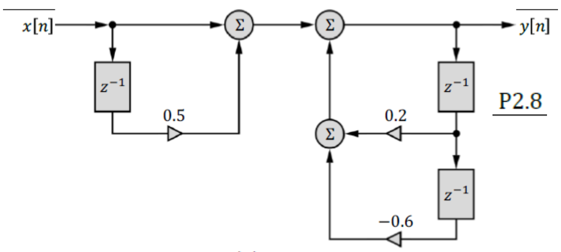
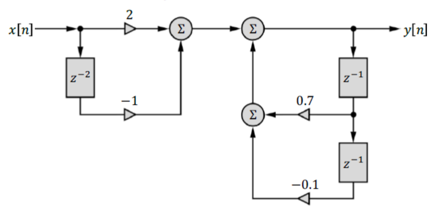
```
############################################ -> (P2.2) <- ############################################
## P2.2 -> gerar series numericas e produzir os respectivos graficos
#################################################(A)#################################################
## (a) x[n] = 2δ[n - 1] + δ[n] - δ[n + 4] + 2δ[n + 7], -10 ≤ n ≤ 5;
using GR
xlabel("x")
ylabel("x[n]")
## importante
# u(x) = x >= 0 ? 1.0 : 0.0;
# δ(x) = x == 0 ? 1.0 : 0.0;
δ(x) = x == 0 ? 1.0 : 0.0;
n = collect(-10:5)
x = 2 * δ.(n .- 1) + δ.(n) - δ.(n .+ 4) + 2 * δ.(n .+ 7)
xlim([-10.25, 5.25])
ylim([-1.1, 2.1])
stem(n, x)
#################################################(B)#################################################
## (b) x[n] = ∑ e-|k| δ[n - 2k] ,-10 ≤ n ≤ 10;
# sum(k = -5 -> 5)
δ(x) = x == 0 ? 1.0 : 0.0;
n = collect(-10:10);
x = [sum([exp(-abs(k)) .* δ.(n .- 2 * k) for k = -5:5]) for n = -10:10]
xlim([-10.5, 10.5])
ylim([-0.05, 1.05])
stem(n, x)
################################################### -> P2.3 <- ######################################
## P2.3 -> Produza os gráficos das seguintes séries periódicas
#################################################(A)#################################################
## (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)
#################################################(B)################################################
## (b) x[n] = {…,x[38],x[39],x[40],x[41],x[42],x[43],x[44],…} = {…,10,0,0,10,0,0,10,…}, -8 ≤ x ≤ 12
n = collect(-8:12)
p = [10,0,0]
## m1 -> onde começa a amostra de p
m1 = 38
x = p[mod.(n .- m1, length(p)) .+ 1]
xlim([-8.25, 12.25]);
ylim([-0.5, 10.5]);
stem(n, x)

############################################### -> P2.5 <- ########################
## Produza os gráficos de x[n], h[n] e x[n] * h[n] em Julia usando a função conv_solve:
####################################################conv_solve#########################
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
##############################################(A)####################################
## (a)  x[n] = 3^-n (u[n + 1] - u[n - 5]),
##      h[n] = u[n] - u[n - 7],
##      -10 ≤ n ≤ 15;
## importante
u(n) = n >= 0 ? 1.0 : 0.0;
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)
```



P2.8

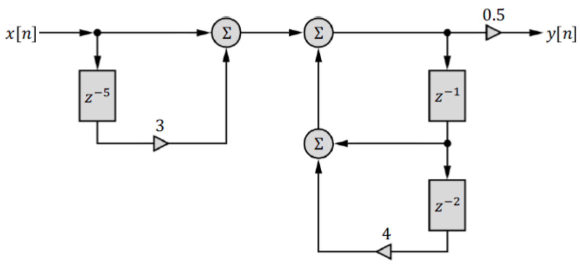
```
##################################################(B)##################
## (b) x[n] = (n/4) (u[n] - u[n - 6]),
##      h[n] = 2(u[n + 2] - u[n - 3]),
##      -10 ≤ n ≤ 10;
u(n) = n >= 0 ? 1.0 : 0.0;
n = collect(-10:10)
x = (n/4.0) .* (u.(n) .- u.(n .-6))    # x[n] = (n/4) (u[n] - u[n - 6])
h = 2.0 .* (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)
```

```
####################################### -> 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 repouso,
# determine manualmente a resposta do sistema H às seguintes excitações:
#############################################(A)##################################################
# (a) x[n] = 2δ[n] - 2δ[n - 1], -1 ≤ n ≤ 10;
# 1º 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]
# 2º para descobrir o vetor x[] (x[n] = 2δ[n] - 2δ[n - 1], -1 ≤ n ≤ 10;)
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
# 3º (assumimos de de y[0, -1, -2, -3, ...] = 0) (e começamos a descobrir o y[n] novos)
# 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}
######################################## -> 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 um 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 (dado?)
α = [2.0, 0.0, -1.0]
## β -> vetor (comprimento = P + 1) dos coeficientes da variável dependente  (dado?)
# 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([-2.05, 2.05])
stem(n, y)
```




```
##################################### -> P2.8 <- ################################################
## Utilizando a função diff_solve em Julia, produza o gráfico da resposta impulsional
## 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 blocos 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([-10.05, 20.05])
ylim([-2.05, 2.05])
stem(n, h)
```

```
########################## -> 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)
```



```
################### 3. ANÁLISE DISCRETA DE FOURIER###################################
######################################## -> P3.1 <- ##################################
## Determine os coeficientes da série de Fourier dos seguintes sinais periódicos:
################################################(A)##################################
## (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))
#################################(B)########################################
## (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))
################################################(D)########################################
## (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(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))
########################################## -> P3.2 <- #####################################
## Determine as séries numéricas cujas séries de Fourier têm os seguintes coeficientes:
################################################(A)########################################
##  (a) a0 = a1 = a2 = a3 = a4 = 0.2 (N = 5)
N = 5        # dado
## fixo -> ωo = 2*π/N;
ωo = 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 parametro
## vetor com os valores e a0,a1, a2, a3, ...
a = fill(0.2, 5);
## fixo -> ϕ = [exp.(im*k*ωo*n) for k in 0:N - 1];
ϕ = [exp.(im * k * ωo * 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)
#######################################(B)######################################
## (b) a0 = 0.8, a1 = a2 = a3 = a4 = -0.2 (N = 5);
N = 5;
ωo = 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*ωo*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)
##################################################(C)##########################################
## (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
ωo = 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*ωo*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)
################################################(D)##########################################
## (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
ωo = 2*π/N;  # fixo
n = collect(-2:(2*N + 1));
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*ωo*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)
#################################### -> 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 0:8) 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))
############################## -> 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

ωo = 2*π/N;
H = [sum(h[m + 1]*exp(-im * k * ωo * 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 * ωo * 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 * ωo * 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))
##################################### -> 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))
```

```
## 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))
```