ANALISE ESPECTRAL POR TRANSFORMADAS DE FOURIER

Aluno: João Victor Alcantara Pimenta 11820812

Professor: Francisco Castilho Alcaraz

1 Introdução

A Transformação de Fourrier que realizaremos é o equivalente discreto à Tranformação de Fourrier Continua, mas para N sinais igualmente espaçadas por Δt. Para um função $f(t)$ continua qualquer, escrevemos a Transformação de Fourrier:

$$F(jw) = \int_{\infty}^{\infty} f(t)e^{-iwt}dt$$

No caso discreto, se o integrando existe somente para os valores do sinal, podemos reescrever:

$$\text{F(iw)} = \int_0^{(N-1)\Delta t} f(t)e^{-iwt}dt$$
$$= \text{f}_0 e^{-iw0\Delta t} + f_1 e^{-iw1\Delta t} + ... + f_k e^{-iwk\Delta t} + ... + f_{N-1}e^{-iw(N-1)\Delta t}$$

Ou equivalentemente:

$$F(iw) = \sum_{k=0}^{N-1} f_k e^{-iwk\Delta t}$$

Dessa forma trocamos o domínio dos dados para uma base de frequências. Importante lembrar que com N pontos é possível somente outra base de N valores independentes. Por isso tomaremos apenas:

$$w = 0, \frac{2\pi}{N\Delta t}, \frac{2\pi}{N\Delta t} \times 2, ..., \frac{2\pi}{N\Delta t} \times (N-1)$$

Logo, em geral:

$$F_n = \sum_{k=0}^{N-1} f_k e^{-i\frac{2\pi}{N}nk}$$

Para $0 \leq n \leq N-1$.

Equivalentemente podemos definir a inversa dessa transformação:

$$f_k = \frac{1}{N}\sum_{n=0}^{N-1} F_n e^{i\frac{2\pi}{N}nk}$$

2 Dados

Precisamos de um sinal. No nosso caso trataremos de um sinal do tipo

$$f_i = a_1 \cos(w_1 t_i) + a_2 \sin(w_2 t_i)$$

Com $t_i = i\Delta t$ e $i = 1, ..., N$. O geramos com o programa:

```fortran
      PROGRAM GENERATE
            IMPLICIT REAL (KIND=8) (a-h,o-z)
            pi = acos(-1.d0)

            N = 200
            dt = 0.04d0
            a1 = 2.d0
            a2 = 4.d0
            w1 = 4*pi
            w2 = 2.5*pi

            CALL ESCREVER(N, dt, a1, a2, w1, w2)

      END PROGRAM GENERATE

      SUBROUTINE ESCREVER (N, dt, a1, a2, w1, w2)
            IMPLICIT REAL (KIND=8) (a-h,o-z)
            COMPLEX (KIND=16) sig, SINAL

            open(newunit=io, file="entradas/entrada-1.1-11820812.in")

            DO 10 i=0,N-1
                  sig = SINAL(a1, a2, w1, w2, dt*i)
                  time = dt*i
                  WRITE (io, *) time, sig
  10        END DO

            close(io)
      END SUBROUTINE ESCREVER

      COMPLEX (KIND=16) FUNCTION SINAL (a1, a2, w1, w2, t)
            IMPLICIT REAL (KIND=8) (a-h,o-z)
            sinal = CMPLX(a1*cos(w1*t) + a2*sin(w2*t), 0.d0, 16)
            RETURN
      END FUNCTION SINAL
```

As quatro configuração estão com os resultados ilustrados na Figura 1. Os parametros eram:

- $a_1 = 2$, $a_2 = 4$, $w_1 = 4\pi Hz$, $w_2 = 2.5\pi Hz$, $N = 200$, $\Delta t = 0.04$;
- $a_1 = 3$, $a_2 = 2$, $w_1 = 4\pi Hz$, $w_2 = 2.5\pi Hz$, $N = 200$, $\Delta t = 0.04$;
- $a_1 = 2$, $a_2 = 4$, $w_1 = 4\pi Hz$, $w_2 = 0.2\pi Hz$, $N = 200$, $\Delta t = 0.4$;
- $a_1 = 3$, $a_2 = 2$, $w_1 = 4\pi Hz$, $w_2 = 0.2\pi Hz$, $N = 200$, $\Delta t = 0.4$;

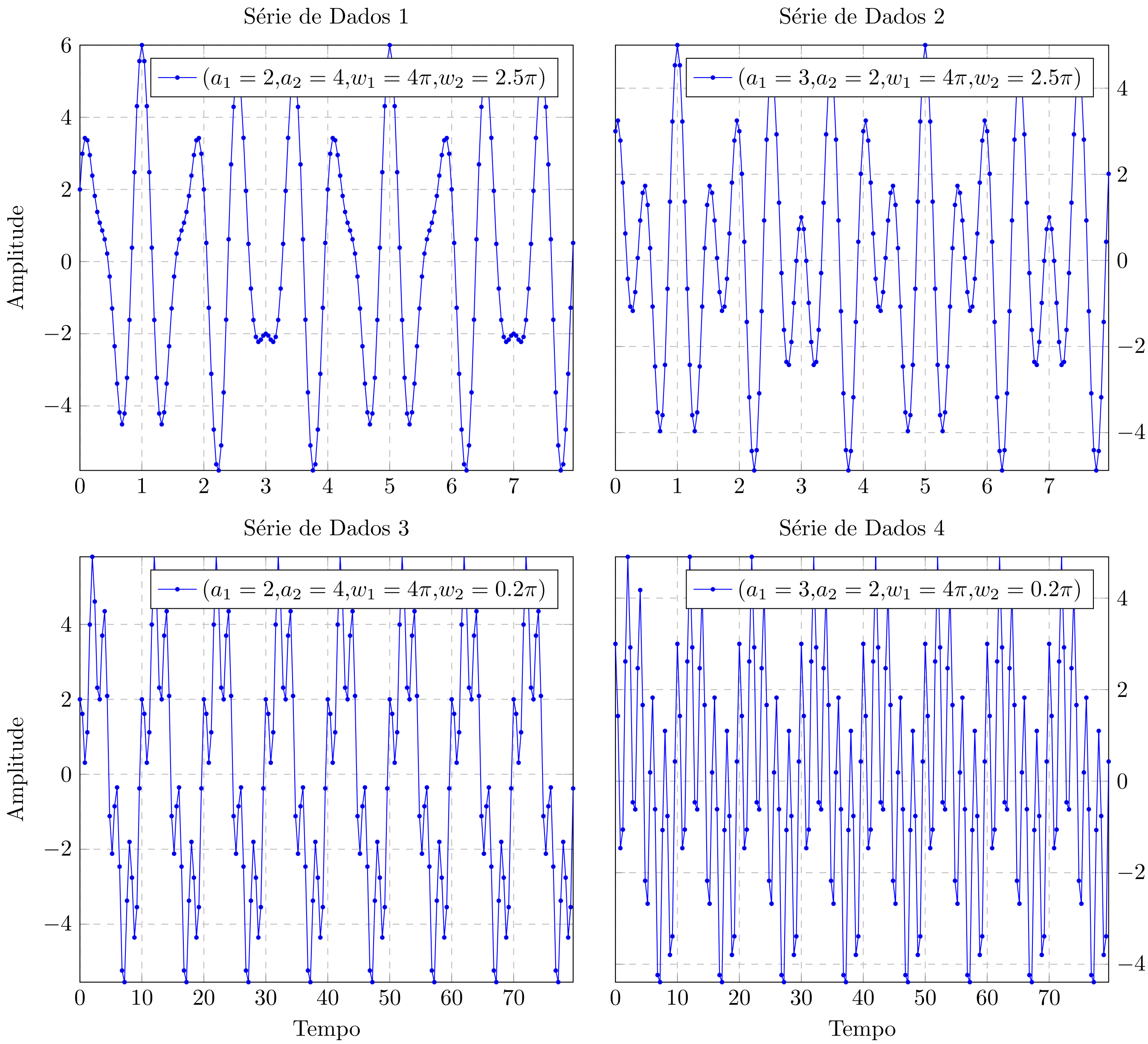


Figure 1: Plot das configurações dadas

Note que dessa vez os dados são reais, mas o programa permite a existência de dados complexos.

3 Transformada

Aplicaremos para as séries de dados descritas acima, as correspondentes representações no espaço de frequência. Para isso, basta aplicar os conceitos discutidos na introdução aos pontos. Isso é atingido pelo programa abaixo.

```
      PROGRAM TRANSFOURRIER
            IMPLICIT REAL (KIND=8) (a-h,o-z)
            COMPLEX (KIND=16), DIMENSION (1000) :: data
            COMMON /DAT/ data

c     Tomar dados do arquivo do sinal
            OPEN(newunit=io, file="entradas/entrada-3.2-11820812.in")

            DO 10 i=1,1000
                  READ(io,*,end=1) time, data(i)
  10        END DO
  1         CONTINUE

            CLOSE(io)

            N = i - 1
            dt = time/(N-1)

c     Chamar a rotina de fourrier que escrevera em data.out
            CALL FOURRIER(N, dt)

      END PROGRAM TRANSFOURRIER

      SUBROUTINE FOURRIER (N, dt)
            IMPLICIT REAL (KIND=8) (a-h,o-z)
            COMPLEX (KIND=16) :: Yk, Ykf

            pi = acos(-1.d0)

            OPEN(newunit=io, file="saidas/saida-3.2-11820812.out")
            DO 10 k=0,N/2-1
                  Yk = Ykf(N, k) / (N/2)
                  freq = k / (N*dt)
                  WRITE(io, *) freq, Yk
  10        END DO
            CLOSE(io)

      END SUBROUTINE FOURRIER

      COMPLEX (KIND=16) FUNCTION Ykf (N, k)
            IMPLICIT REAL (KIND=8) (a-h,o-z)
            COMPLEX (KIND=16), DIMENSION (1000) :: y
            COMMON /DAT/ y
            COMPLEX (KIND=16) :: zi

            pi = ACOS(-1.d0)
            zi = (0.d0, 1.d0)

```

```fortran
            Ykf = 0.d0
            DO 10 j=0,N-1
                  Ykf  = Ykf +  y(j+1) * EXP(2.d0*pi*zi * k * j/N)
 10         END DO

            RETURN
      END FUNCTION Ykf
```

Aplicando sobre os arquivos anteriormente criados, obtemos quatro espectros de frequência demonstrados na Figura 2.

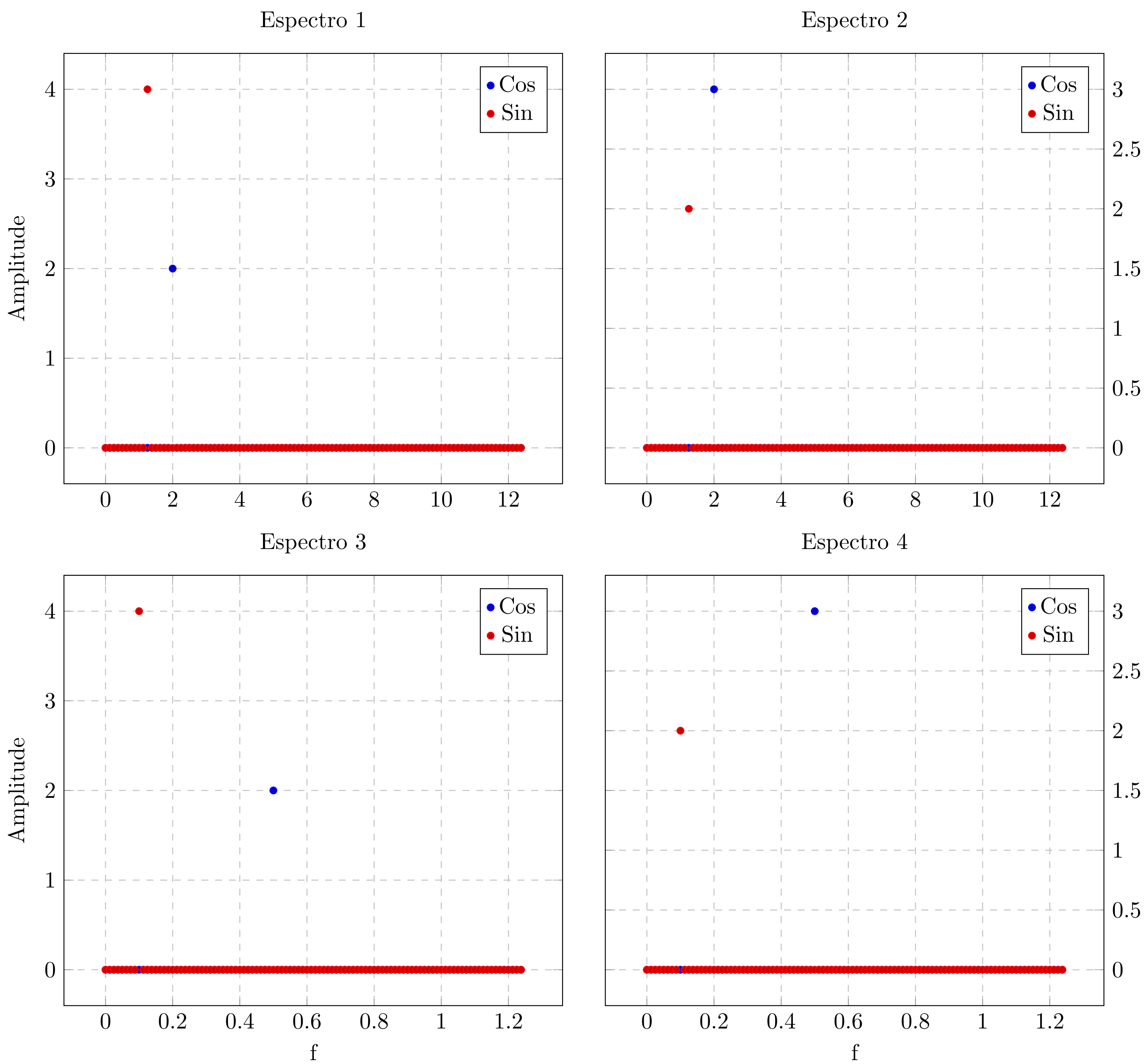


Figure 2: Plot dos espectros

É necessário discutir os resultados obtidos. Note que tanto para (a) quanto para (b) as frequências que usamos para gerar os dados estão incluídas no intervalo de frequência possível de detectar com a transformada para o sample utilizado. Dessa forma, ambos o Seno e Cosseno que foram utilizados foram bem detectados no espectro de frequência.

Para as séries (c) e (d) o cenário é diferente. Com o sample bem menor, as frequências altas (acima de 1.25) não podem ser identificadas e alguns efeitos entram em ação. Primeiramente, a frequência em 0.1 é devidamente identificada em ambos. Existe contudo um artefato em 0.5 causado pela frequência 2 que foi usada. Neste caso, $f_r - f_n = 0.75$. Ou seja, observamos a frequência em $1.25 - 0.75 = 0.5$.

4 Outros casos

Analisaremos agora os casos descritos abaixo e referentes às séries na Figura 3. Aqui se reutiliza o programa para gerar as séries já descrito no relatório.

- $a_1 = 2$, $a_2 = 4$, $w_1 = 4\pi Hz$, $w_2 = 1.4\pi Hz$, $N = 200$, $\Delta t = 0.04$;
- $a_1 = 2$, $a_2 = 4$, $w_1 = 4.2\pi Hz$, $w_2 = 1.4\pi Hz$, $N = 200$, $\Delta t = 0.04$;

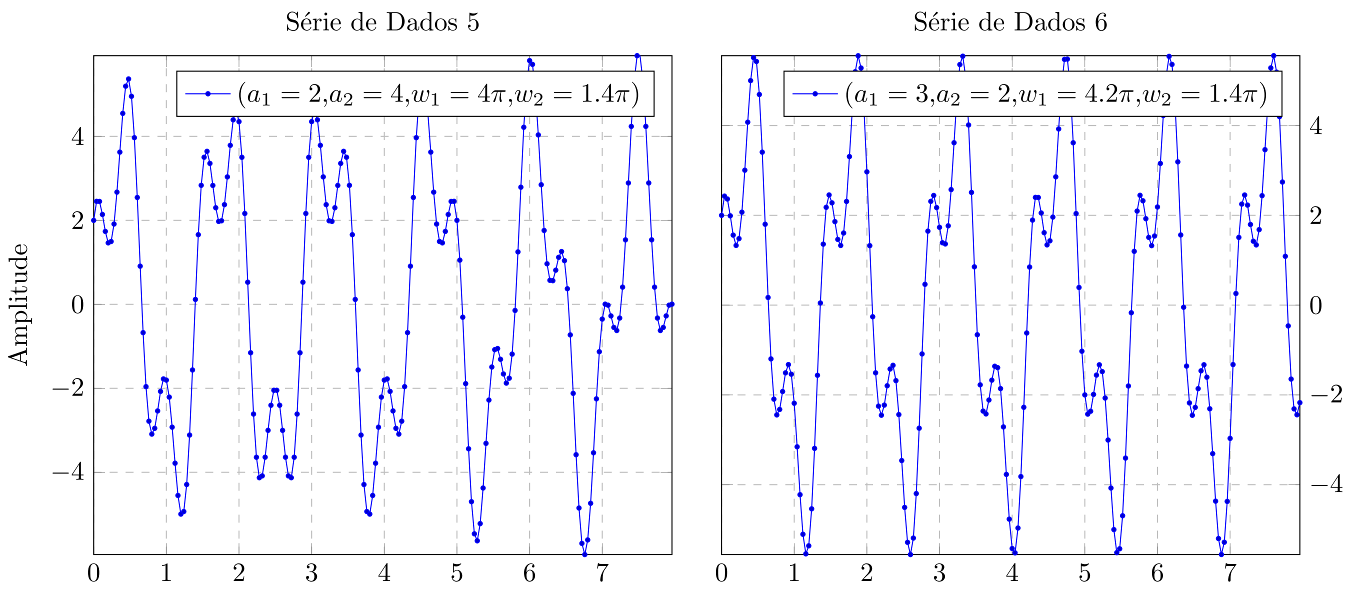


Figure 3: Plot das configurações novas

Agora vamos gerar os espectros correspondentes utilizando da mesma técnica descrita anteriormente. Obtemos algo do formato da Figura 4.

Aqui mais alguns fatores aparecem no nosso espectro de frequência na Figura 4. No espectro da sequência 5 observamos dois picos, um em 0.7 identificado ambos pelo Seno e pelo Cosseno e um identificado pelo Cosseno em 2. Note que a frequência de 2 já havia sido devidamente identificada em cenários parecidos como em (a) e ficou como esperada. A diferença no gráfico vem de medir o pico em 0.7. Percebe-se o entorno com coeficientes não nulos. Isso deve-se ao fato de que a frequência exata 0.7 não é testada na somatória. Com isso a transformação reproduz o mesmo efeito com a contribuição de uma série de outras frequências que, ainda assim, devem dar uma representação ótima dos nossos dados.

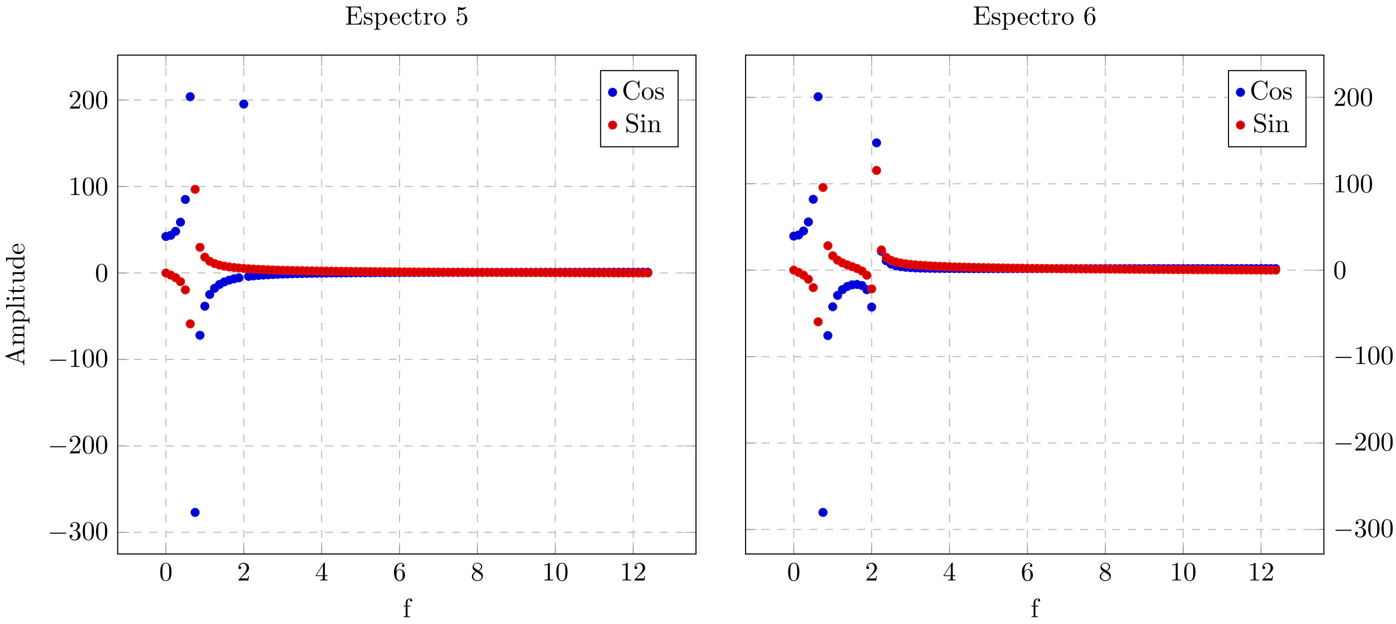


Figure 4: Plot dos novos espectros

Algo muito parecido ocorre no espectro da série 6 ou (f). Além de tudo que foi discutido, existe o fato ainda de que a outra frequência utilizada, 2.1, também compartilha a característica e tem essa dispersão em volta de si.

5 Inversa

Para a inversa basta acrescentar uma subrotina e uma função análogas às que fizemos para o processo de tomar a transformada de Fourrier. Nosso código será:

```
      PROGRAM TRANSINVERSE
            IMPLICIT REAL (KIND=8) (a-h,o-z)
            COMPLEX (KIND=16) , DIMENSION (1000) :: data
            COMMON /DAT/ data

c     Tomar dados do arquivo do espectro
            OPEN(newunit=io, file="saidas/saida-1.1-11820812.out")

            DO 10 i=1,1000
                  READ(io,*,end=1) freq, data(i)
  10        END DO
  1         CONTINUE

            CLOSE(io)

            M = i - 1
            dt  = 0.04d0

c     Chamar a rotina de fourrier que escrevera em data.out
            CALL INVERSE(M, dt)
```

```
    END PROGRAM TRANSINVERSE

    SUBROUTINE INVERSE (M, dt)
          IMPLICIT REAL (KIND=8) (a-h,o-z)
          COMPLEX (KIND=16) :: yj, yjf

          pi = acos(-1.d0)

          OPEN(newunit=io, file="saidas/saida-4-11820812.out")
          DO 10 j=0, (2*M)-1
                yj = yjf(M, j, dt)
                time = j * dt
                WRITE(io, *) time, yj
10        END DO
          CLOSE(io)

    END SUBROUTINE INVERSE

    COMPLEX (KIND=16) FUNCTION yjf (M, j, dt)
          IMPLICIT REAL (KIND=8) (a-h,o-z)
          COMPLEX (KIND=16) , DIMENSION (1000) :: Y
          COMMON /DAT/ Y
          COMPLEX (KIND=16) :: zi

          pi = ACOS(-1.d0)
          zi = (0.d0, 1.d0)

          yjf = 0.d0
          DO 10 k=0,M-1
                arg = k * j / (2.d0 * M)
                yjf = yjf + Y(k+1) * EXP(-zi * 2.d0 * pi * arg)
10        END DO

          RETURN
    END FUNCTION yjf
```

Vamos aplicar o programa sobre o espectro gerado para a série (a) (ou 1). Resultando em uma série do tipo:

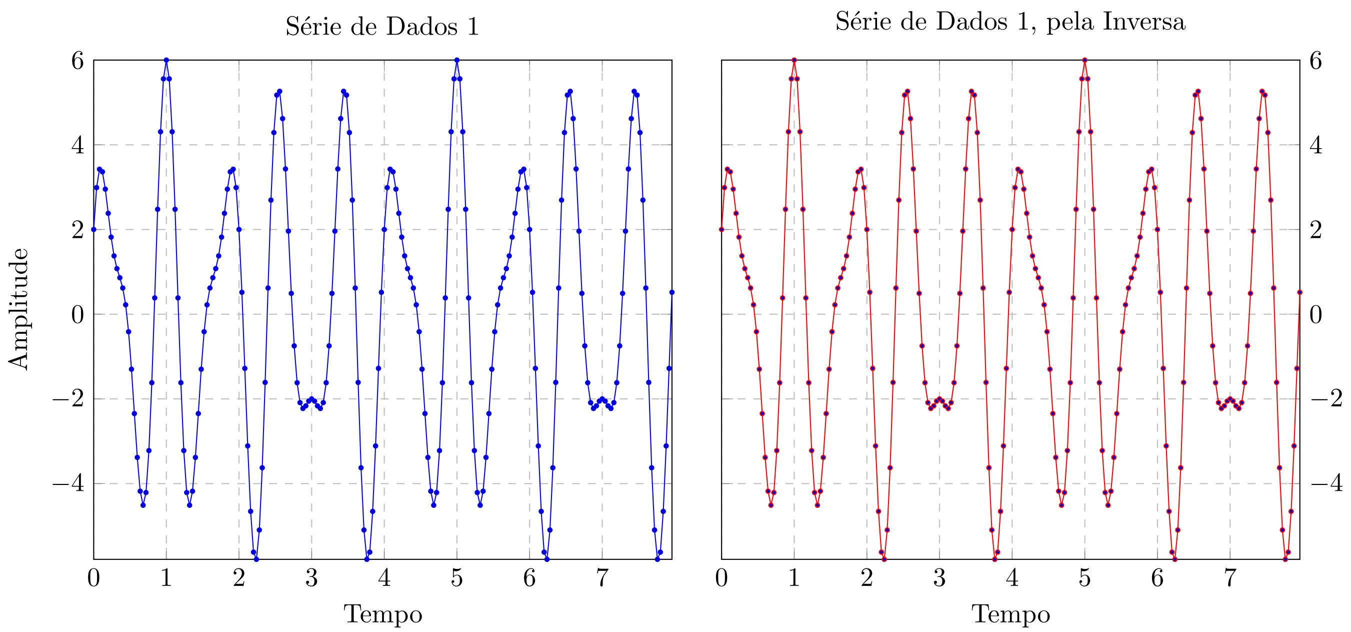
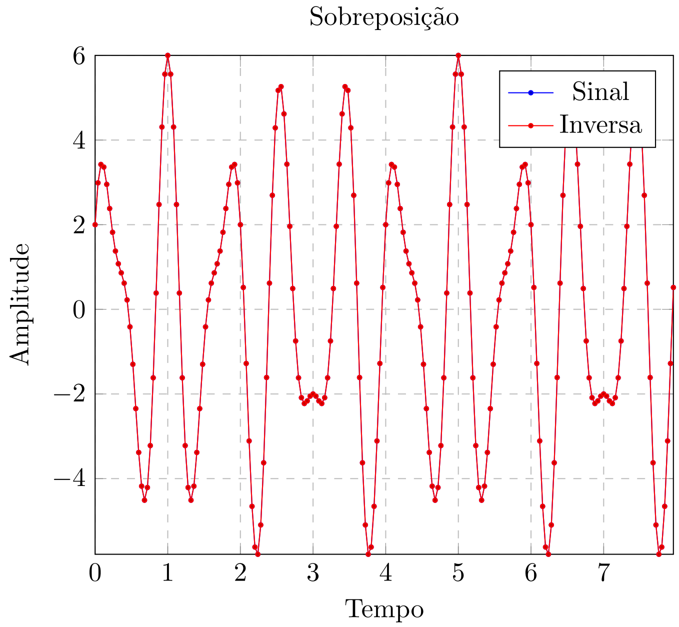


Figure 5: Comparação (a) com (a) recuperada



Finalmente, podemos observar que recuperamos a série temporal inicial sem perda de informação.

6 Tempo

Para avaliar o tempo de execução vamos utilizar a função *dtime*(). Assim poderemos obter o *elapsed* time do programa em cada um das transformadas que realizaremos para valores distintos de N. Note é claro que o programa faz a média de 100 execuções da sub-rotina de Fourrier. O programa é da forma

```
      PROGRAM TIMEFOURRIER
            IMPLICIT REAL (KIND=8) (a-h,o-z)
            COMPLEX (KIND=16), DIMENSION (1000) :: data
            COMMON /DAT/ data
            CHARACTER*200 :: filename, Nin(4)
            INTEGER :: N(4)
            REAL t(2)

            N =  [ 50, 100, 200, 400 ]
            Nin = ['050-11820812.in', '100-11820812.in',
     +             '200-11820812.in', '400-11820812.in' ]


            open(newunit=ioT, file="saidas/saida-5.1-11820812.out")
            open(newunit=ioTL, file="saidas/saida-5.2-11820812.out")
            write(ioT,*) 'N ', 'tempo'
            write(ioTL,*) 'N ', 'SqrtTempo'

            e = dtime( t )
            eaux = e

            DO 10 i=1,4

                  filename = "entradas/entrada-5-"//Nin(i)
```

```
                OPEN(newunit=io, file=filename)
                DO 20 j=1,1000
                      READ(io,*,end=1) time, data(j)
20              END DO
1               CONTINUE

                CLOSE(io)

                M = j
                dt = time/(M-1)

                DO 100 k=1,100
                      CALL FOURRIER(N(i), dt)
100             END DO

                e = dtime( t )
                write(ioT,*) N(i), (e-eaux)/100
                write(ioTL,*) N(i), SQRT((e-eaux)/100)
                eaux = e

10          END DO

      END PROGRAM TIMEFOURRIER

      SUBROUTINE FOURRIER (N, dt)
            IMPLICIT REAL (KIND=8) (a-h,o-z)
            COMPLEX (KIND=16) :: Yk, Ykf

            pi = acos(-1.d0)

            DO 10 k=0,N/2-1
                  Yk = Ykf(N, k) / (N/2)
                  freq = k / (N*dt)
10          END DO

      END SUBROUTINE FOURRIER

      COMPLEX (KIND=16) FUNCTION Ykf (N, k)
            IMPLICIT REAL (KIND=8) (a-h,o-z)
            COMPLEX (KIND=16), DIMENSION (1000) :: y
            COMMON /DAT/ y
            COMPLEX (KIND=16) :: zi

            pi = ACOS(-1.d0)
            zi = (0.d0, 1.d0)

            Ykf = 0.d0
            DO 10 j=0,N-1
                  Ykf  = Ykf +  y(j+1) * EXP(2.d0*pi*zi * k * j/N)
10          END DO

            RETURN
      END FUNCTION Ykf
```

Para as marcações feitas podemos visualizar o tempo de execução no gráfico abaixo

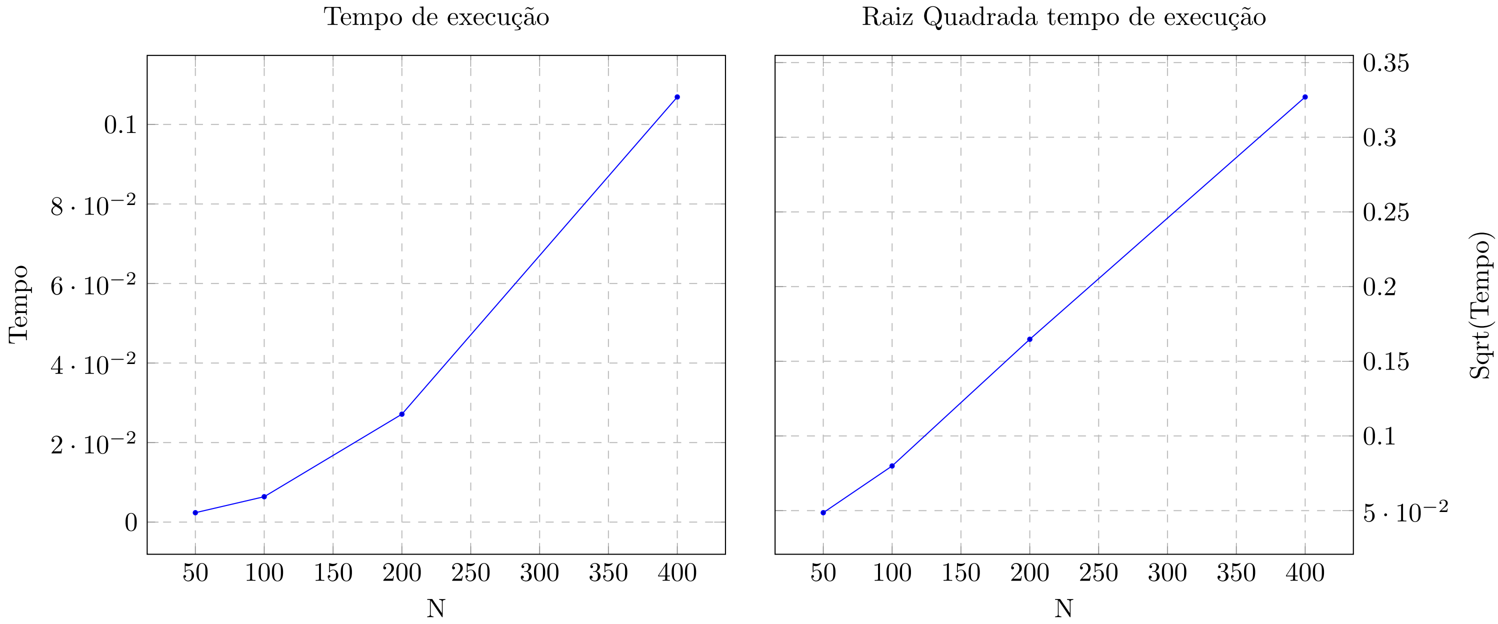


Figure 6: Plot dos novos espectros

Como é possível ver no crescimento do gráfico que representa a raiz quadrada do tempo, o crescimento é razoavelmente linear. Isso significa que o crescimento do Tempo deve ser N^2.