In [5]:
```
import Pkg
```

In [6]:
```
Pkg.add("GR")
```

```
 Resolving package versions...
  Updating `~/Project.toml`
 [no changes]
  Updating `~/Manifest.toml`
 [no changes]
```

In [7]:
```
using GR
```

In [56]:
```
############################ -> P1.1 <- ########################################
###############
## Produza as seguintes séries numéricas (vetores) em Julia utilizando a função ran
ge (ou :):,
## (a) {-3,-1,1,…,25};
println(collect(-3:2:25))
```

```
[-3, -1, 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25]
```

In [57]:
```
## (b) {100,90,80,…,-100};
println(collect(100:-10:-100))
```

```
[100, 90, 80, 70, 60, 50, 40, 30, 20, 10, 0, -10, -20, -30, -40, -50, -60, -70,
-80, -90, -100]
```

In [58]:
```
## (c) {10^-3,0.01,0.1,…,10^12};
println(10.0.^collect(-3:12))
```

```
[0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0, 10000.0, 100000.0, 1.0e6, 1.0e7, 1.
0e8, 1.0e9, 1.0e10, 1.0e11, 1.0e12]
```

In [59]:
```
## (d) {12,…,2,1,0,1,2,…,12};
println([collect(12:-1:0);collect(1:12)])
## ou
println(abs.(collect(-12:12)))
```

```
[12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 1
2]
[12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 1
2]
```

In [60]:
```
## (e) {1+0.5j,2+j,3+1.5j,…,14+7j};
println(collect(1:14) .+ collect(0.5:0.5:7).*im)
```

```
Complex{Float64}[1.0 + 0.5im, 2.0 + 1.0im, 3.0 + 1.5im, 4.0 + 2.0im, 5.0 + 2.5i
m, 6.0 + 3.0im, 7.0 + 3.5im, 8.0 + 4.0im, 9.0 + 4.5im, 10.0 + 5.0im, 11.0 + 5.5i
m, 12.0 + 6.0im, 13.0 + 6.5im, 14.0 + 7.0im]
```

In [61]:
```
## (f) {sin(𝑥):𝑥=-π,-9π/10,8π/10,…,π/2};
## 9π/10 - 8π/10 = π/10
println(sin.(collect(-π:π/10:π/2)))
```

```
[-1.2246467991473532e-16, -0.3090169943749475, -0.5877852522924732, -0.809016994
3749475, -0.9510565162951536, -1.0, -0.9510565162951535, -0.8090169943749475, -
0.5877852522924731, -0.3090169943749474, 0.0, 0.3090169943749474, 0.587785252292
4731, 0.8090169943749475, 0.9510565162951535, 1.0]
```

In [62]:
```
## (g) {e^jn: n = 0,π/8,1π/4,…,3π/2};
## π/8 - 1π/4 = -π/8
print(exp.(im .* collect(0:π/8:3π/2)))
```

```
Complex{Float64}[1.0 + 0.0im, 0.9238795325112867 + 0.3826834323650898im, 0.70710
67811865476 + 0.7071067811865475im, 0.38268343236508984 + 0.9238795325112867im,
6.123233995736766e-17 + 1.0im, -0.3826834323650897 + 0.9238795325112867im, -0.70
71067811865475 + 0.7071067811865476im, -0.9238795325112867 + 0.3826834323650899i
m, -1.0 + 1.2246467991473532e-16im, -0.9238795325112868 - 0.38268343236508967im,
-0.7071067811865477 - 0.7071067811865475im, -0.38268343236509034 - 0.92387953251
12865im, -1.8369701987210297e-16 - 1.0im]
```

In [63]:
```
## (h) {sinx2-cos2x:x=0,0.1,0.2,..,10}.
Xh = collect(0:0.1:10)
println(sin.(Xh.^2) .- cos.(2 .* Xh))
```

```
[-1.0, -0.970066744507075, -0.8810716598162509, -0.7354570657116672, -0.53738850
27329194, -0.2928983466136168, -0.010083521201583678, 0.30065874527091696, 0.626
394963663681, 0.9514892690632297, 1.2576178213550389, 1.5241171188087317, 1.7288
520637329319, 1.8497924044630658, 1.8674338614568264, 1.7680656934883667, 1.5476
502122218794, 1.215744979252614, 0.7985098225890384, 0.3395019597529937, -0.1031
5887444431626, -0.46436695031951697, -0.6845358873324932, -0.7256169532300434, -
0.5871408665563489, -0.31684140201078304, -0.009565184922686043, 0.2104405357145
831, 0.2243363800377256, -0.03615613843585164, -0.5480518014086093, -1.180706876
4238926, -1.7210627891079304, -1.9446648012617247, -1.7142934341258522, -1.06502
1609324432, -0.2248085591196446, 0.4631284424920014, 0.7032355876586658, 0.42368
17243513645, -0.14240328285645176, -0.5529745037105414, -0.41617048687497793, 0.
3268614603878389, 1.299657779834177, 1.8966553734497966, 1.7135496511079389, 0.9
010025280254967, 0.1178366999678152, 0.029134908016131167, 0.7067197789786794,
1.4832550603059476, 1.5049127506456565, 0.5686299323120627, -0.5800067542865854,
-0.9235793921916281, -0.2589022499830838, 0.4856581235974877, 0.2258066185645253
3, -0.9702393625874206, -1.835632812175608, -1.4034756966923365, -0.311248780721
2674, -0.0863830937330069, -1.0918453129614671, -1.8944338445629074, -1.21574083
54943537, 0.11589870810524938, 0.2615074423142134, -0.7980051126120657, -1.09048
98709673054, 0.20680674135378357, 1.259810432498348, 0.5632927357426305, -0.3610
0727056106185, 0.46544070114731584, 1.8098224590733596, 1.3426111070401379, 0.08
152009343939992, 0.5863158375187056, 1.8776855185201753, 1.2368694694215483, -0.
18412632457588418, 0.4046842616798707, 1.4527718422846359, 0.28179432202187604,
-0.9125469159728085, 0.166710315013039, 0.5753802825325527, -1.1190861362380677,
-1.2902047025185341, 0.10745766071275675, -0.7186582585162181, -1.96439834573616
9, -0.6135897402258657, -0.23335533252406404, -1.808534523071684, -1.00941810768
58821, 0.2441995909687813, -1.162867760855728, -0.9144477029231508]
```

In [16]:
```
############################ -> P1.1.2 <- ########################################
################
## Produza as séries numéricas (vetores) de P1.1 em Julia utilizando a criação de v
etores com expressões (comprehension).
## (a) {-3,-1,1,…,25};
println([x for x in -3:2:25])
```

```
[-3, -1, 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25]
```

In [17]:
```
## (b) {100,90,80,…,-100};
println([x for x in 100:-10:-100])
```

```
[100, 90, 80, 70, 60, 50, 40, 30, 20, 10, 0, -10, -20, -30, -40, -50, -60, -70,
-80, -90, -100]
```

In [18]:
```
## (c) {10^-3,0.01,0.1,…,10^12};
println(10.0.^[x for x in -3:12])
```

```
[0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0, 10000.0, 100000.0, 1.0e6, 1.0e7, 1.
0e8, 1.0e9, 1.0e10, 1.0e11, 1.0e12]
```

In [19]:
```
## (d) {12,…,2,1,0,1,2,…,12};
println([[x for x in 12:-1:0]; [x for x in 1:12]])
```

```
[12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 1
2]
```

In [20]:
```
## (e) {1+0.5j,2+j,3+1.5j,…,14+7j};
println([x for x in 1:14] .+ ([x for x in 0.5:0.5:7].*im))
```

```
Complex{Float64}[1.0 + 0.5im, 2.0 + 1.0im, 3.0 + 1.5im, 4.0 + 2.0im, 5.0 + 2.5i
m, 6.0 + 3.0im, 7.0 + 3.5im, 8.0 + 4.0im, 9.0 + 4.5im, 10.0 + 5.0im, 11.0 + 5.5i
m, 12.0 + 6.0im, 13.0 + 6.5im, 14.0 + 7.0im]
```

In [21]:
```
## (f) {sin(𝑥):𝑥=-π,-9π/10,8π/10,…,π/2};
## 9π/10 - 8π/10 = π/10
println(sin.([x for x in -π:π/10:π/2]))
```

```
[-1.2246467991473532e-16, -0.3090169943749475, -0.5877852522924732, -0.809016994
3749475, -0.9510565162951536, -1.0, -0.9510565162951535, -0.8090169943749475, -
0.5877852522924731, -0.3090169943749474, 0.0, 0.3090169943749474, 0.587785252292
4731, 0.8090169943749475, 0.9510565162951535, 1.0]
```

In [22]:
```
## (g) {e^j𝑛: 𝑛 = 0,π/8,1π/4,…,3π/2};
## π/8 - 1π/4 = -π/8
println(exp.([x for x in 0:π/8:3π/2]))
```

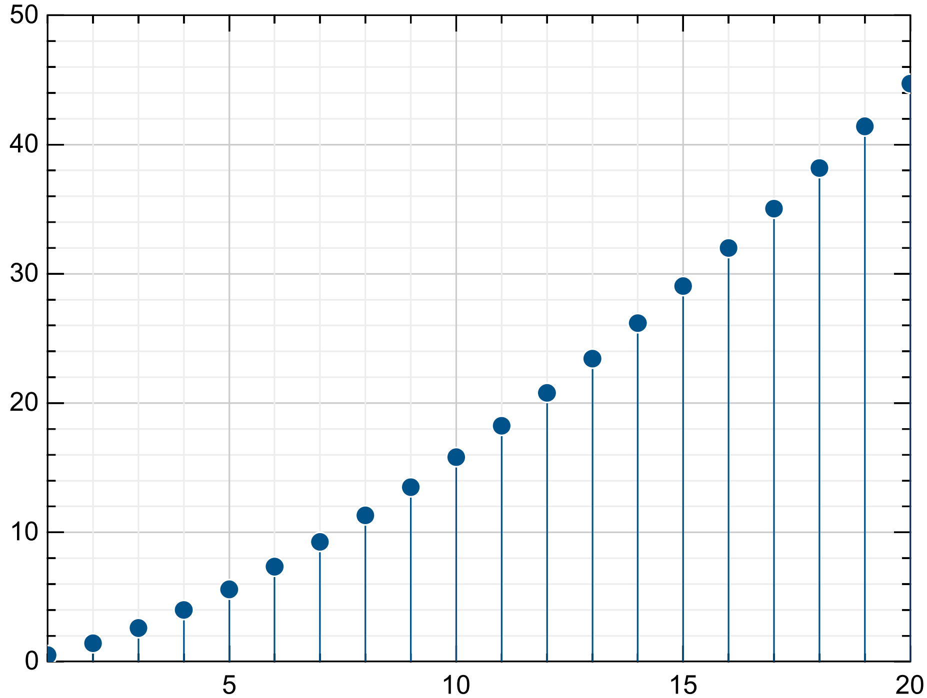
```
[1.0, 1.48097267048991, 2.1932800507380152, 3.2481878138737237, 4.81047738096535
1, 7.124185533219564, 10.550724074197761, 15.625334007766842, 23.14069263277926
7, 34.270733365353294, 50.754019511734924, 75.16531581439104, 111.3177784898562
1]
```

In [23]:
```
## (h) {sinx2-cos2x:x=0,0.1,0.2,..,10}.
println([(sin.(Xh.^2) .- cos.(2 .* Xh)) for Xh in 0:0.1:10])
```

```
[-1.0, -0.970066744507075, -0.8810716598162509, -0.7354570657116672, -0.53738850
27329194, -0.2928983466136168, -0.010083521201583678, 0.30065874527091696, 0.626
394963663681, 0.9514892690632297, 1.2576178213550389, 1.5241171188087317, 1.7288
520637329319, 1.8497924044630658, 1.8674338614568264, 1.7680656934883667, 1.5476
502122218794, 1.215744979252614, 0.7985098225890384, 0.3395019597529937, -0.1031
5887444431626, -0.46436695031951697, -0.6845358873324932, -0.7256169532300434, -
0.5871408665563489, -0.31684140201078304, -0.009565184922686043, 0.2104405357145
831, 0.2243363800377256, -0.03615613843585164, -0.5480518014086093, -1.180706876
4238926, -1.7210627891079304, -1.9446648012617247, -1.7142934341258522, -1.06502
1609324432, -0.2248085591196446, 0.4631284424920014, 0.7032355876586658, 0.42368
17243513645, -0.14240328285645176, -0.5529745037105414, -0.41617048687497793, 0.
3268614603878389, 1.299657779834177, 1.8966553734497966, 1.7135496511079389, 0.9
010025280254967, 0.1178366999678152, 0.029134908016131167, 0.7067197789786794,
1.4832550603059476, 1.5049127506456565, 0.5686299323120627, -0.5800067542865854,
-0.9235793921916281, -0.2589022499830838, 0.4856581235974877, 0.2258066185645253
3, -0.9702393625874206, -1.835632812175608, -1.4034756966923365, -0.311248780721
2674, -0.0863830937330069, -1.0918453129614671, -1.8944338445629074, -1.21574083
54943537, 0.11589870810524938, 0.2615074423142134, -0.7980051126120657, -1.09048
98709673054, 0.20680674135378357, 1.259810432498348, 0.5632927357426305, -0.3610
0727056106185, 0.46544070114731584, 1.8098224590733596, 1.3426111070401379, 0.08
152009343939992, 0.5863158375187056, 1.8776855185201753, 1.2368694694215483, -0.
18412632457588418, 0.4046842616798707, 1.4527718422846359, 0.28179432202187604,
-0.9125469159728085, 0.166710315013039, 0.5753802825325527, -1.1190861362380677,
-1.2902047025185341, 0.10745766071275675, -0.7186582585162181, -1.96439834573616
9, -0.6135897402258657, -0.23335533252406404, -1.808534523071684, -1.00941810768
58821, 0.2441995909687813, -1.162867760855728, -0.9144477029231508]
```

In [24]:
```
############################ -> P1.2 <- #########################################
###############
## Produza os gráficos das seguintes séries numéricas em Julia utilizando a função
stem:
## (a) {(n^1.5)/2 - 1: n=1,2,3,…,20};
## stem(n, y)
stem(collect(1:20), (collect(1:20)).^1.5 / 2)
```

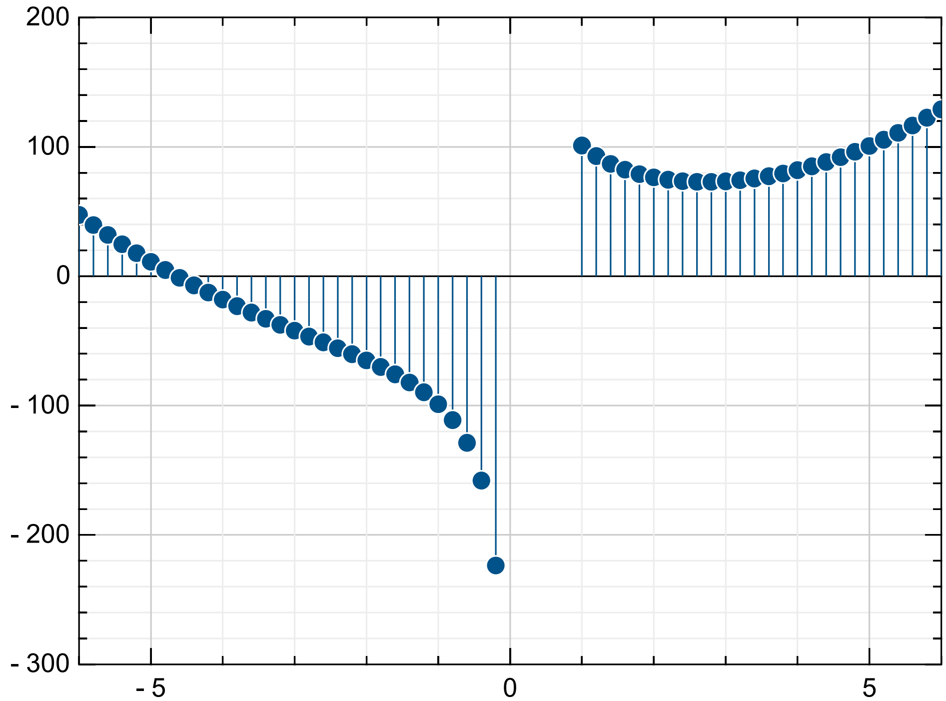
Out[24]:



In [25]:

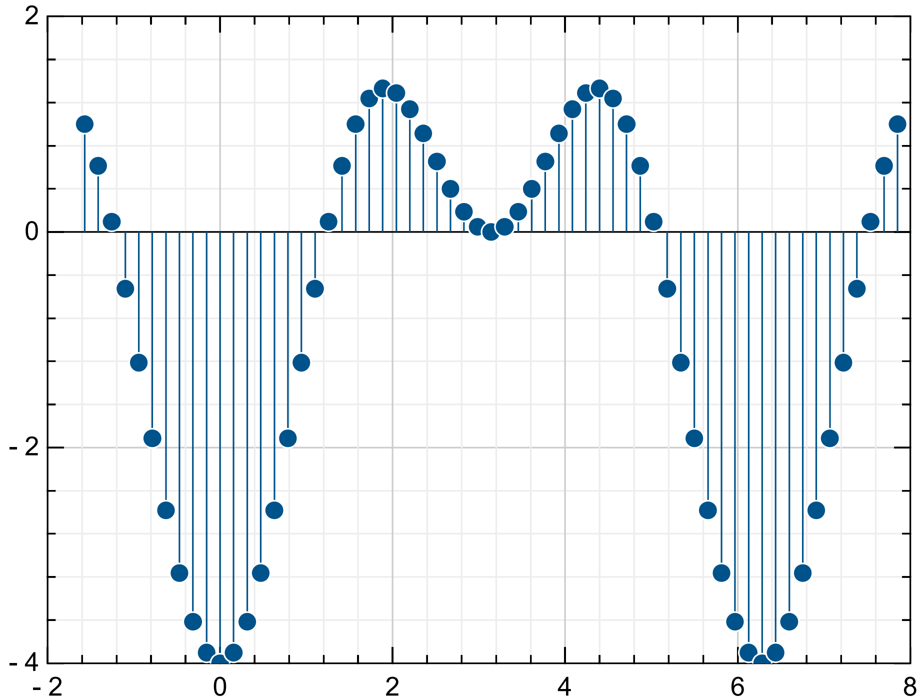
```
## (b) {√|a|(a^2+100/a):a=-6,-5.8,-5.6,…,6};
nb = [collect(-6.0:0.2:0); collect(1:0.2:6)]
stem(nb, (sqrt.(abs.(nb)) .* (nb.^2.0 .+ 100.0./nb)))
```

Out[25]:



In [26]:
```
## (c) {3sin^2x - 2cosx - 2: x = -π/2,-9π/20,-8π/20,…,5π/2};
## π/2 - 9π/20 = π/20
nc = collect(-π/2:π/20:5π/2)
stem(nc, (3.0.*sin.(nc).^2.0 .- 2.0.*cos.(nc) .- 2.0))
```

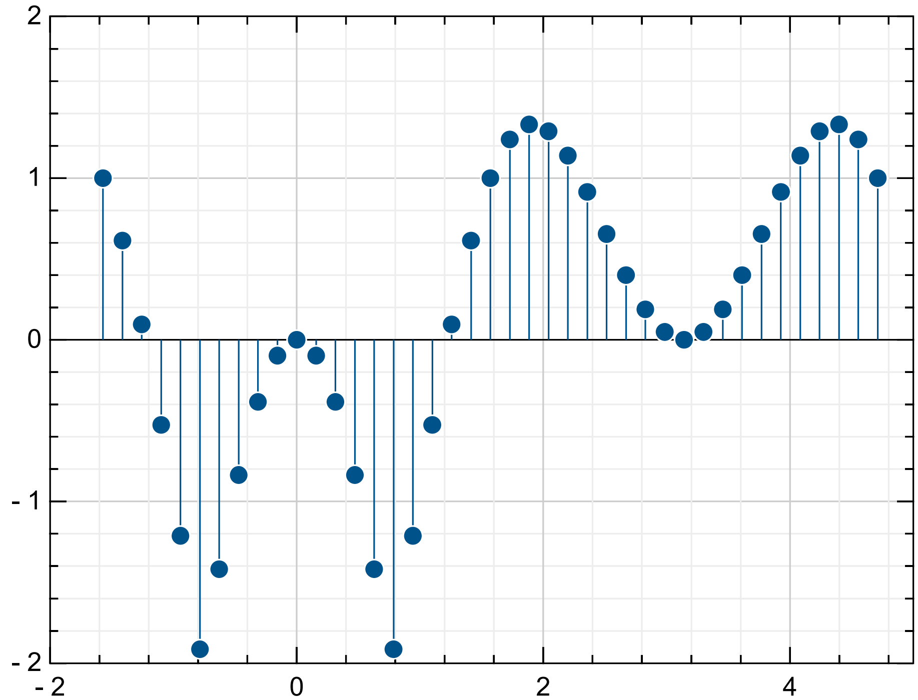
Out[26]:



In [27]:

```
## (d) {|3sin^2𝑥 - 2cos𝑥| - 2 :𝑥 = -π/2,-9π/20,-4π/10,…,3π/2};
## π/2 - 9π/20 = π/20
nd = collect(-π/2:π/20:3π/2)
stem(nd, (abs.(3.0.*sin.(nd).^2.0 .- 2.0.*cos.(nd)) .- 2))
```

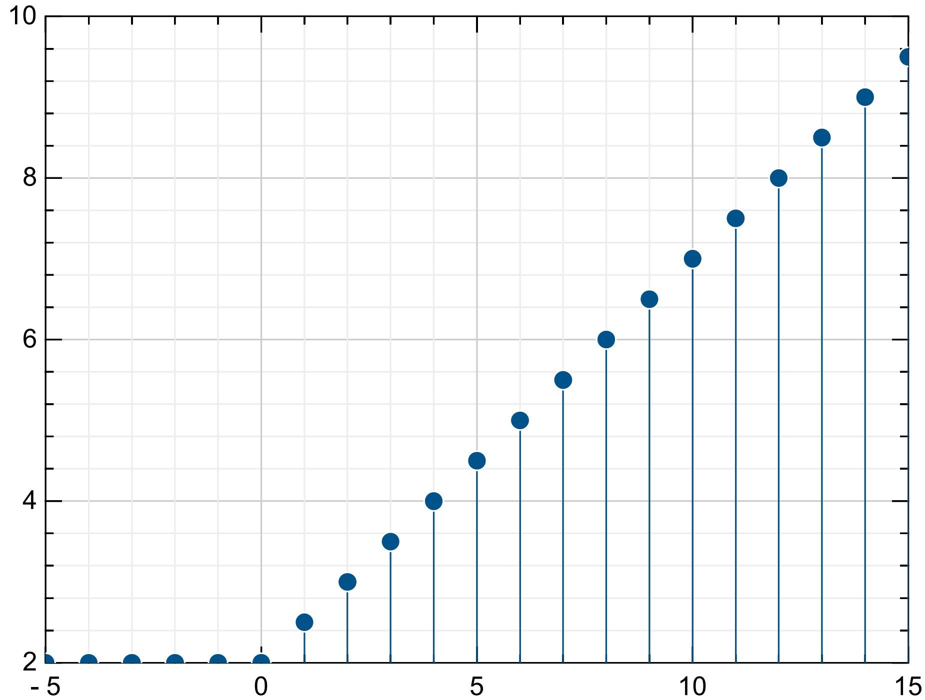
Out[27]:



In [28]:

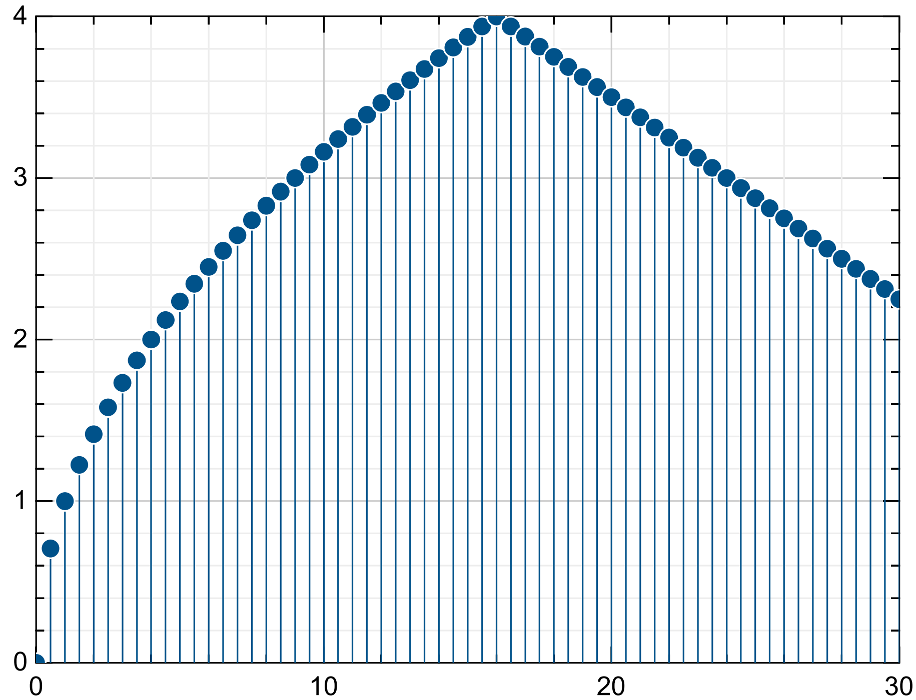
```
## (e) {{    2,      𝑥 < 0
##      {{0.5𝑥 + 2, 𝑥 ≥ 0: 𝑥 = −5,−4,−3,…,15};
ne = collect(-5:15)
stem(ne, ((ne -> ne < 0 ? 2.0 : 0.5*ne + 2).(ne)))
```

Out[28]:



In [29]:
```
## (f) {{ √(x),     x ≤ 16
##     {{ -x/8 + 6, x > 16 :x = 0, 0.5, 1, …, 30}.
nf = collect(0.0:0.5:30.0)
stem(nf, ((nf -> nf <= 16.0 ? sqrt.(nf) : -nf/8 + 6).(nf)))
```

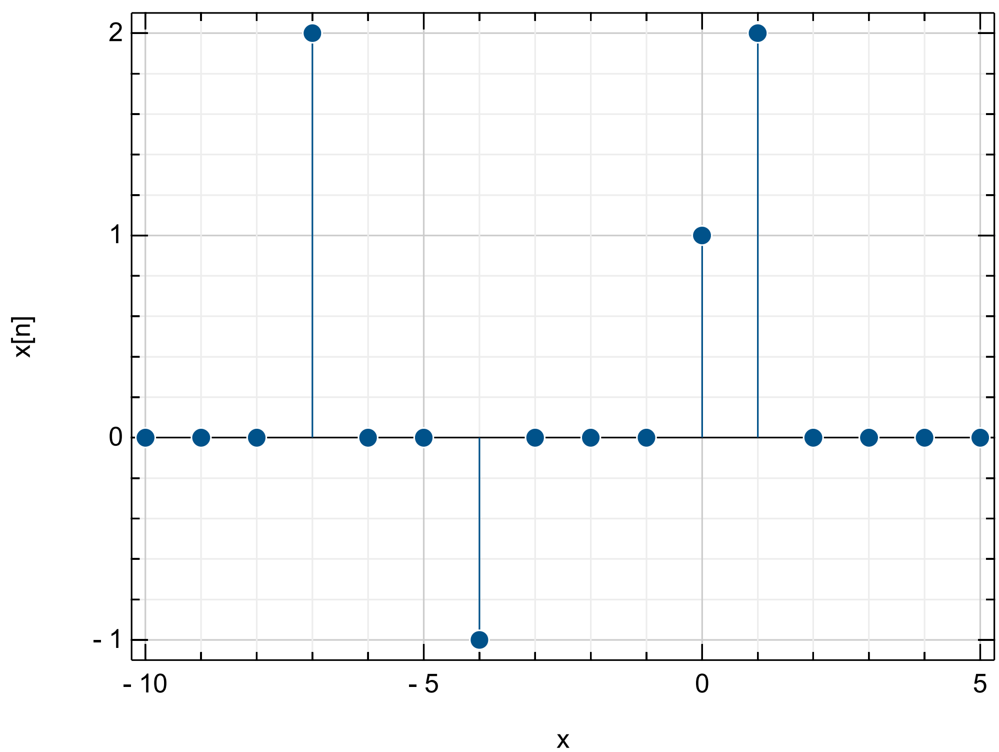
Out[29]:



In [30]:

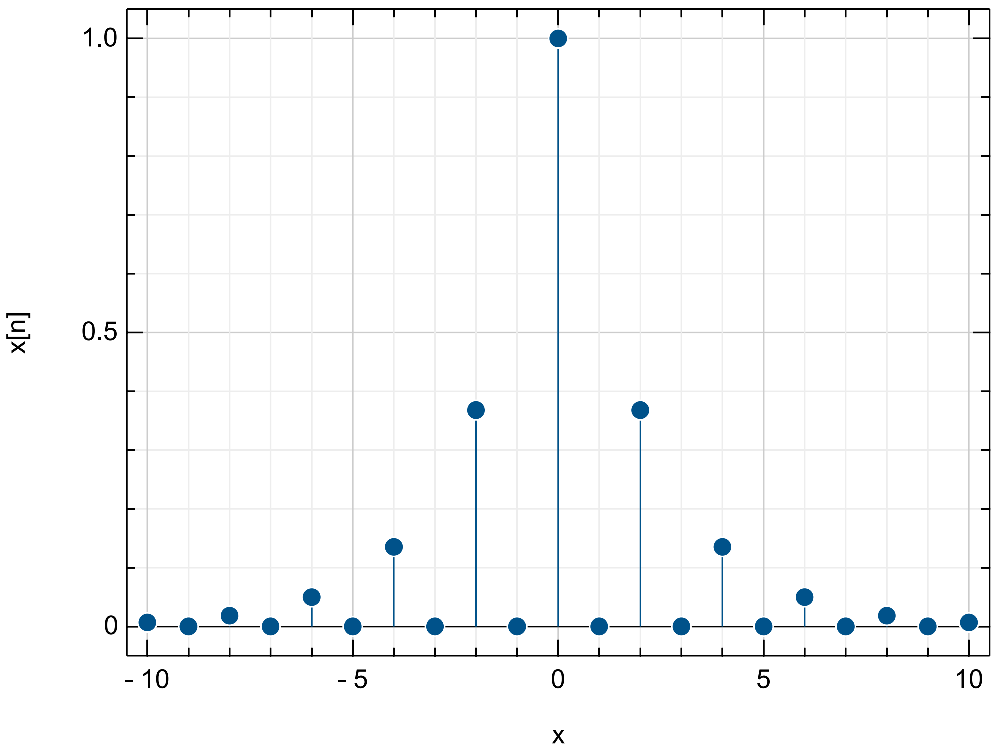
```
############################ -> P2.2 <- ##########################################
###############
## Gere as seguintes séries numérica e produza os respetivos gráficos em Julia:
xlabel("x")
ylabel("x[n]")
## importante
u(x) = x >= 0 ? 1.0 : 0.0;
δ(x) = x == 0 ? 1.0 : 0.0;

## (a) x[n] = 2δ[n − 1] + δ[n] − δ[n + 4] + 2δ[n + 7], −10 ≤ n ≤ 5;
na = collect(-10:5)
xa = 2 .* δ.(na .- 1) .+ δ.(na) .- δ.(na .+ 4) .+ 2 .* δ.(na .+ 7)
xlim([-10.25, 5.25])
ylim([-1.1, 2.1])
stem(na, xa)
```

Out[30]:



In [31]:
```
## (b) x[n] = ∑ e−|k| δ[n − 2k] ,−10 ≤ n ≤ 10;
## sum(k = −5 −> 5)
nb = collect(-10:10);
xb = [sum([exp(-abs(k)) .* δ.(nb .- 2 * k) for k = -5:5]) for nb = -10:10]
xlim([-10.5, 10.5])
ylim([-0.05, 1.05])
stem(nb, xb)
```

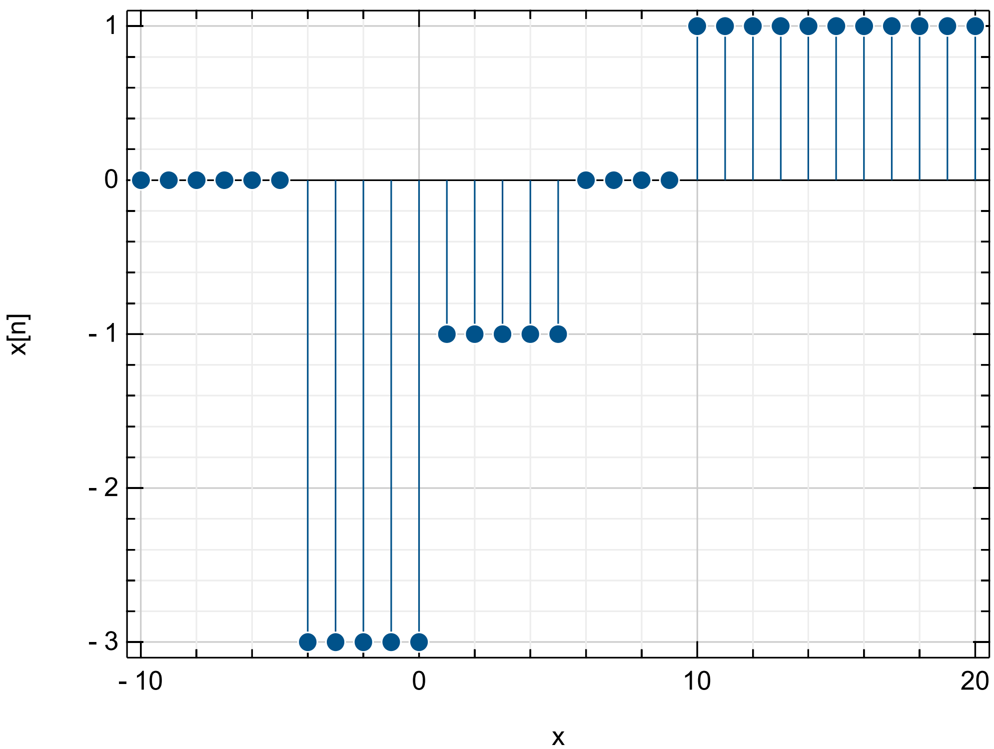
Out[31]:



In [32]:

```
## (c) x[n] = 3u[-n-5] - 2u[-n] - u[-n+5] + u[n-10], -10 ≤ n ≤ 20;
## -10 ≤ n ≤ 20
nc = collect(-10.0:1.0:20.0)
xc = (3.0.*u.(-nc .- 5.0)) .- (2.0.*u.(-nc)) .- (u.(-nc .+ 5.0)) .+ (u.(nc .- 10.
0))
xlim([-10.5, 20.5]);
ylim([-3.1, 1.1]);
stem(nc, xc)
```

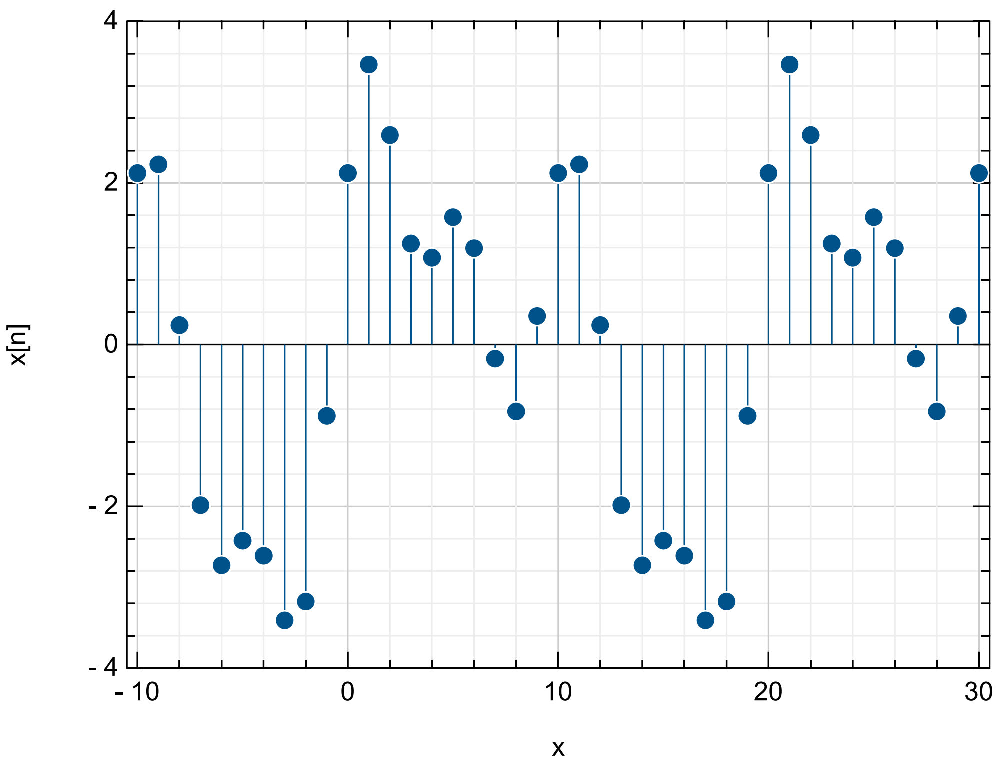
Out[32]:



In [33]:

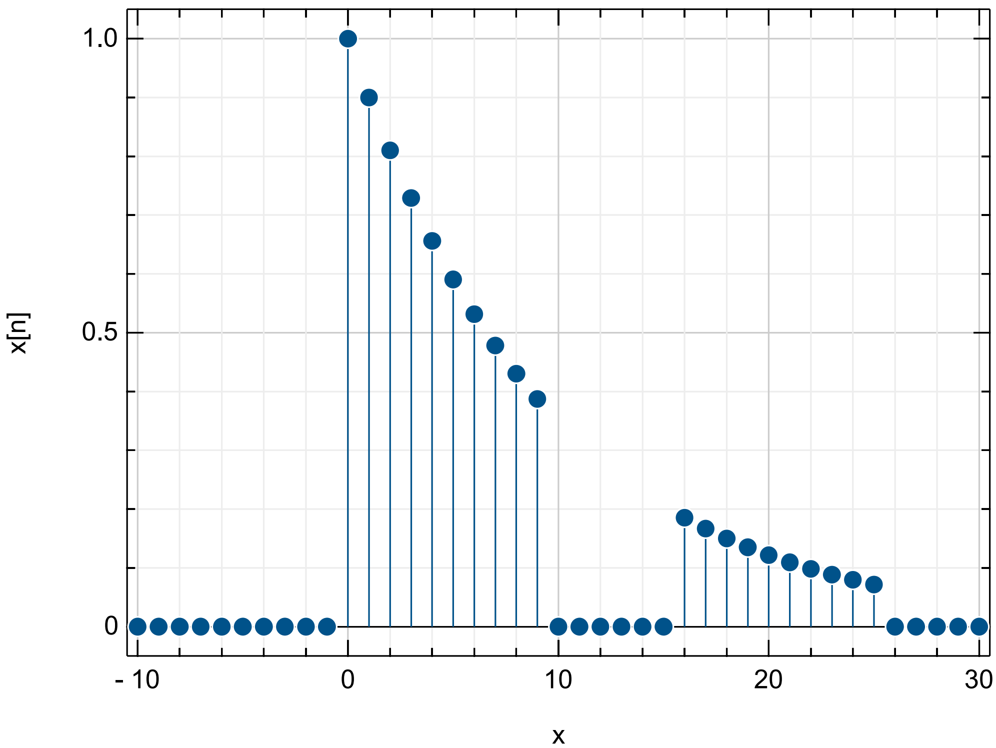
```
## (d) x[n] = 2sin(0.1πn) + 1.8sin(0.2πn + 0.25π) + 1.2cos(0.4πn-0.25π), -10 ≤ n ≤ 30;
## -10 ≤ n ≤ 30
nd = collect(-10.0:1.0:30.0)
xd = (2.0.*sin.(0.1.*π.*nd)) .+ (1.8.*sin.(0.2.*π.*nd .+ 0.25.*π)) .+ (1.2.*cos.(0.4.*π.*nd .- 0.25.*π))
xlim([-10.5, 30.5]);
ylim([-4.0, 4.0]);
stem(nd, xd)
```

Out[33]:



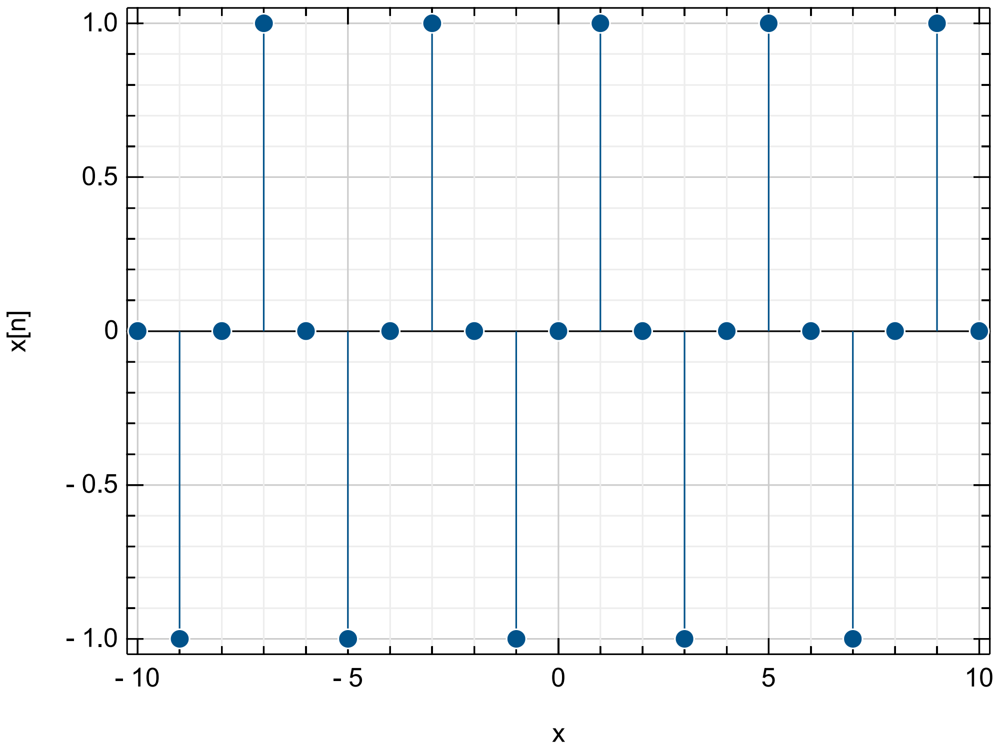
In [34]:
```
## (e) x[n] = 0.9^n(u[n] - u[n-10] + u[-n+25] - u[-n+15]),
## -10 ≤ n ≤ 30
ne = collect(-10.0:1.0:30.0)
xe = 0.9.^ne.*((u.(ne)) .- (u.(ne.-10)) .+ (u.(-ne.+25)) .- (u.(-ne.+15)))
xlim([-10.5, 30.5]);
ylim([-0.05, 1.05]);
stem(ne, xe)
```

Out[34]:



In [35]:
```
############################ -> P2.3 <- ########################################
###############
## Produza os gráficos das seguintes séries periódicas
## (a) x[n] = {…, 0, −1, 0↑, 1, 0, −1, 0, 1, …}, −10 ≤ n ≤ 10;
n = collect(-10:10)
## m1 -> onde começa a amostra de p (zero)
m1 = 0
# amostra
p = [0, 1, 0, -1]
## importante
## formula => x = p[mod.(n .- m1, length(p)) .+ 1]
x = p[mod.(n .- m1, length(p)) .+ 1]
xlim([-10.25, 10.25]);
ylim([-1.05, 1.05]);
stem(n, x)
```

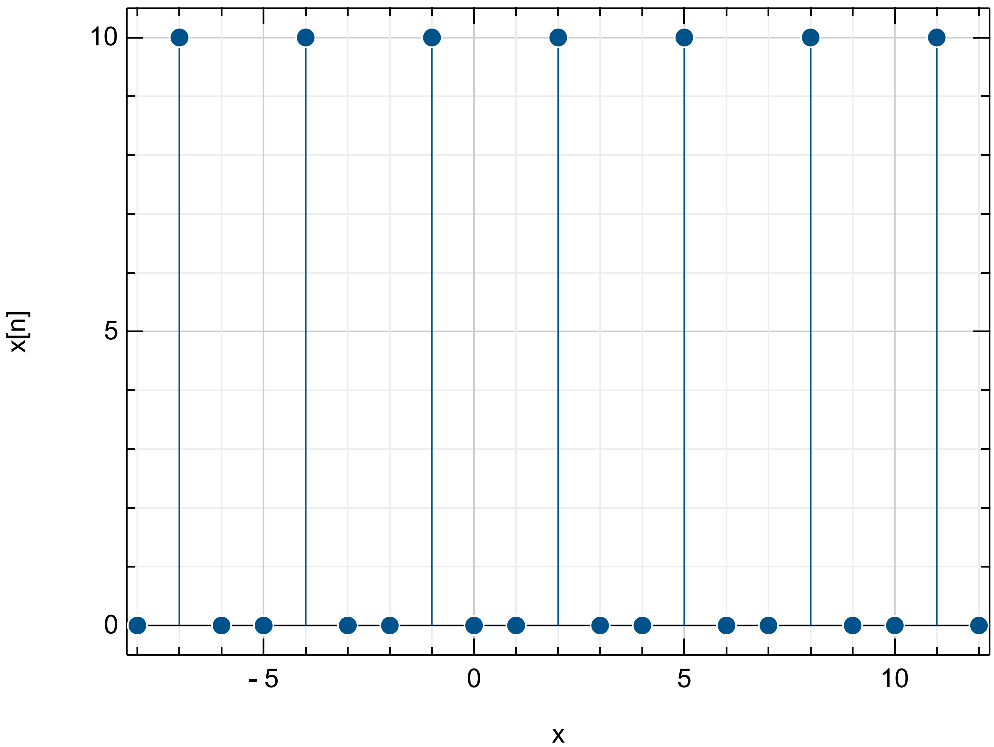
Out[35]:



In [36]:

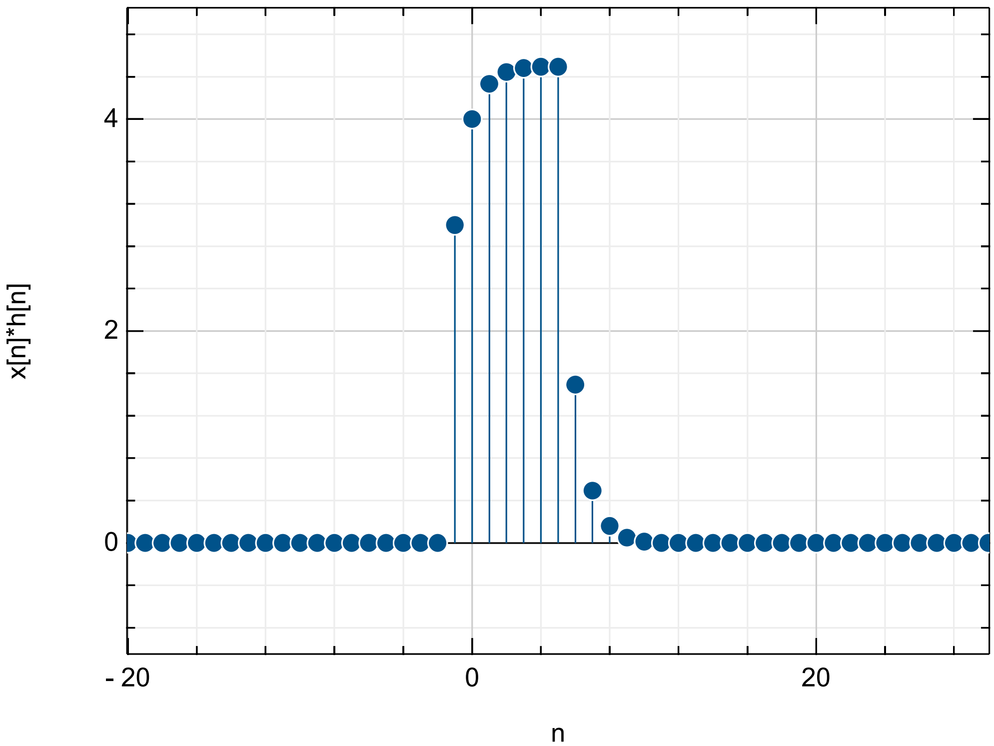
```
## (b) x[n] = {…,x[38],x[39],x[40],x[41],x[42],x[43],x[44],…} = {…,10,0,0,10,0,0,1
0,…}, -8 ≤ x ≤ 12
n = collect(-8:12)
p = [10,0,0]
## m1 -> onde começa a amostra de p (indice)
m1 = 38
x = p[mod.(n .- m1, length(p)) .+ 1]
xlim([-8.25, 12.25]);
ylim([-0.5, 10.5]);
stem(n, x)
```

Out[36]:



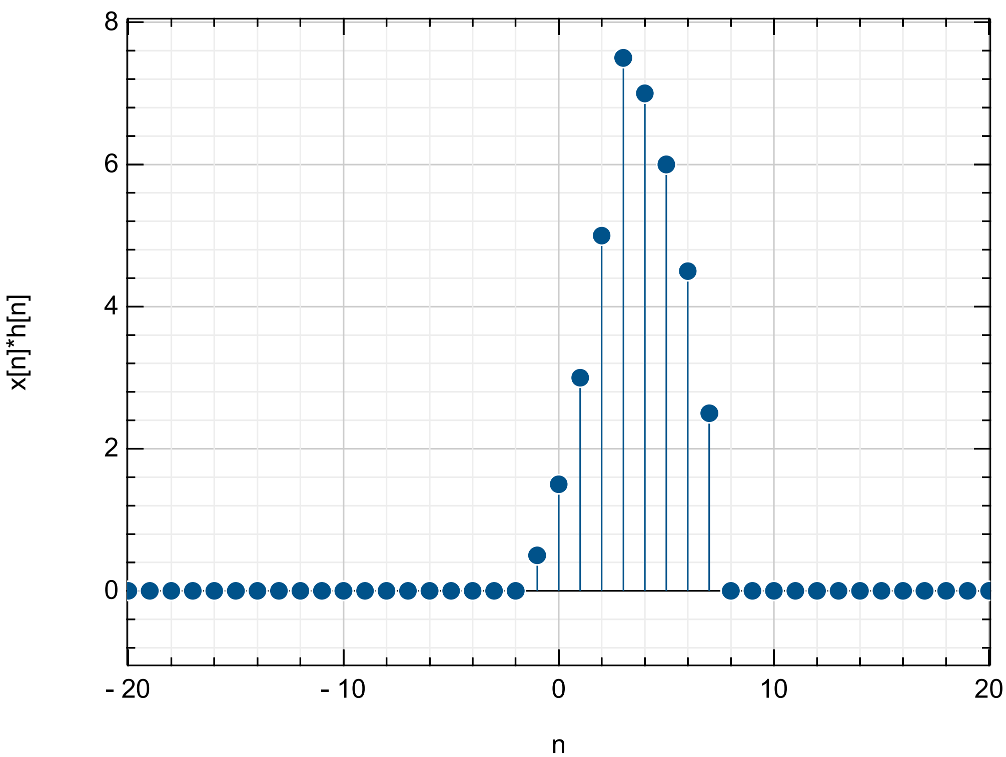
In [37]:
```
############################ -> P2.5 <- ########################################
###############
## Produza os gráficos de x[n], h[n] e x[n] * h[n] em Julia usando a função conv_so
lve:
function conv_solve(n_x, n_h, x, h)
    l_x, l_h = length(x), length(h)
    l_y = l_x + l_h - 1
    y = Vector{promote_type(eltype(x), eltype(h))}(undef, l_y)
    h = h[end:-1:1]
    for i in 1:l_y
        n_i = max(i, l_h)
        n_f = min(i + l_h - 1, l_y)
        y[i] = sum(x[(n_i:n_f) .- l_h .+ 1].*h[(n_i:n_f) .- i .+ 1])
    end
    n = collect(n_x[1] + n_h[1]:n_x[end] + n_h[end])
    return n, y
end

u(x) = x >= 0 ? 1.0 : 0.0;
δ(x) = x == 0 ? 1.0 : 0.0;

## (a)  x[n] = 3^-n (u[n + 1] - u[n - 5]),
##      h[n] = u[n] - u[n - 7],
##      -10 ≤ n ≤ 15;
n = collect(-10:15)
# x[n] = 3^-n (u[n + 1] - u[n - 5])
x = (3.0.^-n) .* (u.(n .+ 1) - u.(n .- 5))
# h[n] = u[n] - u[n - 7]
h = u.(n) - u.(n .- 7)
n_y, y = conv_solve(n, n, x, h)

xlabel("n");
ylabel("x[n]");
xlim([-10.05, 10.05])
ylim([-0.05, 3.05])
stem(n, x)

xlabel("n")
ylabel("h[n]");
xlim([-10.05, 10.05])
ylim([-0.05, 1.05])
stem(n, h)

xlabel("n")
ylabel("x[n]*h[n]");
xlim([-20.05, 30.05])
ylim([-1.05, 5.05])
stem(n_y, y)
```

Out[37]:



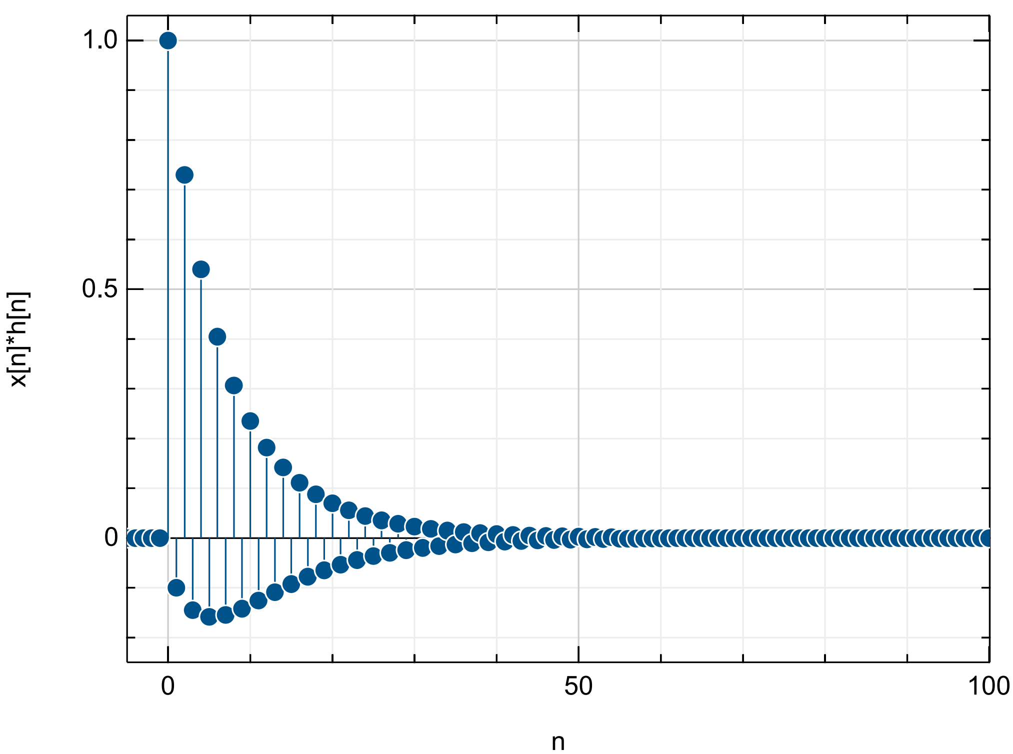
In [38]:
```
## (b) x[n] = (n/4) (u[n] - u[n - 6]),
##     h[n] = 2(u[n + 2] - u[n - 3]),
##     -10 ≤ n ≤ 10;
n = collect(-10:10)
x = (n/4.0) .* (u.(n) .- u.(n .-6))    # x[n] = (n/4) (u[n] - u[n - 6])
h = 2.0 .* (u.(n .+ 2) .- u.(n .- 3)) # h[n] = 2(u[n + 2] - u[n - 3])
n_y, y = conv_solve(n, n, x , h)

xlabel("n");
ylabel("x[n]");
xlim([-10.05, 10.05])
ylim([-1.05, 1.55])
stem(n, x)

xlabel("n")
ylabel("h[n]");
xlim([-10.05, 10.05])
ylim([-0.05, 2.05])
stem(n, h)

xlabel("n")
ylabel("x[n]*h[n]");
xlim([-20.05, 20.05])
ylim([-1.05, 8.05])
stem(n_y, y)
```

Out[38]:



In [39]:
```
## (c) x[n] = 0.8^n u[n],
##     h[n] = (-0.9)^n u[n],
##     -5 ≤ n ≤ 55.
n = collect(-5:55)
x = 0.8.^n .* u.(n)      ## x[n] = 0.8^n u[n]
h = (-0.9).^n .* u.(n)   ## h[n] = (-0.9)^n u[n]
n_y, y = conv_solve(n, n, x , h)

xlabel("n");
ylabel("x[n]");
xlim([-5.0, 60.05])
ylim([-0.05, 1.05])
stem(n, x)

xlabel("n")
ylabel("h[n]");
xlim([-5.0, 60.05])
ylim([-1.05, 1.05])
stem(n, h)


xlabel("n")
ylabel("x[n]*h[n]");
xlim([-5.0, 100.05])
ylim([-0.25, 1.05])
stem(n_y, y)
```

Out[39]:

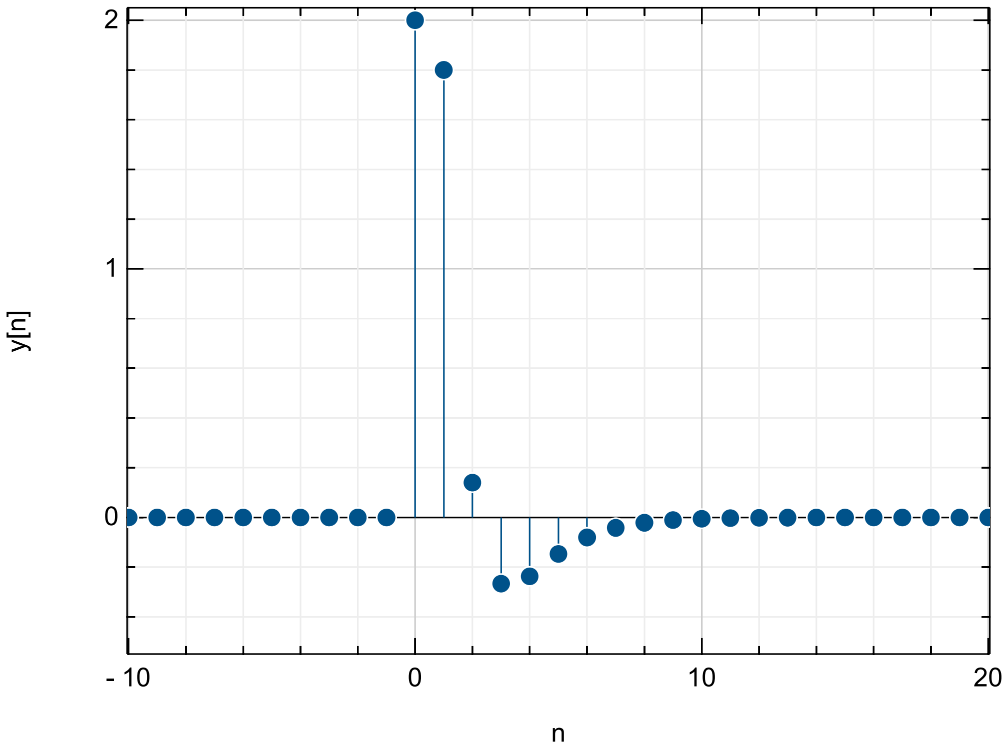


In [40]:
```
############################ -> P2.6 <- ##########################################
###############
##  Considere que a equação diferencial y[n] − y[n − 1] + 0.5y[n − 2] = 2x[n] − x[n
− 1]
##  representa um sistema H. Assumindo que o sistema se encontra inicialmente em r
epouso,
##  determine manualmente a resposta do sistema H às seguintes excitações:
# (a) x[n] = 2δ[n] − 2δ[n − 1], −1 ≤ n ≤ 10;
# 1° simplificar ( y[n] − y[n − 1] + 0.5y[n − 2] = 2x[n] − x[n − 1] )
# y[n] = 2x[n] − x[n − 1] + y[n − 1] − 0.5y[n − 2]
# 2° para descobrir o vetor x[] (x[n] = 2δ[n] − 2δ[n − 1], −1 ≤ n ≤ 10;)
u(x) = x >= 0 ? 1.0 : 0.0;
δ(x) = x == 0 ? 1.0 : 0.0;
n = collect(-1:10)
x = 2 .* δ.(n) - 2 .* δ.(n .- 1)
println(x)
# x[n] = [0.0, 2.0↑, -2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
#         [-1,   0,    1,    2,    3,    4,    5,    6,    7,    8,    9,   10 ] -> posicoes
# 3° (assumimos de de y[0, -1, -2, -3, ...] = 0) (e começamos a descobrir o y[n] no
vos)
# n = -1 -> y[-1] = 2x[-1] - x[-2] + y[-2] - 0.5y[-3] = 2 x 0 - 0 + 0 - 0.5 x 0 = 0
= y[-1]
# n = 0 -> y[0] = 2x[0] - x[-1] + y[-1] - 0.5y[-2] = 2 x 2 - 0 + 0 - 0.5 x 0 = 4 =
y[0]
# n = 1 -> y[1] = 2x[1] - x[0] + y[0] - 0.5y[-1] = 2 x (-2) - 2 + 4 - 0.5 x 0 = -2
= y[1]
# n = 2 -> y[2] = 2x[2] - x[1] + y[1] - 0.5y[0] = 2 x 0 - (-2) + (-2) - 0.5 x 4 = -
2 = y[2]
# n = 3 -> y[3] = 2x[3] - x[2] + y[2] - 0.5y[1] = 2 x 0 - 0 + (-2) - 0.5 x (-2) = -
1 = y[3]
# n = 4 -> y[4] = 2x[4] - x[3] + y[3] - 0.5y[2] = 2 x 0 - 0 + (-1) - 0.5 x (-2) = 0
= y[4]
# .
# até n = 10
# resultado
# y[n] = {0, 4↑, -2, -2, -1, 0, 0.5, 0.5, 0.25, 0, -0.125, -0.125}
#      = {4↑, -2, -2, -1, 0, 0.5, 0.5, 0.25, 0, -0.125, -0.125}
```

```
[0.0, 2.0, -2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

In [41]:
```
############################ -> P2.7 <- #########################################
###############
function diff_solve(x, α, β)
    # x = vetor excitação
    # α = vetor com coeficientes α_0, α_1, α_2, ..., α_M
    # β = vetor com coeficientes β_0, β_1, β_2, ..., β_P
    M = length(α) - 1
    P = length(β) - 1
    l_x = length(x)
    T = promote_type(Float64, eltype(x))
    x = [zeros(T, M); x]
    y = [zeros(T, P); Vector{T}(undef, l_x)]
    for i in 1:l_x
        y[i + P] = 1/β[1]*(sum(α.*x[i + M:-1:i]) - sum(β[2:end].*y[i + P - 1:-1:
i]))
    end
    return y[P + 1:end]
end
## Utilizando a função diff_solve em Julia, produza o gráfico da resposta y[n] de u
m sistema representado pela equação
## y[n] − 0.7y[n − 1] + 0.1y[n − 2] = 2x[n] − x[n − 2] à excitação x[n] = 5^−n * u
[n],
## para −10 ≤ n ≤ 20, assumindo que o sistema está inicialmente em repouso. Esboce
o diagrama de blocos do sistema.
## DADOS
## y[n] = 2x[n] − x[n − 2] − (−0.7y[n − 1] + 0.1y[n − 2])
# valores de alpha e beta talvez sejam retirados de y[n]?? (vetor alpha dados em x,
e vetor beta dados em y)
## α = {2,0,−1}; ## β = {1,0.7,−0.1}; ## x0 = {0,0}; ## y0 = {0,0}

n = collect(-10:20)
## x -> sinal de entrada (vetores x e y têm o mesmo comprimento)
## x = (n -> n == 0 ? 1.0 : 0.0).(n);   # impulso ou degrau
## neste caso temos uma excitação x[n] = 5^−n * u[n]
x = (k -> k >= 0 ? 5.0^-k : 0.0).(n);
## α -> vetor (comprimento = M + 1) dos coeficientes da variável independente (dad
o?)
α = [2.0, 0.0, -1.0]
## β -> vetor (comprimento = P + 1) dos coeficientes da variável dependente  (dad
o?)
# β = [1.0, 0.7, -0.1] -> alterado pelo prof?
β = [1.0, -0.7, 0.1];
# dado: nao utilizado com a nova versão do diff_solve()
x_0 = [0.0, 0.0]
y_0 = [0.0, 0.0]
## y -> sinal de saída (solução da equação de diferenças lineares)
## (o tipo dos elementos de y é o mesmo que o dos elementos de x)
y = diff_solve(x, α, β)

xlabel("n");
ylabel("x[n]");
xlim([-10.05, 20.05])
ylim([0, 1.05])
stem(n, x)

xlabel("n");
ylabel("y[n]");
xlim([-10.05, 20.05])
ylim([-0.55, 2.05])
stem(n, y)
```

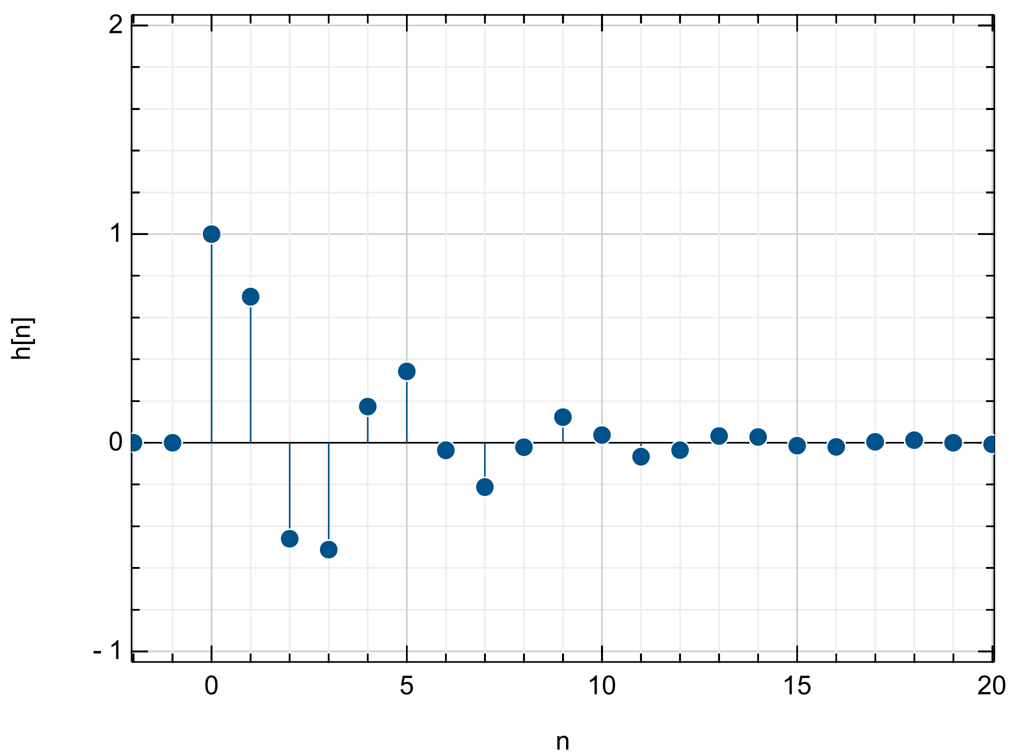
Out[41]:



In [42]:

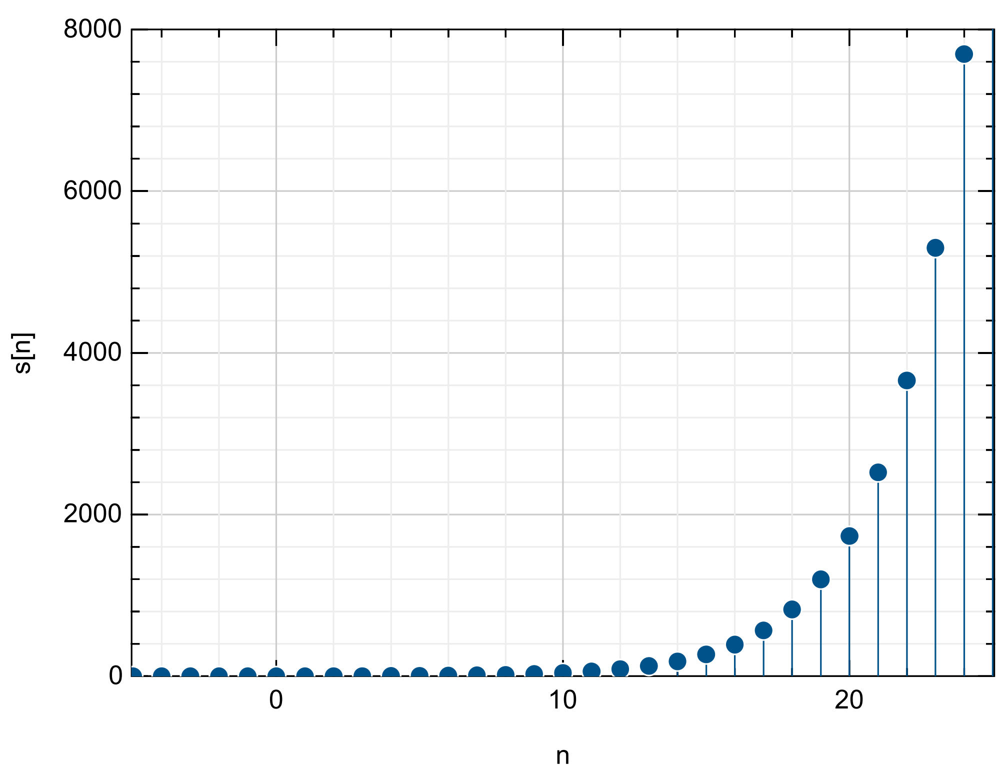
```
################################### -> P2.8 <- ###################################
##############
## Utilizando a função diff_solve em Julia, produza o gráfico da resposta impulsion
al
## h[n] = y[n] de um sistema representado pela equação
## y[n] - 0.2y[n - 1] + 0.6y[n - 2] = x[n[+0.5x[n - 1], para -5 ≤ n ≤ 25,
## assumindo que o sistema está inicialmente em repouso. Esboce o diagrama de bloco
s do sistema.
## DADOS
## y[n] = x[n[+0.5x[n - 1] - (-0.2y[n - 1] + 0.6y[n - 2])
# valores de alpha e beta talvez sejam retirados de y[n]?? (vetor alpha dados em x,
e vetor beta dados em y)
## α = {1,0.5}; ## β = {1,0.2,-0.6}; ## x0 = {0,0}; ## y0 = {0,0};


n = collect(-5:25)
## x = (n -> n == 0 ? 1.0 : 0.0).(n);   # impulso ou degrau
x = (k -> k == 0 ? 1.0 : 0.0).(n);
α = [1.0, 0.5]
β = [1.0, -0.2, 0.6];
# β = [1.0, 0.2, -0.6]
x_0 = [0.0, 0.0]
y_0 = [0.0, 0.0]
h = diff_solve(x, α, β)

ylabel("x[n]");
xlim([-5.05, 20.05])
ylim([-0.05, 1.05])
stem(n, x)

xlabel("n");
ylabel("h[n]");
xlim([-2.05, 20.05])
ylim([-1.05, 2.05])
stem(n, h)
```

Out[42]:



In [43]:
```
######################### -> P2.9 <- #####################################
## Utilizando a função diff_solve em Julia, produza o gráfico da resposta
## s[n] = y[n] ao degrau unitário u[n] de um sistema representado pela
## equação 2y[n] − y[n − 1] − 4y[n − 3] = x[n[+3x[n − 5], para −5 ≤ n ≤ 25,
## assumindo que o sistema está inicialmente em repouso.
## Esboce o diagrama de blocos do sistema.
## DADOS
## y[n] = 1/2 [x[n] + 3x[n − 5] − (−y[n − 1] − 4y[n − 3])]
## α = {1,0,0,0,0,3}; ## β = {2,−1,0,−4}; ## x0 = {0,0,0,0,0,0}; ## y0 = {0,0,0}

n = collect(-5:25)
## degrau
x = (k -> k >= 0 ? 1.0 : 0.0).(n);
α = [1.0, 0.0, 0.0, 0.0, 0.0, 3.0]
β = [2.0, -1.0, 0.0, -4.0]
x_0 = zeros(6)
y_0 = zeros(3)
s  = diff_solve(x, α, β)

xlabel("n");
ylabel("x[n]");
xlim([-5.05, 25.05])
ylim([-0.05, 1.05])
stem(n, x)

xlabel("n");
ylabel("s[n]");
xlim([-5.05, 25.05])
ylim([-0.05, 8000])
stem(n, s)
```

Out[43]:



In [44]:
```
################# 3. ANÁLISE DISCRETA DE FOURIER#################################
########################################### -> P3.1 <- ##########################
## Determine os coeficientes da série de Fourier dos seguintes sinais periódicos:
## (a) x[n] = {  1 k ≤ n ≤ 3 + k        , k = 0, ±1, ±2, …;
##              {  2 4 + k ≤ n ≤ 7 + k
## n <= 7 -> N = 8 (N = tamanho da amostra)
N = 8
## neste passo -> n = collect(0:(2*N-1))
n = collect(-2: 17)
x = [1.0, 1.0, 1.0, 1.0, 2.0, 2.0, 2.0, 2.0][mod.(n, N) .+ 1]

xlim([-2.25, 17.25])
ylim([-0.05, 2.05])
xlabel("n")
ylabel("x[n]")
stem(n, x)

## calcular os quoficientes
## ω0 -> fixo
ω0 = 2 * π /N
## neste passo -> n = collect(0:(N - 1))
n = collect(0:(N - 1))
x = [1.0, 1.0, 1.0, 1.0, 2.0, 2.0, 2.0, 2.0]
# formula
a = [1/N * sum(x .* exp.(-im * k * ω0 * n)) for k in 0:N - 1];
println(round.(real(a), digits = 4))
println(round.(imag(a), digits = 4))
```

```
[1.5, -0.125, -0.0, -0.125, 0.0, -0.125, -0.0, -0.125]
[0.0, 0.3018, -0.0, 0.0518, -0.0, -0.0518, -0.0, -0.3018]
```

In [45]:
```
## (b) x[n] = {…, 0.8, 1, 0↑, 0.2, 0.4, 0.6, 0.8, 1, 0, 0.2,…}
## tamanho da amostra
N = 6
## neste passo -> n = collect(0:(2*N-1))
n = collect(-2:13)
x = [0.0, 0.2, 0.4, 0.6, 0.8, 1.0][mod.(n, N) .+ 1]

xlim([-2.25, 13.25]);
ylim([-0.05, 1.05]);
xlabel("n");
ylabel("x[n]");
stem(n, x)

## calcular os quoficientes
ω0 = 2 * π /N
## neste passo -> n = collect(0:(N - 1))
n = collect(0:(N - 1))
x = [0.0, 0.2, 0.4, 0.6, 0.8, 1.0]
# formula
a = [1/N * sum(x .* exp.(-im * k * ω0 * n)) for k in 0:N - 1];
println(round.(real(a), digits = 4))
println(round.(imag(a), digits = 4))
```

```
[0.5, -0.1, -0.1, -0.1, -0.1, -0.1]
[0.0, 0.1732, 0.0577, -0.0, -0.0577, -0.1732]
```

In [46]:
```
## (c) x[n]={…, 0, 0.5, 1↑, 0.5, 0, -0.5, -1, -0.5, 0, 0.5, 1, 0.5, …}
N = 8
n = collect(-2:17)
x = [1, 0.5, 0, -0.5, -1, -0.5, 0, 0.5][mod.(n, N) .+ 1]
xlim([-2.25, 17.25])
ylim([-1.05, 1.05])
xlabel("n")
ylabel("x[n]")
stem(n, x)

## calcular os quoficientes
ω0 = 2 * π /N
## neste passo -> n = collect(0:(N - 1))
n = collect(0:(N - 1))
x = [1, 0.5, 0, -0.5, -1, -0.5, 0, 0.5]
# formula
a = [1/N * sum(x .* exp.(-im * k * ω0 * n)) for k in 0:N - 1];
println(round.(real(a), digits = 4))
println(round.(imag(a), digits = 4))
```

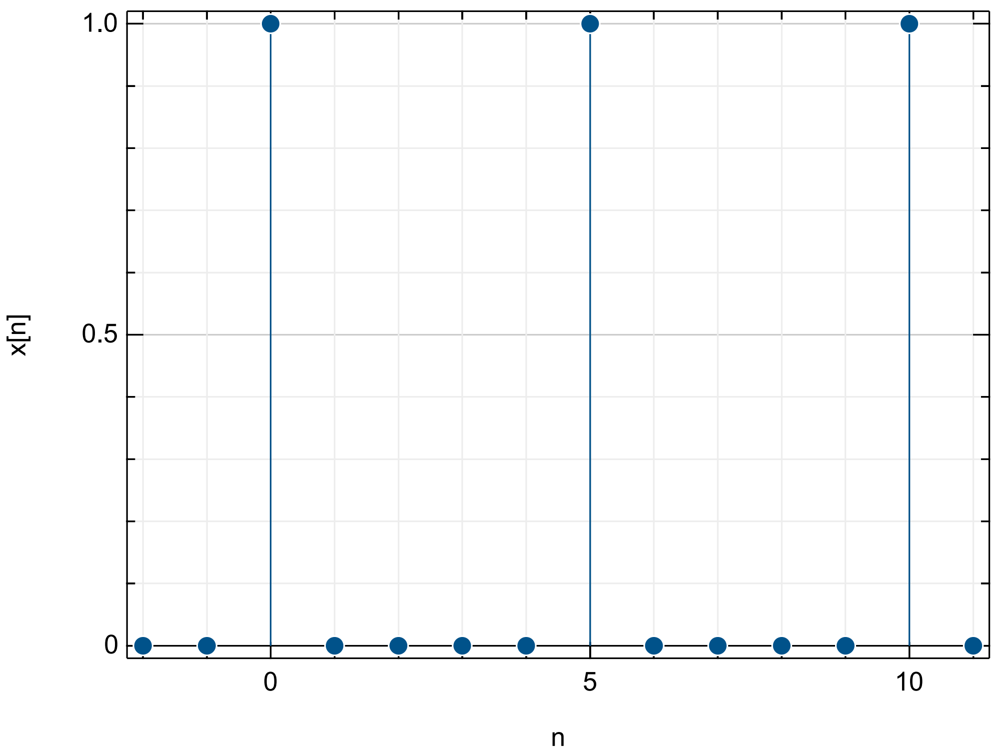
```
[0.0, 0.4268, -0.0, 0.0732, 0.0, 0.0732, 0.0, 0.4268]
[0.0, 0.0, -0.0, -0.0, -0.0, -0.0, -0.0, 0.0]
```

In [47]:
```
## (d) x[n] = cos(0.5πn) + cos(0.25πn - 0.125π)
f(n) = cos(0.5*π*n) + cos(0.25*π*n - 0.125*π);
## 0.25 vêm do sinal -> 2/0.25 = 8
## N = ω0 / 2π = max(2π/0.5π, 2π/0.25π) = (4, 8) = 8
N = 2/0.25;

## n = collect(0:(2*N-1))
n = collect(-2:(2*N-1)) # (0:(2*N-1))
## println(n1)
## teste
## n2 = collect(-2:17)
## println(n2)

x = f.(n)
xlim([-3.25, 17.25]);
ylim([-2.05, 2.05]);
xlabel("n");
ylabel("x[n]");
stem(n, x)

## calcular os quoficientes
ω0 = 2 * π /N
n = collect(0:(N - 1))
x = f.(n)
# formula
a = [1/N * sum(x .* exp.(-im * k * ω0 * n)) for k in 0:N - 1];
println(round.(real(a), digits = 4))
println(round.(imag(a), digits = 4))
```

```
[-0.0, 0.4619, 0.5, 0.0, -0.0, 0.0, 0.5, 0.4619]
[0.0, -0.1913, 0.0, -0.0, -0.0, -0.0, 0.0, 0.1913]
```

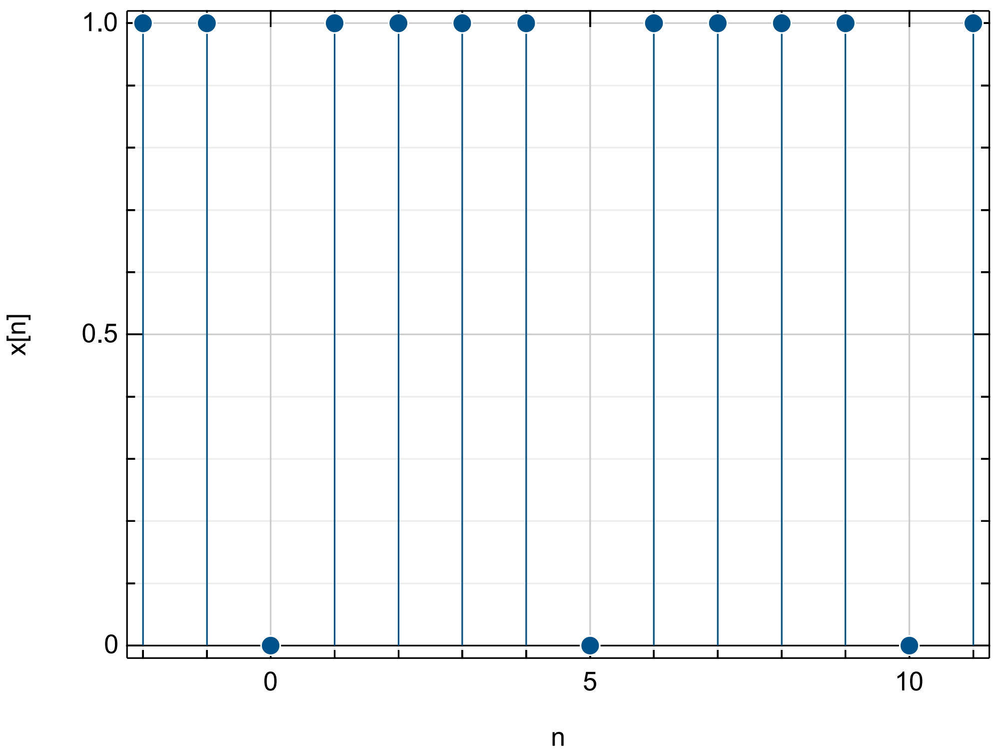
In [48]:
```
########################################## -> P3.2 <- ############################
#####
## Determine as séries numéricas cujas séries de Fourier têm os seguintes coeficien
tes:
##   (a) a0 = a1 = a2 = a3 = a4 = 0.2 (N = 5)
N = 5          # dado
## fixo -> ω₀ = 2*π/N;
ω₀ = 2*π/N;
n = collect(-2:(2*N + 1));
## fill(0.2, 3) = [0.2, 0.2, 0.2] -> vetor de 0.2 com tamanho igual ao segundo para
metro
## vetor com os valores e a0,a1, a2, a3, ...
a = fill(0.2, 5);
## fixo -> φ = [exp.(im*k*ω₀*n) for k in 0:N - 1];
φ = [exp.(im * k * ω₀ * n) for k in 0:(N - 1)];
## fixo -> x = real(sum([a[k + 1]*φ[k + 1] for k in 0:N - 1]));
x = real(sum([a[k + 1]*φ[k + 1] for k in 0:(N - 1)]));
xlim([-2.25, 11.25]);
ylim([-0.02, 1.02]);
xlabel("n");
ylabel("x[n]");
stem(n, x)
```

Out[48]:



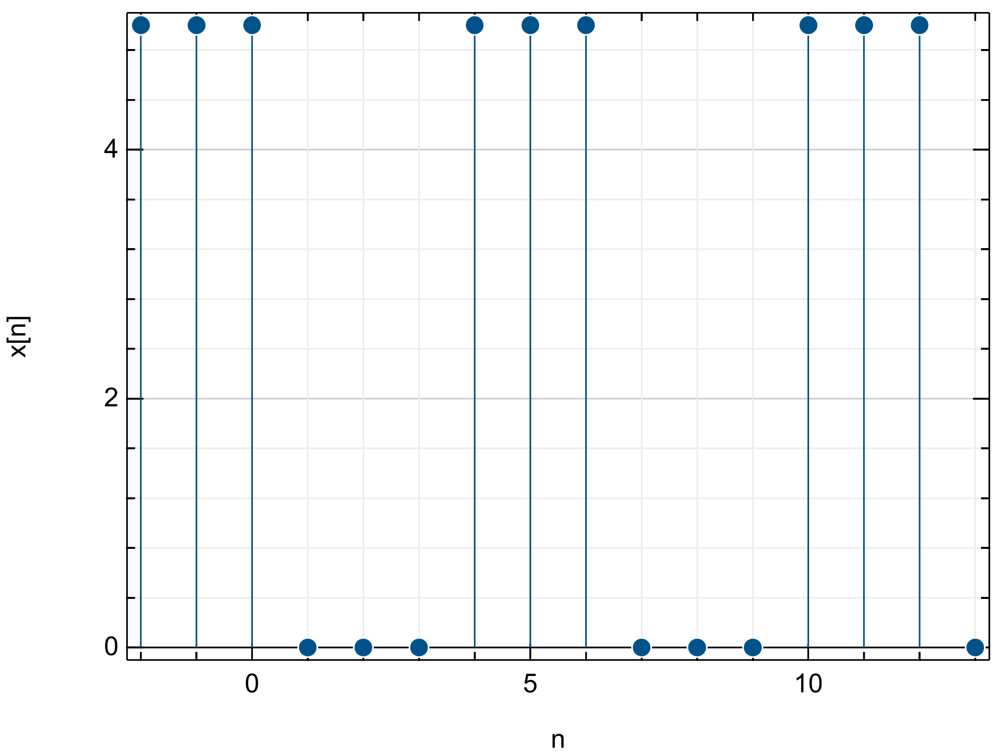
In [49]:
```
## (b) a0 = 0.8, a1 = a2 = a3 = a4 = -0.2 (N = 5);
N = 5;
ω₀ = 2*π/N;
n = collect(-2:(2*N + 1));
## vetor com os valores e a0,a1, a2, a3, ...
a = [0.8; fill(-0.2, 5)];
φ = [exp.(im*k*ω₀*n) for k in 0:N - 1];
x = real(sum([a[k + 1]*φ[k + 1] for k in 0:N - 1]));
xlim([-2.25, 11.25]);
ylim([-0.02, 1.02]);
xlabel("n");
ylabel("x[n]");
stem(n, x)
```

Out[49]:



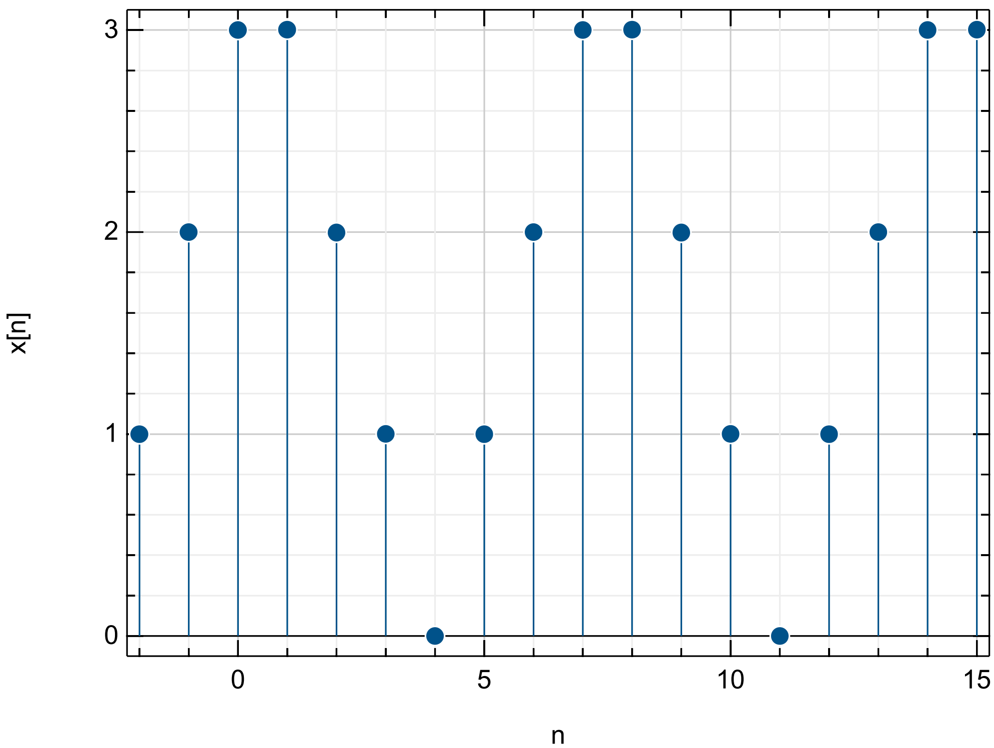
In [50]:
```
## (c) a0 = 2.5, a1 = (5/3)e^j(π/3), a2 = 0, a3 = 5/6 , a4 = 0, a5 = (5/3)e^-j(π/3)
(N = 6);
N = 6
ω₀ = 2*π/N;
n = collect(-2:2*N + 1);
## vetor com os valores e a0,a1, a2, a3, ...
a = [2.5, 5/3*exp(im*(π/3)), 0.0, 5/6, 0.0, 5/3*exp(-im*(π/3))]
φ = [exp.(im*k*ω₀*n) for k in 0:N - 1];
x = real(sum([a[k + 1]*φ[k + 1] for k in 0:N - 1]));
xlim([-2.25, 13.25]);
ylim([-0.1, 5.1]);
xlabel("n");
ylabel("x[n]");
stem(n, x)
```

Out[50]:



In [51]:
```
## (d) a0 = 1.714, a1 = 0.65 - j0.313, a2 = -0.027 + j0.034, a3 = 0.02 - j0.09, a4 =
0.02 + j0.09,
## a5 = -0.027 - j0.034, a6 = 0.65 + j0.313 (N = 7);
N = 7
ω₀ = 2*π/N;  # fixo
n = collect(-2:(2*N + 1));
## a = [a0, a1, a2, a...]
a = [1.714, (0.65 - im*0.313), (-0.027 + im*0.034), (0.02 - im*0.09), (0.02 + im*0.
09), (-0.027 - im*0.034), (0.65 + im*0.313)]
φ = [exp.(im*k*ω₀*n) for k in 0:N - 1];
x = real(sum([a[k + 1]*φ[k + 1] for k in 0:N - 1]));
xlim([-2.25, 15.25]);
ylim([-0.1, 3.1]);
xlabel("n");
ylabel("x[n]");
stem(n, x)
```

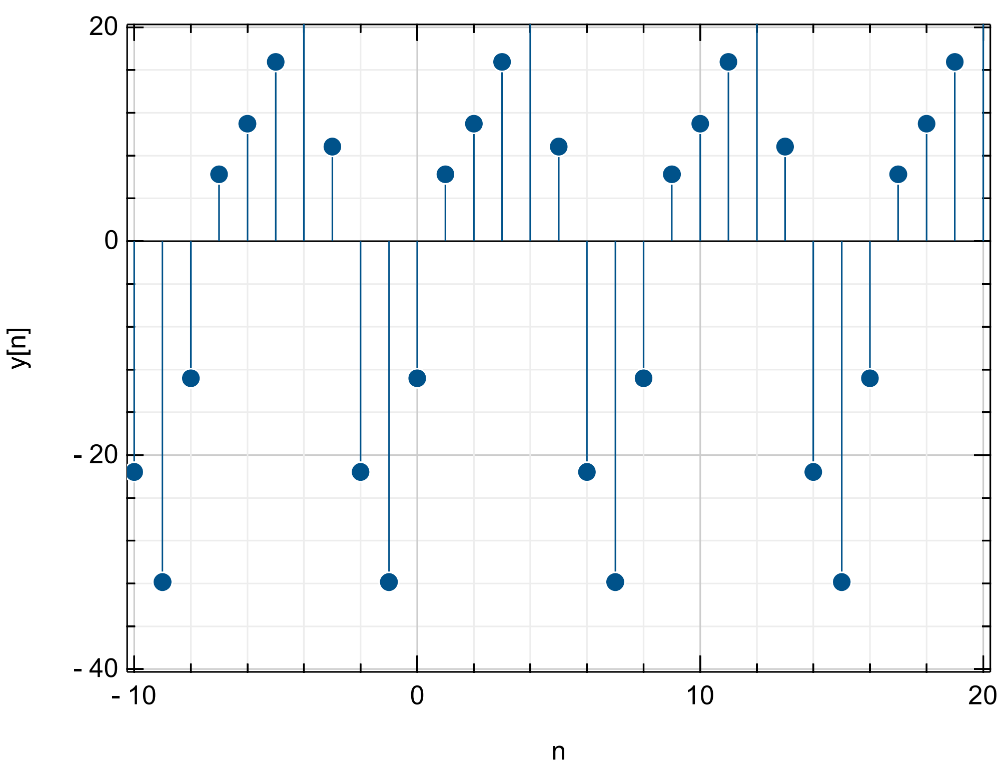
Out[51]:



In [52]:

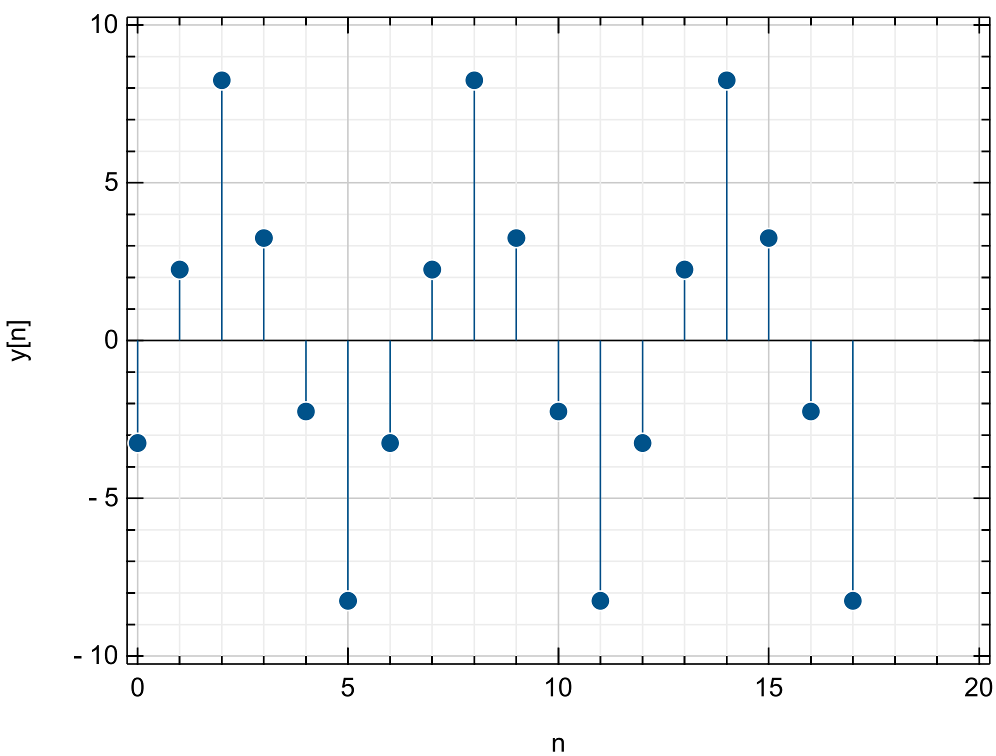
```
################################## -> P3.3_1 <- #########################
## Determine a resposta do sistema do sistema H, cuja resposta impulsional
## é h[n] = {8↑, 8, 7, 6, 4, 2, 0.5},
## ao sinal x[n] = sin(0.25πn) + cos(0.5πn + 0.125π), -10 ≤ n ≤ 20

# N = max(2π/0.25π, 2π/0.5π) = 8
N = 8;
## h[n]
## h = [8, 8, 7, 6, 4, 2, 0.5]
h = [8, 12, 7, 2, 0, -1, -0.5, 0, 0.5]
## fixo
H = [sum(h[m + 1]*exp(-im*k*2*π/N*m) for m in 0:N) for k in 0:N - 1];
## fixo
n = collect(0:(N - 1));

# x[n] = sin(0.25πn) + cos(0.5πn + 0.125π)
x = (n -> sin(0.25*π*n) + cos(0.5*π*n + 0.125*π)).(n);
# vetor x de cima, vai ser usado para calcular vetor 'a' de baixo
a = [1/N*sum(x.*exp.(-im*k*2*π/N*n)) for k in 0:N - 1];
b = a.*H; n = collect(-10:20);

x = x[mod.(n, N) .+ 1];
y = sum(b[k + 1]*exp.(im*k*2*π/N*n) for k in 0:N - 1);

xlabel("n");
ylabel("x[n]");
xlim([-10.25, 20.25]);
ylim([-2.25, 1.25]);
stem(n, x)

xlabel("n");
ylabel("y[n]");
xlim([-10.25, 20.25]);
ylim([-40.25, 20.25]);
stem(n, real(y))
```

Out[52]:



In [53]:
```
########################## -> P3.3_2 <- ###################################
## Determine a resposta do sistema do sistema H, cuja resposta impulsional
## é h[n] = {5,4,3,2,1,0,-0.5,-0.25},
## ao sinal periódico x[n] = {…,-1,-1,1↑ ,1,1,-1,-1,-1,1,1,…}, 0≤ n ≤ 17.
h = [5, 4, 3, 2, 1, 0, -0.5, -0.25]        # h[n] = {5,4,3,2,1,0,-0.5,-0.25}
# M é o tamanho do vetor h?
M = 8
x = [1,1,1,-1,-1,-1]  # amostrta x[n]
N = 6                 # tamanho da amostra de x

ω₀ = 2*π/N;
H = [sum(h[m + 1]*exp(-im * k * ω₀ * m) for m in 0:M - 1) for k in 0:N - 1];

n = collect(0: (N-1))
a = [1/N*sum(x.*exp.(-im * k * ω₀ * n)) for k in 0:N - 1];
b = a.*H;

n = collect(0:17);
x = x[mod.(n, N) .+ 1];
y = sum(b[k + 1]*exp.(im * k * ω₀ * n) for k in 0:N - 1);

xlabel("n");
ylabel("x[n]");
xlim([-0.25, 20.25]);
ylim([-1.05, 1.05]);
stem(n, x)

xlabel("n");
ylabel("y[n]");
xlim([-0.25, 20.25]);
ylim([-10.25, 10.25]);
stem(n, real(y))
```

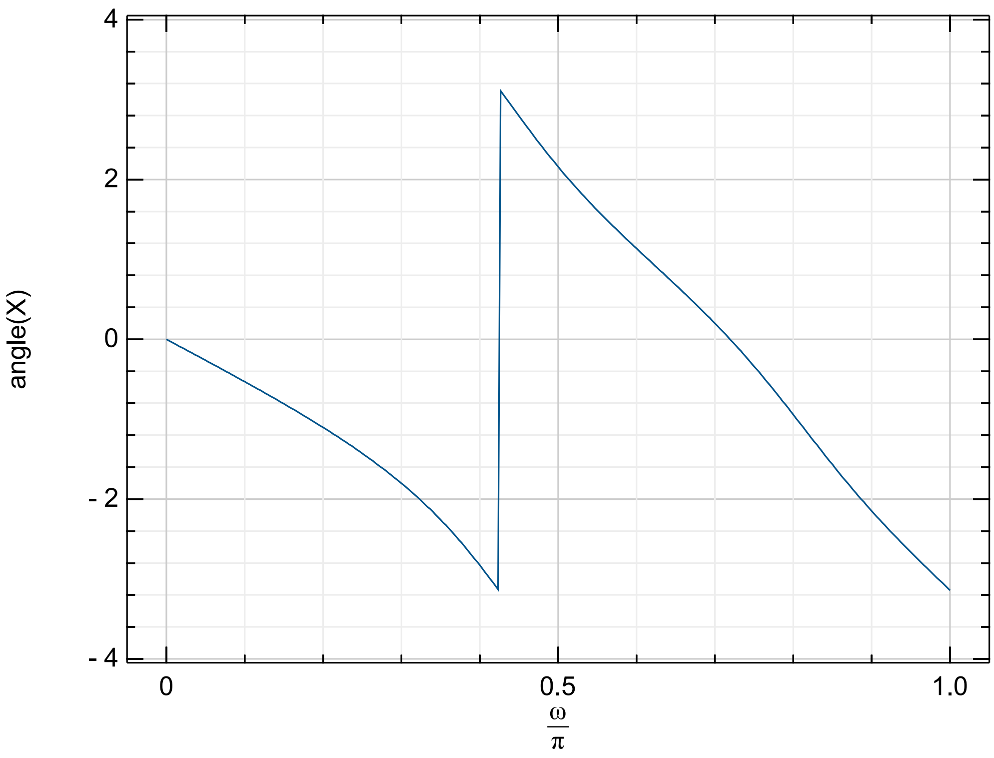
Out[53]:



In [54]:

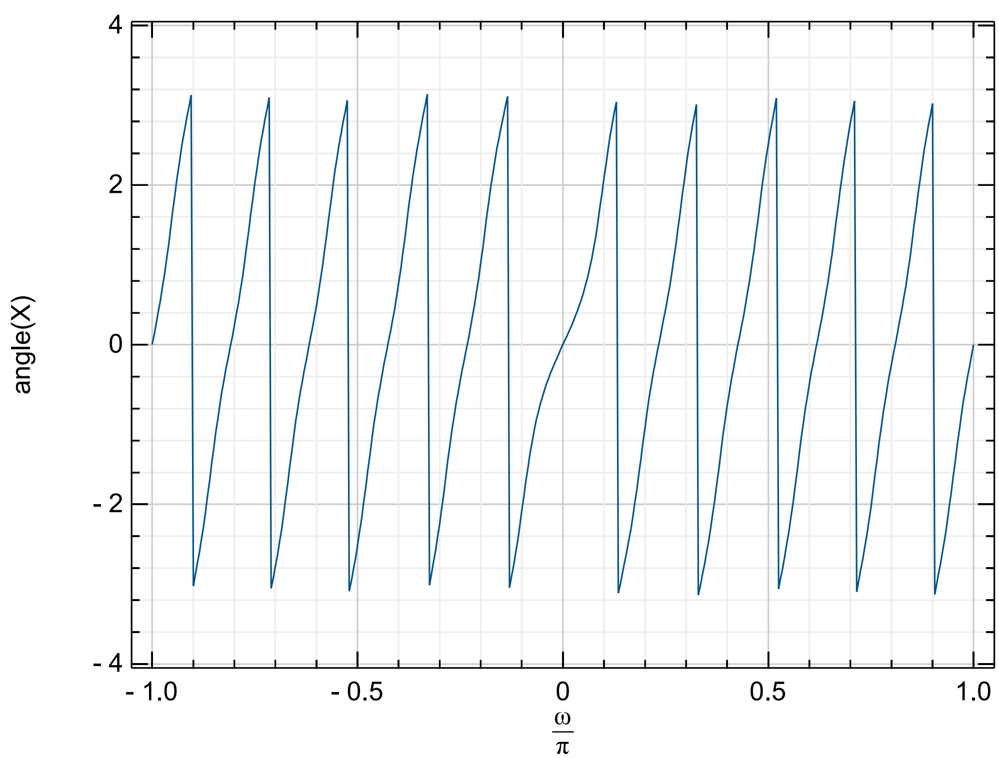
```
################################# -> P3.4 <- #######################
## Determine a transformada discreta de Fourier para o sinal
## x[n] = {1, 2↑, 3, 4, 5}
## para 301 pontos equidistantes entre 0 e π e produza gráficos
## da magnitude e do ângulo do resultado da transformada.

## fixo
dft(ω, n, x) = sum(x.*exp.(-im * ω * n));

n = collect(-1:3);
## tamanho de x[n]
x = collect(1:5);
## 301 pontos
ω = collect(0:300)*π/300;
X = dft.(ω, (n,), (x,));

xlabel("\\omega/\\pi");
ylabel("abs(X)");
xlim([-0.05, 1.05]);
ylim([-0.05, 15.05]);
plot(ω/π, abs.(X))

xlabel("\\omega/\\pi");
ylabel("angle(X)");
ylim([-4.05, 4.05]);
plot(ω/π, angle.(X))
```

Out[54]:



In [55]:
```
####################################### -> P3.5 <- ###################
## Determine a transformada discreta de Fourier para o sinal
## x[n] = 0.9^n, -10 ≤ n ≤ 10, para 401 pontos
## equidistantes entre -π e π e produza gráficos da magnitude
## e do ângulo do resultado da transformada.

n = collect(-10:10)       # -10 ≤ n ≤ 10
x = (n -> 0.9^n).(n);     # x[n] = 0.9^n
## 401 pontos
ω = collect(-200:200)*π/200;
X = dft.(ω, (n,), (x,));

xlabel("\\omega/\\pi");
ylabel("abs(X)");
xlim([-1.05, 1.05]);
ylim([-1.05, 30.05]);
plot(ω/π, abs.(X))

xlabel("\\omega/\\pi");
ylabel("angle(X)");
xlim([-1.05, 1.05]);
ylim([-4.05, 4.05]);
plot(ω/π, angle.(X))
```

Out[55]:



In [ ]: