1[n]*h2[n])')
plt.xlabel('n')
plt.ylabel('x1[n]*(h1[n]*h2[n])')

plt.tight_layout()
plt.show()
```

Mynd 4:

```
    def kerfi1(x):
    w = np.zeros_like(x)
    for n in range(len(x)):
        w[n] = x[n] - x[n-1] -x[n-2]
    return w


n = np.arange(0, 6)


delta = np.zeros_like(n)
delta[0] = 1


x1 = delta
x2 = np.roll(delta, 1)
x3 = delta + 2 * np.roll(delta, 1)


```

```
w1 = kerfi1(x1)
w2 = kerfi1(x2)
w3 = kerfi1(x3)
w_sum = w1 + 2 * w2


plt.figure(figsize=(8, 4))

plt.subplot(2, 2, 1)
plt.stem(n, w1, 'r', markerfmt='ro', basefmt=" ", linefmt='r')
plt.title('w1[n]')
plt.xlabel('n')
plt.ylabel('w1[n]')
plt.grid(True)

plt.subplot(2, 2, 2)
plt.stem(n, w2, 'g', markerfmt='go', basefmt=" ", linefmt='g')
plt.title('w2[n]')
plt.xlabel('n')
plt.ylabel('w2[n]')
plt.grid(True)

plt.subplot(2, 2, 3)
plt.stem(n, w3, 'b', markerfmt='bo', basefmt=" ", linefmt='b')
plt.title('w3[n]')
plt.xlabel('n')
plt.yticks([-2, -1, 0, 1,2], ['-2', '-1', '0', '1','2'])
plt.ylabel('w3[n]')
plt.ylim(top=2)
plt.grid(True)

plt.subplot(2, 2, 4)
plt.stem(n, w_sum, 'm', markerfmt='mo', basefmt=" ", linefmt='m')
plt.title('w1[n] + 2w2[n]')
plt.xlabel('n')
plt.ylim(top=2)
plt.yticks([-2, -1, 0, 1,2], ['-2', '-1', '0', '1','2'])
plt.ylabel('w1[n] + 2w2[n]')
plt.grid(True)

plt.tight_layout()
plt.show()
```

Mynd 5:

```
    def kerfi2(x):
    y = np.zeros_like(x)
    for n in range(len(x)):
        y[n] = np.cos(x[n])
    return y




y1 = kerfi2(x1)
y2 = kerfi2(x2)
y3 = kerfi2(x3)
y_sum = y1 + 2 * y2

```

```
plt.figure(figsize=(8, 4))

plt.subplot(2, 2, 1)
plt.stem(n, y1, 'r', markerfmt='ro', basefmt=" ", linefmt='r')
plt.title('y1[n]')
plt.xlabel('n')
plt.ylabel('y1[n]')
plt.grid(True)

plt.subplot(2, 2, 2)
plt.stem(n, y2, 'g', markerfmt='go', basefmt=" ", linefmt='g')
plt.title('y2[n]')
plt.xlabel('n')
plt.ylabel('y2[n]')
plt.grid(True)

plt.subplot(2, 2, 3)
plt.stem(n, y3, 'b', markerfmt='bo', basefmt=" ", linefmt='b')
plt.title('y3[n]')
plt.xlabel('n')
plt.ylabel('y3[n]')
plt.grid(True)

plt.subplot(2, 2, 4)
plt.stem(n, y_sum, 'm', markerfmt='mo', basefmt=" ", linefmt='m')
plt.title('y1[n] + 2y2[n]')
plt.xlabel('n')
plt.ylabel('y1[n] + 2y2[n]')
plt.grid(True)

plt.tight_layout()
plt.show()
```

Mynd 6:

```
    def kerfi3(x):
    z = np.zeros_like(x)
    for n in range(len(x)):
        z[n] = n*(x[n])
    return z


z1 = kerfi3(x1)
z2 = kerfi3(x2)
z3 = kerfi3(x3)
z_sum = z1 + 2 * z2


plt.figure(figsize=(8, 4))

plt.subplot(2, 2, 1)
plt.stem(n, z1, 'r', markerfmt='ro', basefmt=" ", linefmt='r')
plt.title('z1[n]')
plt.xlabel('n')
plt.ylabel('z1[n]')
plt.grid(True)

plt.subplot(2, 2, 2)
```

```
plt.stem(n, z2, 'g', markerfmt='go', basefmt=" ", linefmt='g')
plt.title('z2[n]')
plt.xlabel('n')
plt.ylabel('z2[n]')
plt.grid(True)

plt.subplot(2, 2, 3)
plt.stem(n, z3, 'b', markerfmt='bo', basefmt=" ", linefmt='b')
plt.title('z3[n]')
plt.xlabel('n')
plt.ylabel('z3[n]')
plt.grid(True)

plt.subplot(2, 2, 4)
plt.stem(n, z_sum, 'm', markerfmt='mo', basefmt=" ", linefmt='m')
plt.title('z1[n] + 2z2[n]')
plt.xlabel('n')
plt.ylabel('z1[n] + 2z2[n]')
plt.grid(True)

plt.tight_layout()
plt.show()
```

Mynd 7

```
    h = ([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
x = ([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
n = np.arange(0, 19)

a = 0
b = 3

c = 2
d = 5

x[a] = 1
x[b] = 1
h[c] = 1
h[d] = 1

a = np.convolve(x, h,'full')
plt.figure()
plt.stem(n, a)
plt.xlabel('n')
plt.ylabel('$y[n]$')
plt.grid()
```

Mynd 8

```
    n = np.arange(-2, 39)
l = len(n)
x = np.zeros(41)
h = np.zeros(41)


x[(n >= 0) & (n < 25)] = np.power(1/2, n[(n >= 0) & (n < 25)] - 2)
h[(n >= -2) & (n < 14)] = 1

a = np.convolve(x, h, 'full')

```

```
12 plt.figure()
13 plt.stem(n, a[:l])
14 plt.xlabel('n')
15 plt.ylabel('$y[n]$')
16 plt.grid()
17 plt.show()
```