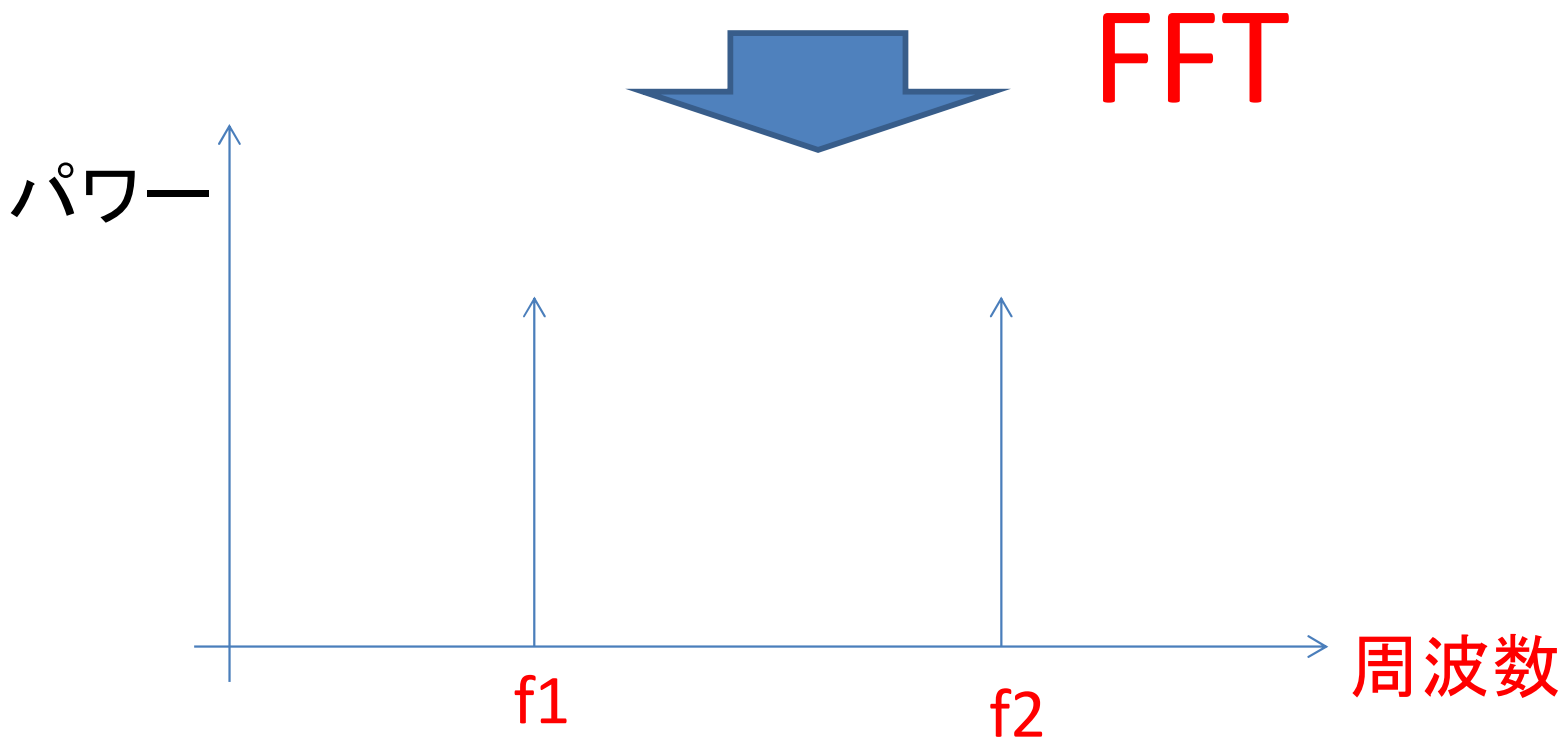
フーリエ変換

FFT (Fast Fourier Transformation)

- １次元のFFTにより信号の周波数スペクトルを表示することができる

$$y(t) = \sin(2\pi f_1 \mathrm{t}) + \sin(2\pi f_2 \mathrm{t})$$



今回の目的

- 画像データに対して二次元FFTを実行し，画像に含まれる周波数成分を表示する

- FFTを利用した周波数フィルタリングを行う

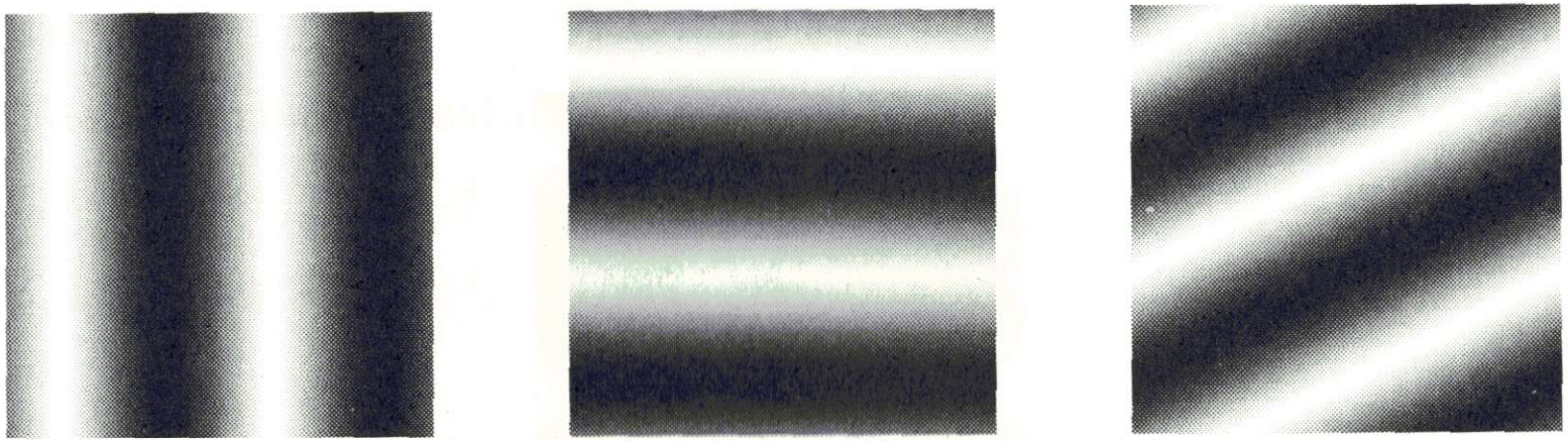
１次元信号と2次元信号

$$f(x) = a \cdot \sin\left(\frac{2\pi}{N}(x-\theta)\right)$$

X, Yに独立な周波数成分を持つ正弦波

$$f(x,y)$$
$$= a_0 + a \cdot \sin\left(\frac{2\pi}{N_x}(x-\theta_x) + \frac{2\pi}{N_y}(y-\theta_y)\right)$$

二次元空間の正弦波関数の例



（横・縦のサイズはいずれも 256 画素）

(a) $a_0 = 128, \quad a = 128$
$N_x = 128, \quad \theta_x = 0$
$N_y = 0, \quad \theta_y = 0$

(b) $a_0 = 128, \quad a = 128$
$N_x = 0, \quad \theta_x = 0$
$N_y = 128, \quad \theta_y = 0$

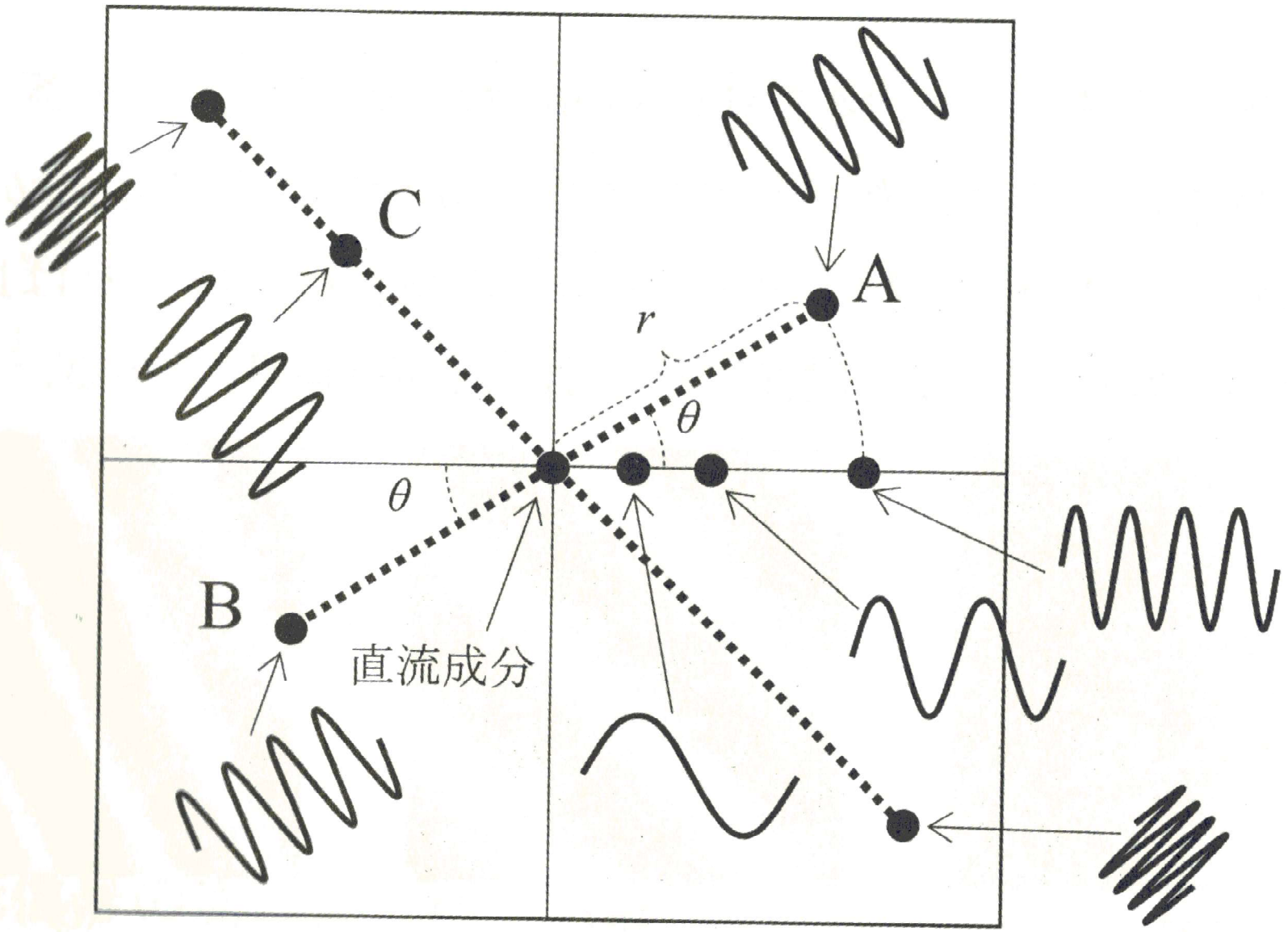
(c) $a_0 = 128, \quad a = 128$
$N_x = 256, \quad \theta_x = 0$
$N_y = 128, \quad \theta_y = 0$

注：N_x=0 または N_y=0 はその方向の波がないことを表している．

図 5.3　2 次元空間の正弦波関数の例

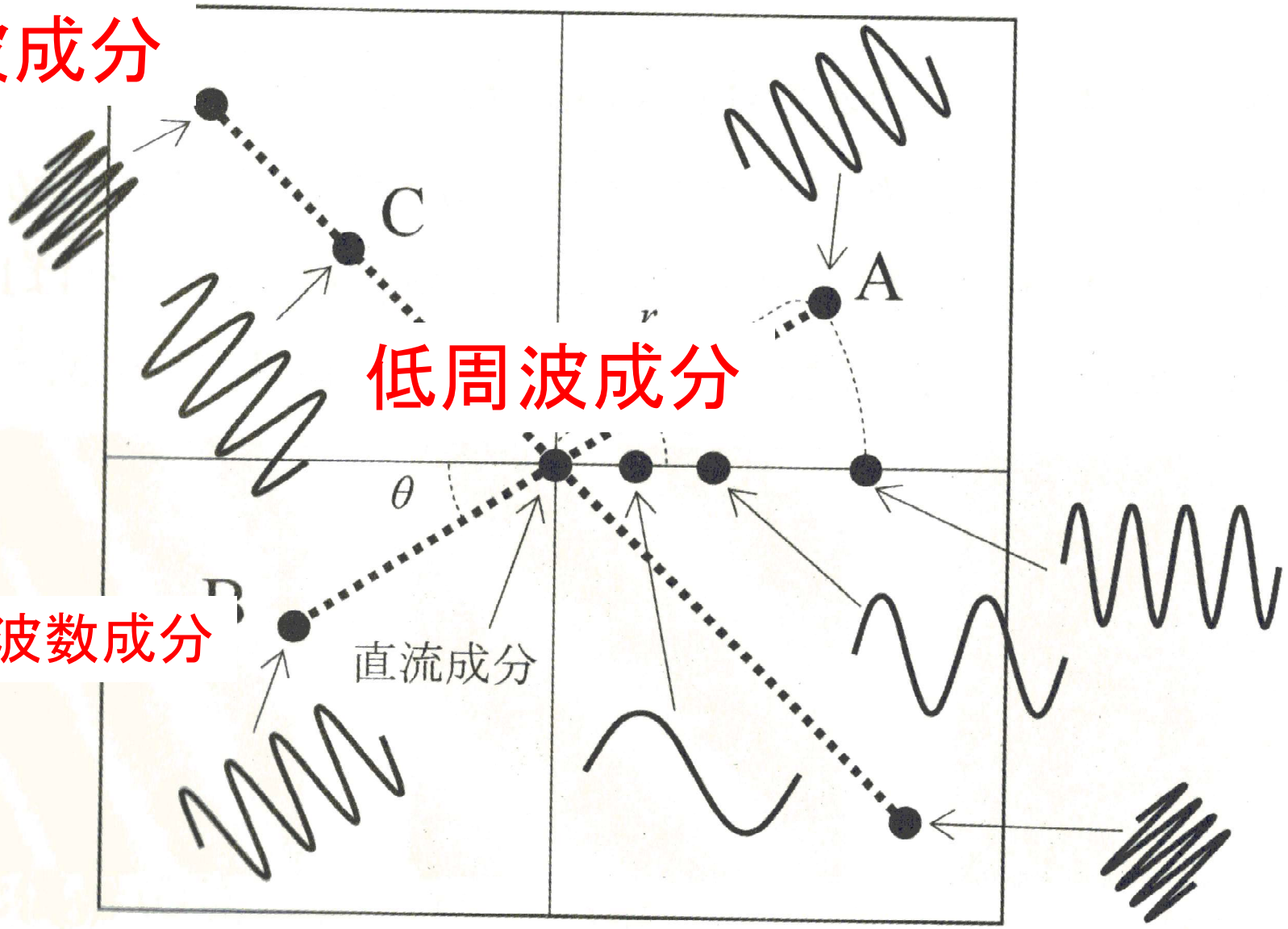
画像の振幅スペクトルによる周波数解析

FFTによる周波数スペクトル



FFTによる周波数スペクトル

高周波成分



低周波成分

中間周波数成分

FFTによる周波数フィルタリング

① 原画像をFFTしてスペクトル表示

② 特定の周波数成分を削除

③ IFFTにより画像を復元

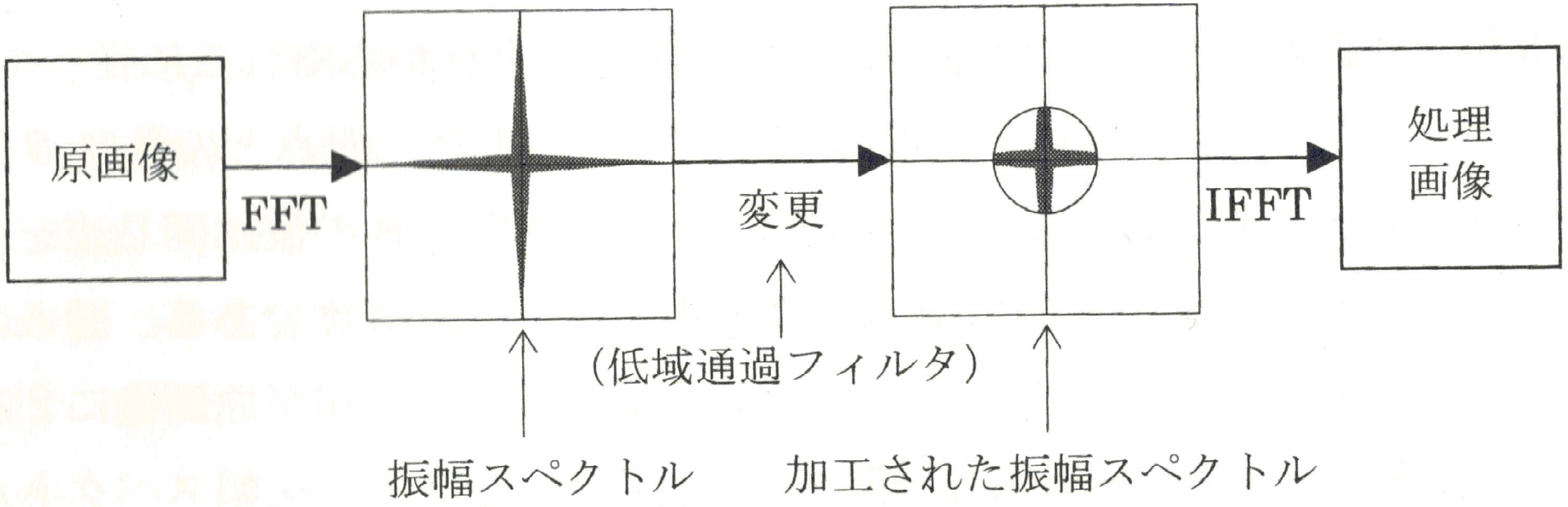


図 5.8　画像の周波数フィルタリング（低域通過フィルタの例）

課題1

- 任意の周波数を持つ正弦波画像を作成する

課題2

- 作成した画像をFFTし，周波数スペクトルを表示する

課題1

- 任意の周波数を持つ正弦波画像を作成する

main関数

```
int main() {
    char file[256] = "";
    make_original_image(0);
    save_image(0, file);
    return 0;
}
```

ヘッダとプロトタイプ宣言の追加

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

#define PI 3.14159265

void make_original_image(int n);
void add_sin_waves(int n, double a0, double a, double Nx, double
theta_x, double Ny, double theta_y);
```

make_original_image関数

```
void make_original_image(int n) {
      width[n] = 256;
      height[n] = 256;

      init_image(n, 0);
      add_sin_waves(n, 128.0, 128.0,
                        128.0, 0.0, 0.0, 0.0);

}
```

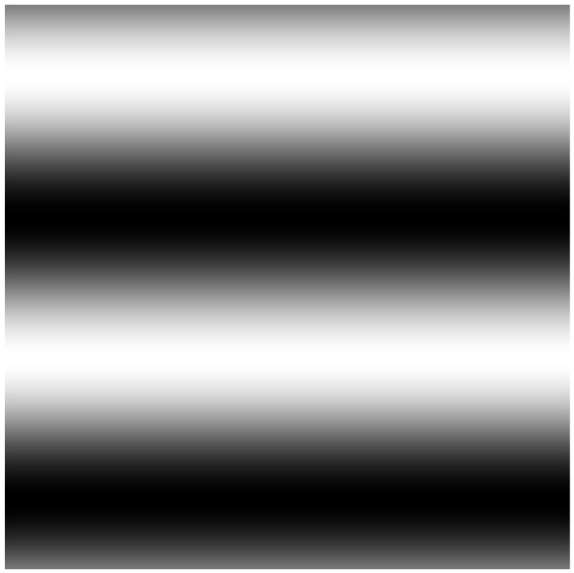
add_sin_waves関数

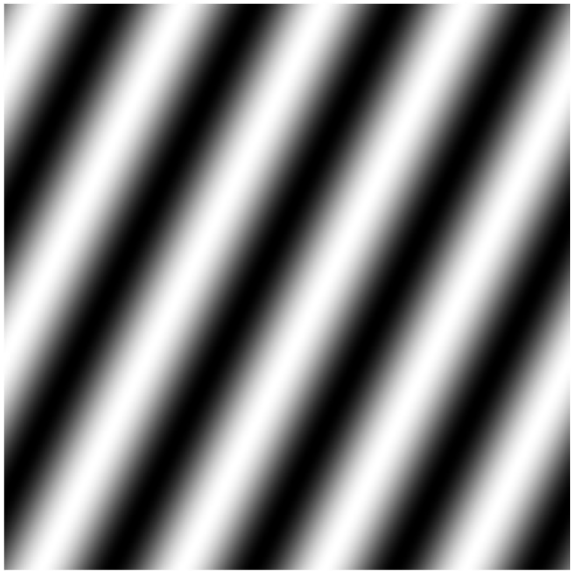
```
void add_sin_waves(int n, double a0, double a, double Nx,
                          double theta_x, double Ny, double theta_y) {
        int x, y, brightness;
        double term_x, term_y, value;

        for (y = 0; y < height[n]; y++) {
                for (x = 0; x < width[n]; x++) {
                        if (Nx == 0)
                                term_x = 0.0;
                        else
                                term_x = 2.0 * PI / Nx * (x - theta_x);
                        if (Ny == 0)
                                term_y = 0.0;
                        else
                                term_y = 2.0 * PI / Ny * (y - theta_y);
```

```
                        value = a0 + a * sin(term_x + term_y);
                        brightness = image[n][x][y] + value;
                        if (brightness > 255)
                                brightness = 255;
                        else if(brightness < 0)
                                brightness = 0;
                        image[n][x][y] = brightness;
                }
        }
}
```

実行例

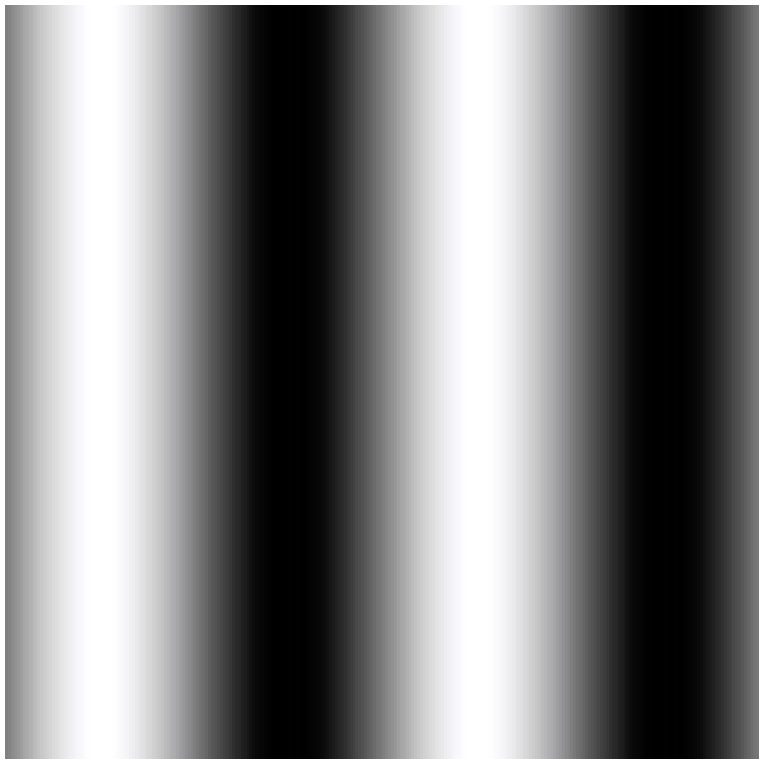
- いくつか条件を変更して正弦波画像を作成する





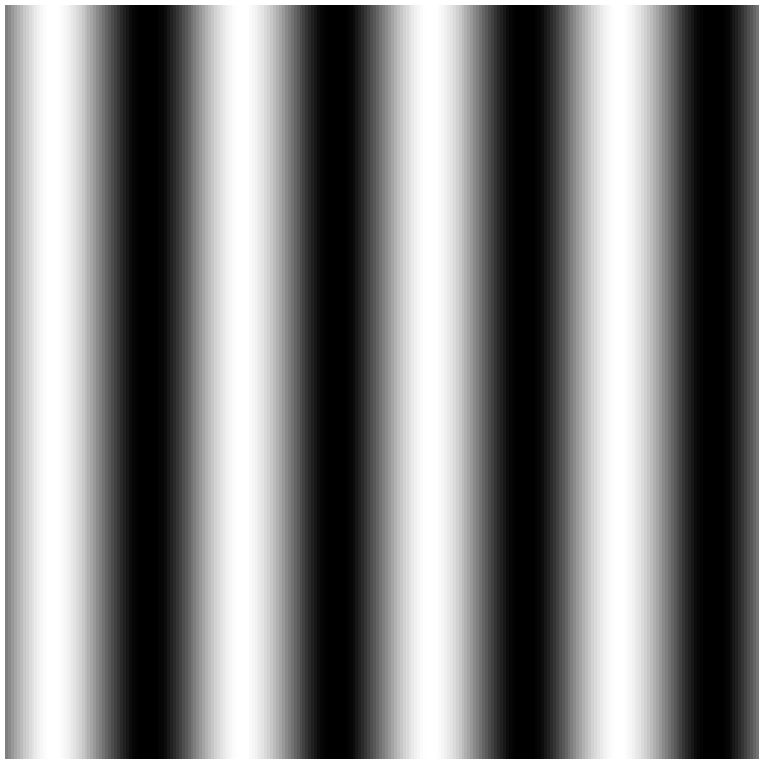
0.pgm

```
add_sin_waves(n, 128.0, 128.0, 128.0, 0.0, 0.0, 0.0);
```



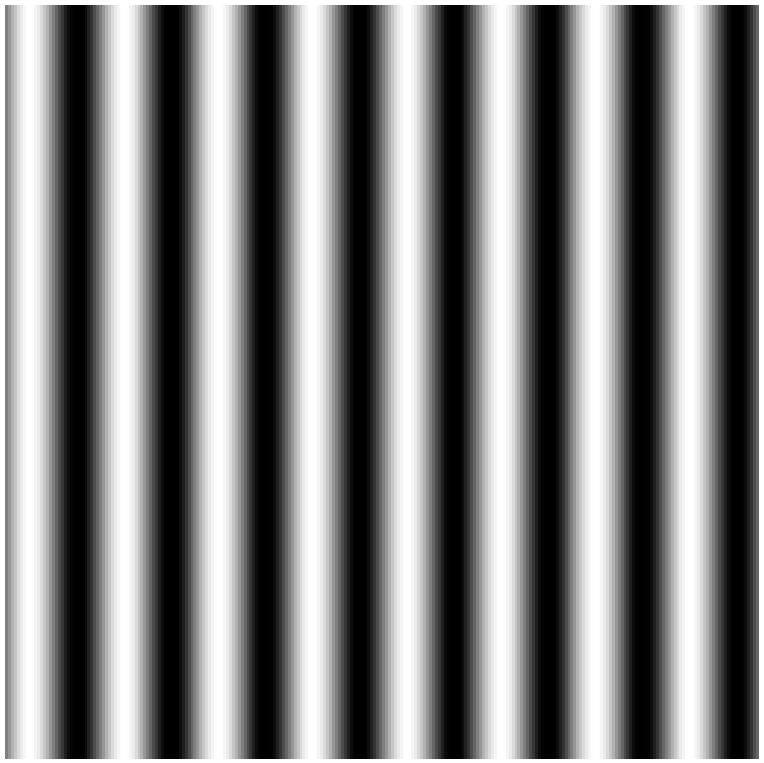
1.pgm

```
add_sin_waves(n, 128.0, 128.0, 64.0, 0.0, 0.0, 0.0);
```



2.pgm

```
add_sin_waves(n, 128.0, 128.0, 32.0, 0.0, 0.0, 0.0);
```

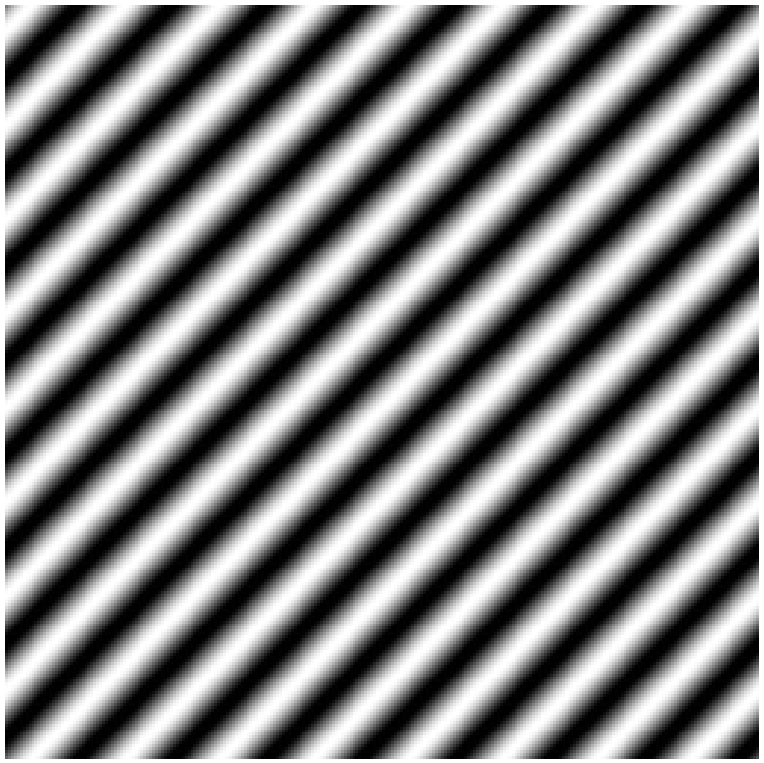


3.pgm

```
add_sin_waves(n, 128.0, 128.0, 0.0, 0.0, 32.0, 0.0);
```

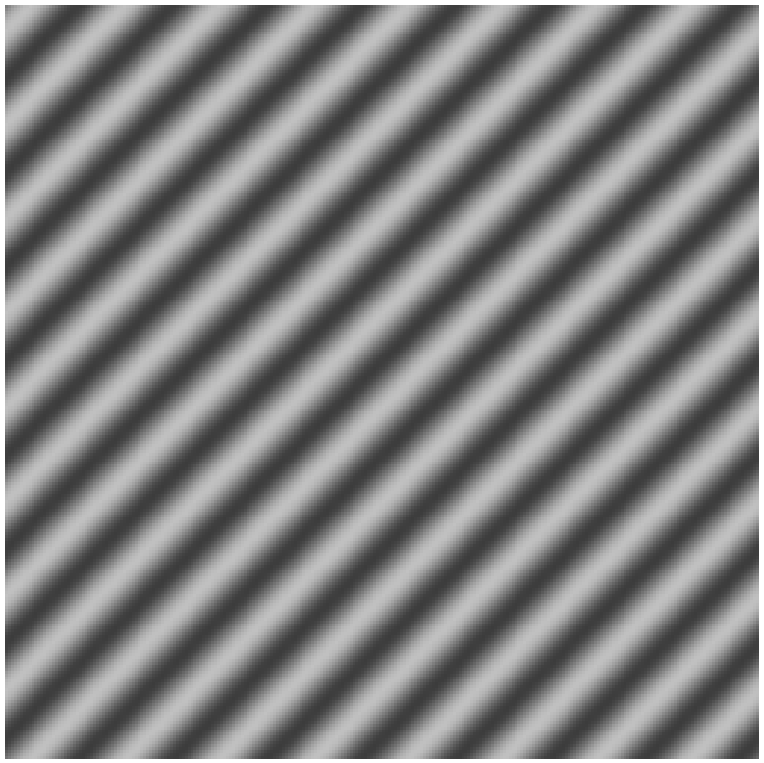
4.pgm

```
add_sin_waves(n, 128.0, 128.0, 32.0, 0.0, 32.0, 0.0);
```



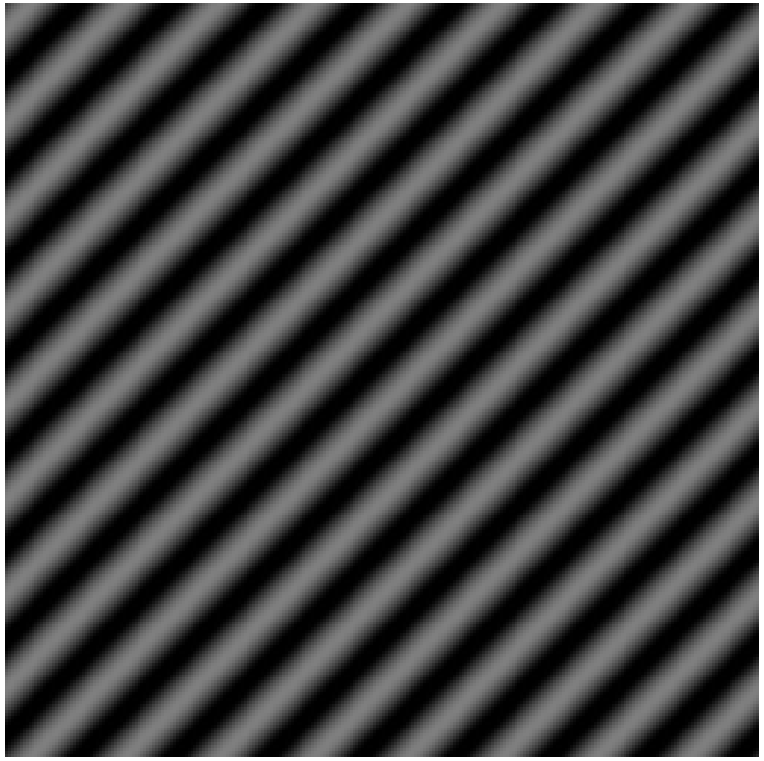
5.pgm

```
add_sin_waves(n, 128.0, 64.0, 32.0, 0.0, 32.0, 0.0);
```



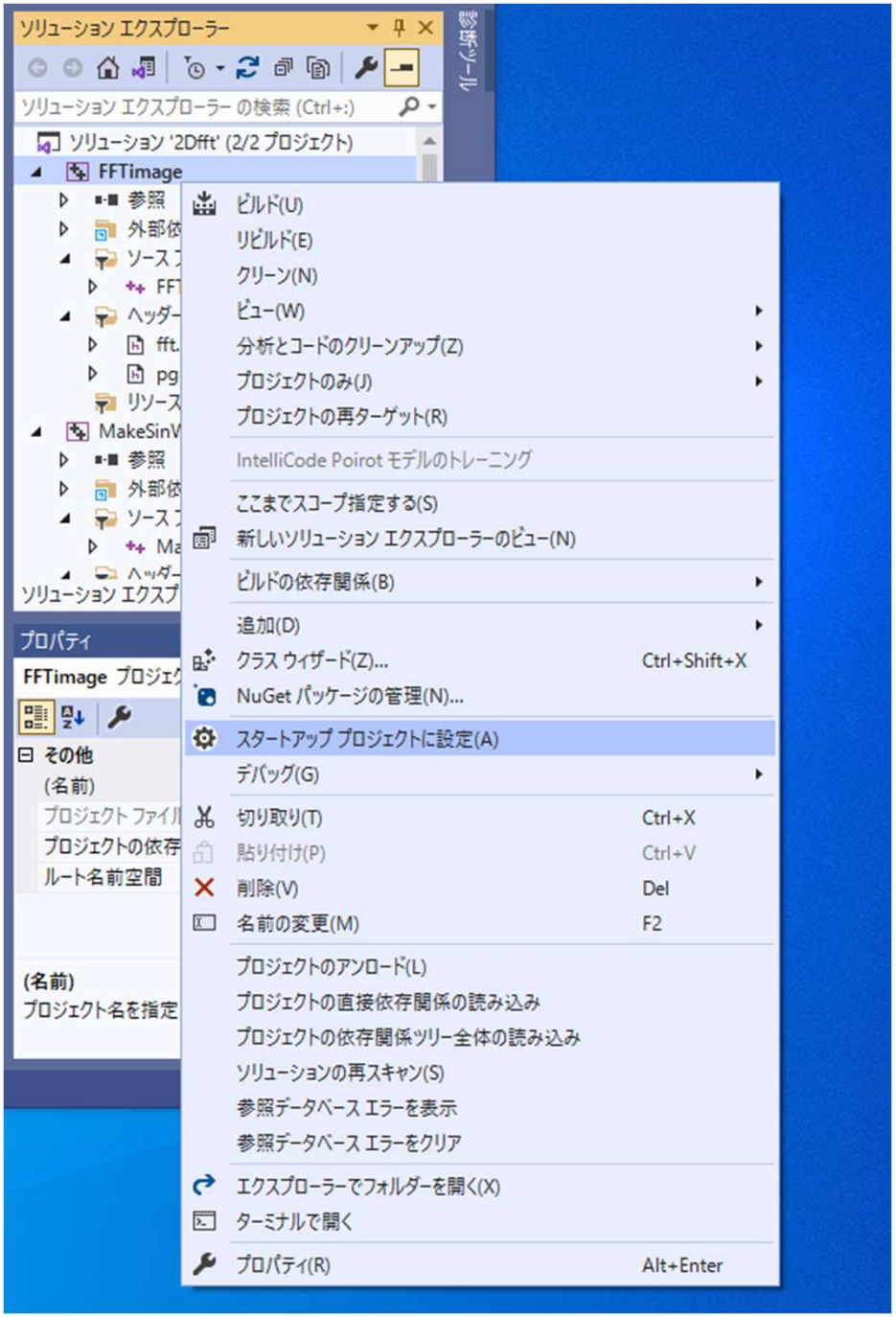
6.pgm

```
add_sin_waves(n, 64.0, 64.0, 32.0, 0.0, 32.0, 0.0);
```



課題2

- 作成した画像をFFTし，周波数スペクトルを表示する


ソリューション エクスプローラー
ソリューション エクスプローラー の検索 (Ctrl+;)
ソリューション 'Dfft' (2/2 プロジェクト)
FFTimage
参照
外部依
ソース
FF
ヘッダー
fft.
pg
リソース
MakeSinW
参照
外部依
ソース
Ma
ヘッダー
ソリューション エクスプ
診断ツール
プロパティ
FFTimage プロジェク
その他
(名前)
プロジェクト ファイル
プロジェクトの依存
ルート名前空間
(名前)
プロジェクト名を指定
ビルド(U)
リビルド(E)
クリーン(N)
ビュー(W)
分析とコードのクリーンアップ(Z)
プロジェクトのみ(J)
プロジェクトの再ターゲット(R)
IntelliCode Poirot モデルのトレーニング
ここまでスコープ指定する(S)
新しいソリューション エクスプローラーのビュー(N)
ビルドの依存関係(B)
追加(D)
クラス ウィザード(Z)... Ctrl+Shift+X
NuGet パッケージの管理(N)...
スタートアップ プロジェクトに設定(A)
デバッグ(G)
切り取り(T) Ctrl+X
貼り付け(P) Ctrl+V
削除(V) Del
名前の変更(M) F2
プロジェクトのアンロード(L)
プロジェクトの直接依存関係の読み込み
プロジェクトの依存関係ツリー全体の読み込み
ソリューションの再スキャン(S)
参照データベース エラーを表示
参照データベース エラーをクリア
エクスプローラーでフォルダーを開く(X)
ターミナルで開く
プロパティ(R) Alt+Enter

main関数

```
int main() {
        char file[256] = "";
        load_image(0, file);
        make_original_data(0);
        FFT2(1);
        save_spectrum(1);
        save_image(1, file);
        return 0;
}
```

FFT2(1);
２次元のfftを実行(fft.h)に定義

ヘッダとプロトタイプ宣言

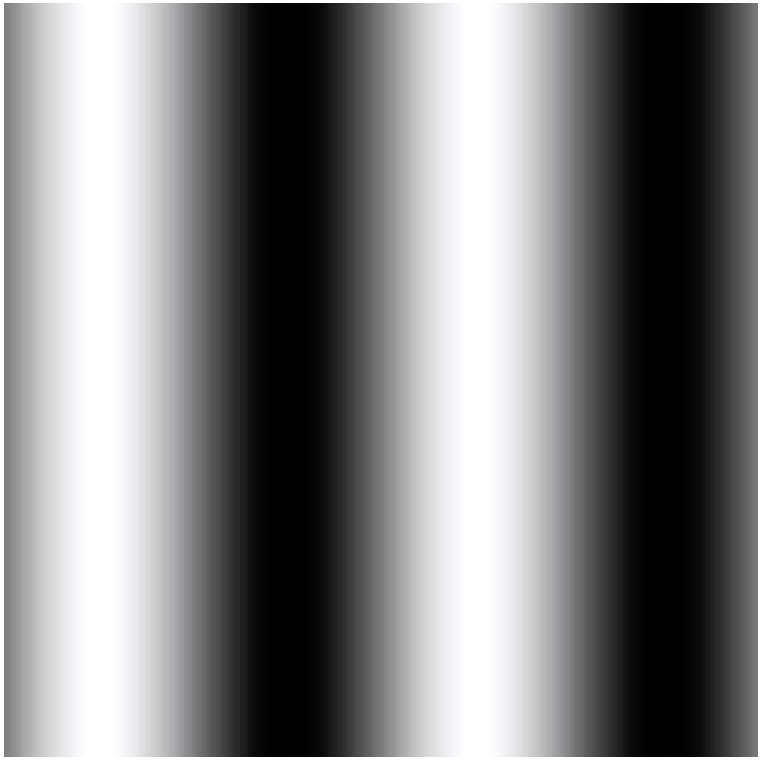
```
#include <stdio.h>
#include "pgmlib.h"
#include <math.h>
#include "fft.h"


void make_original_data(int n);
void save_spectrum(int n);
```

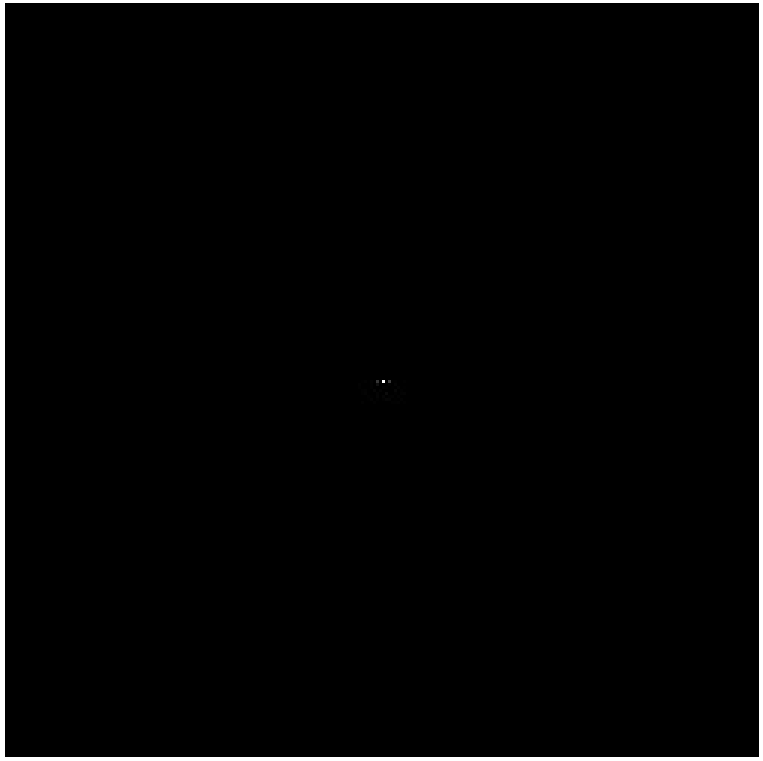
```
void make_original_data(int n);
```
画像データを実部に代入し，虚部＝0とする

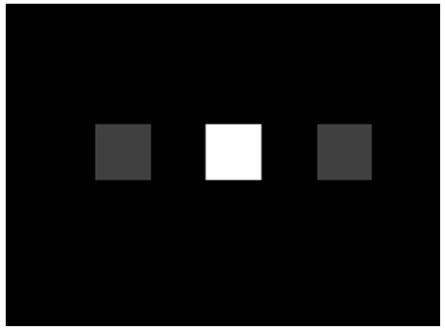
```
void save_spectrum(int n);
```
スペクトルデータを画像ファイルとして出力する

0.pgm

```
add_sin_waves(n, 128.0, 128.0, 128.0, 0.0, 0.0, 0.0);
```



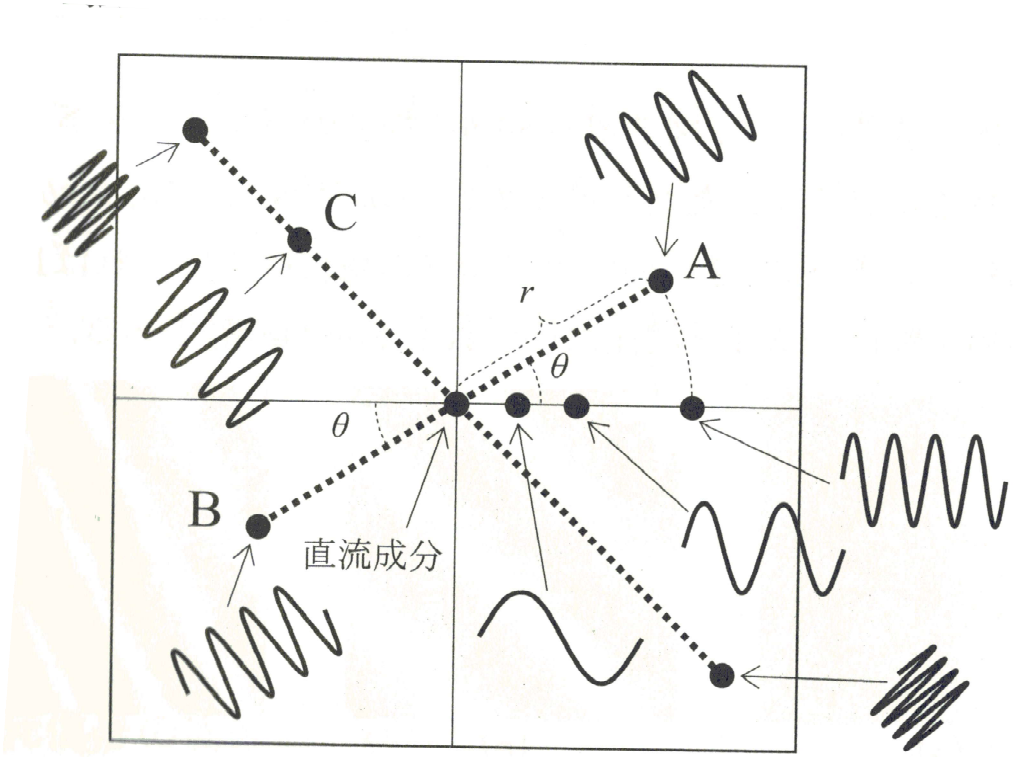


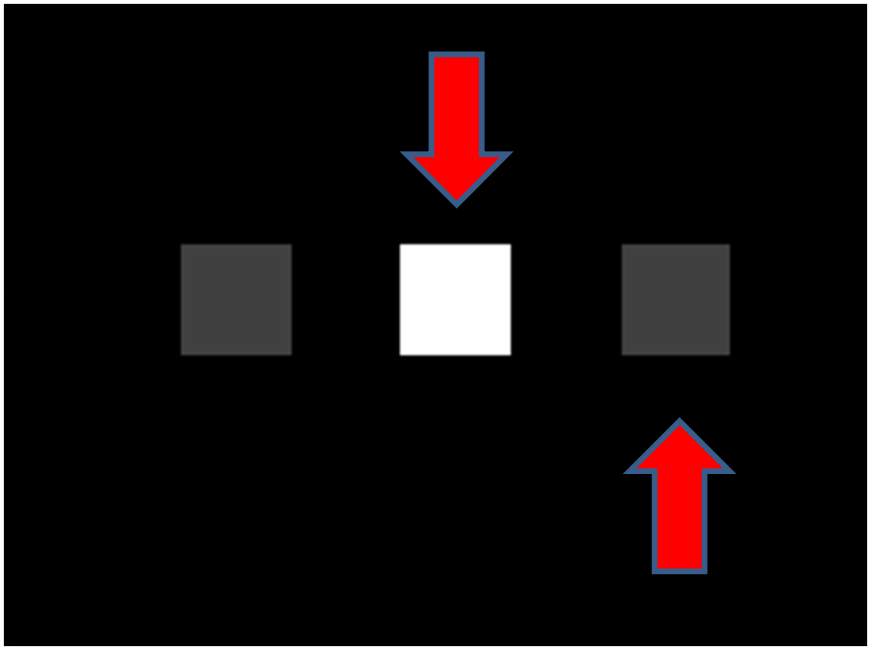


0.pgm

add_sin_waves(n, 128.0, 128.0, 128.0, 0.0, 0.0, 0.0);

直流成分 a0

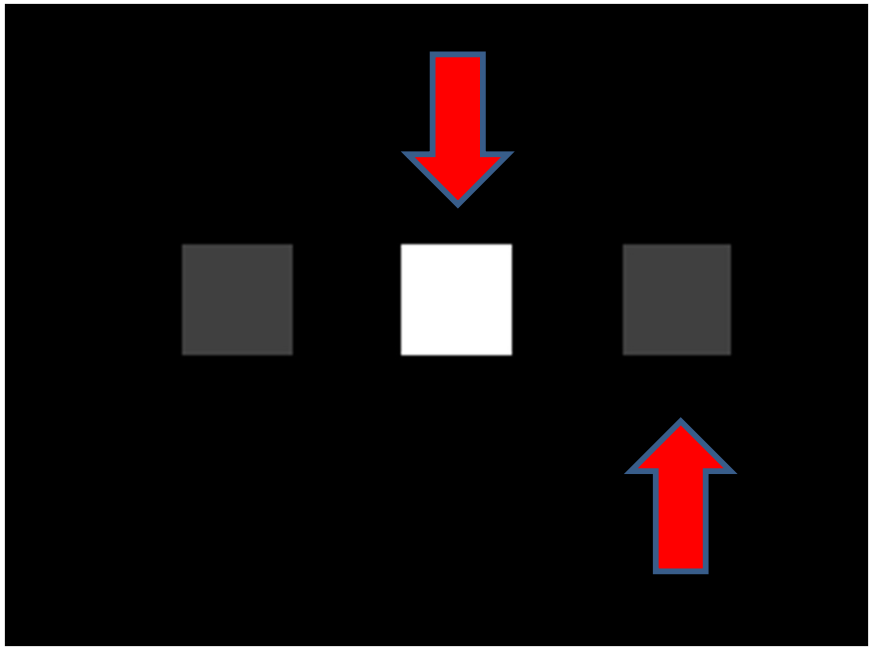




周期 128

0.pgmと1.pgmの比較

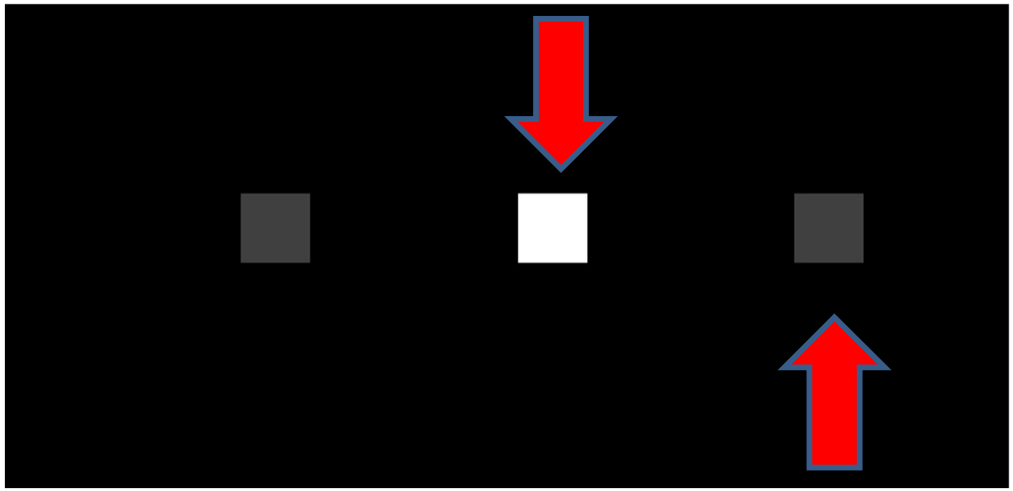
直流成分　a0



周期　128

```
add_sin_waves(n, 128.0, 128.0, 128.0,
0.0, 0.0, 0.0);
```

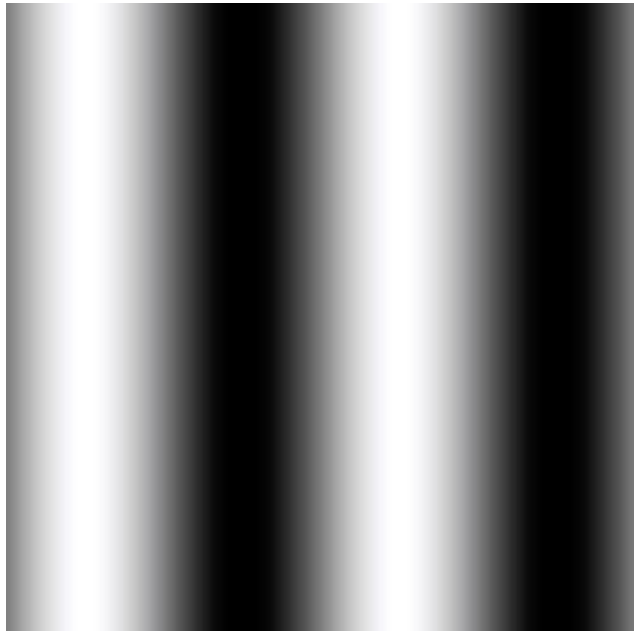
直流成分　a0



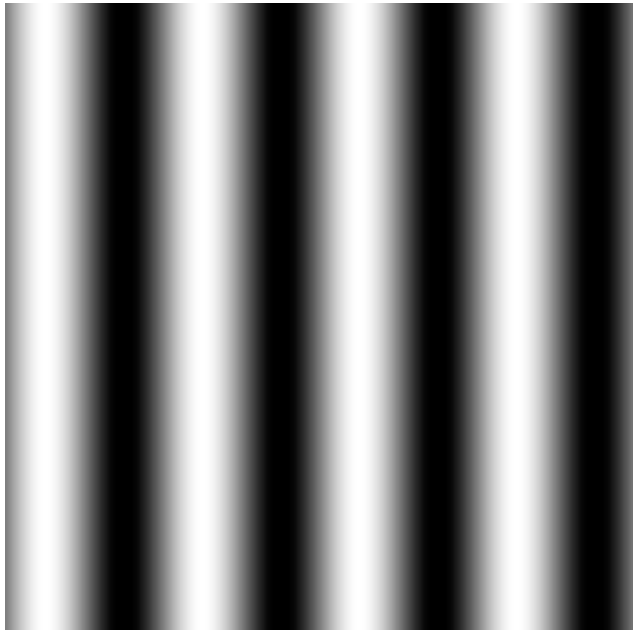
周期　64

```
add_sin_waves(n, 128.0, 128.0, 64.0,
0.0, 0.0, 0.0);
```

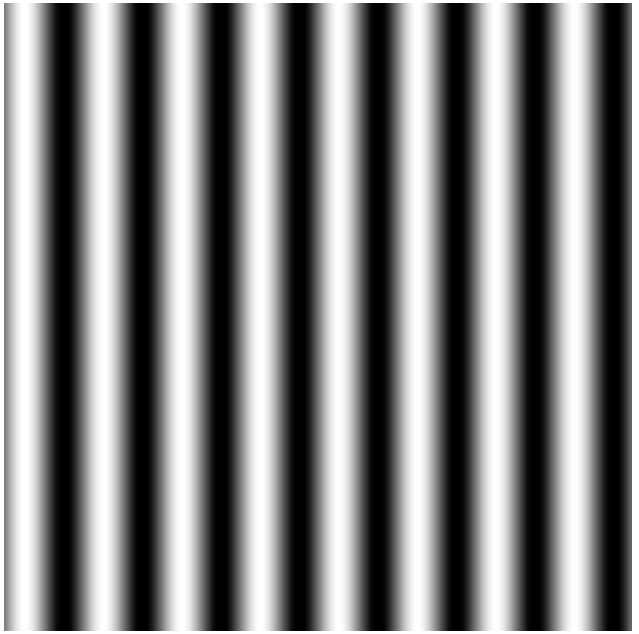
0.pgm

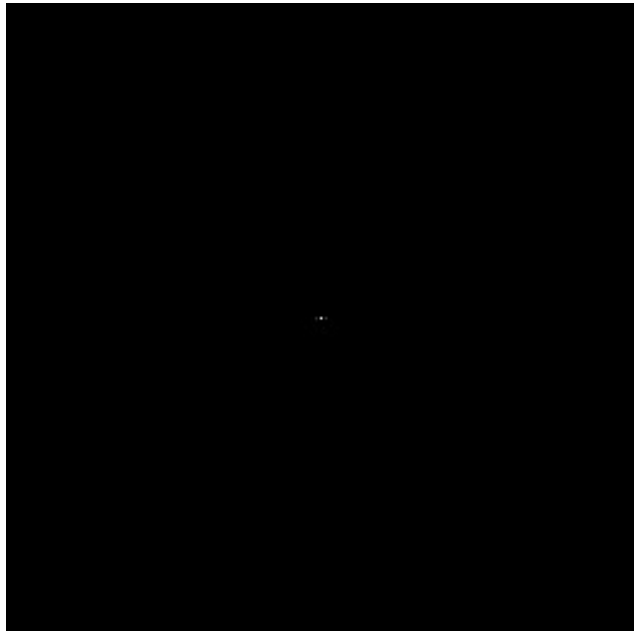


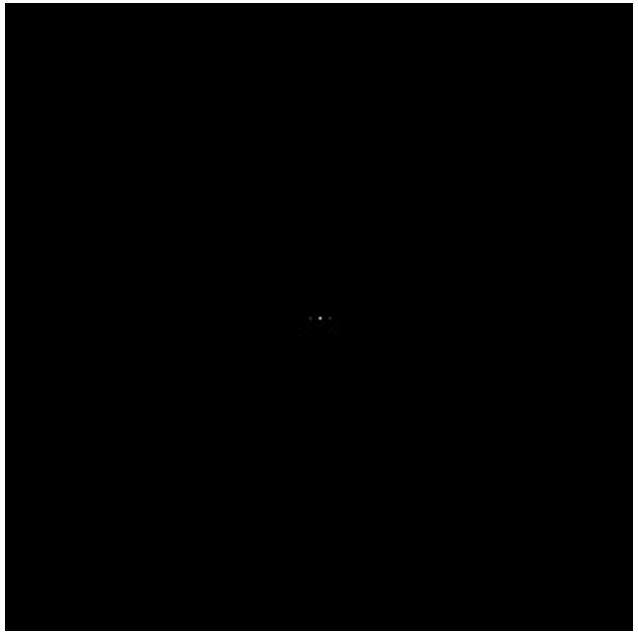
1.pgm

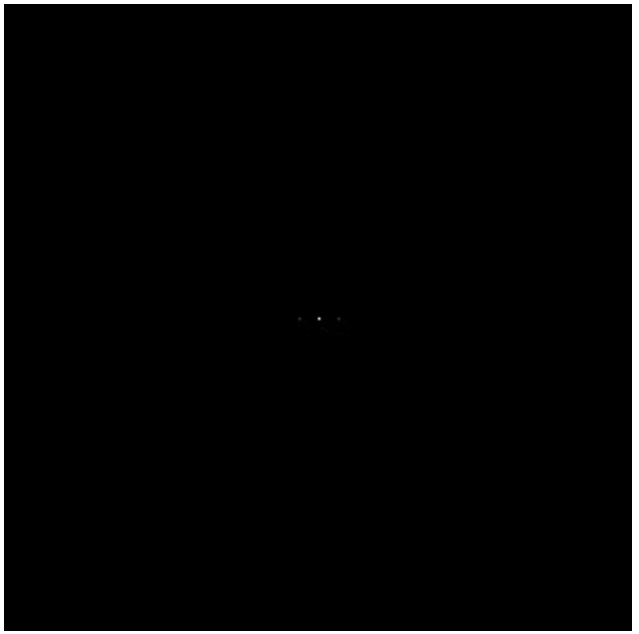


2.pgm



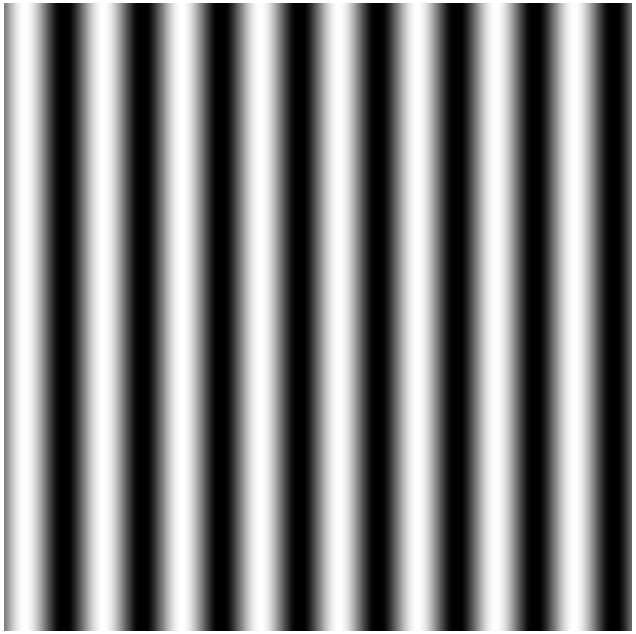






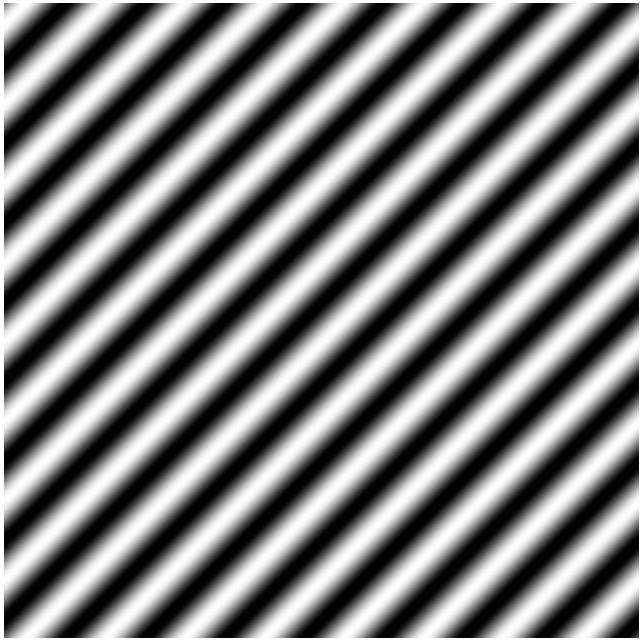
中心部を拡大して様子を観察する

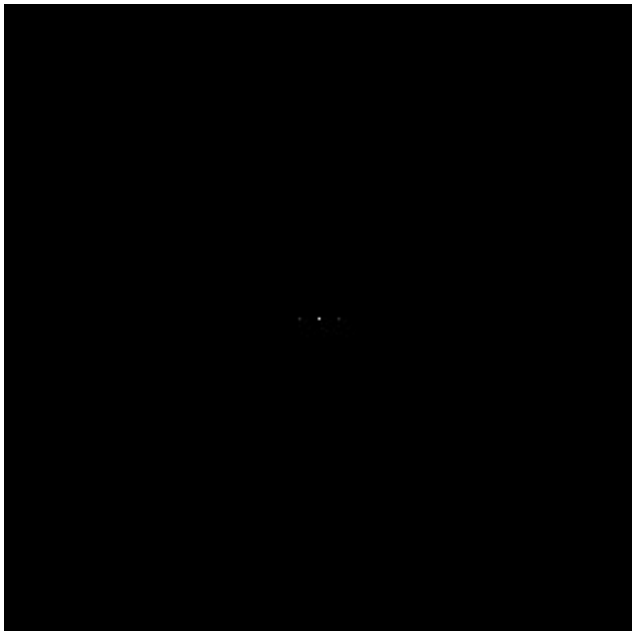
2.pgm

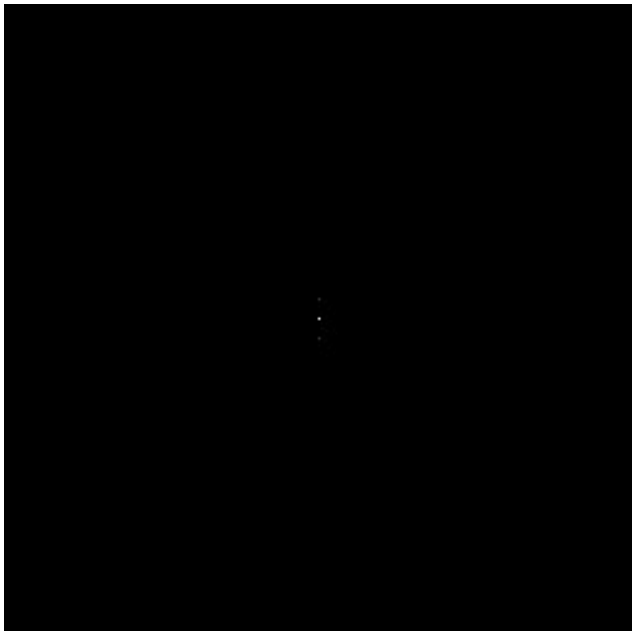


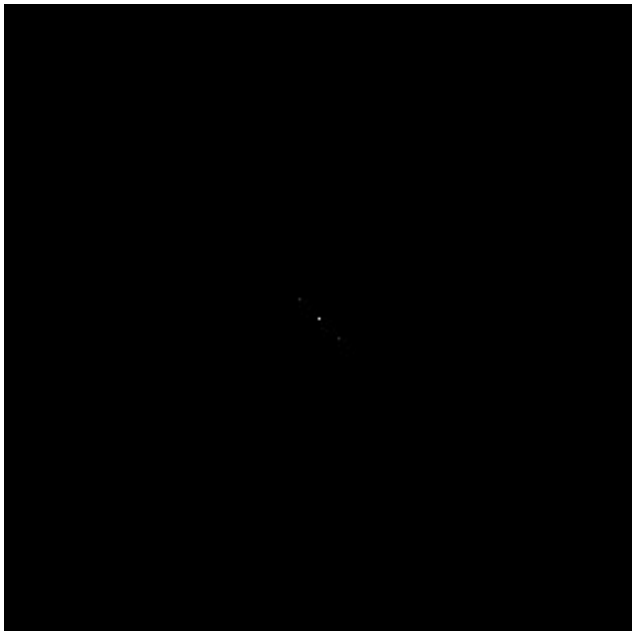
3.pgm

4.pgm









中心部を拡大して様子を観察する