Computational Implementation

Giuliano Graziani and Pietro Rossi

Abstract

The document has to be intended as the computational translation of **Implementation details**.In each explanation of the functions, we recall the sections of the **Implementation details** on which they reference.
The final aim is to publish something self-contained and accessible.
In case of suggestions and tips, please contact us at the followings:

pietro.rossi3@unibo.it
giuliano.graziani2@studio.unibo.it

Contents

1 The Cmatrixmultip function

1.1 The function

The function is the computational implementation of section 1.2 and section 1.3: the matrix equation $Cu = v$,where C is a Circular matrix.

As showed by Theorem 1.1, the eigenfunctions of any Circular matrix are in the form of $exp(\frac{i2\pi nj}{N})$ and therefore the matrix C can be always inverted.

1.2 Input and Output

The inputs required are two: **c** and **vec**. The former is the first column of the Matrix C and the latter can be either u or v of the expression $Cu = v$.

The Matlab version of the fft does not require a power of 2 length, however the zero padding operated by the fft function can decrease the final precision.

The final output is either u or v, according to the parameter selected (see following section).

1.3 Parameters

The user can choose to calculate either u or v thanks to the parameter **Ways**. The two options are **reg**(the default option) and **inv**. The **reg** command calculates v , while **inv** u .

1.4 Examples

The function generates always a warning in order to remind to the user that c has to be the first column of the Circular matrix.

The calculation of v

```
c=(1:5);
u=(6:10);
Cmatrixmultip(a,x,'ways','reg')
ans =
    120    125    125    120    110
```

The calculation of u

```
c=(1:5);
v=(6:10);
Cmatrixmultip(a,x,'ways','inv')
ans =
    1.3333    0.3333    0.3333    0.3333    0.3333
```

2 The Toeplitz function

2.1 The function

The function is the computational translation of section 1.5 of *Implementation Details*: it can be used to embed a Toepltiz matrix into a Circular one.

The function $findQ-M$, called inside the Toeplitz function, finds the values of Q and M that satisfy the system:

$$\begin{cases} 2N + Q = 2^M \\ M \quad \text{smallest integer such that} \quad 2^M \geq 2N \end{cases}$$

2.2 Input and Output

The required inputs are the first row (**row**) and the first column (**col**) of the Toeplitz matrix because, as shown in section 1.4, they are sufficient to fully define the Toeplitz. The function requires and checks that the inputs have the same initial element and same cardinality.

The final output is the column vector **r** (equation 8).

3 The Tmatrixmultiplip function

3.1 The function

The function allows the calculation of the expression $Tu = v$ where T is a n-by-n Toeplitz matrix .

The *Tmatrixmultiplip* function takes advantage of the *Toeplitz* function to embed the starting Toeplitz matrix into the Circular Matrix.

3.2 Input and output

The function requires three inputs: the first row (**Trow**) and column (**Tcol**) of the Toeplitz matrix and the vector (**vec**) that can be either u or v .
The final output is an n-dimension vector that gives the first n values obtained by:

$$\begin{pmatrix} z \\ u \end{pmatrix} = C(r) \begin{pmatrix} x \\ 0 \end{pmatrix}$$

because are the ones of the left corner of C (see section 1.5).

It is worth noticing that the starting Toeplitz matrix can be of any dimension because,by the procedure,the resulting vector r is always a power of two and therefore the full precision of the fft function is preserved.

3.3 Parameters

See Parameters section of *Cmatrixmultip*.

4 The FRFT function

4.1 The function

The function allows the user to calculate the Fractional Fast Fouier Transform expressed as the equation $f = Tq$ (see section 1.1).

4.2 Input and Output

The function has two required inputs: the starting vector $\widehat{p}_n$ (**pn**) and ϵ (**e**) factor described in equation (5).
It is worth remarking that the FRFT requires before the negative frequencies and then the positive ones.

The final output is an n-dimension vector that gives back the first n values obtained by

$$\begin{pmatrix} z \\ u \end{pmatrix} = C(r) \begin{pmatrix} x \\ 0 \end{pmatrix}.$$

The values are the ones of the left corner of C (see section 1.5).

Even though the starting $\widehat{p}_n$ vector can be of any cardinality, the procedure achieves its full precision working with a 2^n base.

4.3 Implementation Details

The *frft* function takes advantage of the special symmetry of the specific Toeplitz matrix of section 1.1 in the function *fromToep2Circ* that creates the embedding only for the Fractional FFT.
Therefore the user, interested in using the function *frft*, has to download only the functions:
-*frft*
-*findQ-M*
-*fromToep2Circ*.

5 The alphachar function

5.1 The function

The function calculates N-points of the α-stable characteristic function. There are different ways to represent the characteristic function of the α stable distribution: the one we are taking into consideration, is the following (based on Zolotarev (1986) M parameterization):

$$\varphi_\alpha = \begin{cases} e^{i\mu_0 t - |\sigma t|^\alpha \left[1 + i\beta sign(t) tan(\frac{\pi\alpha}{2})\left(|\sigma t|^{1-\alpha} - 1\right)\right]}, \ \alpha \neq 1 \\ e^{i\mu_0 t - |\sigma t| \left[1 + i\beta sign(t) \frac{2}{\pi} ln(|\sigma t|)\right]}, \ \alpha = 1 \end{cases}$$

where μ_0 and μ are related trough:

$$\mu = \begin{cases} \mu_0 - \beta tan(\frac{\pi\alpha}{2})\sigma, \ \alpha \neq 1 \\ \mu_0 - \beta \frac{2}{\pi}\sigma ln(\sigma), \ \alpha = 1 \end{cases}$$

5.2 Input and Output

The function has 7 inputs: the first four are the parameters on which the α distribution is defined:
- α : *stability* $0 < \alpha \leq 2$
- β : *skewness* $-1 \leq \beta \leq 1$
- σ : *scale* $0 < \sigma < \infty$
- μ : *location* $-\infty < \mu < \infty$

The remaining 3 are :
-**xc**: the bounds of the function
-**wc**: the shifting factor
-**N**: the number of points

The output is a N-vector.

5.3 Examples

In the final example, we combine the *FRFT* function with *alphachar*. The output is the Probability Density Function of an alpha stable distribution with $\alpha = 1.75$ and $\beta = -1$.In order to check the correctness of the obtained results, we have selected as counter-check the package **Stable Distribution** (see https://it.mathworks.com/help/stats/stable-distribution.html) because the PDF values are obtained with a different procedure.

```
N=2^13;
xc=50;
wc=0  ;
dk=1/(2*xc);
dx=(2*xc)/N   ;
e=0.15;
a=1.75;
b=-1;
s=1;
m=0;
f=frft(alphachar(a,b,s,m,xc,wc,N),e);
%% the x-axis values
F=zeros(N,1);
for i  = 1 : N
 F(i)=wc+e*((i-1)*dx-xc);
end
%%
pd=makedist('Stable','alpha',a,'beta',b,...
'gam',s,'delta',m);
y=pdf(pd,F);
```