Processamento Digital de Sinais
Filtro Personalizado

Davidson de Faria
ICEx - UFF

Prof. Gustavo Luís F. Vicente

26 de julho de 2016

1 Objetivo

O objetivo deste trabalho foi a criação de um filtro de sinais personalizado com o uso da transformada rápida de Fourier (FFT).

2 Introdução

2.1 O que são filtros personalizados?

Diferentemente dos filtros padrões representados por expressões matemáticas, os filtros personalizados são representados por valores contidos em vetores com comportamento determinado pelo criador do filtro. Um vetor guarda os valores da magnitude de cada frequência que podem ser representados em um gráfico de Amplitude por Frequência e outro guarda a fase de cada componente espectral representado por um gráfico de Fase por Frequência.

2.2 Implementação

A implementação de um filtro é feita através da transformada inversa de Fourier que gera a resposta ao impulso no domínio do tempo e, logo após, é feito o janelamento gerando o kernel do filtro. Por fim, é feita a convolução com o sinal que se deseja filtrar.

Porém, neste trabalho, a convolução com o sinal não foi feita, pois apenas queríamos verificar a qualidade e precisão do filtro. Assim, o último passo feito foi a transformada rápida de Fourier para a verificação do filtro gerado em relação ao filtro lido.

3 Desenvolvimento

A magnitude das componentes de um filtro é composta por partes real e imaginária, seguindo as equações:

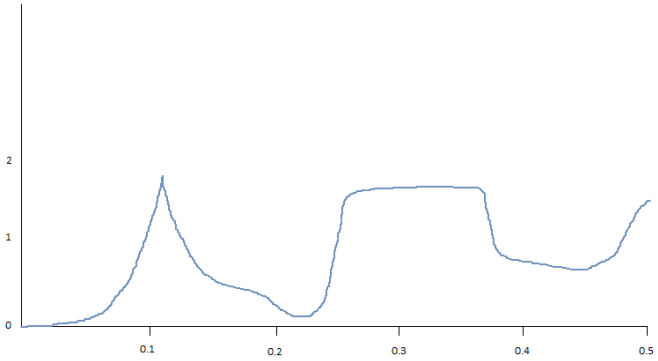
$$\begin{cases} ReX[i] = MagX[i]\, cos(FaseX[k]) \\ ImX[i] = MagX[i]\, sen(FaseX[k]) \end{cases}$$

Mas como esta implementação usa apenas valores reais, então, tomamos a fase como zero.

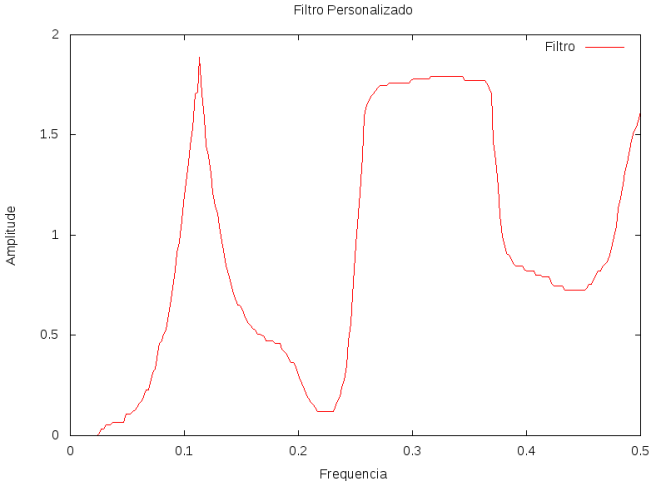
Para a implementação, foi necessário o uso de dois vetores, rex e imx, de tamanho $N = 4(TAM - 1)$, onde TAM é o tamanho do vetor que contém o filtro. Como a fase é zero, o vetor rex recebe os TAM valores da amplitude no início e zero no resto, enquanto o vetor imx é inteiramente preenchido com zeros.

3.1 Criação do filtro

Foi necessário preencher o vetor índice à índice com o uso de um programa de edição de imagens, verificando quais pontos que melhor se adequavam à curva do gráfico, gerado através de um desenho.



A quantidade de pontos obtida foi de 257 e, após ajuste dos pontos lidos e da escala, foi gerado o seguinte filtro:



3.2 Transformada Inversa de Fourier

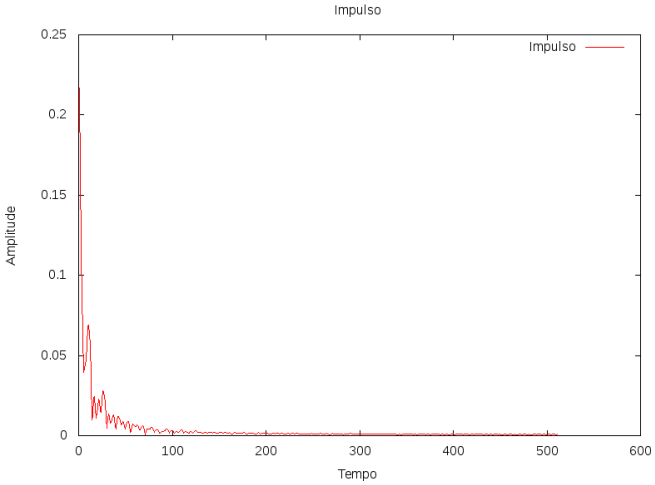
Para o cálculo da transformada inversa, foi necessário inverter o sinal do imx, calcular a FFT (seção 3.4), dividir o resultado de rex e imx por N e inverter o sinal de imx novamente.

A transformada inversa irá converter a resposta em frequência em resposta ao impulso no domínio do tempo, transformando $N/2$ pontos dos vetores rex e imx no domínio da frequência em um vetor de tamanho N no domínio do tempo.

A magnitude gerada pela FFT foi calculada a partir de:

$$magx[i] = \sqrt{rex^2[i] + imx^2[i]} \tag{1}$$

E, em função do tempo, o impulso tem como gráfico:

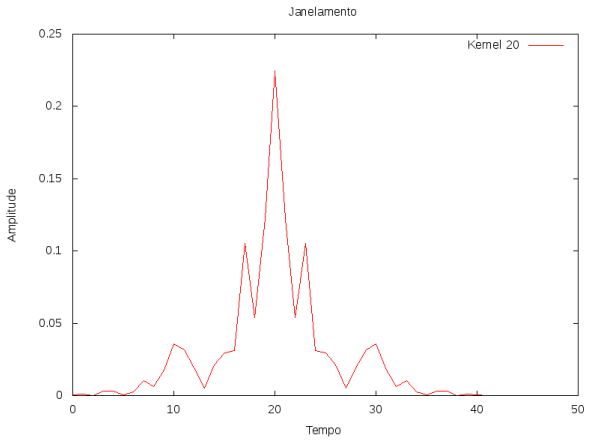


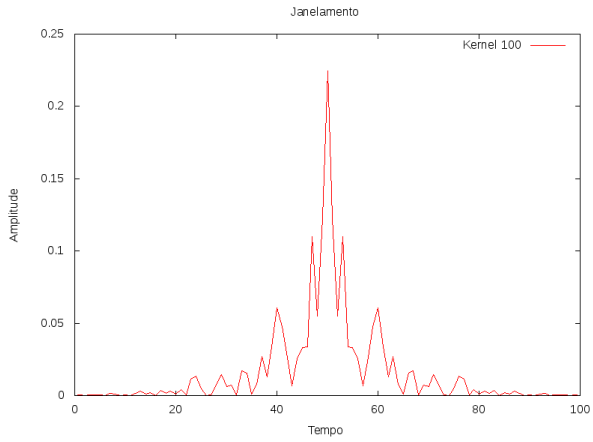
3.3 Janelamento

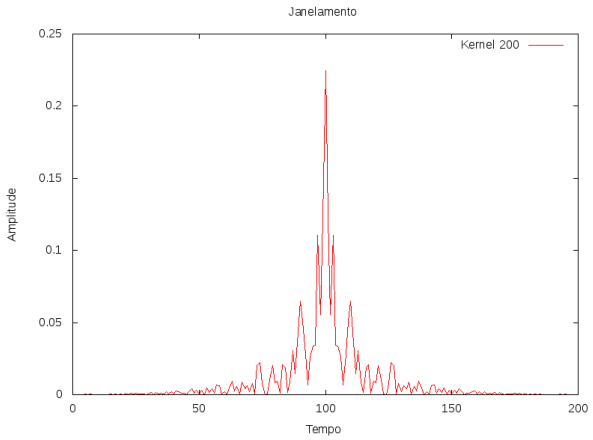
A resposta ao impulso não é conveniente para a utilização em um filtro, portanto, é necessário deslocá-lo em $NF/2$ pontos para direita, onde NF é o tamanho do kernel desejado, enjanelá-lo e completando-o com zeros. Além disso, levamos as frequências negativas que estavam no final do impulso para as $NF/2$ primeiras posições do kernel.

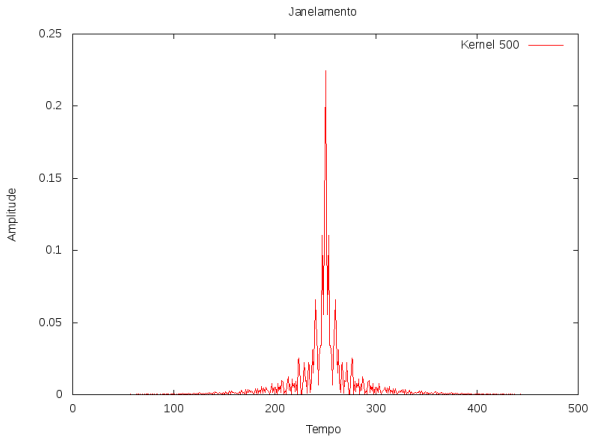
O tamanho do kernel pode variar, porém, não pode ser maior que N. Quanto maior NF, maior a precisão do kernel, com o custo de maior tempo de processamento.

A magnitude do kernel é calculada de acordo com a equação (1) em domínio do tempo. Uma comparação com diferentes tamanhos pode ser vista abaixo:









3.4 Transformada Rápida de Fourier

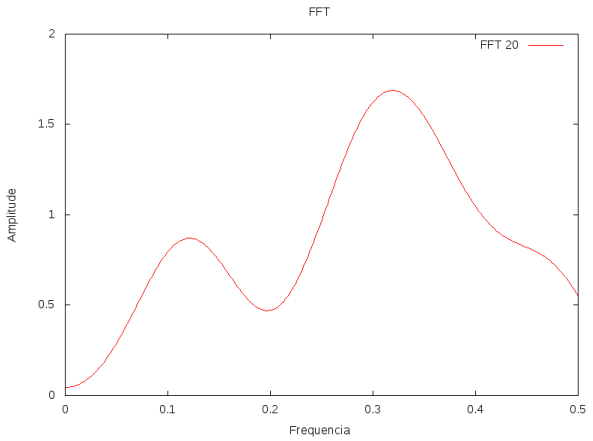
O objetivo da FFT é transformar o kernel no mais próximo do filtro inicial, dependendo do seu tamanho. Para o cálculo da FFT, foi necessário:

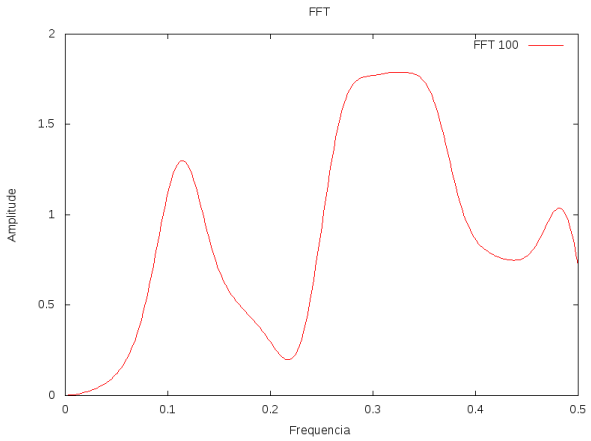
- Reordenar os pontos do sinal de acordo com a ordem reversa de bits das suas posições;
- Decompor o sinal de N pontos em N sinais de um ponto;
- Calcular o espectro de frequência de cada sinal de um ponto, gerando N frequências;
- Sintetizar o espectro de frequências único: Intercalando zeros entre amostras no tempo para a duplicação de pontos da frequência e deslocando de uma amostra no tempo.

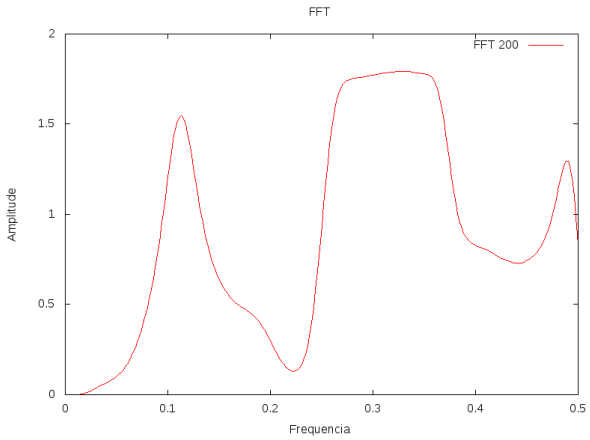
A amplitude gerada pela equação (1) foi normalizada, portanto apresenta apenas valores entre 0 e 1. Para retornar ao valor original, é necessário multiplicar por 2. E para gerar os pontos da frequência, foi dividido os valores de i por $N/2$, devido ao tamanho do vetor $magx$.

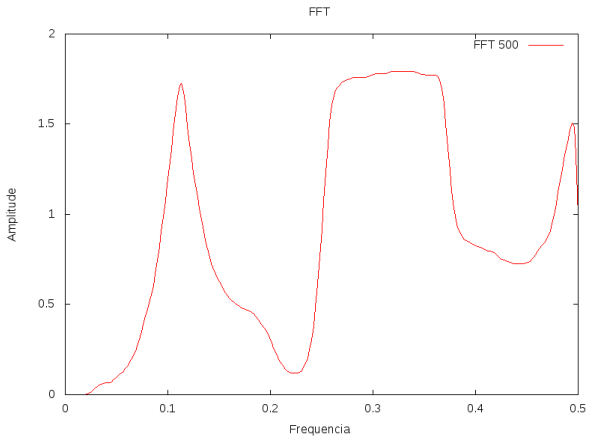
Foram plotados $N/4 + 1$ pontos, que é a quantidade de pontos lidos inicialmente. As outras componentes são zeros.

Abaixo é apresentado os filtros de acordo com o tamanho do seu kernel.









4 Conclusão

O filtro lido pelos vetores apresenta algumas diferenças devido ao truncamento na hora de discretizar a curva original, em azul. Porém, os gráficos acima mostram que quanto maior o tamanho do kernel, maior a precisão da curva e mais próxima ao filtro inicial.

Para ganhar mais precisão no filtro final, seria necessário aumentar o número de pontos lidos da curva original e aumentar o número do kernel.

Um detalhe importante, é a rapidez do algoritmo da transformada rápida de Fourier que apresenta complexidade logarítmica, devido a decomposição do sinal em $log(N)/log(2)$ passos e ao reordenamento dos pontos do sinal de acordo com a ordem reversa de bits de suas posições.

5 Código

```
#include <stdio.h>
#include <math.h>

#define pi 4*atan(1)
#define TAM 257          //quantidade de pontos do rex
#define NF 500           //numero do filtro: 20, 100, 500

main(){
  int i, j, k, N;
  N = 4*(TAM-1);
  int nm1,nd2,m,l,le,inp,le2,jm1, ind;
  double tr,ti,ur,si,sr,ui;
  double fx[TAM], fy[TAM];
  double rex[N], imx[N], temp[N];

  //arquivo de leitura do filtro
  FILE *farq;
  farq = fopen("pontoAjustado.dat", "r");
  for(i=0; i<TAM; i++){
    fscanf(farq, "%lf\t%lf\n", &fx[i], &fy[i]);
  }
  fclose(farq);

  //le rex
  for(i=0; i<N; i++){
    rex[i] = 0;
    imx[i] = 0;
  }
  for(i=0; i<TAM; i++){
    rex[i] = fy[i];
  }

//FFT INVERSA
  //inverte sinal da imx
  for(k=0; k<N; k++)
    imx[k] = -imx[k];

 //calcula FFT
 nm1 = N-1;
 nd2 = N/2;
 m = log(N)/log(2);
 j = nd2;
 k=0;

 //bit reverse
  for(i=1; i<N-1; i++){
    if(i < j){
      tr = rex[j];
      ti = imx[j];
      rex[j] = rex[i];
      imx[j] = imx[i];
      rex[i] = tr;
      imx[i] = ti;
   }
```

```
      k = nd2;
      while(k<=j){
        j = j-k;
        k = k/2;
      }
      j = j+k;
  }
  //loop dos estagios
  for(l=1; l<=m; l++){
    le = pow(2, l);
    le2 = le/2;
    ur = 1.0;
    ui = 0.0;
    sr = cos(pi/le2);            //calcula valores de senos e cossenos
    si = -sin(pi/le2);
    for(j=1; j<=le2; j++){       //calcula cada sub dft
      jm1 = j-1;
      for(i=jm1; i<=nm1; i+=le){                                  //loop para
          cada butterfly
        inp = i+le2;
        tr = rex[inp]*ur - imx[inp]*ui;          //calculo da butterfly
        ti = rex[inp]*ui + imx[inp]*ur;
        rex[inp] = rex[i] - tr;
        imx[inp] = imx[i] - ti;
        rex[i] = rex[i] + tr;
        imx[i] = imx[i] + ti;
      }
      tr = ur;
      ur = tr*sr - ui*si;
      ui = tr*si + ui*sr;
    }
  }

  //divide dominio por N e muda sinal de imx
  for (i=0;i<N;i++){
    rex[i] = rex[i]/(double)N;
    imx[i] = -imx[i]/(double)N;
  }
  //impressao teste Impulso
  FILE *imparq;
  imparq = fopen("Impulso500.dat", "w");
  for(i=0; i<N/2; i++){
    fprintf(imparq, "%d\t%f\n", i,  sqrt(rex[i]*rex[i] + imx[i]*imx[i]));
  }
  fclose(imparq);

//Janelamento
  for(i=0; i<N; i++){
    ind = i+NF/2;
    if(ind > N-1)
      ind = ind - N;
    temp[ind] = rex[i];
    rex[i] = 0;
    imx[i] = 0;
  }
  for(i=0; i<N; i++)
```

```
    rex[i] = temp[i];

 for(i=0; i<N; i++){
   if (i<=NF){
     rex[i] = rex[i]*(0.54 - 0.46*cos((2*pi*i)/NF));
   }
   else{
     rex[i] = 0;
     imx[i] = 0;
   }
 }
 FILE *rexarq;
 rexarq = fopen("rexJanelamento500.dat", "w");
 for(i=0; i<N/2; i++){
   fprintf(rexarq, "%d\t%f\n", i, sqrt(rex[i]*rex[i]+imx[i]*imx[i]));
 }
 fclose(rexarq);

//calcula FFT
nm1 = N-1;
nd2 = N/2;
m = log(N)/log(2);
j = nd2;
k=0;

//bit reverse
 for(i=1; i<N-1; i++){
   if(i < j){
     tr = rex[j];
     ti = imx[j];
     rex[j] = rex[i];
     imx[j] = imx[i];
     rex[i] = tr;
     imx[i] = ti;
  }
     k = nd2;
     while(k<=j){
       j = j-k;
       k = k/2;
     }
     j = j+k;
 }
 //loop dos estagios
 for(l=1; l<=m; l++){
   le = pow(2, l);
   le2 = le/2;
   ur = 1.0;
   ui = 0.0;
   sr = cos(pi/le2);            //calcula valores de senos e cossenos
   si = -sin(pi/le2);
   for(j=1; j<=le2; j++){       //calcula cada sub dft
     jm1 = j-1;
     for(i=jm1; i<=nm1; i+=le){                                  //loop para
         cada butterfly
       inp = i+le2;
       tr = rex[inp]*ur - imx[inp]*ui;          //calculo da butterfly
```

```
        ti = rex[inp]*ui + imx[inp]*ur;
        rex[inp] = rex[i] - tr;
        imx[inp] = imx[i] - ti;
        rex[i] = rex[i] + tr;
        imx[i] = imx[i] + ti;
      }
      tr = ur;
      ur = tr*sr - ui*si;
      ui = tr*si + ui*sr;
    }
  }

  //divide dominio por N e muda sinal de imx
  for (i=0;i<N;i++){
    rex[i] = rex[i]/(double)N;
    imx[i] = -imx[i]/(double)N;
  }
  FILE *fftarq;
  fftarq = fopen("FFT500.dat", "w");
  for(i=0; i<(N/4)+1; i++){
    fprintf(fftarq, "%f\t%f\n",((float) i/(N/2)), 2*sqrt(rex[i]*rex[i]+imx[i
        ]*imx[i]));
  }
  fclose(fftarq);
}
```

6 Referências

Esse trabalho foi feito baseado nas notas de aula do curso de Processamento de Sinais.