Trabalho Sinais e Sistemas

September 13, 2024

1 Tranformada Discreta de Fourier - DFT

Aluna: Luísa Brito Dias - 20.1.8107

O objetivo deste trabalho foi implementar, do zero, os códigos da Transformada Discreta de Fourier (DFT) e sua inversa (IDFT). Para isso, foi desenvolvido os algoritmos seguindo as fórmulas matemáticas, utilizando apenas operações básicas com números complexos. Depois de implementados, foram testados os códigos com sinais amostrados que foram fornecidos, para verificar se estavam funcionando corretamente. Comparamos os sinais no tempo e na frequência, e conseguimos mostrar que os algoritmos desenvolvidos foram eficazes, j á q ue a r econstrução d o sinal original a partir da sua versão no domínio da frequência funcionou bem.

A Transformada Discreta de Fourier (DFT) converte sinais discretos do domínio do tempo para o domínio da frequência, permitindo a análise das componentes de frequência que formam o sinal.

Transformada Discreta de Fourier:

$$X(m) = \sum_{n=0}^{N-1} x(n) \left[\cos\left(2\pi n \frac{m}{N}\right) - j \sin\left(2\pi n \frac{m}{N}\right)\right]$$

[75]:
```
# Inicializando as bibliotecas utilizadas
import numpy as np
import matplotlib.pyplot as plt
```

[76]:
```
def calcular_dft(sinal_amostral):
    # N é o número total de amostras no sinal
    N = len(sinal_amostral)
    # Inicializa a lista que armazenará os resultados da DFT
    dft = []

    # Laço externo: percorre cada valor de m (índice da frequência)
    for m in range(N):
        # Inicializa a variável X para armazenar o valor complexo da DFT para a␣
 ↪frequência m
        X = 0
        # Inicializa as variáveis para as partes real e imaginária da soma
        real = 0
        imaginario = 0
```

```
        # Laço interno: percorre cada valor de n (índice temporal ou de amostras do sinal)
        for n in range(N):
            # Calcula o ângulo correspondente para o termo da exponencial complexa
            angulo = 2 * np.pi * n * m / N
            # Atualiza a parte real da DFT usando cosseno
            real = real + (sinal_amostral[n] * np.cos(angulo))
            # Atualiza a parte imaginária da DFT usando seno
            imaginario = imaginario + (sinal_amostral[n] * np.sin(angulo))
            # Calcula o valor complexo combinando a parte real e imaginária (imaginário multiplicado por -1j)
            X = real - (imaginario * 1j)

        # Adiciona o valor de X (o resultado da DFT para a frequência m) na lista de resultados
        dft.append(X)

    # Retorna a lista com os valores da DFT para todas as frequências
    return dft
```

A Inversa da Transformada Discreta de Fourier (IDFT) reconstrói o sinal original no domínio do tempo a partir de suas componentes de frequência obtidas pela DFT. Ela faz o processo inverso, convertendo o sinal de volta do domínio da frequência para o tempo.

Transformada Discreta de Fourier Inversa:

$$x(n) = \frac{1}{N} \sum_{m=0}^{N-1} X(m) \left[\cos\left(2\pi \frac{mn}{N}\right) + j \sin\left(2\pi \frac{mn}{N}\right) \right]$$

[77]:
```
def calcular_idft(dft):
    # N é o número total de componentes da DFT (tamanho do sinal original)
    N = len(dft)
    # Inicializa a lista que armazenará os resultados da IDFT
    idft = []

    # Laço externo: percorre cada valor de n (índice temporal, ou seja, cada amostra do sinal reconstruído)
    for n in range(N):
        # Inicializa a variável X para armazenar o valor resultante da IDFT para o índice temporal n
        X = 0

        # Laço interno: percorre cada valor de m (índice de frequência da DFT)
        for m in range(N):
```

```
            # Calcula o ângulo correspondente para o termo da exponencial␣
↪complexa
            angulo = 2 * np.pi * m * n / N
            # Atualiza X somando a parte real e subtraindo a parte imaginária␣
↪do termo da DFT
            # Para a IDFT, usamos cosseno para a parte real e seno para a parte␣
↪imaginária
            X = X + ((dft[m].real * np.cos(angulo)) - (dft[m].imag * np.
↪sin(angulo)))

        # Multiplicamos o resultado por 1/N para concluir a IDFT (normalização)
        idft.append(X / N)

    # Ajusta os valores muito próximos de zero (erro numérico) para 0
    idft = [0 if abs(x) < 1e-10 else x for x in idft]

    # Retorna a lista com o sinal reconstruído a partir da IDFT
    return idft
```

1ª Amostra:

$x[n] = [0, 6.1982, 4.6707, 5.6899, 4.7456, 6.4052, 10.4355, 10.5058, 10.4316, 10.5058, 10.4355, 6.4052, 4.7456, 5.6899, 4.$

Frequência de amostragem:

fs = 1920 Hz

[78]:
```
# Sinal amostral dado e a frequência de amostragem (fs)
sinal_amostral = [0, 6.1982, 4.6707, 5.6899, 4.7456, 6.4052, 10.4355, 10.5058,␣
↪10.4316, 10.5058, 10.4355, 6.4052, 4.7456, 5.6899, 4.6707, 6.1982, 0.0000,␣
↪-6.1982, -4.6707, -5.6899, -4.7456, -6.4052, -10.4355, -10.5058, -10.4316,␣
↪-10.5058, -10.4355, -6.4052, -4.7456, -5.6899, -4.6707, -6.1982]

fs = 1920 # Frequência de amostragem do sinal em Hz (1920 amostras por segundo)
```

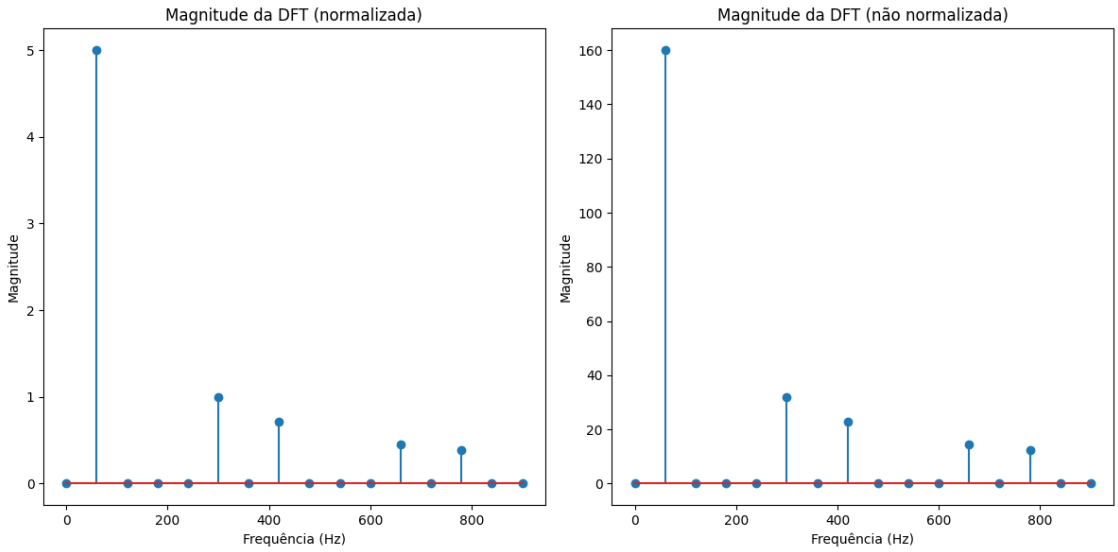
Magnitude e fase da DFT:

[79]:
```
# Função para calcular a DFT
dft = calcular_dft(sinal_amostral)
dft = np.array(dft)  # Converte o resultado da DFT para um array NumPy para␣
↪facilitar o processamento

# Calcula a magnitude e a fase da DFT
magnitude = np.abs(dft)
fase = np.angle(dft)

# Número de amostras no sinal
```

```
N = len(sinal_amostral)

# Calcula as frequências correspondentes para a DFT
frequencias = np.fft.fftfreq(N, 1/fs)

# Normalizar a DFT fora da função
resultado_dft_normalizado = dft / N

# Obter a magnitude e a fase da DFT normalizada
magnitude_normalizada = np.abs(resultado_dft_normalizado)
```

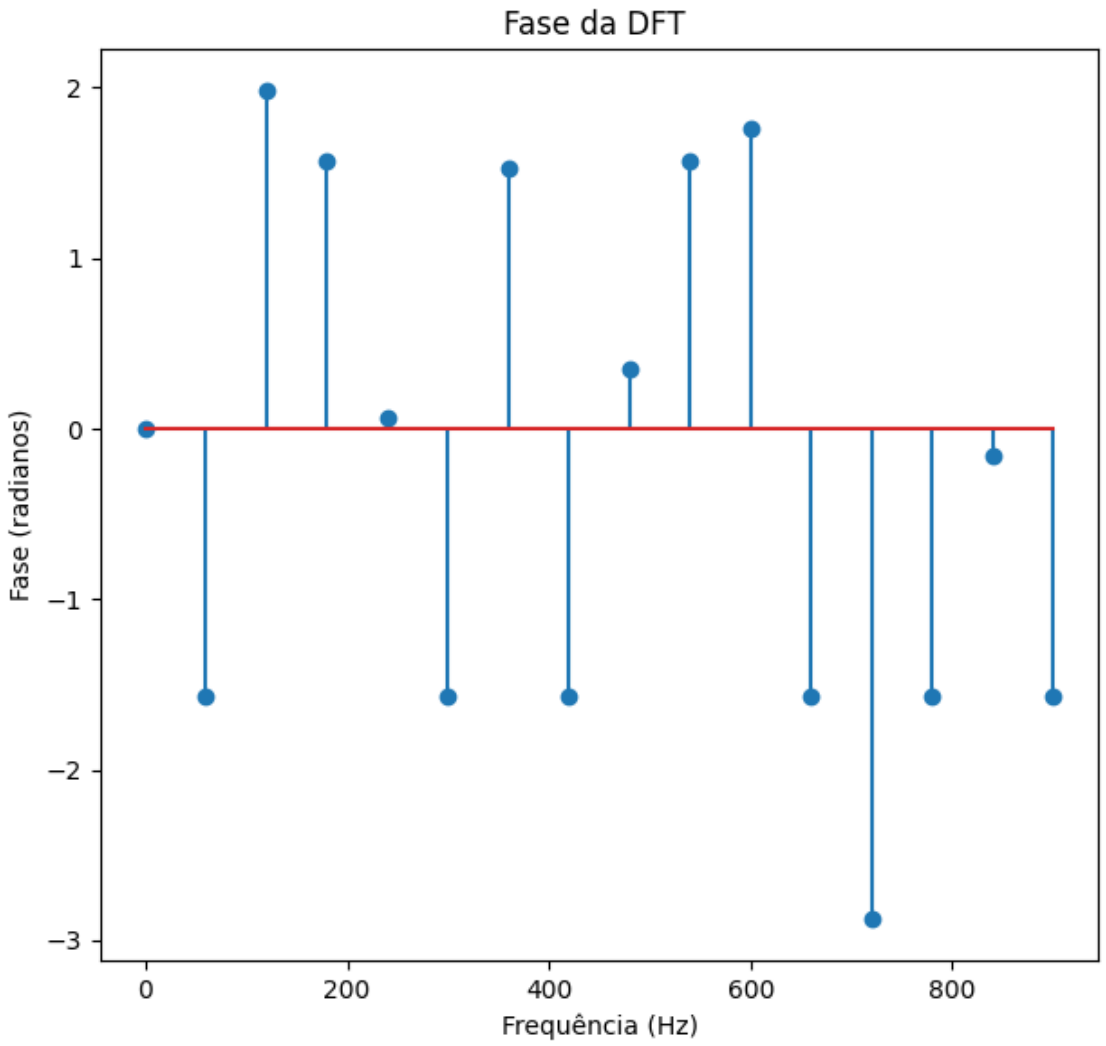
Magnitude:

[80]:
```
# Criação da figura com um tamanho de 12x6 polegadas para os gráficos
plt.figure(figsize=(12, 6))

# Primeiro gráfico: Magnitude da DFT normalizada
plt.subplot(1, 2, 1)  # Define o primeiro subplot (1 linha, 2 colunas, primeiro gráfico)
# Plota a magnitude apenas para as frequências positivas (até N//2)
plt.stem(frequencias[:N//2], magnitude_normalizada[:N//2])
plt.title('Magnitude da DFT (normalizada)')  # Título do gráfico
plt.xlabel('Frequência (Hz)')  # Rótulo do eixo X
plt.ylabel('Magnitude')  # Rótulo do eixo Y

# Segundo gráfico: Magnitude da DFT não normalizada
plt.subplot(1, 2, 2)  # Define o primeiro subplot (1 linha, 2 colunas, primeiro gráfico)
# Plota a magnitude apenas para as frequências positivas (até N//2)
plt.stem(frequencias[:N//2], magnitude[:N//2])
plt.title('Magnitude da DFT (não normalizada)')  # Título do gráfico
plt.xlabel('Frequência (Hz)')  # Rótulo do eixo X
plt.ylabel('Magnitude')  # Rótulo do eixo Y

# Ajusta o layout da figura para evitar sobreposição de elementos
plt.tight_layout()

# Exibe a figura com os gráficos de Magnitude e Fase
plt.show()
```



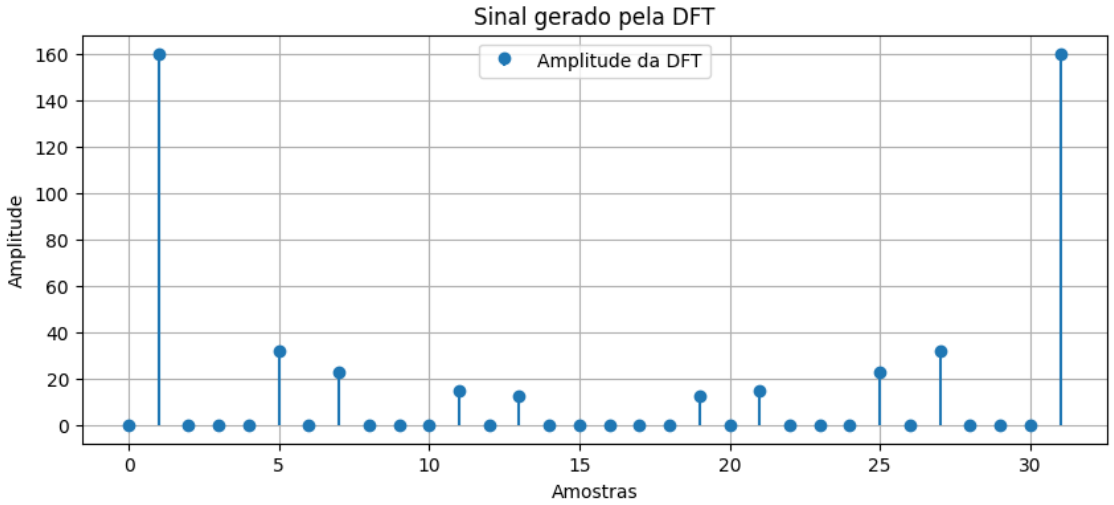
Fase:

[81]:
```
# Criação da figura com um tamanho de 12x6 polegadas para os gráficos
plt.figure(figsize=(12, 6))

# Segundo gráfico: Fase da DFT
plt.subplot(1, 2, 2)  # Define o segundo subplot (1 linha, 2 colunas, segundo gráfico)
# Plota a fase apenas para as frequências positivas (até N//2)
plt.stem(frequencias[:N//2], fase[:N//2])
plt.title('Fase da DFT ')  # Título do gráfico
plt.xlabel('Frequência (Hz)')  # Rótulo do eixo X
plt.ylabel('Fase (radianos)')  # Rótulo do eixo Y

# Ajusta o layout da figura para evitar sobreposição de elementos
plt.tight_layout()

# Exibe a figura com os gráficos de Magnitude e Fase
plt.show()
```



Sinal gerado pela DFT:

[82]:
```
# Cria um gráfico da DFT (Amplitude x Amostras)
plt.figure(figsize=(10, 4))
plt.stem(magnitude, label="Amplitude da DFT", basefmt=" ")  # Plota a magnitude␣
↪da DFT
plt.title("Sinal gerado pela DFT")
plt.xlabel("Amostras")
plt.ylabel("Amplitude")
plt.legend()
plt.grid(True)
plt.show()
```

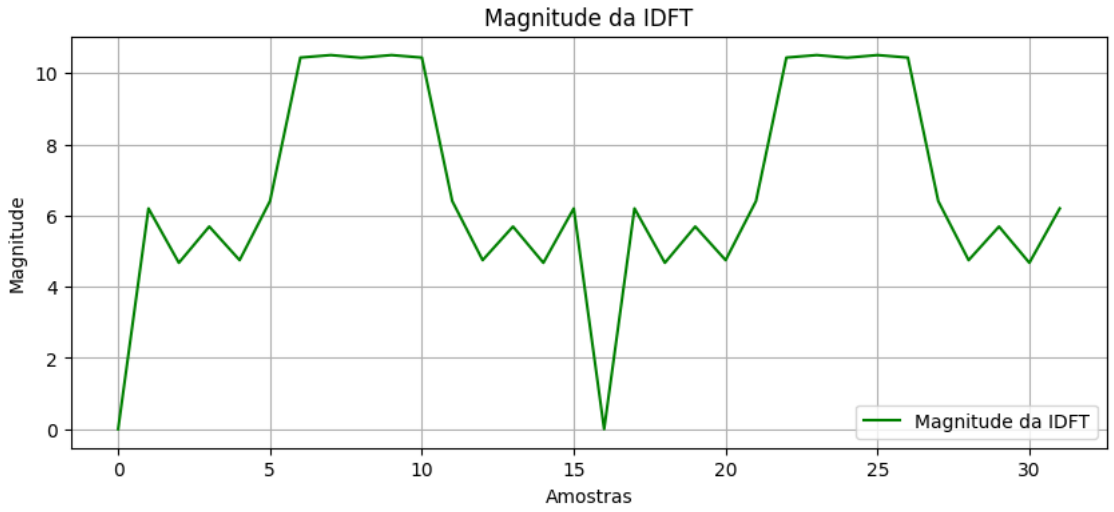


Transformada Discreta de Fourier Inversa:

[83]:
```
# Calcula a IDFT (Inversa da DFT) para reconstruir o sinal no domínio do tempo
idft = calcular_idft(dft)
```

Magnitude de IDFT:

[84]:
```
# Calcula a magnitude dos valores da IDFT
magnitude_idft = np.abs(idft)

# Cria um gráfico da magnitude da IDFT
plt.figure(figsize=(10, 4))
plt.plot(magnitude_idft, label="Magnitude da IDFT", color='green')  # Plota a␣
 ↪magnitude da IDFT
plt.title("Magnitude da IDFT")
plt.xlabel("Amostras")
plt.ylabel("Magnitude")
plt.legend()
plt.grid(True)
plt.show()
```



Sinal reconstruído pela IDFT:

[85]:
```
# Cria um gráfico da IDFT (Amplitude x Amostras)
plt.figure(figsize=(10, 4))
plt.plot(np.real(idft), label="Sinal reconstruído (IDFT)", color='blue')  #␣
 ↪Plota a parte real da IDFT
plt.title("Sinal reconstruído pela IDFT")
plt.xlabel("Amostras")
plt.ylabel("Amplitude")
plt.legend()
plt.grid(True)
plt.show()
```

2ª Amostra:

$x[n] = [0, 3.1265, 2.2714, 2.9741, 2.0743, 4.5833, 13.3536, 12.8402, 13.3205, 12.8402, 13.3536, 4.5833, 2.0743, 2.9741, 2.$

Frequência de amostragem:

fs = 1920 Hz

[86]:
```
# Sinal amostral dado e a frequência de amostragem (fs)
sinal_amostral2 = [0, 3.1265, 2.2714, 2.9741, 2.0743, 4.5833, 13.3536, 12.8402,␣
 ↪13.3205, 12.8402, 13.3536, 4.5833, 2.0743, 2.9741, 2.2714, 3.1265, 0, -3.
 ↪1265, -2.2714, -2.9741, -2.0743, -4.5833, -13.3536, -12.8402, -13.3205, -12.
 ↪8402, -13.3536, -4.5833, -2.0743, -2.9741, -2.2714, -3.1265]

# Frequência de amostragem do sinal em Hz (1920 amostras por segundo)
fs2 = 1920
```

Magnitude e fase da DFT:

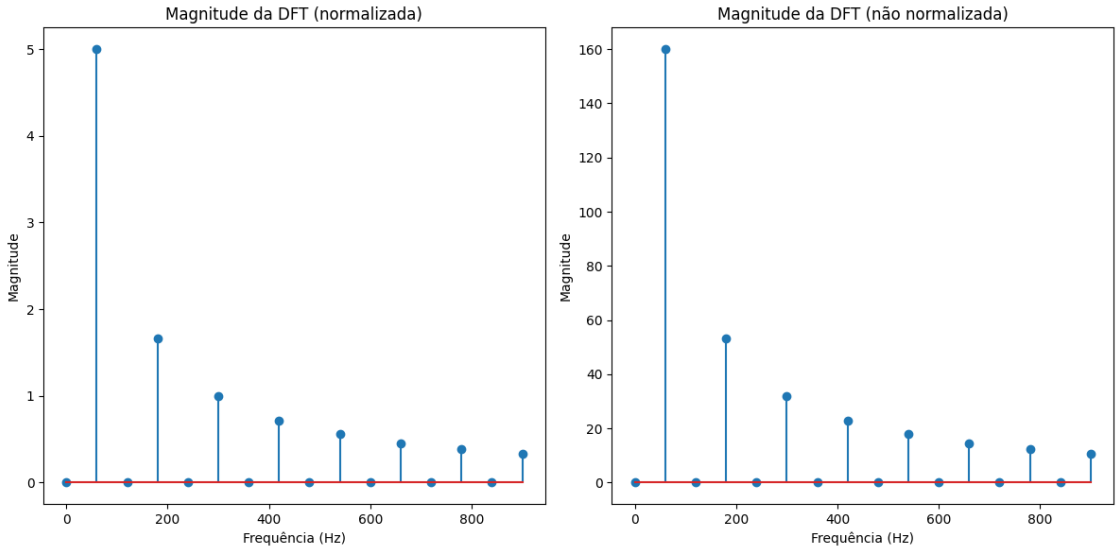
[87]:
```
# Função para calcular a DFT
dft2 = calcular_dft(sinal_amostral2)
dft2 = np.array(dft2)  # Converte o resultado da DFT para um array NumPy para␣
 ↪facilitar o processamento

# Calcula a magnitude e a fase da DFT
magnitude2 = np.abs(dft2)
fase2 = np.angle(dft2)

# Número de amostras no sinal
N2 = len(sinal_amostral2)
print(N2)

# Calcula as frequências correspondentes para a DFT
frequencias2 = np.fft.fftfreq(N2, 1/fs)

# Normalizar a DFT fora da função
resultado_dft_normalizado2 = dft2 / N2

# Obter a magnitude e a fase da DFT normalizada
magnitude_normalizada2 = np.abs(resultado_dft_normalizado2)
```

```
32
```

Magnitude:

[88]:
```
# Criação da figura com um tamanho de 12x6 polegadas para os gráficos
plt.figure(figsize=(12, 6))
```

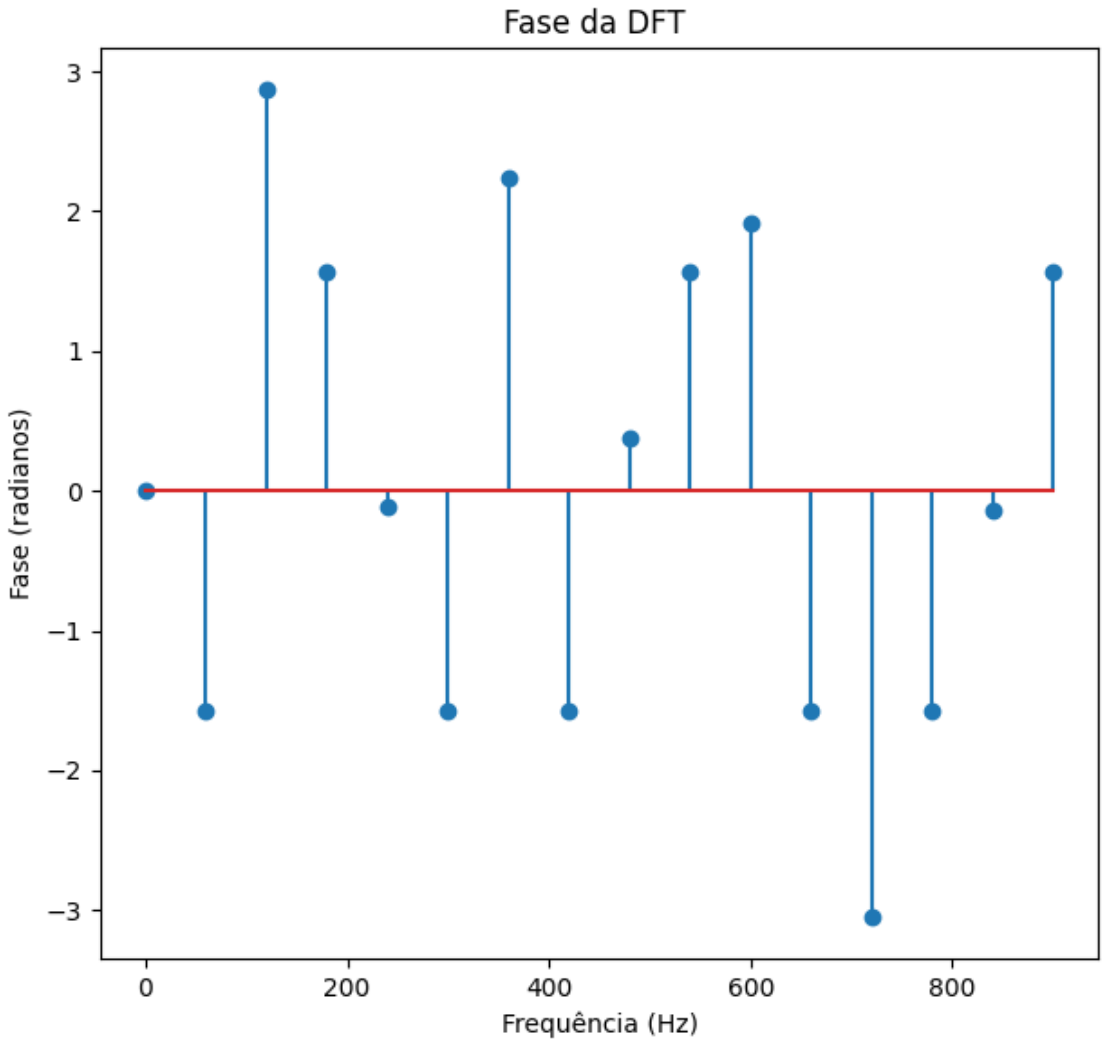
```
# Primeiro gráfico: Magnitude da DFT normalizada
plt.subplot(1, 2, 1)  # Define o primeiro subplot (1 linha, 2 colunas, primeiro gráfico)
# Plota a magnitude apenas para as frequências positivas (até N//2)
plt.stem(frequencias2[:N2//2], magnitude_normalizada2[:N2//2])
plt.title('Magnitude da DFT (normalizada)')  # Título do gráfico
plt.xlabel('Frequência (Hz)')  # Rótulo do eixo X
plt.ylabel('Magnitude')  # Rótulo do eixo Y

# Segundo gráfico: Magnitude da DFT não normalizada
plt.subplot(1, 2, 2)  # Define o primeiro subplot (1 linha, 2 colunas, primeiro gráfico)
# Plota a magnitude apenas para as frequências positivas (até N//2)
plt.stem(frequencias2[:N2//2], magnitude2[:N2//2])
plt.title('Magnitude da DFT (não normalizada)')  # Título do gráfico
plt.xlabel('Frequência (Hz)')  # Rótulo do eixo X
plt.ylabel('Magnitude')  # Rótulo do eixo Y

# Ajusta o layout da figura para evitar sobreposição de elementos
plt.tight_layout()

# Exibe a figura com os gráficos de Magnitude e Fase
plt.show()
```



Fase:

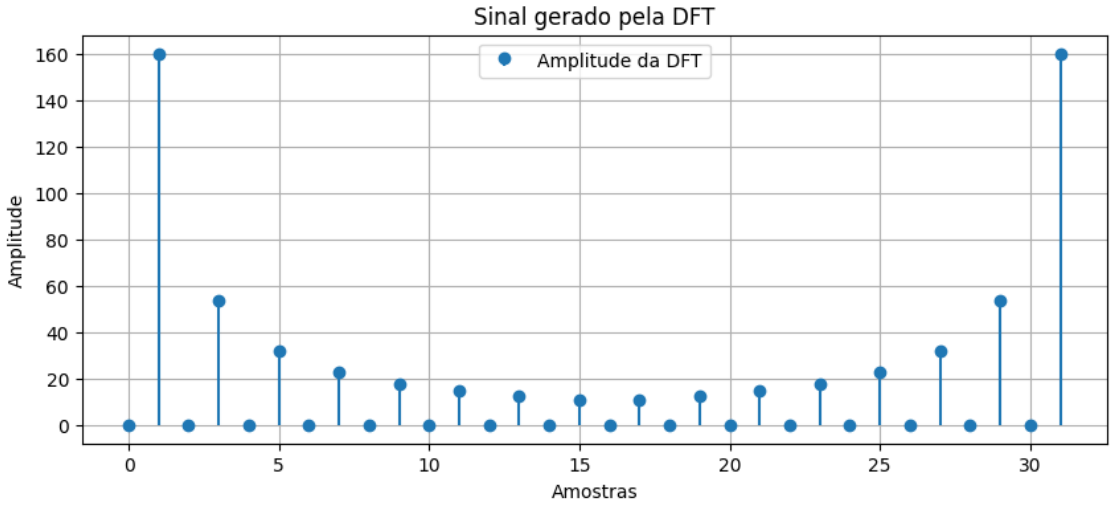
[89]:
```
# Criação da figura com um tamanho de 12x6 polegadas para os gráficos
plt.figure(figsize=(12, 6))
```

```
# Segundo gráfico: Fase da DFT
plt.subplot(1, 2, 2)  # Define o segundo subplot (1 linha, 2 colunas, segundo␣
 ↪gráfico)
# Plota a fase apenas para as frequências positivas (até N//2)
plt.stem(frequencias2[:N2//2], fase2[:N2//2])
plt.title('Fase da DFT ')  # Título do gráfico
plt.xlabel('Frequência (Hz)')  # Rótulo do eixo X
plt.ylabel('Fase (radianos)')  # Rótulo do eixo Y

# Ajusta o layout da figura para evitar sobreposição de elementos
plt.tight_layout()

# Exibe a figura com os gráficos de Magnitude e Fase
plt.show()
```



Sinal gerado pela DFT:

[90]:
```
# Cria um gráfico da DFT (Amplitude x Amostras)
plt.figure(figsize=(10, 4))
plt.stem(magnitude2, label="Amplitude da DFT", basefmt=" ")  # Plota a␣
↪magnitude da DFT
plt.title("Sinal gerado pela DFT")
plt.xlabel("Amostras")
plt.ylabel("Amplitude")
plt.legend()
plt.grid(True)
plt.show()
```



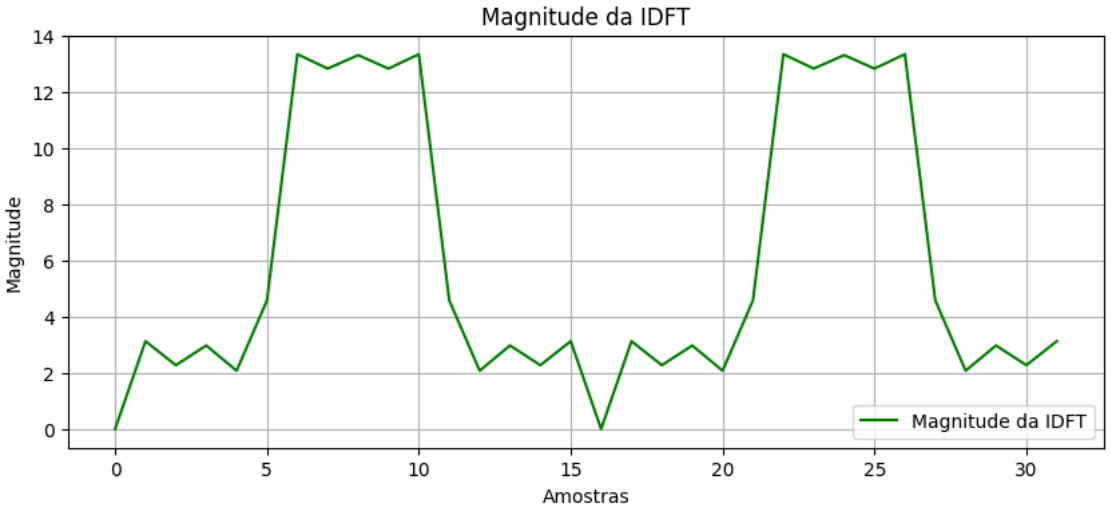
Transformada Discreta de Fourier Inversa:

[91]:
```
# Calcula a IDFT (Inversa da DFT) para reconstruir o sinal no domínio do tempo
idft2 = calcular_idft(dft2)
```

Magnitude de IDFT:

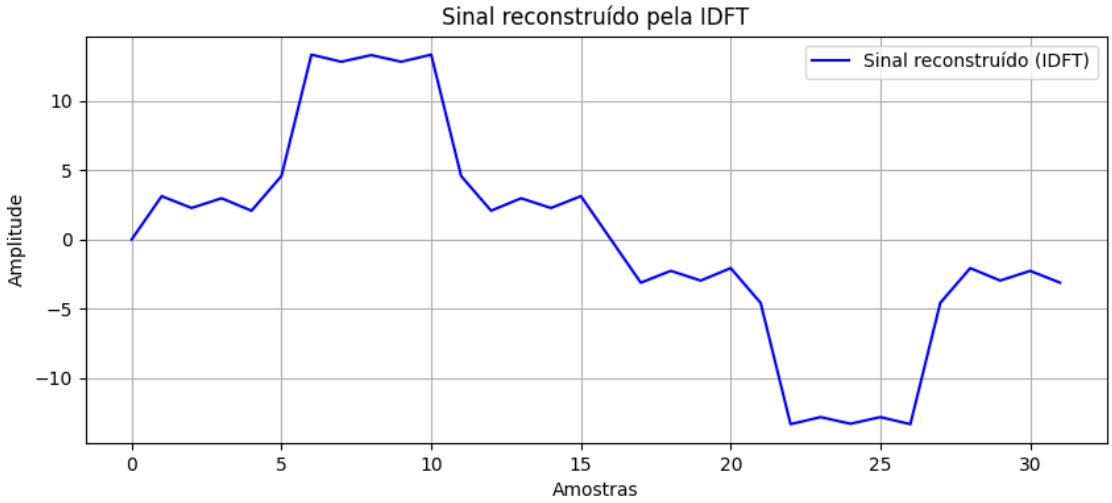
[92]:
```
# Calcula a magnitude dos valores da IDFT
magnitude_idft2 = np.abs(idft2)

# Cria um gráfico da magnitude da IDFT
plt.figure(figsize=(10, 4))
plt.plot(magnitude_idft2, label="Magnitude da IDFT", color='green')  # Plota a␣
↪magnitude da IDFT
plt.title("Magnitude da IDFT")
plt.xlabel("Amostras")
```

```
plt.ylabel("Magnitude")
plt.legend()
plt.grid(True)
plt.show()
```



Sinal reconstruído pela IDFT:

[93]:
```
# Cria um gráfico da IDFT (Amplitude x Amostras)
plt.figure(figsize=(10, 4))
plt.plot(np.real(idft2), label="Sinal reconstruído (IDFT)", color='blue')  #␣
 ↪Plota a parte real da IDFT
plt.title("Sinal reconstruído pela IDFT")
plt.xlabel("Amostras")
plt.ylabel("Amplitude")
plt.legend()
plt.grid(True)
plt.show()
```


Sinal reconstruído pela IDFT
Sinal reconstruído (IDFT)
Amplitude
Amostras
10
5
0
−5
−10
0
5
10
15
20
25
30