

UNIVERSIDADE DE PERNAMBUCO (UPE)

[VINICIUS JOSE FERNANDES RIBEIRO](mailto:vjfr@poli.br)

**Atividade Avaliativa**

**Sistemas de Comunicação**

Recife

2024

# Resumo

Documento com o intuito de alocar as informações necessárias para realizar as atividades avaliativas propostas pelo Prof. Diego Rátiva, na disciplina de Sistemas de comunicação.

## Atividades

Seguirei com a alocação dos exercícios e a transcrição dos respectivos exercícios Propostos

### Exercício 1, 2, 3, em range:

3.10 EXERCÍCIOS COM O MATLAB

## **Cálculo de Transformadas de Fourier**

Nesta seção de exercícios baseados em computador, consideremos dois exemplos para ilustrar o uso de DFT no cálculo da transformada de Fourier. Usaremos MATLAB para calcular a DFT com o algoritmo de FFT. No primeiro exemplo, o sinal é *g*(*t*) = *e*

−2*t u*(*t*), com início em *t* = 0, e no segundo, *g*(*t*) = Π(*t*), com início em *t* = −½.



EXEMPLO COMPUTACIONAL C3.1

Empreguemos a DFT (implementada pelo algoritmo de FFT) para calcular a transformada de Fourier de *e* −2*t u*(*t*) e, a seguir, tracemos o gráfico do resultante espectro de Fourier.

Primeiro, devemos determinar *T s* e *T* 0 . A transformada de Fourier de *e* −2*t u*(*t*) é 1/(2*πf* + 2). Esse sinal passa­faixa não é limitado em frequência. Tomemos sua largura de banda essencial como a frequência em que |*G*(*f*)| se torna igual a 1% do valor de pico, que ocorre em *f* = 0. Observemos que



O pico de |*G*(*f*)| ocorre em *f* = 0, em que |*G*(0)| = 0,5. Portanto, a largura de banda essencial *B* corresponde a *f = B*, com



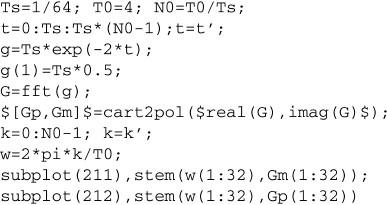
e, da Eq. (3.105b),



Arredondemos esse valor para *T s* = 0,015625 segundo, de modo que tenhamos 64 amostras por segundo. Agora, devemos determinar *T* 0 . O sinal não é limitado no tempo. Precisamos truncá­lo em *T* 0 , tal que *g*(*T* 0 ) ≪ 1. Escolhamos *T* 0 = 4 (oito constantes de tempo do sinal), o que resulta em *N* 0 = *T* 0 /*T s* = 256, que é uma potência de 2. Vale ressaltar que há

muita flexibilidade na determinação de *T s* e *T* 0 , dependendo da precisão desejada e da capacidade computacional disponível. Poderíamos ter escolhido *T* 0 = 8 e *T s* = 1/32, o que também resultaria em *N* 0 = 256, mas implicaria um erro de mascaramento ligeiramente maior.

Como o sinal tem uma descontinuidade do tipo degrau em *t* = 0, o valor da primeira amostra (em *t* = 0) é 0,5, média dos valores nos dois lados da descontinuidade. O programa de MATLAB que implementa a DFT com o algoritmo de FFT é o seguinte:



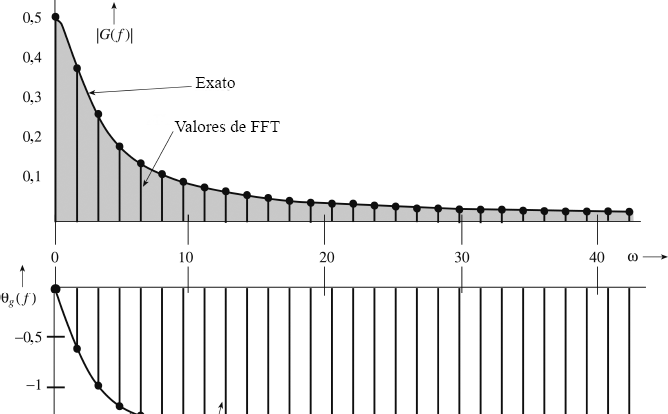
Como *Gq* tem período *N* 0 , *Gq = G* (*q*+256) , de modo que *G* 256 = *G* 0 . Portanto, basta traçar o gráfico de *Gq* no intervalo *q*

= 0 a *q* = 255 (e não 256). Além disso, devido à periodicidade, *G* −*q* = *G* (−*q*+256) , ou seja, os valores de *Gq* no intervalo *q* =

−127 a *q* = −1 são idênticos aos valores de *Gq* no intervalo *q* = 129 a *q* = 255. Logo, *G* −127 = *G* 129 , *G* −126 = *G* 130 ,..., *G* −1 = *G* 255 . Adicionalmente, devido à propriedade de simetria conjugada da transformada de Fourier, *G* −*q* = *G\* q* ; assim, *G* 129 = *G*\* 127 , *G* 130 = *G*\* 126 ,..., *G* 255 = *G*\* 1 . Consequentemente, para sinais de valores reais, não é necessário marcar no gráfico os valores de *Gq* com *q* maior que *N* 0 /2 (128, neste caso), pois são os complexos conjugados dos valores de *Gq* com *q* = 0 a 128.

O gráfico do espectro de Fourier na Fig. 3.40 mostra amplitude e fase das amostras de *G*(*f*) tomadas em intervalos de 1/*T* 0

= 1/4 Hz, ou *ω* 0 = 1,5708 rad/s. Na Fig. 3.40, mostramos apenas os primeiros 28 pontos (em vez dos 128 pontos), para evitar o acúmulo excessivo de dados no gráfico.





**Figura 3.40** Transformada de Fourier discreta de um sinal exponencial *e* −2*t u*(*t*). O eixo horizontal é *ω* (em radianos por segundo).

Neste exemplo, dispúnhamos da expressão analítica de *G*(*f*), o que nos permitiu fazer escolhas INTELIGENTES para *B* (ou frequência de amostragem *f s* ). Na prática, em geral, não conhecemos *G*(*f*). Na verdade, isso é exatamente o que desejamos calcular. Nesses casos, para determinar *B* ou *f s* , devemos lançar mão de evidências circunstanciais. Devemos, sucessivamente, reduzir o valor de *T s* e calcular a transformada até que o resultado satisfaça o desejado número de algarismos significativos.



A seguir, calcularemos a transformada de Fourier de *g*(*t*) = 8 Π(*t*).

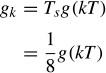


EXEMPLO COMPUTACIONAL C3.2

Empreguemos a DFT (implementada pelo algoritmo de FFT) para calcular a transformada de Fourier de 8 Π(*t*) e tracemos o gráfico do resultante espectro de Fourier.

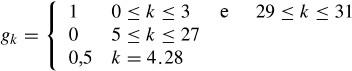
Essa função retangular e sua transformada de Fourier são mostradas na Fig. 3.41a e b. Para determinar o valor do intervalo de amostragem *T s* , devemos, primeiro, definir a largura de banda essencial *B*. Da Fig. 3.41b, vemos que *G*(*f*) decai lentamente com *f*. Consequentemente, a largura de banda essencial *B* é bastante grande. Por exemplo, em *B* = 15,5 Hz (97,39 rad/s), *G*(*f*) = −0,1643, o que corresponde a cerca de 2% do valor de pico, *G*(0). Poderíamos, então, tomar a largura de banda essencial como 16 Hz. No entanto, deliberadamente, tomaremos *B* = 4 Hz, por dois motivos: (1) mostrar o efeito de mascaramento e (2) o uso de *B* > 4 implicaria enorme número de amostras, que não poderiam ser mostradas de forma adequada em uma página de livro sem perda de detalhes fundamentais. Portanto, aceitaremos a aproximação para que possamos esclarecer conceitos de DFT por meio de gráficos.

A escolha *B* = 4 resulta em um intervalo de amostragem *T s* = 1/2*B* = 1/8 segundos. Examinando novamente o espectro na Fig. 3.41b, vemos que a escolha da resolução de frequência *f* 0 = 1/4 Hz é razoável, e corresponde a quatro amostras em cada lóbulo de *G*(*f*). Neste caso, *T* 0 = 1/*f* 0 = 4 segundos, e *N* 0 = *T* 0 /*T s* = 32. A duração de *g*(*t*) é de apenas 1 segundo. Devemos repetir *g*(*t*) a cada 4 segundos, como indicado na Fig. 3.41c, e tomar amostras a cada 0,125 segundo. Isso nos dará 32 amostras (*N* 0 = 32). Também temos

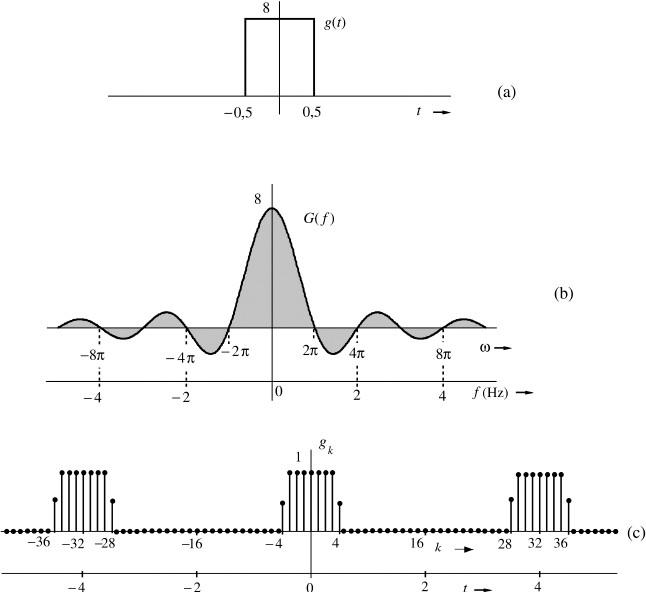


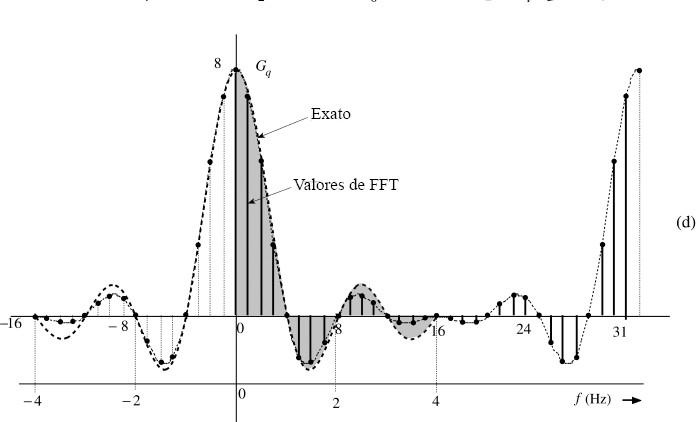
Como *g*(*t*) = 8 Π(*t*), os valores de *g k* são 1, 0 ou 0,5 (nos pontos de descontinuidade), como mostrado na Fig. 3.41c; nessa figura, por conveniência, *g k* é mostrado como função de *t* e de *k*.

Na dedução da DFT, supomos que *g*(*t*) tem início em *t* = 0 (Fig. 3.39a) e tomamos *N* 0 amostras no intervalo (0, *T* 0 ). No caso em consideração, contudo, *g*(*t*) tem início em *t* = −½. Essa dificuldade é facilmente resolvida quando observamos que a DFT obtida por este procedimento é, na verdade, a DFT de *g k* repetido a cada *T* 0 segundos. Da Fig. 3.41c, fica claro que a repetição periódica do segmento de *g k* no intervalo de −2 a 2 segundos é equivalente à repetição do segmento de *g k* no intervalo de 0 a 4 segundos. Portanto, a DFT das amostras colhidas entre −2 e 2 segundos é igual à DFT das amostras colhidas entre 0 e 4 segundos. Assim, independentemente do instante em que *g*(*t*) tem início, sempre podemos tomar as amostras de *g*(*t*) e repetilas periodicamente no intervalo de 0 a *T* 0 . No presente exemplo, os valores das 32 amostras são







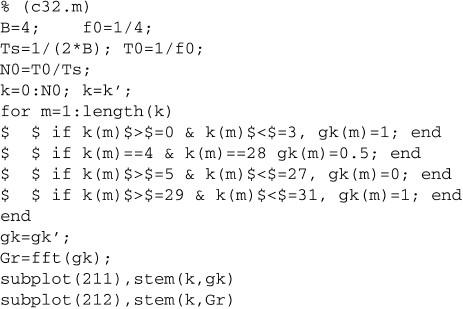


**Figura 3.41** Transformada de Fourier discreta de um pulso retangular.

Vale ressaltar que a última amostra é tomada em *t* = 31/8 e não em *t* = 4, pois a repetição do sinal reinicia em *t* = 4, de modo que a amostra em *t* = 4 é igual à amostra em *t* = 0. Com *N* 0 = 32, Ω 0 = 2*π*/32 = *π*/16. Logo, [ver a Eq. (3.103a)],



O programa MATLAB que usa o algoritmo de FFT para calcular a DFT é dado a seguir. Primeiro, escrevemos um programa MATLAB para gerar 32 amostras de *g k* e, então, calculamos a DFT.



A Fig. 3.41d mostra o gráfico de *Gq* .

As amostras *Gq* são espaçadas de *f* 0 = 1/*T* 0 Hz. Neste exemplo, *T* 0 = 4 segundos, de modo que a resolução de frequência *f* 0 é ¼ Hz, como desejado. A frequência de dobramento *f s* /2 = *B* = 4 Hz corresponde a *q = N* 0 /2 = 16. Como *Ga* tem período *N* 0 (*N* 0 = 32), os valores de *Gq* para *q* entre −16 e −1 são iguais àqueles para *q* entre 16 e 31. A DFT nos fornece amostras do espectro *G*(*f*).

Para facilitar a comparação, a Fig. 3.41d também mostra a curva hachurada 8 sinc(π*f*), que é a transformada de Fourier de 8 Π(*t*). Os valores de *Gq* calculados pela DFT exibem erro de mascaramento, o que fica claro quando comparamos os

dois gráficos. O erro em *G* 2 é da ordem de apenas 1,3%. No entanto, o erro de mascaramento aumenta rapidamente com *r*. Por exemplo, o erro em *G* 6 é de cerca de 12%, e o erro em *G* 10 , 33%. O erro em *G* 14 é de assustadores 72%. O erro percentual aumenta de forma muito rápida nas proximidades da frequência de dobramento (*r* = 16), pois *g*(*t*) tem uma descontinuidade degrau, o que faz com que *G*(*f*) decaia muito lentamente, como 1/*f*. Assim, nas proximidades da frequência de dobramento, a cauda invertida (devido ao mascaramento) é quase igual a *G*(*f*). Além disso, os valores extremos são a diferença entre os valores exato e da parte que sofreu dobra (quase iguais aos exatos). Consequentemente, o erro percentual nas proximidades da frequência de dobramento (*r* = 16, neste exemplo) é muito alto, embora o erro absoluto seja muito pequeno. Fica claro que, para sinais com descontinuidades do tipo degrau, o erro de mascaramento nas proximidades da frequência de dobramento sempre será grande (em termos percentuais), qualquer que seja o valor escolhido para *N* 0 . Para garantir erro de mascaramento desprezível para qualquer valor de *q*, devemos assegurar que *N* 0

≫ *q*. Essa observação se aplica a todos os sinais com descontinuidade do tipo degrau.



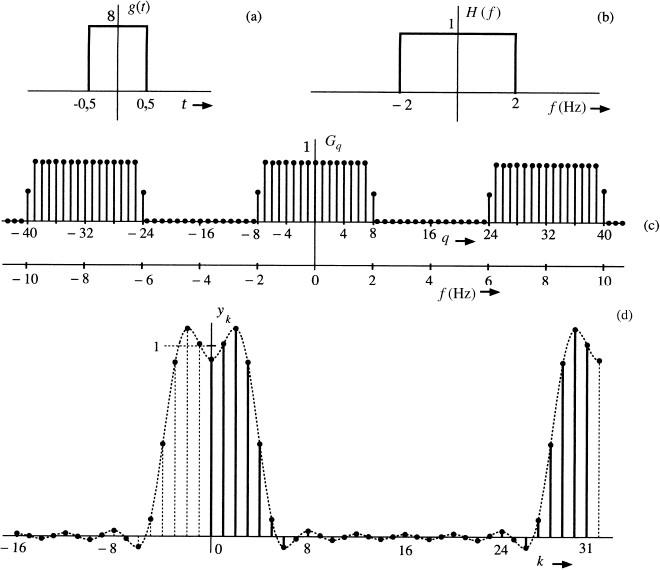
## **Filtragem**

Quando pensamos em filtragem, em geral, o fazemos em termos de uma solução orientada a hardware (ou seja, montagem de um circuito com componentes *RLC* e amplificadores operacionais). Contudo, a filtragem também admite uma solução orientada a software [algoritmo computacional que fornece a saída filtrada *y*(*t*), para uma dada entrada *g*(*t*)]. Isso pode ser implementado de modo conveniente via DFT. Seja *g*(*t*) o sinal a ser filtrado; então, os valores *Gq* , DFT de *g k* , são calculados. O espectro *Gq* é formatado (filtrado) como desejado através da multiplicação de *Gq* por *Hq* , em que *Hq* são as amostras da função de transferência do filtro, *H*(*f*) [*Hq = H*(*qf* 0 )]. Por fim, calculamos a DFT inversa (ou IDFT) de *Gq Hq* e obtemos a saída filtrada *y k* [*y k* = *T s y*(*kT*)]. O próximo exemplo ilustra este procedimento.



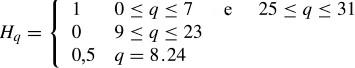
EXEMPLO COMPUTACIONAL C3.3

O sinal *g*(*t*) na Fig. 3.42a é aplicado a um filtro passa­baixos ideal, cuja função de transferência *H*(*f*) é mostrada na Fig. 3.42b. Usemos a DFT para calcular a saída do filtro.



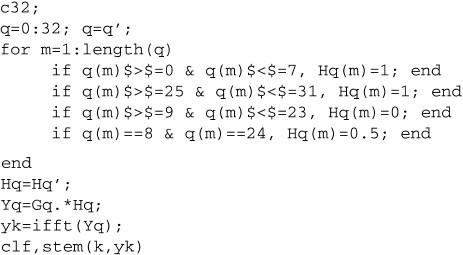
**Figura 3.42** Filtragem de *g*(*t*) por *H*(*f*).

Já calculamos a DFT de *g*(*t*) com 32 amostras (Fig. 3.41d). Agora, devemos multiplicar *Gq* por *Hq* . Para calcular *Hq* , recordemos que, na determinação da DFT de *g*(*t*) com 32 amostras, usamos *f* 0 = 0,25 Hz. Como *Gq* tem período *N* 0 = 32, *Hq* deve ter o mesmo período e, portanto, amostras espaçadas de 0,25 Hz. Isso significa que *Hq* deve se repetir a cada 8 Hz ou 16*π* rad/s (ver Fig. 3.42c). Assim, as 32 amostras de *Hq* são produzidas, no intervalo 0 ≤ *f* ≤ 8, como

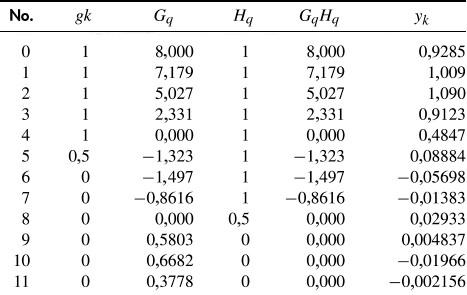


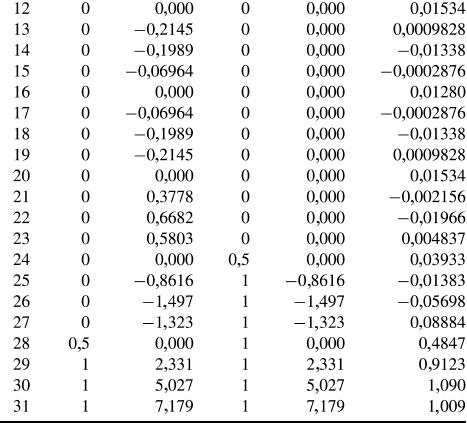
Multiplicamos *Gq* por *Hq* e calculamos a DFT inversa. O resultante sinal de saída é mostrado na Fig. 3.42d. A Tabela 3.4 lista valores de *g k* , *Gq* , *Hq* , *Y q* e *y k* .

No Exemplo C.32, já calculamos a DFT de *g*(*t*) com 32 amostras (*Gq* ). O programa MATLAB do Exemplo C3.2 pode ser armazenado como um arquivo.m (por exemplo, “c32.m”). Podemos importar *Gq* no ambiente MATLAB via comando “c32”. A seguir, geramos 32 amostras de *Hq* , multiplicamos *Gq* por *Hq* e, para obter *y k* , calculamos a DFT inversa. Também podemos obter *y k* calculando a convolução de *g k* e *h k* .



**Tabela 3.4**





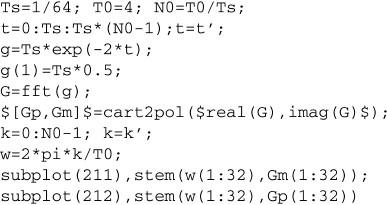


# REFERÊNCIAS

1. R. V. Churchill and J. W. Brown, *Fourier Series and Boundary Value Problems*, 3rd ed., McGraw­Hill, New York, 1978.
2. R. N. Bracewell, *Fourier Transform and Its Applications*, rev. 2nd ed., McGraw­Hill, New York, 1986.
3. B. P. Lathi, *Signal Processing and Linear Systems*, Oxford University Press, 2000.
4. E. A. Guillemin, *Theory of Linear Physical Systems*, Wiley, New York, 1963.
5. F. J. Harris, “On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform,” *Proc. IEEE*, vol. 66, pp. 51– 83, Jan. 1978.
6. J. W. Tukey and J. Cooley, “An Algorithm for the Machine Calculation of Complex Fourier Series,” *Mathematics of Computation*, Vol. 19, pp. 297–301, April 1965.

### Códigos (Imagem e transcrição)

#### Exercício 1



#### 

#### Transcript:

Ts=1/64; T0=4; N0=T0/TS;

t=0:Ts:Ts\* (N0-1); t=t';

g=Ts\*exp(-2\*t);

g(1) Ts\*0.5;

G=fft (g);

$ [Gp, Gm] $=cart2pol ($real (G), imag (G) $);

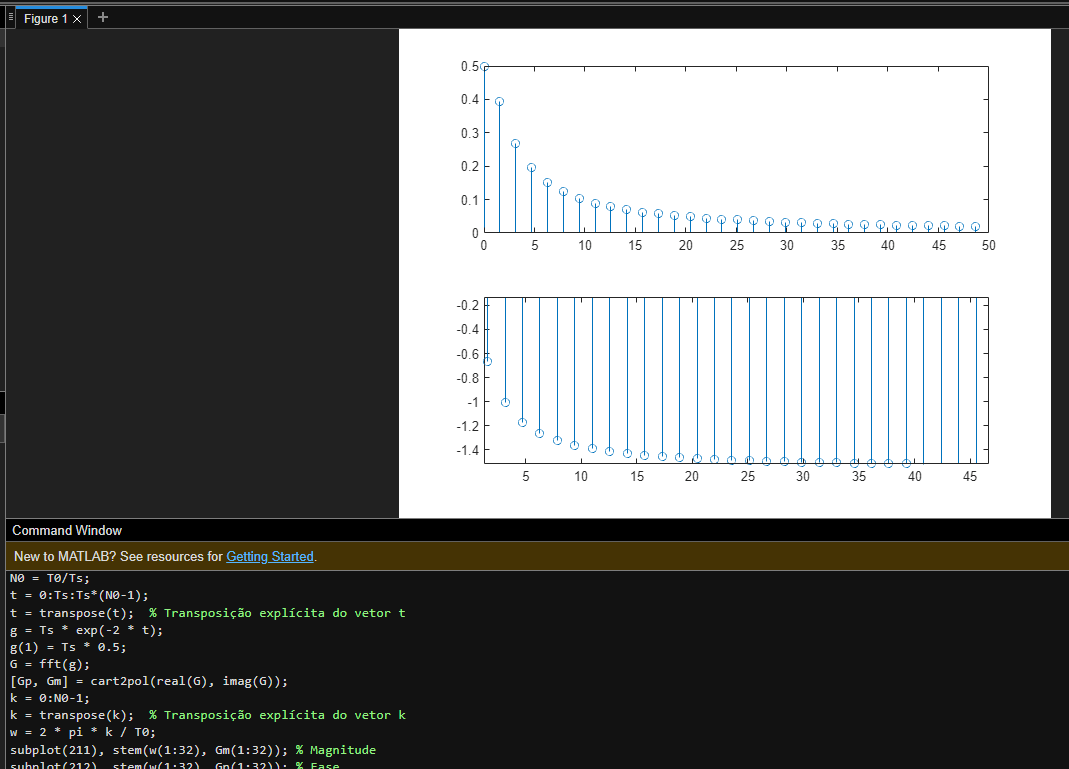
k=0:N0-1; k=k';

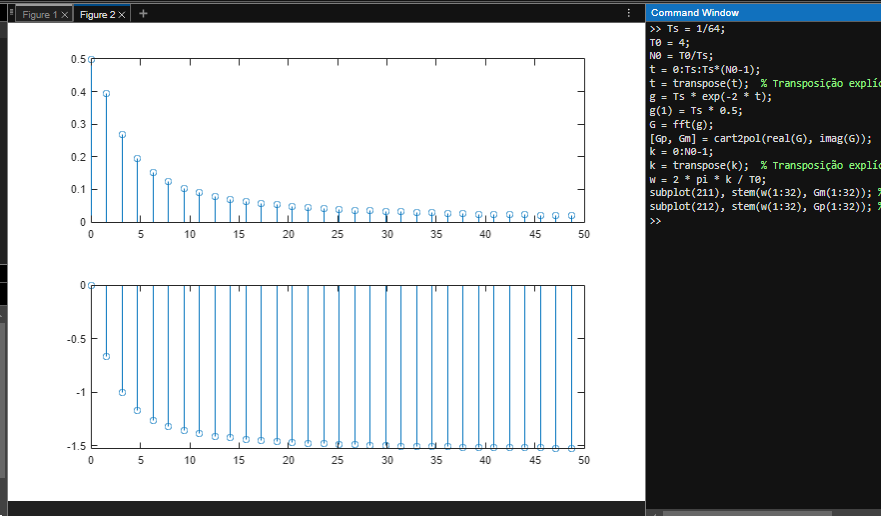
w=2\*pi\*k/TO;

subplot (211), stem (w (1:32), Gm (1:32));

subplot (212), stem (w (1:32), Gp (1:32));

##### Resultado:





##### corrigido:

Ts = 1/64;

T0 = 4;

N0 = T0/Ts;

t = 0:Ts:Ts\*(N0-1);

t = transpose(t); % Transposição explícita do vetor t

g = Ts \* exp(-2 \* t);

g(1) = Ts \* 0.5;

G = fft(g);

[Gp, Gm] = cart2pol(real(G), imag(G));

k = 0:N0-1;

k = transpose(k); % Transposição explícita do vetor k

w = 2 \* pi \* k / T0;

subplot(211), stem(w(1:32), Gm(1:32)); % Magnitude

subplot(212), stem(w(1:32), Gp(1:32)); % Fase

#### Exercício 2

#### 

#### 

#### Transcript:

﻿

B=4;

f0=1/4;

Ts=1/(2\*B); T0=1/f0;

N0=T0/Ts;

k=0:N0; k=k';

for m=1:length(k)

$ $ if k (m) $>$=0 & k (m) $<$=3, gk (m)=1; end

$ $ if k (m)==4 & k (m) ==28 gk (m)=0.5; end

$ $ if k (m) $>$=5 & k (m) $<$=27, gk (m) =0; end

$ $ if k (m) $>$=29 & k (m) $<$=31, gk (m) =1; end

end

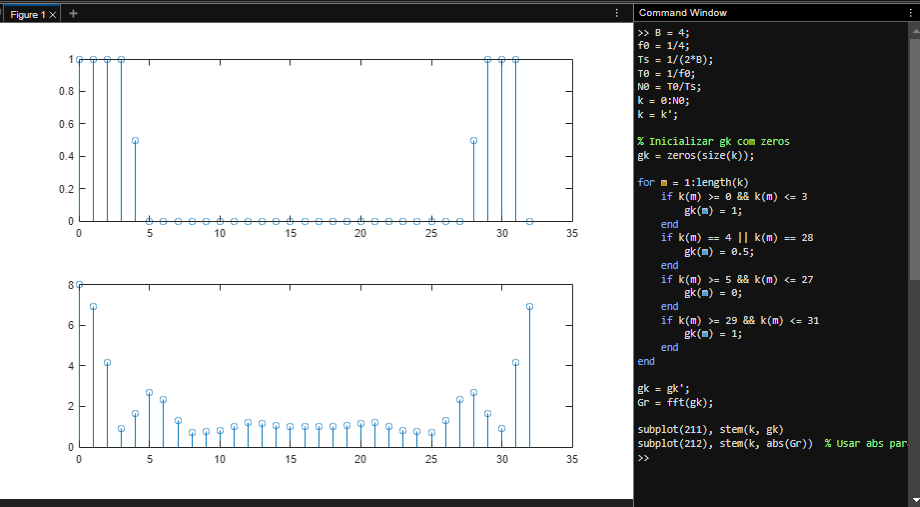
gk=gk';

Gr=fft (gk);

subplot (211), stem (k,gk)

subplot (212), stem (k, Gr)

##### Resultado:



##### Corrigido:

B = 4;

f0 = 1/4;

Ts = 1/(2\*B);

T0 = 1/f0;

N0 = T0/Ts;

k = 0:N0;

k = k';

% Inicializar gk com zeros

gk = zeros(size(k));

for m = 1:length(k)

if k(m) >= 0 && k(m) <= 3

gk(m) = 1;

end

if k(m) == 4 || k(m) == 28

gk(m) = 0.5;

end

if k(m) >= 5 && k(m) <= 27

gk(m) = 0;

end

if k(m) >= 29 && k(m) <= 31

gk(m) = 1;

end

end

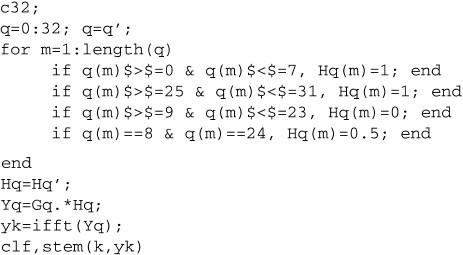
gk = gk';

Gr = fft(gk);

subplot(211), stem(k, gk)

subplot(212), stem(k, abs(Gr)) % Usar abs para a magnitude da FFT

#### Exercício 3



#### Transcript:

% c32;

q=0:32; q=q';

for m=1:length(q)

if q(m) $>$=0 & q(m) $<$=7, Hq (m)=1; end

if q(m) $>$=25 & q(m) $<$=31, Hq (m)=1; end

if q(m) $>$=9 & q(m) $<$=23, Hq (m)=0; end

if q(m) ==8 & q(m) ==24, Hq (m)=0.5; end

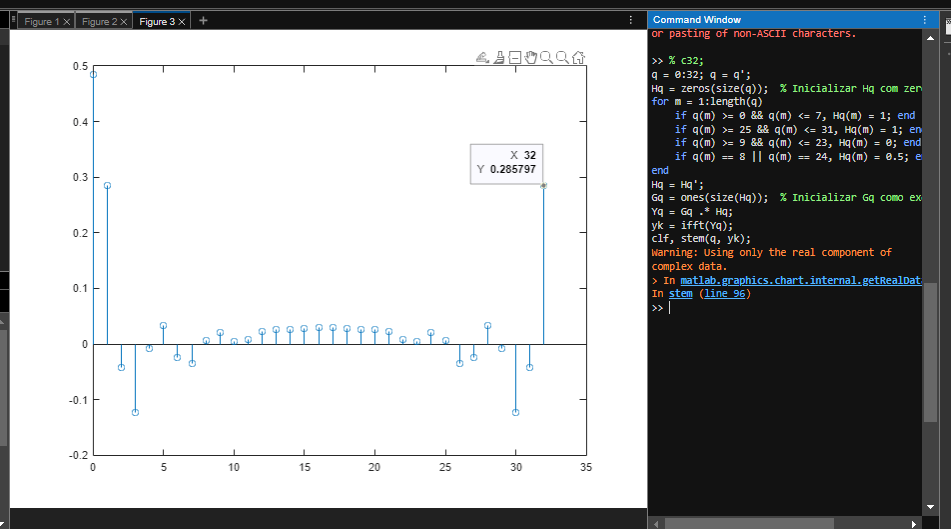
Hq=Hq';

Yq=Gq. \*Hq;

yk=ifft (Yq);

clf, stem (k, yk);

##### Resultado:



##### Corrigido:

% c32;

q = 0:32; q = q';

Hq = zeros(size(q)); % Inicializar Hq com zeros

for m = 1:length(q)

if q(m) >= 0 && q(m) <= 7, Hq(m) = 1; end

if q(m) >= 25 && q(m) <= 31, Hq(m) = 1; end

if q(m) >= 9 && q(m) <= 23, Hq(m) = 0; end

if q(m) == 8 || q(m) == 24, Hq(m) = 0.5; end

end

Hq = Hq';

Gq = ones(size(Hq)); % Inicializar Gq como exemplo

Yq = Gq .\* Hq;

yk = ifft(Yq);

clf, stem(q, yk);

—

Google Colab

Atalho - Atualizado



# Título: Execução dos Exercícios. Semestre 2024.1

## Entrege até

10/06/2024, 10:30

### Aluno(s):

 Vinícius José

### Professor:

 Diego Rátiva

# Base do Experimeto

 Livro

# Documentos - Resultado

 [Vídeo](Upar a playlist) - Atualizar  Repo

 Documento

 Colab - Atualizar! (feito)  Apresentação

# Sugestões do professor

Avaliação:

1. Compartilhe o código com [diego.rativa@ecomp.poli.br](mailto:diego.rativa@ecomp.poli.br)
2. Salve um vídeo por cada exercício de no máximo 2min explicando o exercício.
3. Salve o PDF do Colab e adicione na entrega desta tarefa.
4. Serão sorteados estudantes para apresentar os exercícios durante o horário da aula.

Explicação.

1. Não foque no código, (a disciplina não é de programação) e sim na metodología e os resultados na hora de explicar os fundamentos.
2. Não esqueça de fundamentar-se no livro.

# Execução

**!**pip install numpy

**!**pip install matplotlib

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

Requirement already satisfied: numpy in c:\users\vjfr\appdata\local\packages\pythonsoftware foundation.python.3.11\_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (1.2

5.0)

Requirement already satisfied: matplotlib in c:\users\vjfr\appdata\local\packages\pythonsof twarefoundation.python.3.11\_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (3.8.2)

Requirement already satisfied: contourpy>=1.0.1 in c:\users\vjfr\appdata\local\packages\pyt honsoftwarefoundation.python.3.11\_qbz5n2kfra8p0\localcache\local-packages\python311\site-pa ckages (from matplotlib) (1.2.0)

Requirement already satisfied: cycler>=0.10 in c:\users\vjfr\appdata\local\packages\pythons oftwarefoundation.python.3.11\_qbz5n2kfra8p0\localcache\local-packages\python311\site-packag es (from matplotlib) (0.12.1)

Requirement already satisfied: fonttools>=4.22.0 in c:\users\vjfr\appdata\local\packages\py thonsoftwarefoundation.python.3.11\_qbz5n2kfra8p0\localcache\local-packages\python311\site-p ackages (from matplotlib) (4.45.0)

Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\vjfr\appdata\local\packages\py thonsoftwarefoundation.python.3.11\_qbz5n2kfra8p0\localcache\local-packages\python311\site-p ackages (from matplotlib) (1.4.5)

Requirement already satisfied: numpy<2,>=1.21 in c:\users\vjfr\appdata\local\packages\pytho nsoftwarefoundation.python.3.11\_qbz5n2kfra8p0\localcache\local-packages\python311\site-pack ages (from matplotlib) (1.25.0)

Requirement already satisfied: packaging>=20.0 in c:\users\vjfr\appdata\local\packages\pyth onsoftwarefoundation.python.3.11\_qbz5n2kfra8p0\localcache\local-packages\python311\site-pac kages (from matplotlib) (23.1)

Requirement already satisfied: pillow>=8 in c:\users\vjfr\appdata\local\packages\pythonsoft warefoundation.python.3.11\_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from matplotlib) (9.5.0)

Requirement already satisfied: pyparsing>=2.3.1 in c:\users\vjfr\appdata\local\packages\pyt honsoftwarefoundation.python.3.11\_qbz5n2kfra8p0\localcache\local-packages\python311\site-pa ckages (from matplotlib) (3.1.1)

Requirement already satisfied: python-dateutil>=2.7 in c:\users\vjfr\appdata\local\packages

\pythonsoftwarefoundation.python.3.11\_qbz5n2kfra8p0\localcache\local-packages\python311\sit e-packages (from matplotlib) (2.8.2)

Requirement already satisfied: six>=1.5 in c:\users\vjfr\appdata\local\packages\pythonsoftw arefoundation.python.3.11\_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages

(from python-dateutil>=2.7->matplotlib) (1.16.0)

## Exercício 1

### Base Matemática

Utilização da Transformada de Fourier.

*N*−1

*X*[*f*] = ∑ *x*[*t*] ⋅ *e*

*n*=0

−*j* 2*π nk*

In [73]:

*# Execução: Parâmetros*

Ts **=** 1**/**64 T0 **=** 4

N0 **=** int(T0 **/** Ts)

*# Vetor de tempo*

t **=** np**.**arange(0, Ts **\*** N0, Ts)

*# Sinal g*

g **=** Ts **\*** np**.**exp(**-**2 **\*** t) g[0] **=** Ts **\*** 0.5

*# Transformada de Fourier*

G **=** np**.**fft**.**fft(g)

*# Conversão para coordenadas polares*

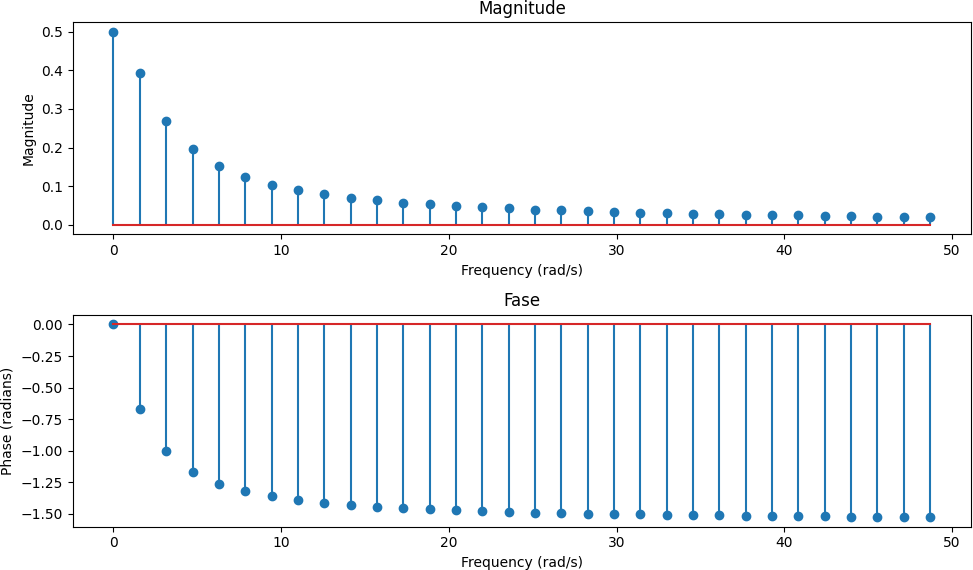
Gp **=** np**.**angle(G) Gm **=** np**.**abs(G)

*# Vetor k e frequência angular*

k **=** np**.**arange(N0)

w **=** 2 **\*** np**.**pi **\*** k **/** T0

In [74]:



## Exercício 2

### Base Matemática

Utilização de conceitos de amostragem e DFT para calcular a DFT de um sinal discreto definido por partes.

*N*−1

*G*[*k*] = ∑ *g*[*n*] ⋅ *e*

*n*=0

−*j* 2*π nk*



In [75]: *# Execução: Parâmetros*

f0 **=** 1**/**4

Ts **=** 1 **/** (2 **\*** B) T0 **=** 1 **/** f0

N0 **=** int(T0 **/** Ts)

*# Vetor k*

k **=** np**.**arange(N0 **+** 1)

*# Inicializar gk com zeros*

gk **=** np**.**zeros\_like(k)

**for** m **in** range(len(k)):

**if** 0 **<=** k[m] **<=** 3:

gk[m] **=** 1

**if** k[m] **==** 4 **or** k[m] **==** 28: gk[m] **=** 0.5

**if** 5 **<=** k[m] **<=** 27:

gk[m] **=** 0

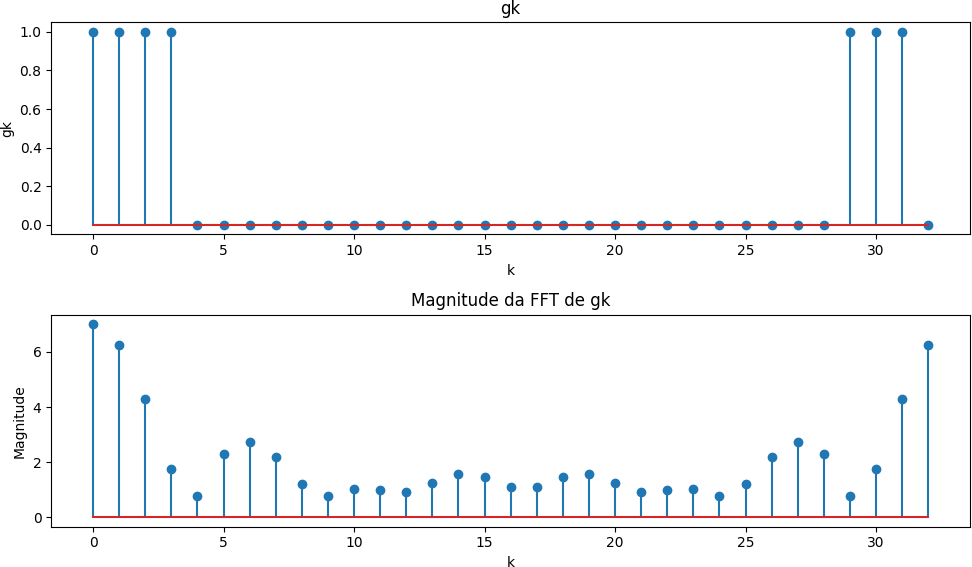
**if** 29 **<=** k[m] **<=** 31:

gk[m] **=** 1

*# Transformada de Fourier*

Gr **=** np**.**fft**.**fft(gk)

In [76]:



## Exercício 3

### Base matemática

Aplicação de conceitos de filtros no domínio da frequência.

*Y* [*k*] = *H*[*k*] ⋅ *G*[*k*]

In [77]:

In [78]: