7．フーリエ解析&FFT

Maple では、フーリエ解析を数式的に実行するコマンドと、数値的に実行するコマンド(高速フーリエ変換/FFT)が用意されています。ここでは、両者の簡単な使用例を紹介します。

フーリエ変換などの積分変換には inttrans パッケージが用意されています。

数式処理によるフーリエ変換

```
> restart;
> with(inttrans);
```

[*addtable*, *fourier*, *fouriercos*, *fouriersin*, *hankel*, *hilbert*, *invfourier*, *invhilbert*, *invlaplace*, *invmellin*, *laplace*, *mellin*, *savetable*] (1.1)

次のコマンドは公式を表示します。fourier コマンドはこのような数式を使い、計算を進めます。
解ける問題は多くはありませんが、解析的な考察を行うことができます。

```
> convert(fourier(f(x),x,u),int);
```

$$\int_{-\infty}^{\infty} f(x)\, e^{-1xu}\, dx \tag{1.2}$$

次の数式のフーリエ変換を求めてみます。Heaviside はステップ関数で、fx は矩形波になります。

```
> fx:=Heaviside(x+1/2)-Heaviside(x-1/2);
```

$$fx := \text{Heaviside}\left(x+\frac{1}{2}\right) - \text{Heaviside}\left(x-\frac{1}{2}\right) \tag{1.3}$$

```
> fu:=simplify(fourier(fx,x,u));
```

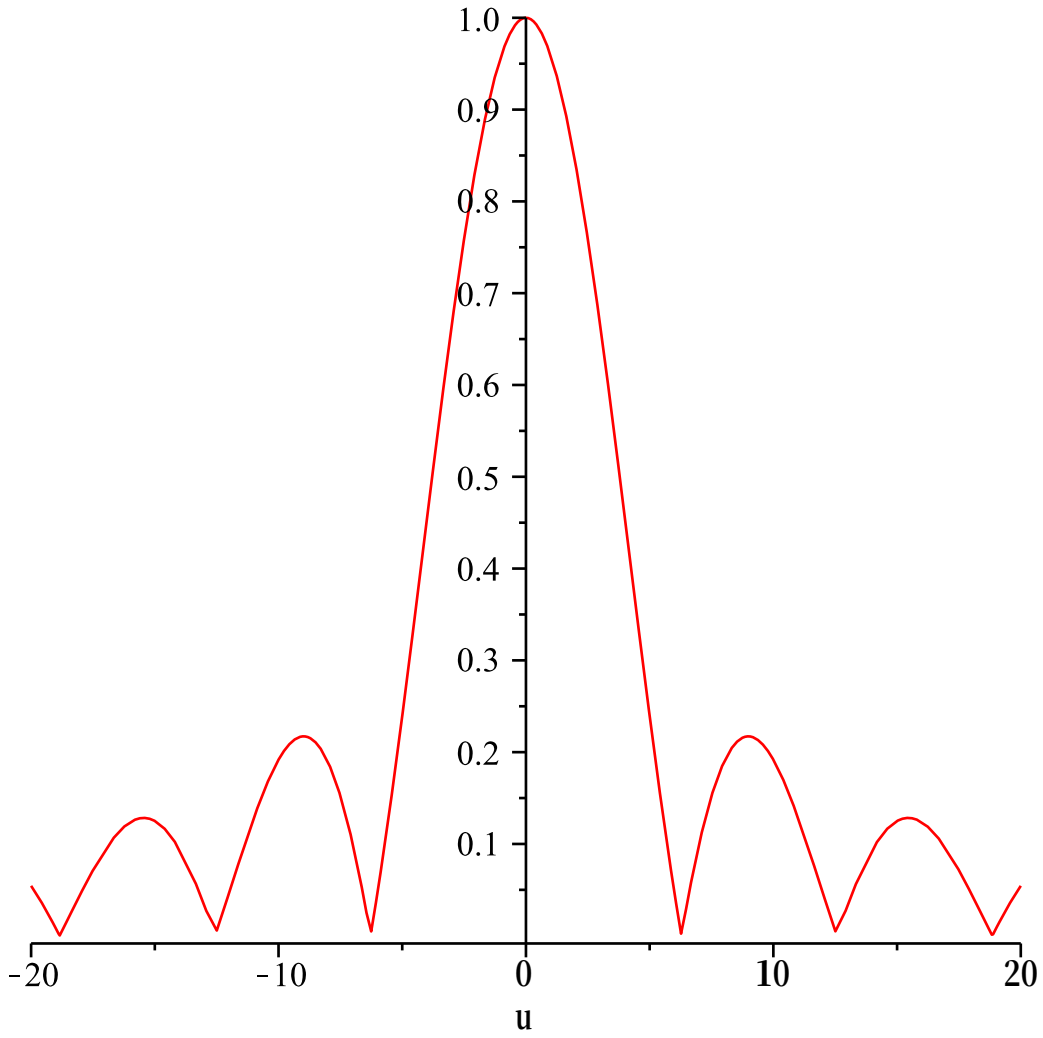
$$fu := \frac{2\sin\left(\frac{1}{2}u\right)}{u} \tag{1.4}$$

変換後の強度をプロットします。

```
> Intensity:=abs(fu);
```

$$Intensity := 2\left|\frac{\sin\left(\frac{1}{2}u\right)}{u}\right| \tag{1.5}$$

```
> plot(Intensity,u=-20..20);
```



高速フーリエ変換

同じ処理を、FFT を使って行います。

```
> restart;
> fx:=Heaviside(x+1/2)-Heaviside(x-1/2);
```

$$fx := \text{Heaviside}\left(x + \frac{1}{2}\right) - \text{Heaviside}\left(x - \frac{1}{2}\right) \tag{2.1}$$

今後の処理で使われるパラメータを定義しておきます。

```
> m:=6:
  n:=2^m:
  sc:=10:
```

FFT コマンドに与えるデータを作成します。

```
> datax:=Array([seq(eval(fx,x=(i/n)*sc-sc/2),i=0..n-1)]);
```

$$datax := \begin{bmatrix} 1 \text{ .. } 64 \text{ Array} \\ \text{Data Type: anything} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{bmatrix} \tag{2.2}$$

```
> nn:=add(i,i=convert(datax,list));
```

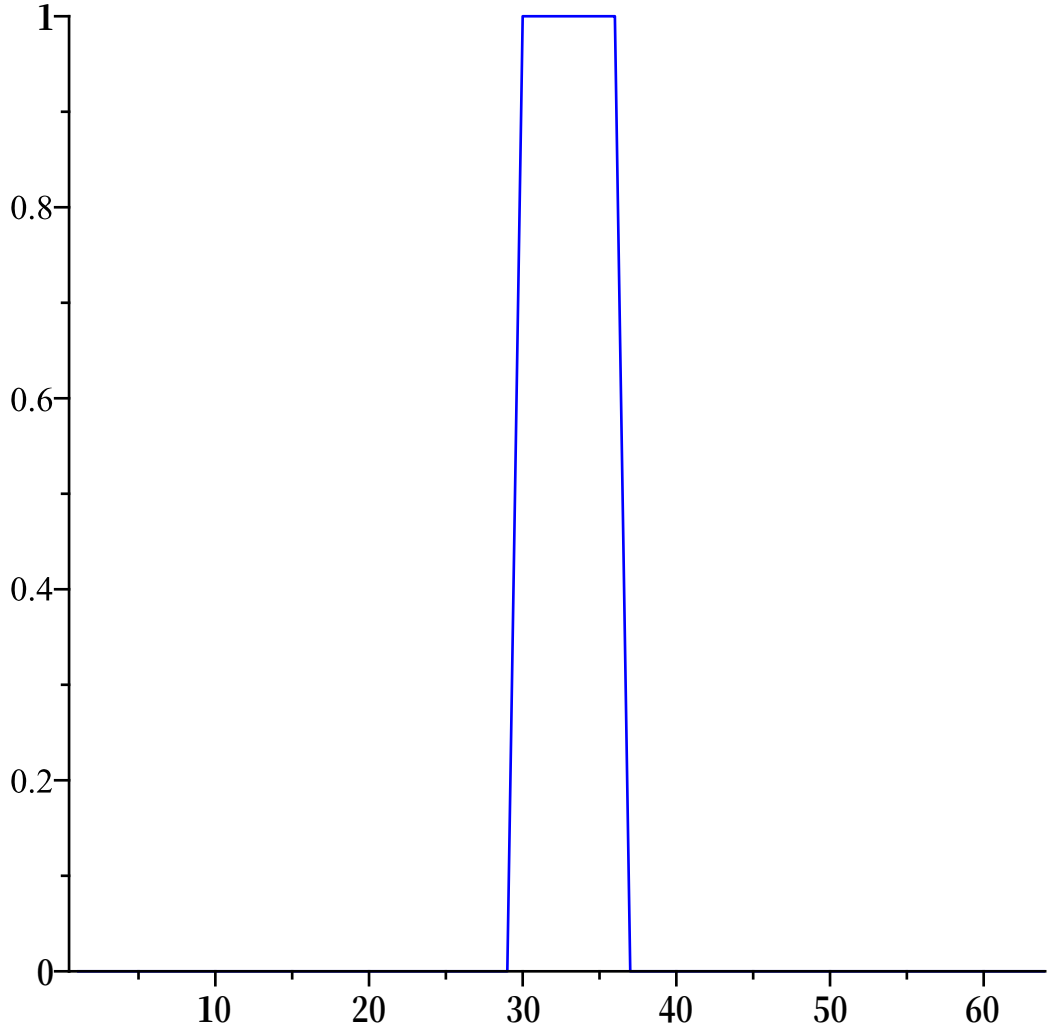
(2.3)

$$nn := 7 \tag{2.3}$$

```
> datay:=Array(1..n,storage=sparse);
```

$$datay := \begin{bmatrix} 1 \text{ .. } 64 \text{ Array} \\ \text{Data Type: anything} \\ \text{Storage: sparse} \\ \text{Order: Fortran\_order} \end{bmatrix} \tag{2.4}$$

```
> p1:=plots[listplot](datax,color=blue):p1;
```




```
> FFT(m,datax,datay);
```

$$64 \tag{2.5}$$

```
> print(datax);
```

$$\begin{bmatrix} 1 \text{ .. } 64 \text{ Array} \\ \text{Data Type: anything} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{bmatrix} \tag{2.6}$$

```
> print(datay);
```

(2.7)

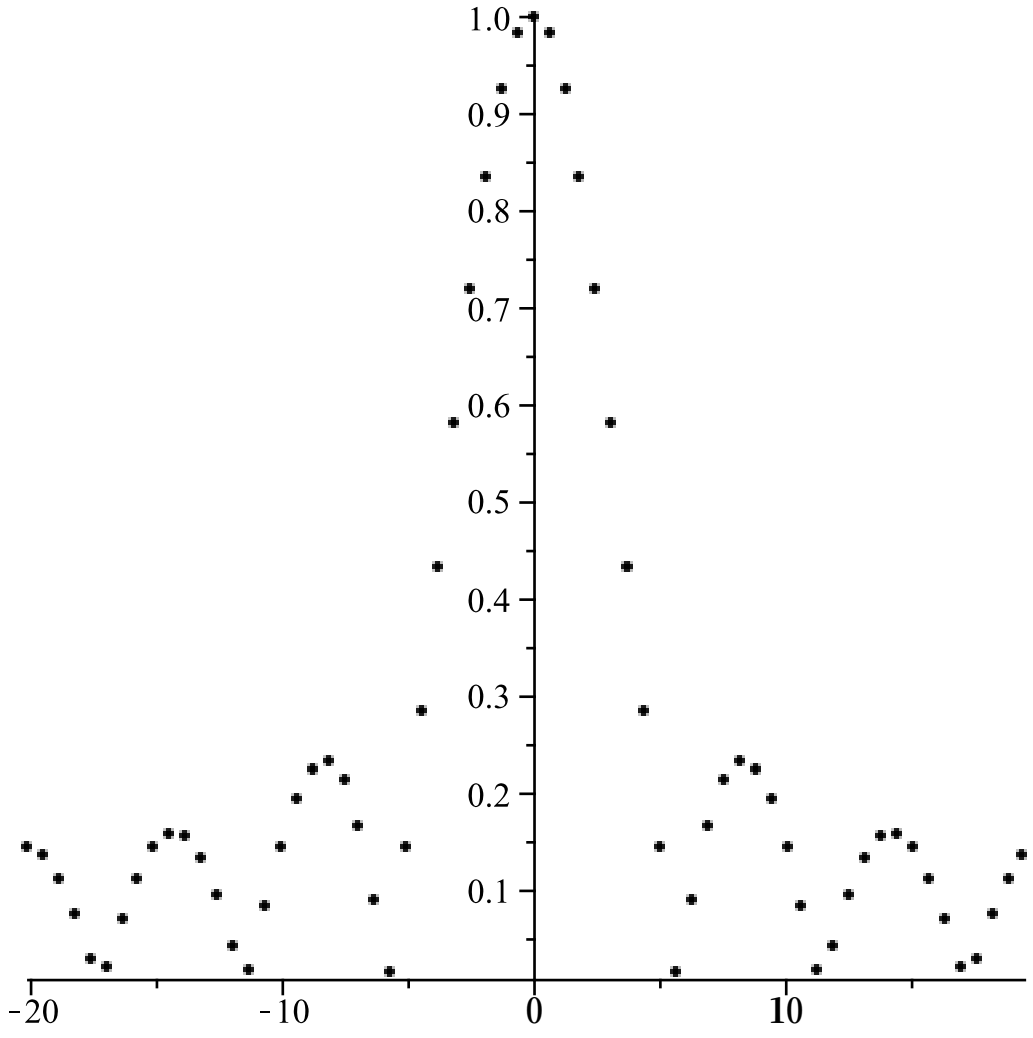
$$\begin{bmatrix} 1 .. 64\ Array \\ Data\ Type: anything \\ Storage: sparse \\ Order: Fortran\_order \end{bmatrix} \tag{2.7}$$

強度をプロットし確認します。

```
> Intensity2:=zip((x,y)->sqrt(x^2+y^2)/nn,datax,datay);
```

$$Intensity2 := \begin{bmatrix} 1 .. 64\ Array \\ Data\ Type: anything \\ Storage: rectangular \\ Order: Fortran\_order \end{bmatrix} \tag{2.8}$$

```
> shift_int:=[seq([(i-n-1)/sc*2*Pi,Intensity2[i]],i=n/2+1..n),seq(
  [(i-1)/sc*2*Pi,Intensity2[i]],i=1..n/2)]:
> plots[pointplot](shift_int,style=point,symbol=solidcircle) ;
```



離散点ですが、数式処理（上記式 1.5）と同じ結果が得られます。

逆高速フーリエ変換

```
> iFFT(m,datax,datay);
```

$$64 \tag{2.1.1}$$

```
> Intensity3:=zip((x,y)->sqrt(x^2+y^2),datax,datay):
```

```
> plots[listplot](Intensity3);
```