Projeto 01: Análise Espectral Por Transformada de Fourier

7600073 - Física Estatística Computacional - 2024/01
23/03/2024

Prof. Dr. Francisco Castilho Alcaraz
Guilherme Santana de Almeida (12694668)

Resumo

Nesta prática montaremos o algorítimo conhecido como Transformada Discreta de Fourier (DFT). Basicamente, queremos transformar sinais de funções periódicas do seu espaço temporal para um espaço de frequências, onde podemos analisar o sinal por outra perspectiva. Também queremos estudar alguns aspectos e fenômenos importantes desse algorítimo e da matemática por trás.

Introdução

Sem cerimônias, o algorítimo da transformada de Fourier é escrito de forma compacta através da seguinte expressão discreta, onde N é o número de iterações e Y_m é o nosso sinal no espaço de frequências e y_n no espaço temporal.

$$Y_m = \sum_{n=0}^{N-1} y_n e^{2\pi mn/N} \tag{1}$$

Ou seja, estamos fazendo um loop duplo, já que para cada m do sinal espectral, precisamos percorrer todo nosso sinal original y_n. E a transformada inversa, para recuperarmos esse sinal, é:

$$y_n = \frac{1}{N} \sum_{m=0}^{N-1} Y_m e^{-2\pi mn/N} \tag{2}$$

Detalhes sobre a origem dessas expressões se encontram na referência e são redundantes aqui.

Tarefa 1

I - Faça um programa que calcule de forma direta (sem qualquer método de aceleração ou otimização) transformadas de Fourier discreta, assim como transformada inversa, para uma série temporal de N dados obtidos em intervalo temporal Δt. A série temporal a ser analizada será dada em um arquivo "data.in" e a resultante colocada em "data.out".

Para facilitar, coloquei todo o algorítimo em um *MODULE*, chamado *utils*. Dentro, tenho três rotinas: Fourier, InvFourier e GerarSerie. Cada uma faz exatamente o que o nome sugere.

A rotina GerarSerie cria uma série temporal usando a expressão

$$a_1 \cos(\omega_1 t_i) + a_2 \sin(\omega_2 t_i),\ t_i = 0, 1, 2, ..., N-1, \tag{3}$$

em intervalos de Δt, e salvando no arquivo "data.in". Ou seja, os argumentos da rotina são: N, Δt, a_1, a_2, ω_1 e ω_2.

Já Fourier, executa o algorítimo Discrete Fourier Transform (DFT) para gerar uma série espectral, salvando no arquivo 'data.out', tendo como argumentos apenas N e Δt. O Δt como argumento é opcional e não precisaria ser implementado no código, tendo em vista que as únicas informações necessárias são o tamanho do meu sinal, N, e os valores de amplitude da minha série temporal, y_n. Mas como gostaria de armazenar a frequência, e o tempo no caso da transformada inversa, preciso da informação do Δt.

E por fim, a rotina InvFourier é análoga à rotina Fourier, gerando de volta a série temporal original e salvando no arquivo 'datainv.out', para não se preocupar com arquivos sobrepostos e rodar o programa apenas uma vez.

Como dito antes, as únicas informações necessárias para realizar o algorítimo são o tamanho do sinal, N, e a amplitude do mesmo. A amplitude resultante/necessária da transformada/transformada inversa é um número complexo. Por isso, salvo na primeira coluna o valor da tupla de complexos gerada pelo FORTRAN, e outras informações nas colunas restantes (detalhes em comentários no código). Assim, posso ler a primeira coluna e ignorar as outras, isso ajuda a diminuir a quantidade de variáveis *dummy* e deixa o código mais eficiente e limpo.

```fortran
MODULE utils
  INTEGER      , PARAMETER, PUBLIC    :: dp = SELECTED_REAL_KIND(p=15)
  REAL(dp)     , PARAMETER, PUBLIC    :: PI = 4d0*ATAN(1d0)
  COMPLEX(dp) , PARAMETER, PRIVATE   :: i  = (0d0,1d0)

  CONTAINS
  SUBROUTINE Fourier(N, Dt)
    INTEGER , INTENT(IN)   :: N
    REAL(dp), INTENT(IN)   :: Dt

    COMPLEX(dp) :: ym(N)
    REAL(dp)    :: yn(N)
    INTEGER     :: j, k

    ym = (0,0)

    OPEN(UNIT=1, FILE="data.in", STATUS="UNKNOWN")
    OPEN(UNIT=2, FILE="data.out", STATUS="UNKNOWN")

    DO j=1, N
      READ(1,*) yn(j)
    END DO

    DO j=1, N
      DO k=1, N
        ym(j) = ym(j) + yn(k)*ZEXP(2*PI*i*(j-1)*(k-1)/N)
      END DO
      !! Salvando: tupla de complexo | iteração | frequencia | parte real | parte imaginária
      WRITE(2,*) ym(j), j, (j-1)/(N*Dt), REALPART(ym(j)), IMAGPART(ym(j))
    END DO

    CLOSE(1)
    CLOSE(2)
  END SUBROUTINE

  SUBROUTINE InvFourier(N, Dt)
    INTEGER    , INTENT(IN)   :: N
    REAL(dp)   , INTENT(IN)   :: Dt
    COMPLEX(dp)               :: ym(N), yn(N)
    INTEGER                   :: j, k

    ym = (0,0)

    OPEN(UNIT=1, FILE="data.out", STATUS="UNKNOWN")
    OPEN(UNIT=2, FILE="datainv.out", STATUS="UNKNOWN")

    DO j=1, N
      READ(1,*) yn(j)
    END DO

    DO j=1, N
      DO k=1, N
        ym(j) = ym(j) + (1d0/N)*yn(k)*ZEXP(-2*PI*i*(j-1)*(k-1)/N)
      END DO
      !! Salvando: tupla de complexo | iteração | tempo | parte real | parte imaginária
      WRITE(2,*) ym(j), j, (j-1)*Dt, REALPART(ym(j)), IMAGPART(ym(j))
    END DO

    CLOSE(1)
    CLOSE(2)
  END SUBROUTINE

  SUBROUTINE GerarSerie(N, Dt, a1, a2, w1, w2)
    REAL(dp), INTENT(IN) :: Dt, a1, a2, w1, w2
    INTEGER , INTENT(IN) :: N

    OPEN(UNIT=1, FILE="data.in", STATUS="UNKNOWN")

    DO j=0, N-1
      WRITE(1,*) a1*DCOS(w1*Dt*j) + a2*DSIN(w2*Dt*j), j*Dt
    END DO

    CLOSE(1)
  END SUBROUTINE
END MODULE
```

Figura 1: Programa para criar séries temporais, e realizar as transformadas de Fourier e sua inversa.

Tarefa 2

II - Teste seu programa gerando as séries:

$$y_i = a_1 \cos(\omega_1 t_i) + a_2 \sin(\omega_2 t_i), t_i = i\Delta t, i = 1, \ldots, N. \quad (1)$$

Escolha:

(a) $N = 200$, $\Delta t = 0.04$, $a_1 = 2$, $a_2 = 4$, $\omega_1 = 4\pi\text{Hz}$, $\omega_2 = 2.5\pi\text{Hz}$

(b) $N = 200$, $\Delta t = 0.04$, $a_1 = 3$, $a_2 = 2$, $\omega_1 = 4\pi\text{Hz}$, $\omega_2 = 2.5\pi\text{Hz}$

(c) $N = 200$, $\Delta t = 0.4$, $a_1 = 2$, $a_2 = 4$, $\omega_1 = 4\pi\text{Hz}$, $\omega_2 = 0.2\pi\text{Hz}$

(d) $N = 200$, $\Delta t = 0.4$, $a_1 = 3$, $a_2 = 2$ $\omega_1 = 4\pi\text{Hz}$, $\omega_2 = 0.2\pi\text{Hz}$

Compare graficamente e discuta as diferenças, se houver, entre as séries: (a) e (b); (c) e (d); (a) e (c); (b) e (d).

Utilizando o *MODULE* utils, o código dessa tarefa se resume a:

```
PROGRAM tarefa2
  USE :: utils

  !! Os arquivos são salvos em "data.in", é preciso renomeá-los depois
  CALL GerarSerie(200, 0.04d0, 2d0, 4d0, 4d0*PI, 2.5d0*PI)
  ! CALL gerarserie(200, 0.04d0, 3d0, 2d0, 4d0*PI, 2.5d0*PI)
  ! CALL gerarserie(200, 0.4d0, 2d0, 4d0, 4d0*PI, 0.2d0*PI)
  ! CALL gerarserie(200, 0.4d0, 3d0, 2d0, 4d0*PI, 0.2d0*PI)

  !! Os arquivos são salvos em "data.out", é preciso renomeá-los depois
  CALL Fourier(200, 0.04d0)
  ! CALL Fourier(200, 0.4d0)
END PROGRAM
```

Figura 2: Código da tarefa 2.

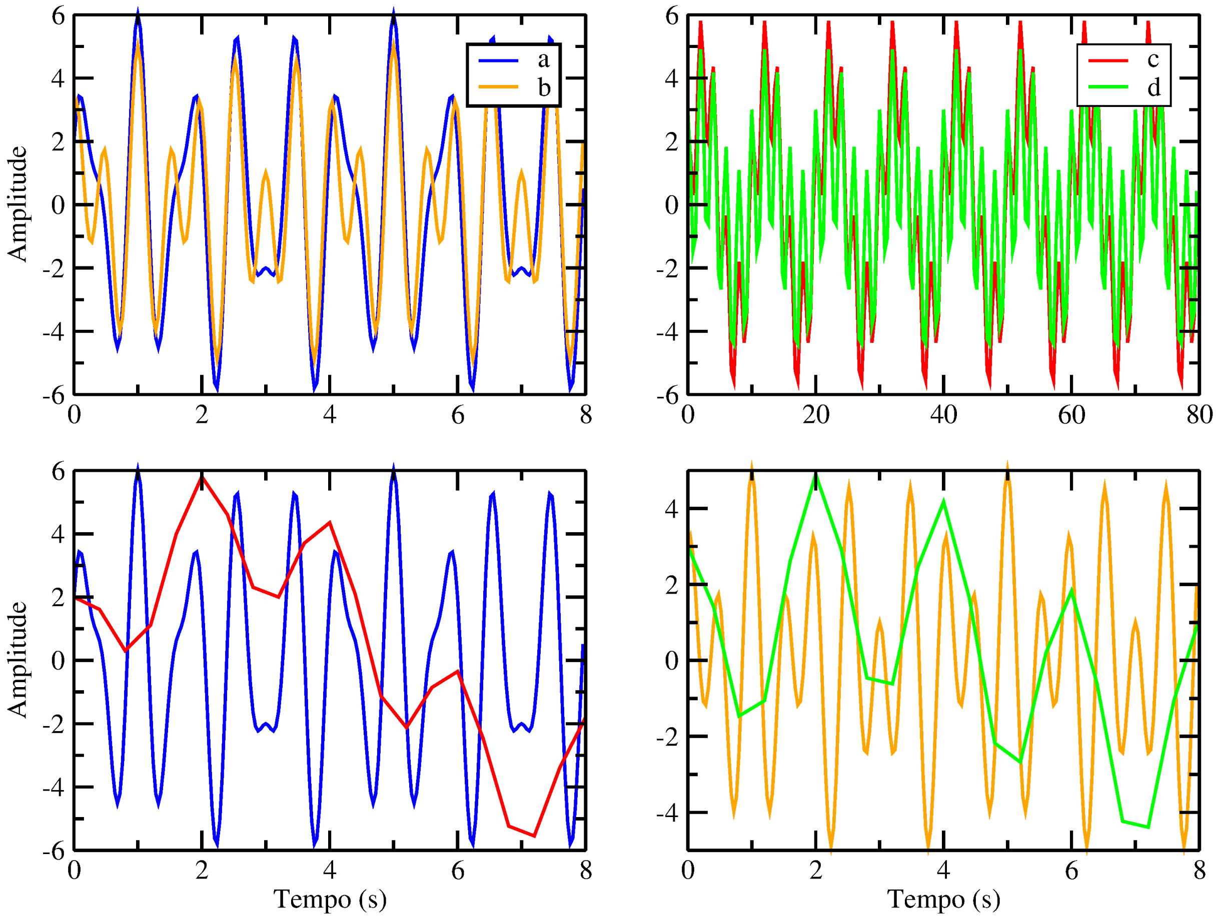

Figura 3: Séries temporais organizadas da forma como o enunciado pede.

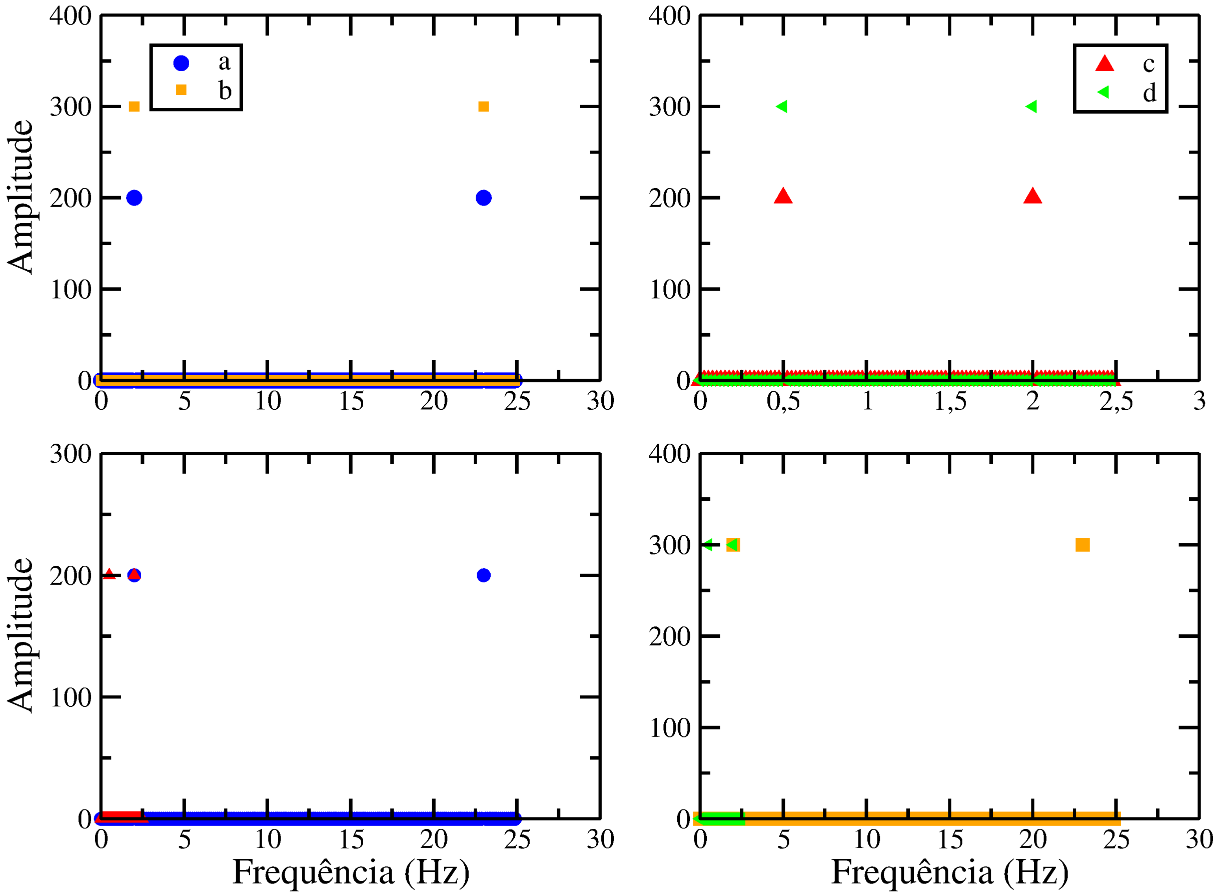

Figura 4: Parte real da transformada de Fourier do gráfico anterior.

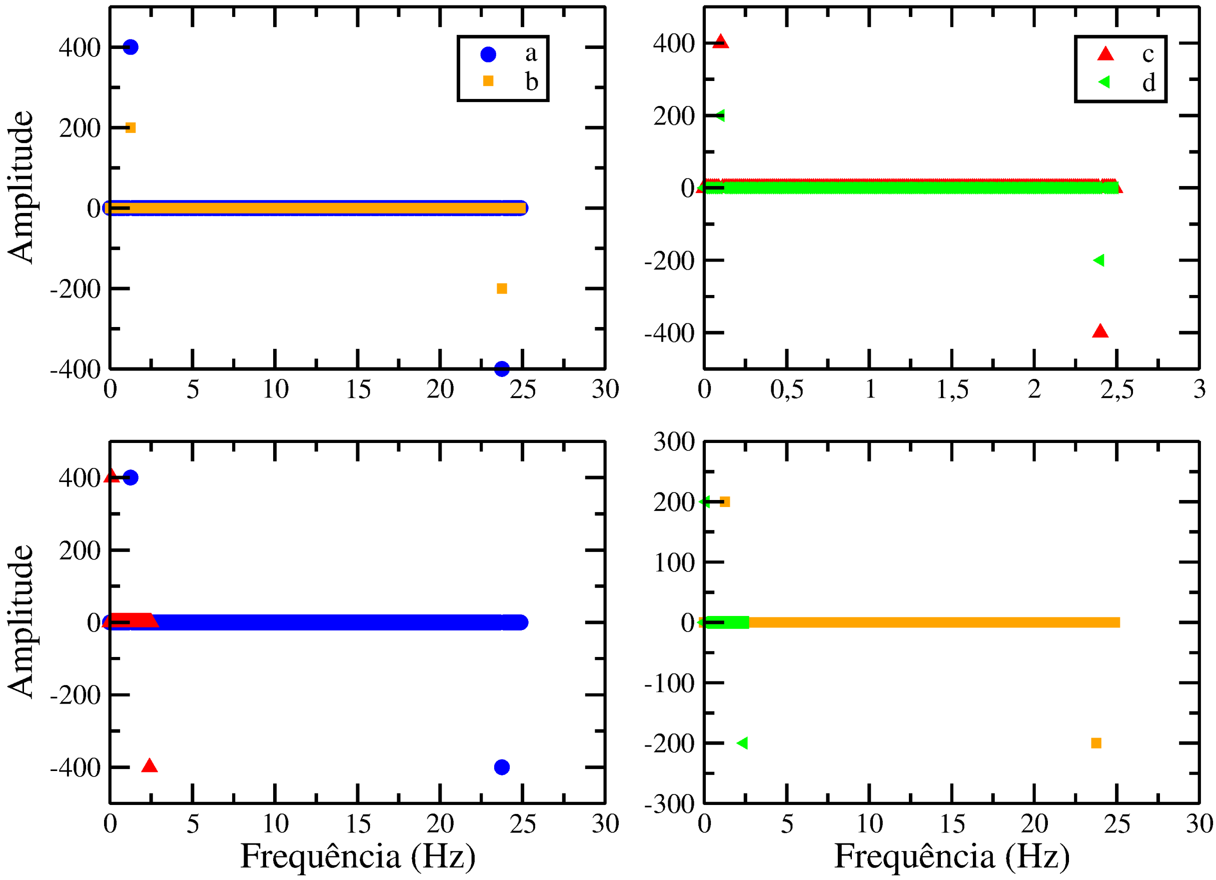

Figura 5: Parte imaginária da transformada de Fourier do gráfico anterior.

As diferenças entre as séries (a), (b), (c) e (d), são as seguintes:

Entre (a) e (b), o que muda é somente a amplitude. O período é igual e as frequências também. Logo, no espaço de frequências, a única diferença é a amplitude das deltas (lembrando que a transformada não está normalizada). Entre (c) e (d) é a mesma coisa. Podemos verificar também que, como nosso sinal corresponde a um cosseno com frequência $f_1 = \omega_1/(2\pi)$ e um seno de frequência $f_2 = \omega_2/(2\pi)$, as deltas do espaço de frequências se encontram exatamente nesses valores para cada série. No caso, a parte real representa a frequência f_1 e a parte imaginaria f_2.

Agora, entre (a) e (c), e (b) e (d), mudamos a resolução das séries temporais e também as frequências. No espaço de frequências podemos perceber que as séries espectrais de (c) e (d) são menores e ficam espremidas no canto esquerdo do gráfico. Para explicar esse comportamento, podemos começar falando sobre o motivo de haverem duas deltas de alturas iguais. Não coincidentemente, o centro das deltas de uma mesma série temporal, fica exatamente na frequência de Nyquist ($f_{Nyquist} = 1/(2\Delta t)$). Para $\Delta t = 0.04$ e $\Delta t = 0.4$ temos $f_{Nyquist} = 12.5\,Hz$ e $f_{Nyquist} = 1.25\,Hz$, respectivamente. Frequências acima de $f_{Nyquist}$ são o espelho das abaixo, por isso, a partir de agora podemos apenas plotar nossos espectros até $f_{Nyquist}$, sem perda de informação. Mas para realizarmos a transformada inversa, precisamos de toda a informação.

O motivo da parte imaginária estar com sinal invertido e do porquê da segunda delta se encontrar em $2f_{Nyquist} - f_k = f_{N-k}$, onde $k = 1, 2$, pode ser explicado expandindo a transformada de Fourier. O valor não zero de Y_m é quando $m = k$, ou seja, quando estivermos em uma das duas frequências f_k. Então:

$$\begin{aligned}
\mathrm{Re}(Y_k) &= \sum_{n=0}^{N-1} \cos(\omega_1 t_j)\cos(2\pi f_k t_j) = \sum_{n=0}^{N-1} \cos(\omega_1 t_j)\cos\left(\frac{2\pi kj}{N}\right) \\
\mathrm{Re}(Y_{N-k}) &= \sum_{n=0}^{N-1} \cos(\omega_1 t_j)\cos(2\pi f_{N-k} t_j) = \sum_{n=0}^{N-1} \cos(\omega_1 t_j)\cos\left(2\pi\frac{(N-k)j}{N}\right) \\
&= \sum_{n=0}^{N-1} \cos(\omega_1 t_j)\cos\left(2\pi j - \frac{2\pi kj}{N}\right) = \sum_{n=0}^{N-1} \cos(\omega_1 t_j)\cos\left(\frac{2\pi kj}{N}\right)
\end{aligned}$$

Ou seja, são a mesma frequência. O termos cruzados $\sin(x)\cos(y)$ são claramente zero e não foram considerados. E para a parte imaginária:

$$\begin{aligned}
\mathrm{Im}(Y_k) &= \sum_{n=0}^{N-1} \sin(\omega_2 t_j)\sin(2\pi f_k t_j) = \sum_{n=0}^{N-1} \sin(\omega_2 t_j)\sin\left(\frac{2\pi kj}{N}\right) \\
\mathrm{Im}(Y_{N-k}) &= \sum_{n=0}^{N-1} \sin(\omega_2 t_j)\sin(2\pi f_{N-k} t_j) = \sum_{n=0}^{N-1} \sin(\omega_2 t_j)\sin\left(2\pi\frac{(N-k)j}{N}\right) \\
&= \sum_{n=0}^{N-1} \sin(\omega_2 t_j)\sin\left(2\pi j - \frac{2\pi kj}{N}\right) = \sum_{n=0}^{N-1} -\sin(\omega_2 t_j)\sin\left(\frac{2\pi kj}{N}\right)
\end{aligned}$$

E o sinal é trocado.

Tarefa 3

III - Escolha agora na expressão (1) as séries:

(e) $N = 200$, $\Delta t = 0.04$, $a_1 = 2$, $a_2 = 4$, $\omega_1 = 4\pi\mathrm{Hz}$, $\omega_2 = 1.4\pi\mathrm{Hz}$

(f) $N = 200$, $\Delta t = 0.04$, $a_1 = 2$, $a_2 = 4$, $\omega_1 = 4.2\pi\mathrm{Hz}$, $\omega_2 = 1.4\pi\mathrm{Hz}$

Compare o comportamento espectral dos dois gráficos obtidos com aqueles das séries (a) e (c), discuta as diferenças.

```fortran
PROGRAM tarefa3
  USE :: utils

  !! Séries (e) e (f) da tarefa 3 (atual)
  CALL GerarSerie(200, 0.04d0, 2d0, 4d0, 4d0*PI, 1.4d0*PI)
  ! CALL GerarSerie(200, 0.04d0, 2d0, 4d0, 4.2d0*PI, 1.4d0*PI)


  !! Séries (a) e (c), da tarefa 1
  ! CALL GerarSerie(200, 0.04d0, 2d0, 4d0, 4d0*PI, 2.5d0*PI)
  ! CALL GerarSerie(200, 0.4d0, 2d0, 4d0, 4d0*PI, 0.2d0*PI)

  CALL Fourier(200, 0.04d0)
  ! CALL Fourier(200, 0.4d0)
END PROGRAM
```

Figura 6: Código da tarefa 3.

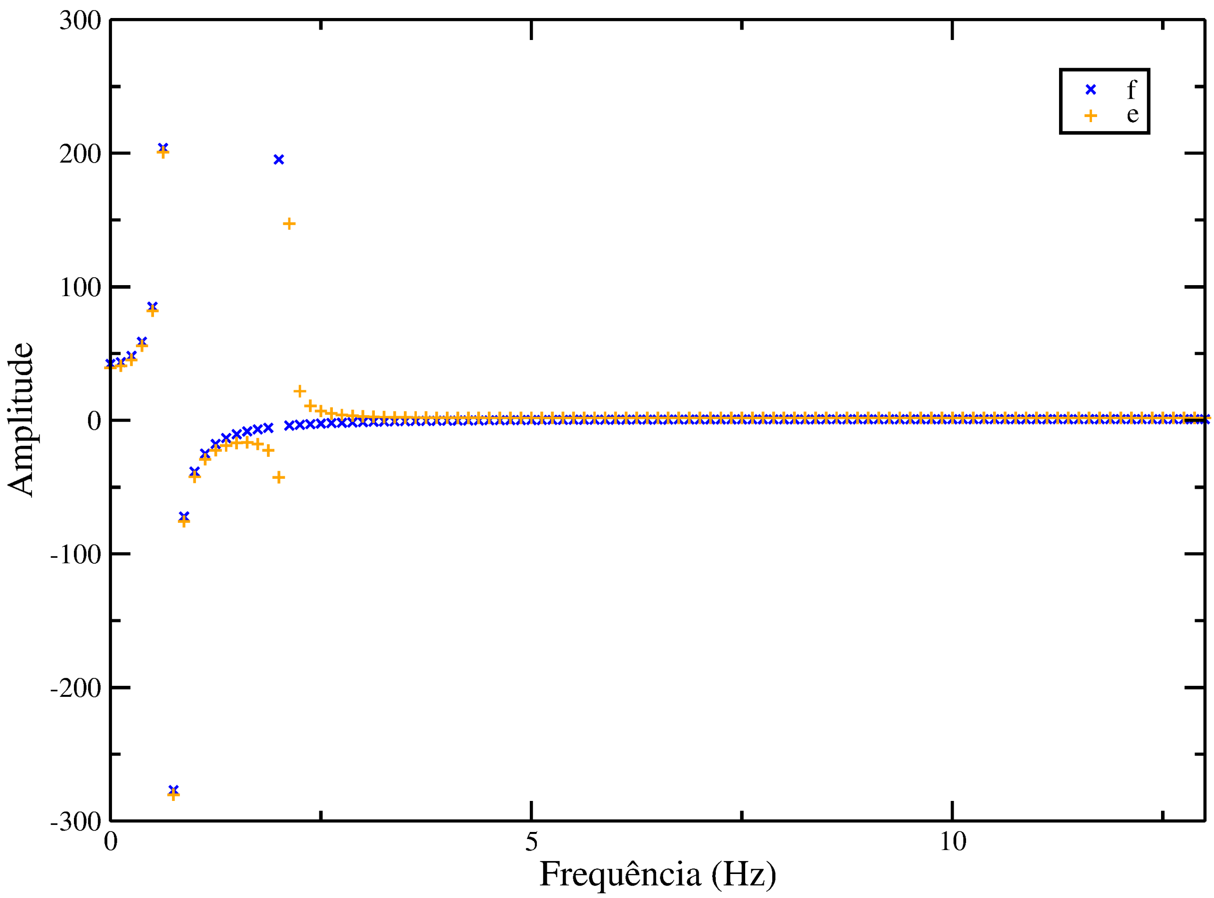

Figura 7: Parte real da transformada de Fourier para as séries (f) e (e).

O fenômeno que ocorre aqui é devido às frequências do nosso sinal não poderem ser representadas adequadamente pelas nossas frequências discretas $f_j = j/(N\Delta t)$. Ou seja, caso as frequências do nosso sinal (f_k) não sejam próximas o suficiente de um múltiplo de $1/(N\Delta t)$, o algorítimo é obrigado a representar essas frequências como a soma de componentes ao longo de uma largura, analogamente ao alargamento de uma delta, devido a motivos de incerteza. Também é bom notar que as amplitudes mudam de acordo. Mesmo assim, não há perda de informação e o nosso sinal continua válido, sendo possível fazer a transformada inversa sem problemas.

Isso pode ser notado com a frequência $f_{1,f} = 4\pi/(2\pi)$ e $f_{1,e} = 4.2\pi/(2\pi)$. A primeira é um múltiplo inteiro de $1/(N\Delta t)$, mas a segunda não. Por isso $f_{1,e}$ tem um comportamento diferente. Também podemos perceber que $f_2 = 0.7\,Hz$ está presente no gráfico da parte real, isso ocorre porque foi assim que o algorítimo escolheu guardar essas informações, enfatizando que não há perda de informação, somente, talvez, perda de padrão, já que esperávamos a parte real e imaginária corresponderem a parte par e impar do nosso sinal original 3.

Tarefa 4

IV - Tome os resultados provenientes da série (a) ("data.out") e o utilize como dado de entrada para o cálculo da transformada inversa de Fourier. Compare a série obtida com a série (a) inicialmente usada. Discuta.

```
PROGRAM tarefa4
  USE :: utils

  CALL GerarSerie(200, 0.04d0, 2d0, 4d0, 4d0*PI, 2.5d0*PI) !! Gerar serie em 'data.in'
  CALL Fourier(200, 0.04d0) ! Transformar séries 'data.in' em série espectral 'data.out'
  CALL InvFourier(200, 0.04d0) ! Recuperar série temporal em 'datainv.out' através de 'data.out'
END PROGRAM
```

Figura 8: Código da tarefa 4.

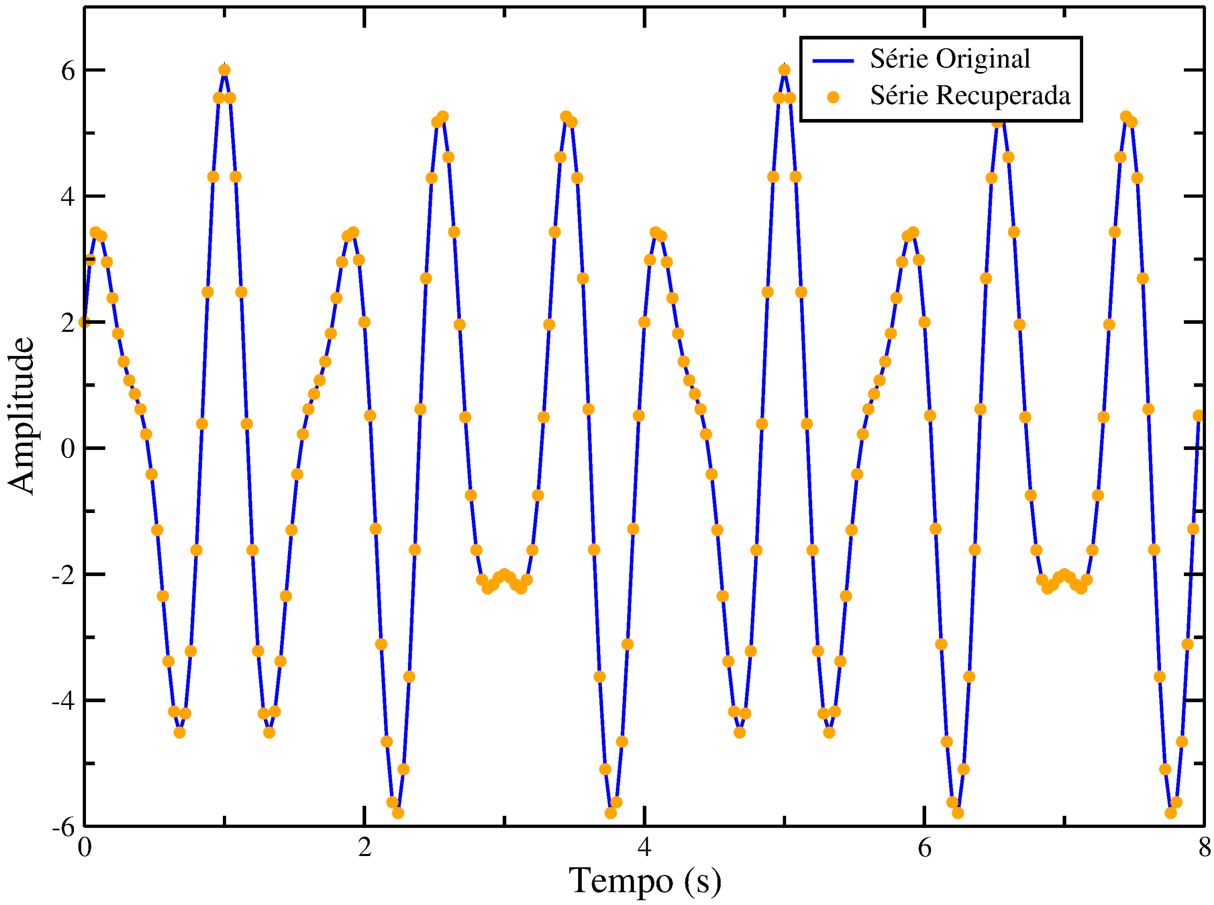

Figura 9: Séries original em azul e o sinal recuperado através da transformada inversa de Fourier em laranja.

Podemos ver que o nosso sinal adquirido da transformada inversa corresponde quase que identicamente ao sinal original. Isso mostra claramente não só o sucesso do nosso algorítimo, como também do método da Transformada Discreta de Fourier.

Tarefa 5

V - Use os parâmetros da série (a) para $N = 50, 100, 200$ e 400 e mostre que o tempo de cálculo computacional cresce com N^2.

```fortran
PROGRAM tarefa5
  USE :: utils
  INTEGER :: N(8)

  OPEN(UNIT=10, FILE="saida-5-tempo.dat", STATUS="UNKNOWN")

  N = (/ 50, 100, 200, 400, 800, 1600, 3600, 7200/) !! Indo um pouco mais além...

  DO j=1, 8
    CALL GerarSerie(N(j), 0.04d0, 2d0, 4d0, 4d0*PI, 2.5d0*PI)

    CALL CPU_TIME(t1)
    CALL Fourier(N(j), 0.04d0)
    CALL CPU_TIME(t2)

    WRITE(10,*) N(j), (t2-t1)
  END DO
END PROGRAM
```

Figura 10: Código da tarefa 5.

Modelando o comportamento do algorítimo como uma função quadrática: $T = aN^2$. Podemos plotar um gráfico Log-Log e analisar qualitativamente se a curva se aproxima de uma reta. E, de fato, é o que acontece.

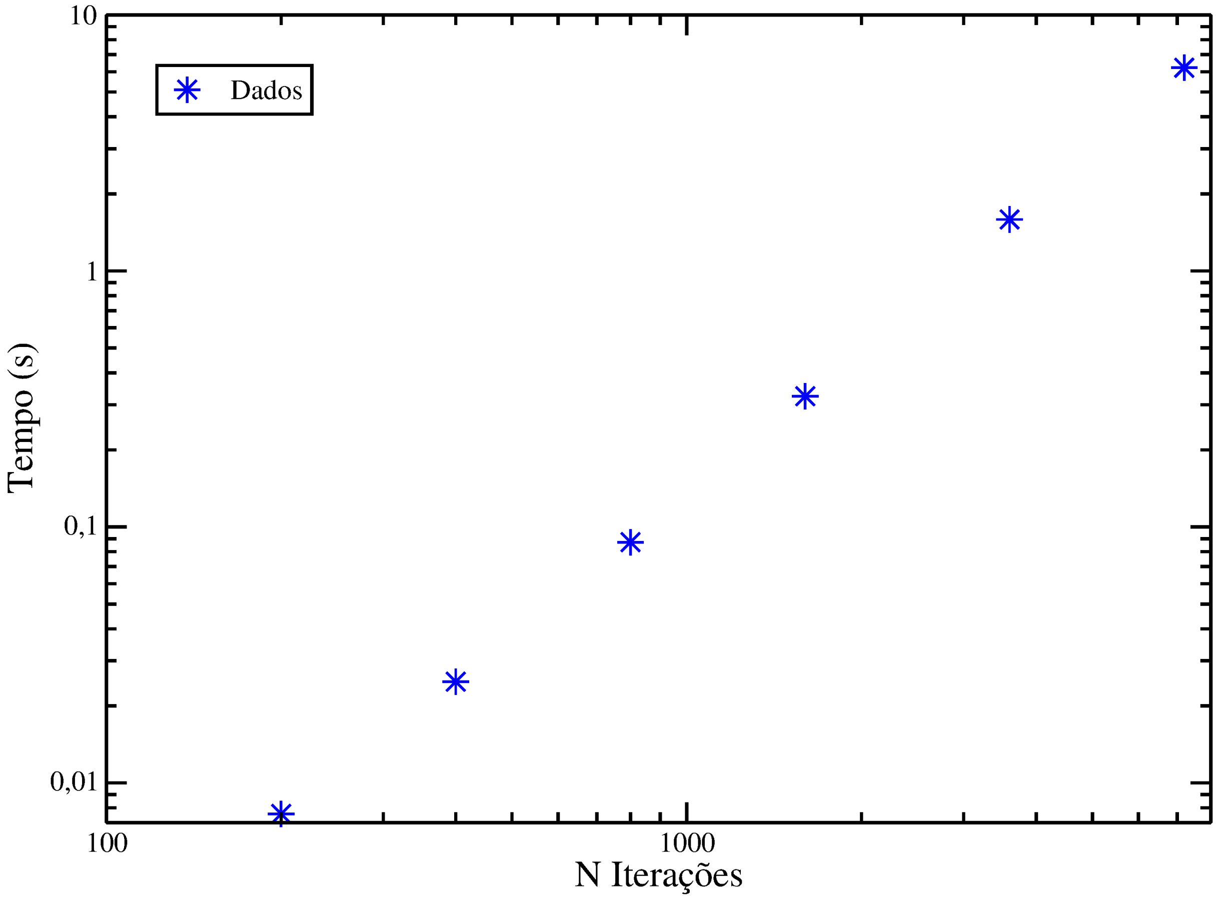

Figura 11: Gŕafico Log-Log do tempo de computação pelo número de iterações, N, usado no nosso algorítimo mostrando o comportamento exponencial do tempo.

Referências

[1] Nicholas J. Giordano e Hisao Nakanishi. *Computational Physics*. 2ª ed. Upper Saddle River (NJ): Prentice Hall, 2006.