3. 課題演習（基礎編）

By Satoru Yamamoto

3 課題演習（基礎編）

3.1 輝度値データから反射率データの作成

3.2 太陽距離を補正した反射率の作成

3.3 コンティニューム(連続光成分)処理

3 課題演習（基礎編）

3.1 (演習a)輝度値データから反射率データの作成

3.1 輝度値データから反射率データの作成

・内容：輝度値データを太陽光のデータで割り算する

$$R(n) = \frac{I(n)}{F(n)} \frac{1}{\pi}$$

R：拡散反射率

I：輝度値 [W/m²/μm/sr]

F：太陽光 [W/m²/μm/sr]

SP in Cを使って

(1) 太陽光データの読み込み

(2) 輝度値データと太陽光データの割り算処理

(3) データ出力

3.1 輝度値データから反射率データの作成

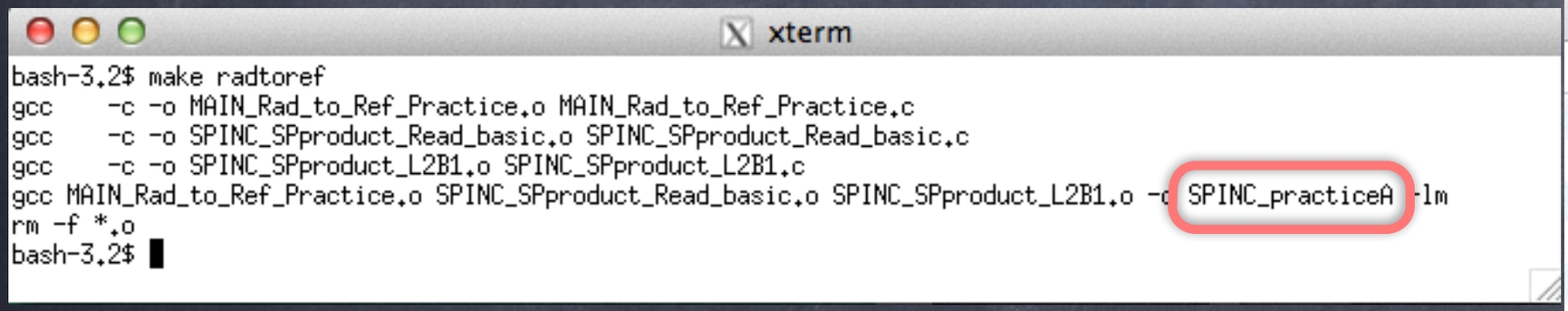
手順1：spinc実行ファイルの作成

・MAIN_Rad_to_Ref_Practice.c

が対象のmainファイル

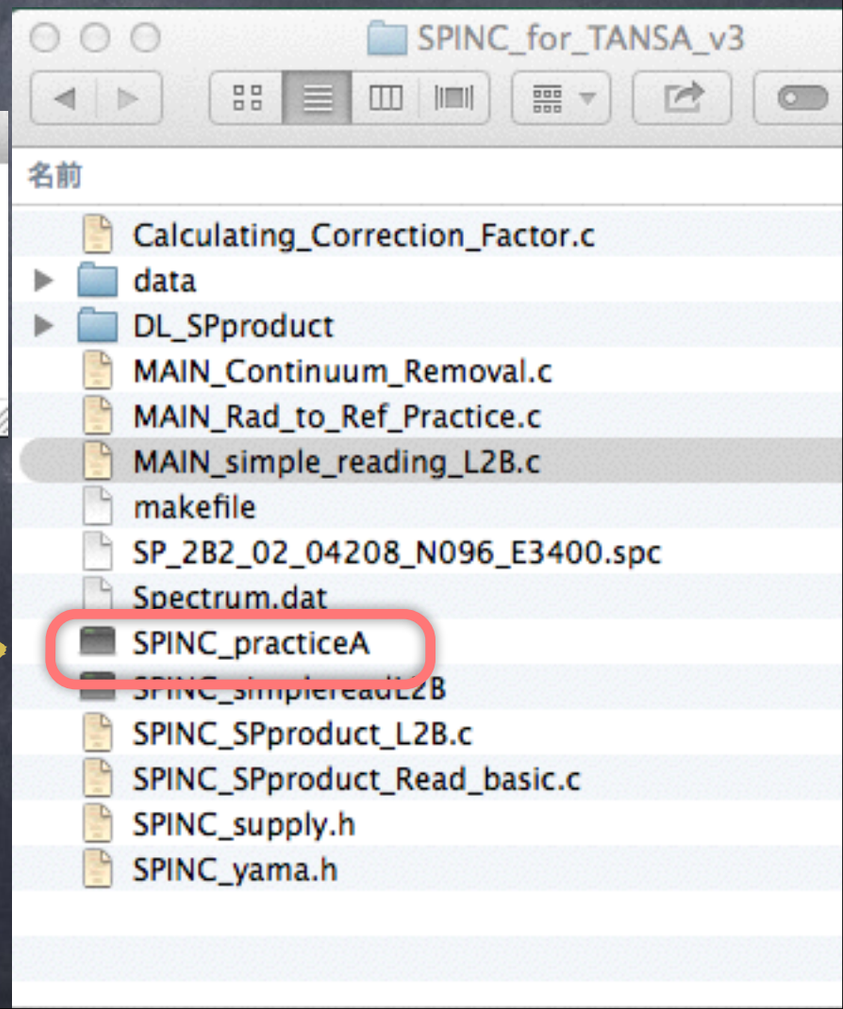
・make radtorefによるコンパイル

> make radtoref



実行ファイルSPINC_practiceA

が生成される



3.1 輝度値データから反射率データの作成

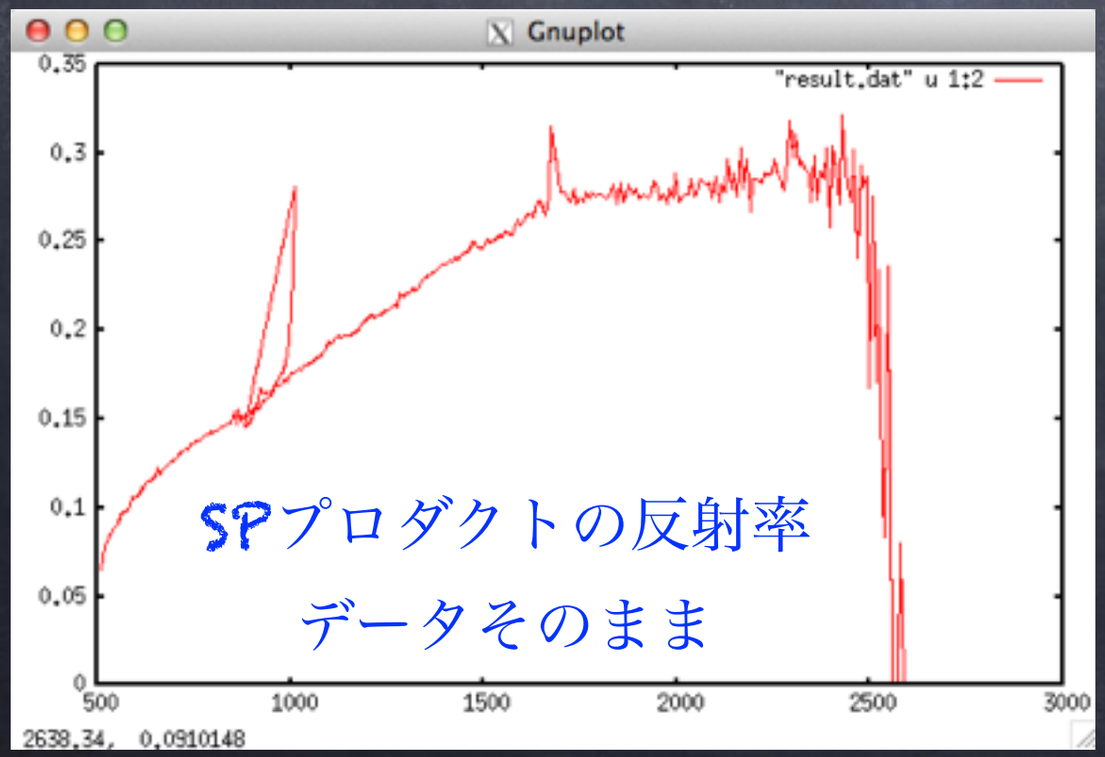
手順２：spinc_practiceAの実行

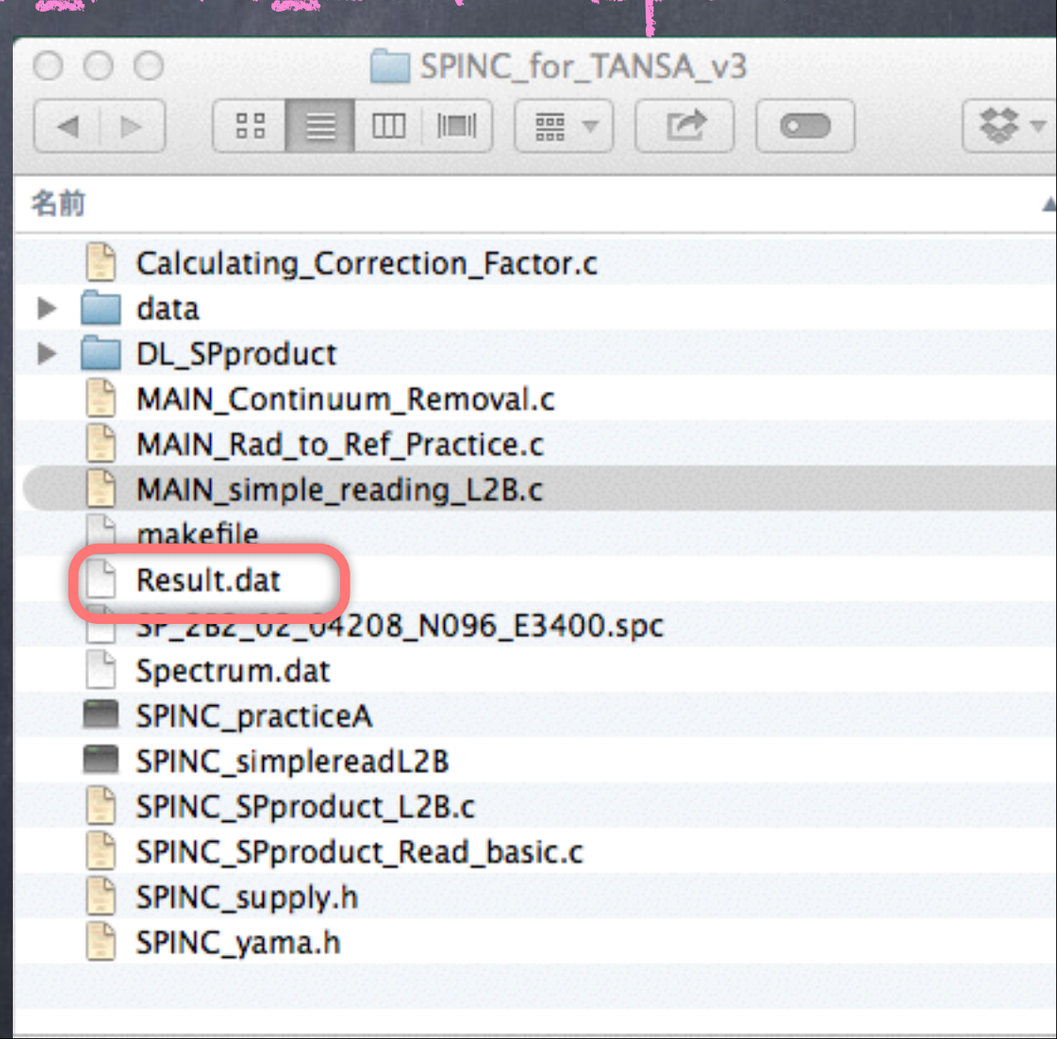
（使い方）spinc_practiceA［spcファイル］［側線番号］

（例）

```
>./SPINC_practiceA SP_2B2_01_04208_N096_E3400.spc 10
```

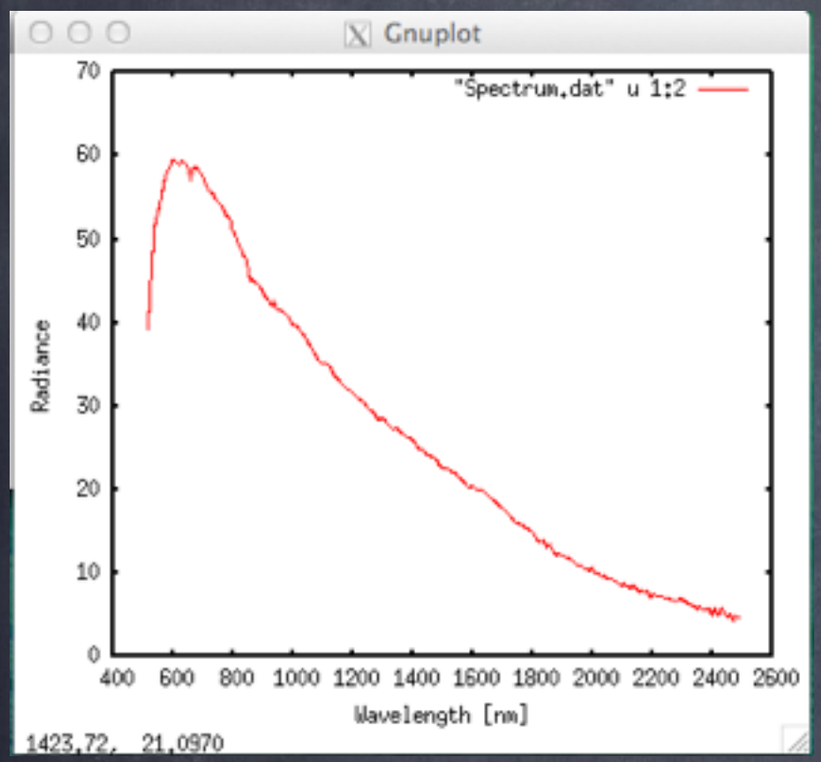
→ result.datにデータが書き出される





3.1 輝度値データから反射率データの作成

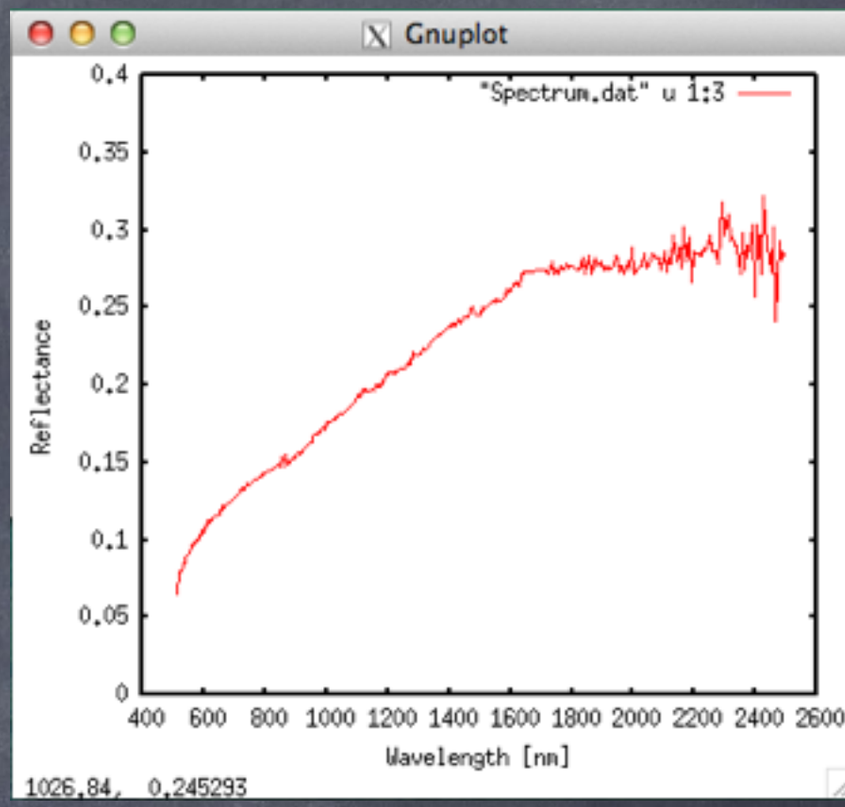
輝度値データを使って、この反射率を再現してみよう！



太陽データ

$F(n)$

$$R(n) = \frac{I(n)}{F(n)} \frac{1}{\pi}$$



3.1 輝度値データから反射率データの作成

手順３：MAIN_Rad_to_Ref_Practice.cの修正

・MAIN_Rad_to_Ref_Practice.cをエディターで開く

→ MAIN_simple_reading_L2B.cの(2)以降を書き換えたもの。

・オリジナルでは、SPプロダクトからの値をそのまま与えている→修正を加える。

3.1 輝度値データから反射率データの作成

(1) 52行目を以下に書き換える。

反射率=輝度値データ／太陽光データ

```
reflectance_data[ px ] = radiance_data[ px ]/solar_data[ px ];
```

(2) 44行目の出力ファイル名も変更

・"Result_A.dat" → "Result_A.dat"
(名前はなんでもよい)

・最初の結果と一致するか比較するため。

```
35  /*****************************************************************************/
36  // (2)  Calulating Reflectance from Radiance data.
37  /*****************************************************************************/
38  double wavelength_data[ BAND_NUM ];
39  double radiance_data[ BAND_NUM ];
40  double solar_data[ BAND_NUM ];
41  double reflectance_data[ BAND_NUM ];
42
43  INPUT_FILE_NAME = fopen("data/Solarflux_for_SP.txt","r");
44  OUTPUT_FILE_NAME = fopen("Result_A.dat","w");
45
46  for(px=0; px<BAND_NUM; px+=1){
47    //(1) Reading Solar Data
48    fscanf(INPUT_FILE_NAME, "%lf %lf %*lf", &wavelength_data[ px ], &solar_data[ px ]);
49    //(2) Giving Radiance
50    radiance_data[ px ] = (double)(AG.data[input_line].radiance[px]* AG.file_head.SCALING_FACT
51    //(3) Here go for it!
52    reflectance_data[ px ] = radiance_data[ px ]/solar_data[ px ];
53    //(4) Output
54    fprintf(OUTPUT_FILE_NAME, "%.1f %.4f\n", wavelength_data[ px ], reflectance_data[ px ]);
55  }
56
57  fclose( INPUT_FILE_NAME );
58  fclose( OUTPUT_FILE_NAME );
```

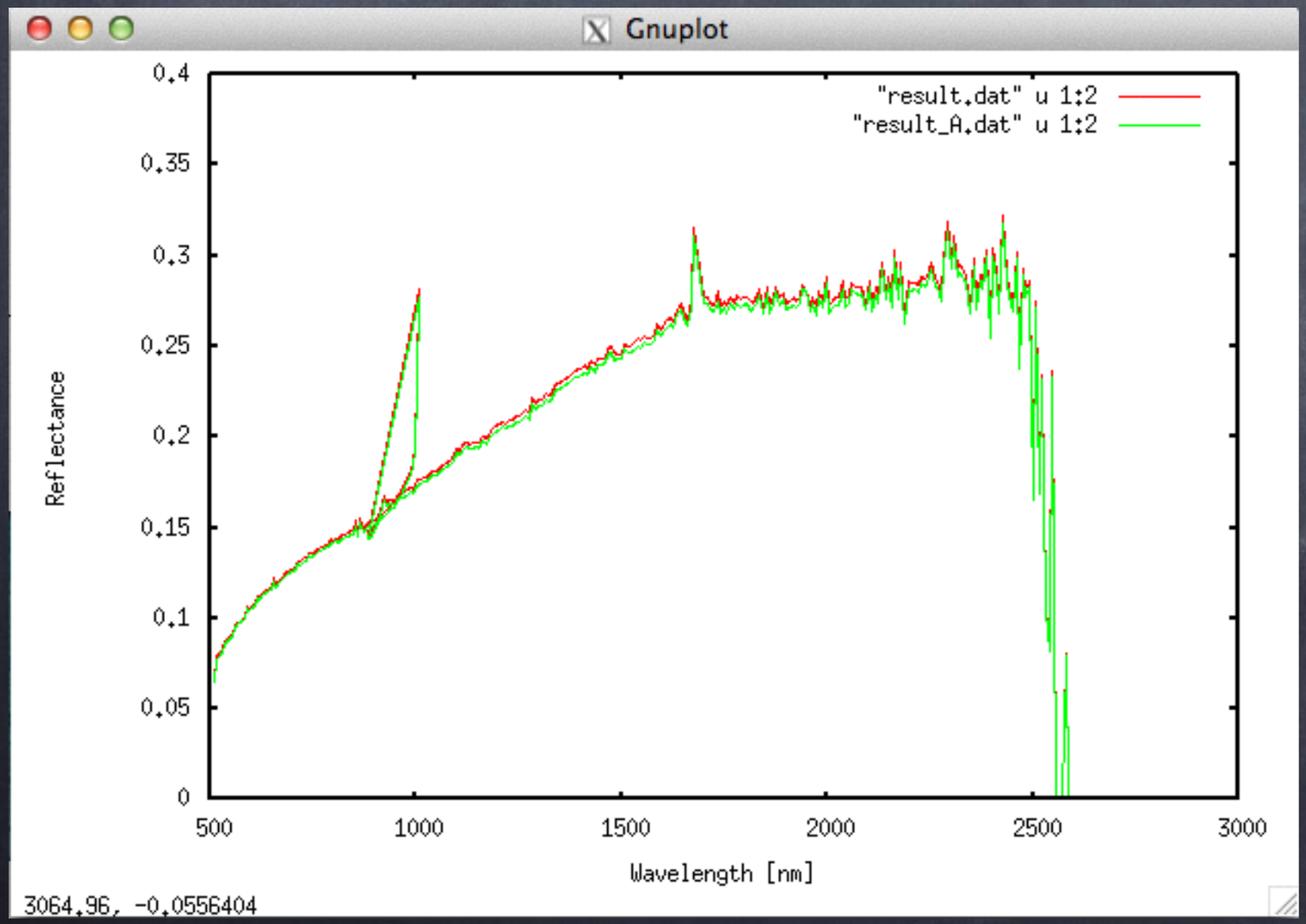
3.1 輝度値データから反射率データの作成

手順４：再コンパイル＆出力結果を検証

(1) make radtoref

(2) ./SPINC_practiceA [SPデータ] [測点番号]

(3) gnuplot> plot "result.dat" u 1:2 w l, "Result_A.dat" u 1:2 w l



一致しない!?

3.1 輝度値データから反射率データの作成

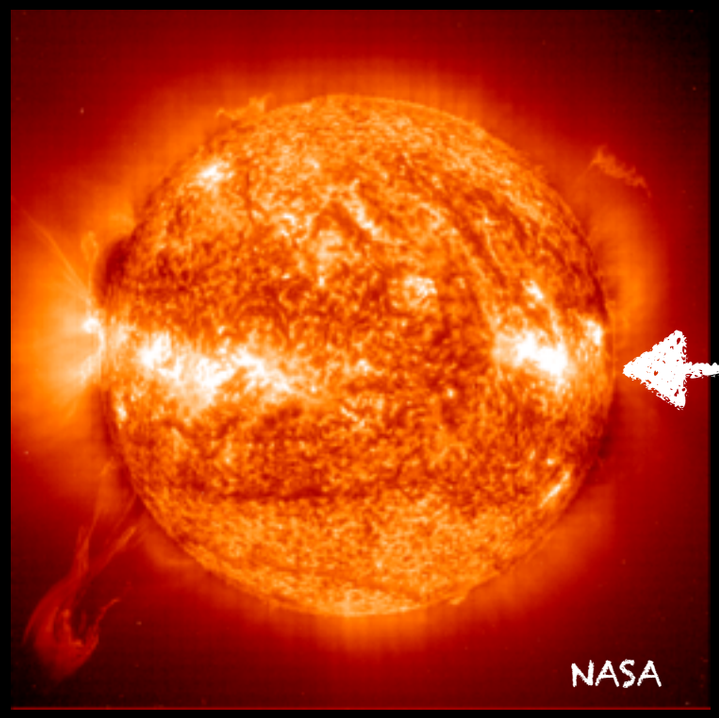
なぜ一致しない！？？？

```
>diff result.dat Result_A.dat
```

→ 数字が全て異なる！

（太陽光データはSPプロダクトで使用したものと全く同じデータ）

☆ 太陽−月距離の違いが補正されていない！



太陽−月距離は時間変動

太陽光データは1AU (= 149,597,870 km)での値

3.2（演習b）太陽距離を補正した反射率の作成

3.2 太陽距離を補正した反射率の作成

・作業：太陽光のデータに対して月ー太陽距離の違いを補正する用に変更

$$F'(n) = F(n) \cdot \left(\frac{[\mathrm{AU}]}{a} \right)^2$$

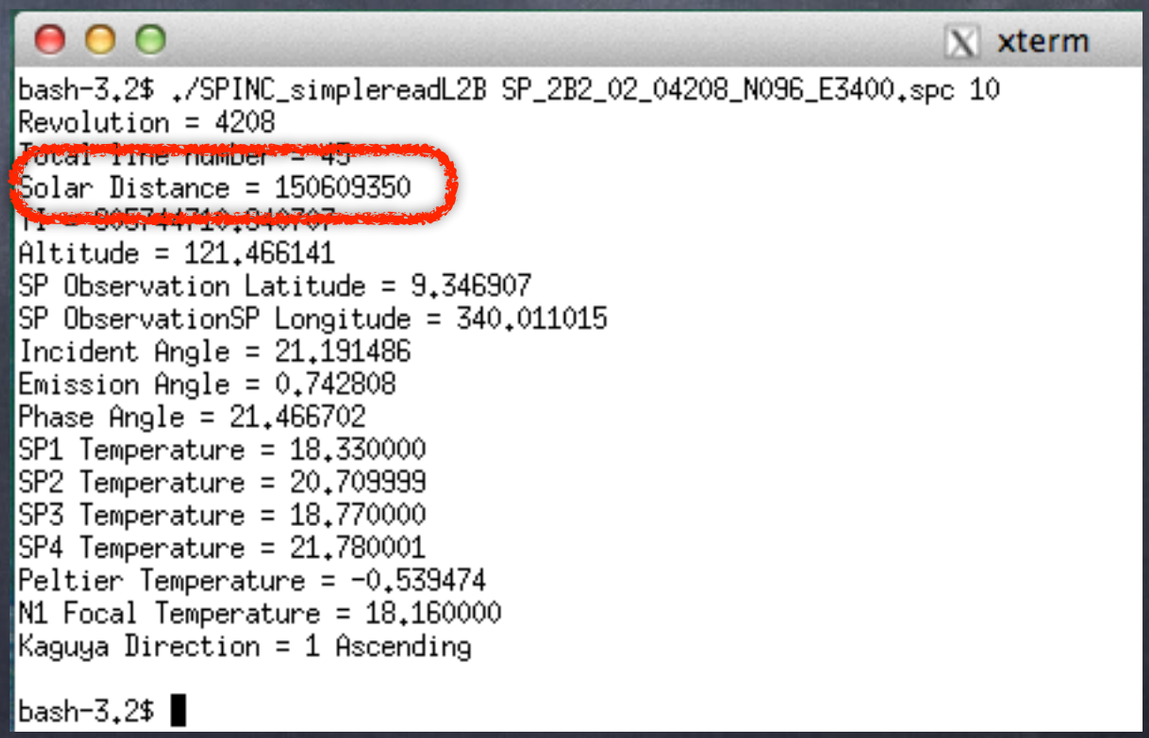
F'(n)：a[AU]での太陽光フラックス

a：太陽-月距離

手順５：MAIN_Rad_to_Ref_Practice.cにおいて上の式を反映させる

太陽-月間距離を知るには？

→ SPInC_simpleL2B



3.2 太陽距離を補正した反射率の作成

$$F'(n) = F(n) \cdot \left(\frac{[\mathrm{AU}]}{a} \right)^2$$

SP in Cのライブラリを使用

(1)太陽-月距離［km］→ AG.file_head.MOON_SUN_DISTANCE

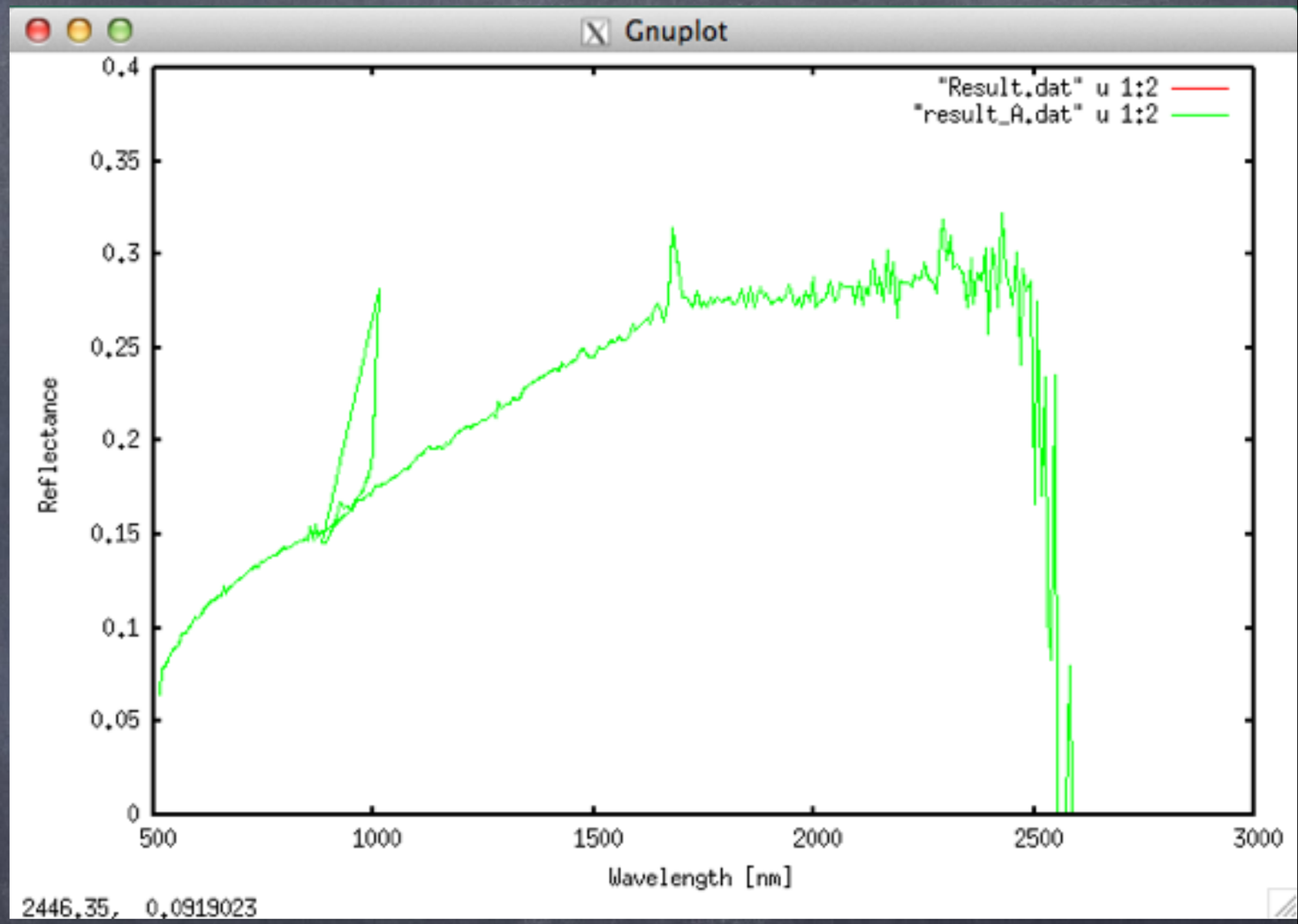
(2) 1AUの値（149,597,870 km）→ AU_UNIT

・solar_data[]に以下を。

(AU_UNIT/AG.file_head.MOON_SUN_DISTANCE)*(AU_UNIT/AG.file_head.MOON_SUN_DISTANCE)

3.2 太陽距離を補正した反射率の作成

手順６：再びmake、実行、評価



一致すれば課題修了！

・さらにdiffコマンド

```
>diff result_A.dat result.dat
```

→ 何も表示されなければOK!

(注) 太陽フラックスデータはSPプロダクトにより若干異なる。

3.3 （演習c)コンティニューム（連続光成分）処理

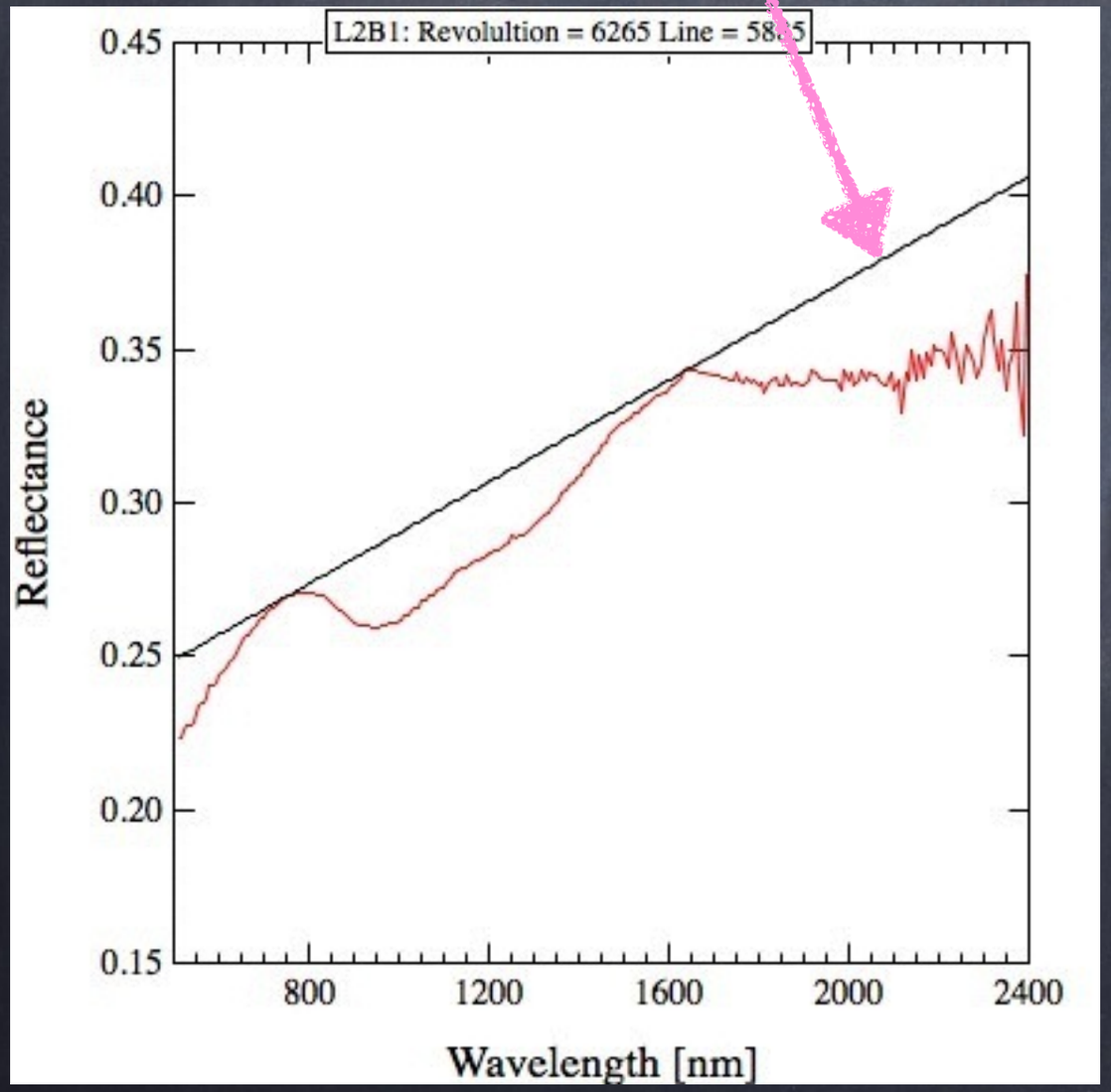
3.3 コンティニューム（連続光成分）処理

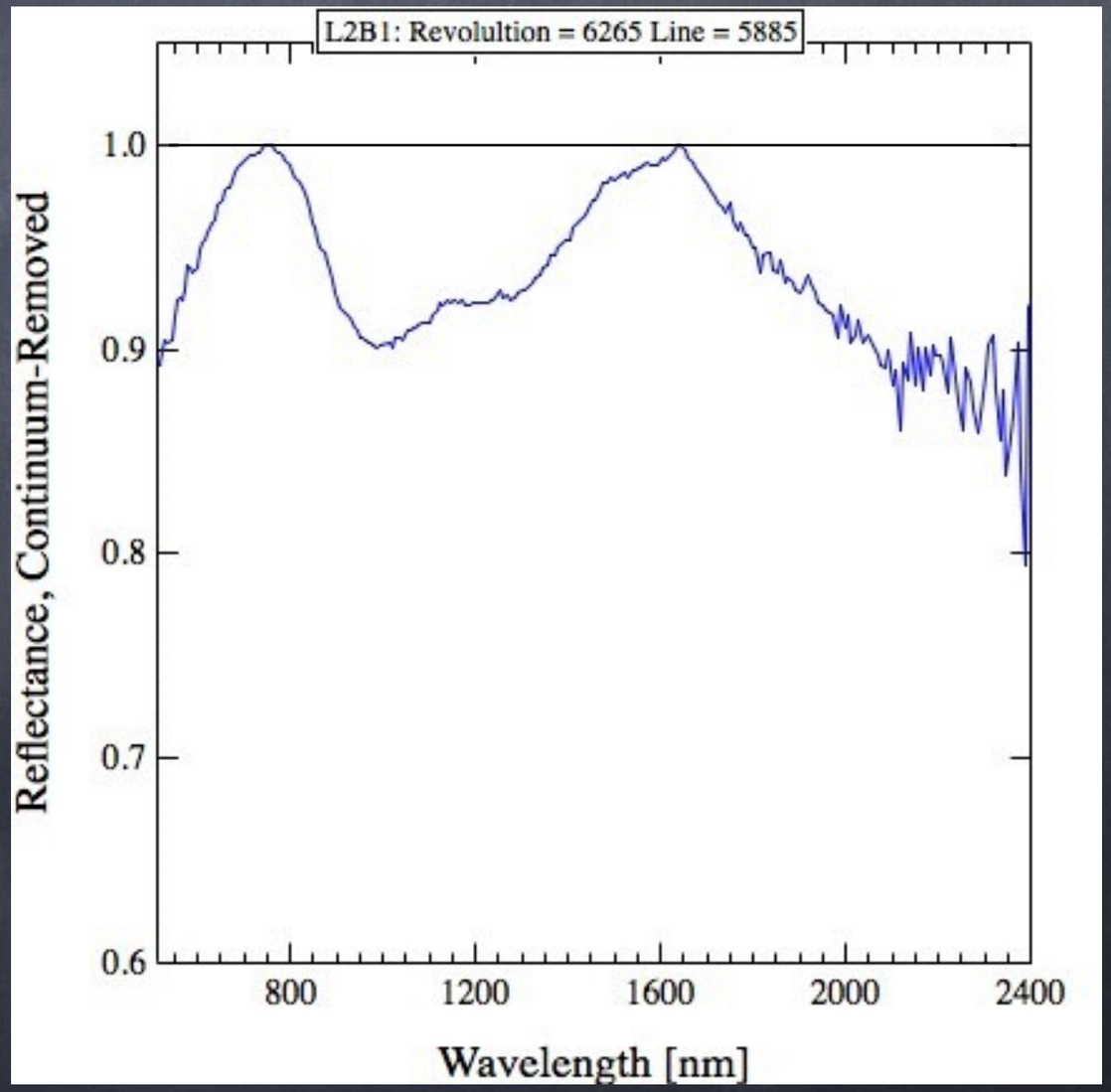
- コンティニュームで反射スペクトルデータを割る

→吸収帯を強調

コンティニューム（直線）

Ref = 8.29843x10^{-5}*wave + 0.207229

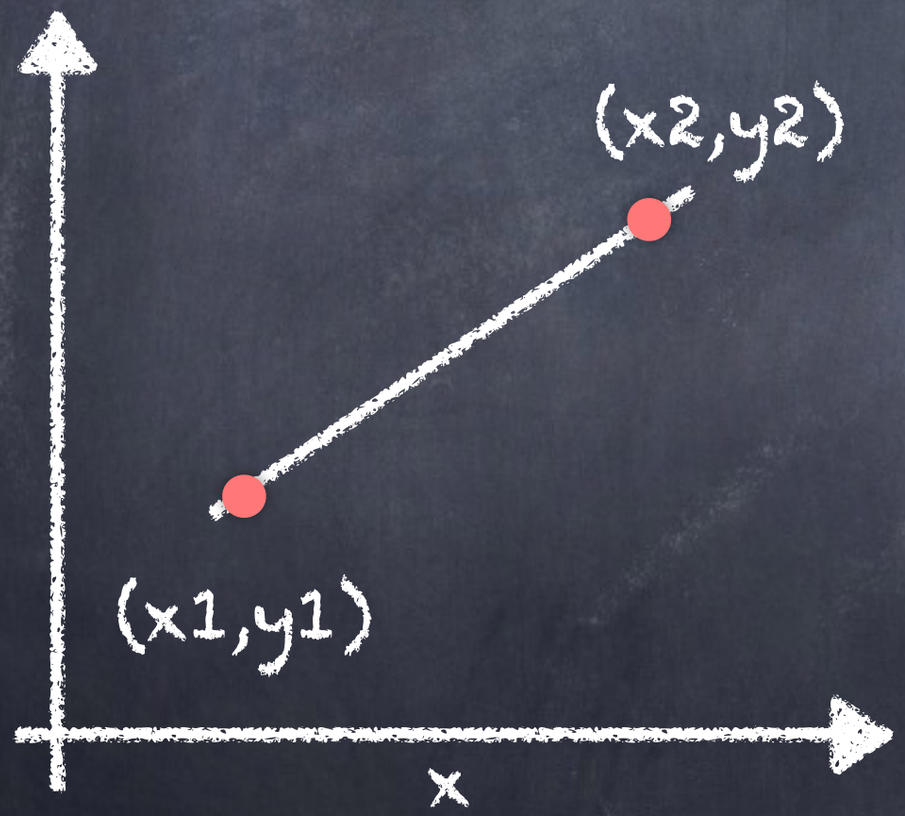




作業：

(1)反射率データからコンティニュームを決定

(2)反射率をコンティニュームで割る処理



$$y=\frac{y2-y1}{x2-x1}(x-x1)+y1$$

3.3 コンティニューム（連続光成分）処理

手順1：MAIN_Continuum_Removal.cを開く

・MAIN_Continuum_Removal.c

が対象のmainファイル

3.3 コンティニューム（連続光成分）処理

手順2：処理の書き足し

反射率データ(reflectance_data)とpx1, px2を使って、コンティニューム(linear_continuum_data)とコンティニュームで割った反射率(ContinuumRemoved_reflectance)を計算

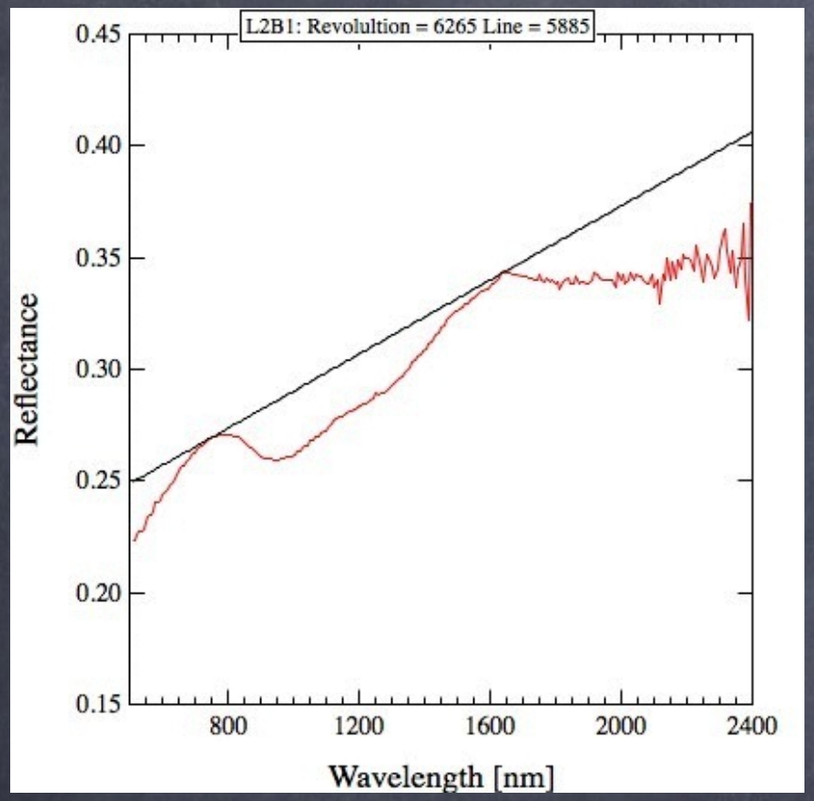
```
35  __/*****************************************************************************/↓
36  __// (2) Calculating Continuum Removed Reflectance.↓
37  __/*****************************************************************************/↓
38  __double wavelength_data[ BAND_NUM ];↓
39  __double reflectance_data[ BAND_NUM ];↓
40  __double linear_continuum_data[ BAND_NUM ];↓
41  __double ContinuumRemoved_reflectance[ BAND_NUM ];↓
42  __double slope, intercept;↓
43  __int px1 = atoi(argv[3]);↓
44  __int px2 = atoi(argv[4]);↓
45  __↓
46  __// Giving Wavelength & reflectance↓
47  __for(px=0; px<BAND_NUM; px+=1){↓
48  ____wavelength_data[ px ] = (double)(AG.data[input_line].wavelength[px] * AG.file_head.SCALING_FACTOR[SCALING_NUM_WAVELENGTH] +AG.file_head.OFFSET[SCALING_NUM_WAVELENGTH]);↓
49  ____reflectance_data[ px ] = (double)(AG.data[input_line].reflectance[px] * AG.file_head.SCALING_FACTOR[SCALING_NUM_REFLECTANCE]+ AG.file_head.OFFSET[SCALING_NUM_REFLECTANCE]);↓
50  __}↓
51  __// Continuum-Removed Reflectance↓
52  __// Here Go for It!↓
53  __↓
54  __↓
55  __// Output↓
56  __OUTPUT_FILE_NAME = fopen("result.dat","w");↓
57  __for(px=0; px<BAND_NUM; px+=1){↓
58  ____fprintf(OUTPUT_FILE_NAME, "%.1f %.4f %.4f %.4f\n", wavelength_data[ px ], reflectance_data[ px ], linear_continuum_data[ px ], ContinuumRemoved_reflectance[ px ]);↓
59  __}↓
60  __fclose( OUTPUT_FILE_NAME );↓
61  __↓
62  __return 0; ↓
63  }↓
64  //EOF↓
```

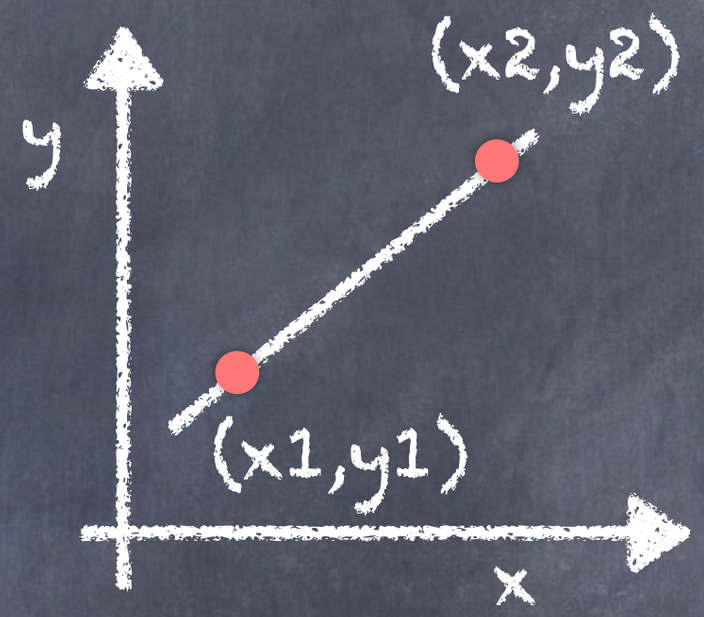
(1) px1とpx２は引数として与える。

☆ここに処理を追加

手順3：コンティニュームの計算

px1とpx2を使って、linear_continuum_dataを計算





$$y=\frac{y2-y1}{x2-x1}(x-x1)+y1$$

y → linear_continuum_data

x → wavelength_data

y1 = reflectance_data[px1]
y2 = reflectance_data[px2]
x1 = wavelength_data[px1]
x2 = wavelength_data[px2]

3.3 コンティニューム（連続光成分）処理

求め方は自由。

（例）

```
slope = (reflectance_data[ px1 ] - reflectance_data[ px2 ])/(wavelength_data[ px1 ] - wavelength_data[ px2 ]);
intercept = - slope * wavelength_data[ px1 ] + reflectance_data[ px1 ];
```

手順４：反射率をコンティニュームで割る処理

ContinuumRemoved_reflectance
reflectance_data
linear_continuum_data

を使用。

手順５：make、実行、評価

(1)make continuum

実行ファイルSPINC_practiceBが生成

(2) ./SPINC_practiceB［SPプロダクト］［測点番号］［px1］［px2］

px1、px2は対応する波長の素子番号を手動で与える（一番最初はpx=0）。

例）752.8nm → px1=40、1555.5nm → px2＝168

→ result.datにデータが書き出される

result.datの中身

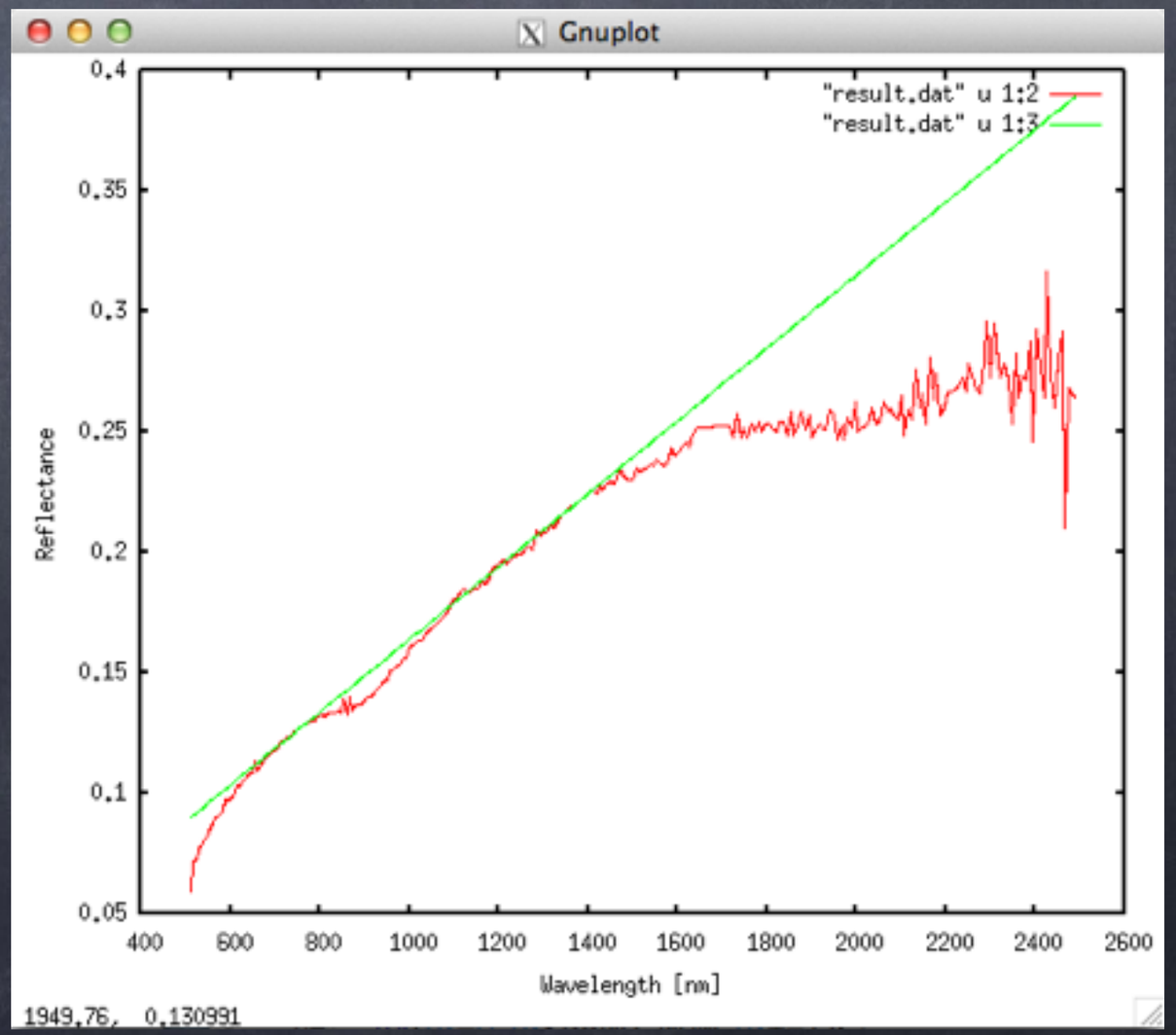
(1)波長 (2)反射率 (3)コンティニューム (4)コンティニューム処理後の反射率

手順６：グラフで確認

反射率データとコンティニュームの関係をチェック

→ 入力 px を変えながら調整

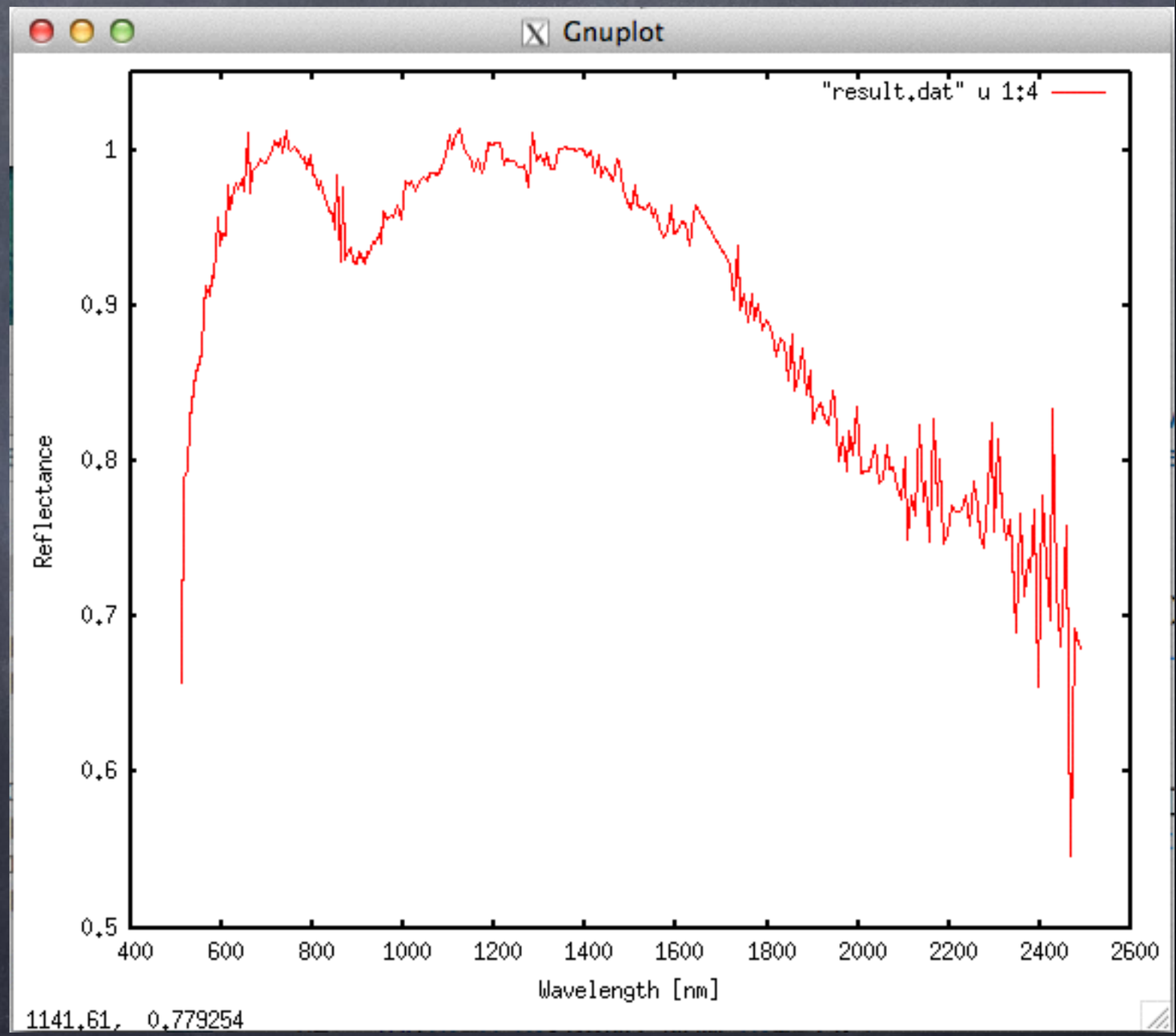
```
gnuplot> p "result.dat" u 1:2 w l, "result.dat" u 1:3 w l
```



手順７：最後にコンティニューム処理後スペクトルをチェック

```
gnuplot> p "result.dat" u 1:4 w l
```

コンティニューム処理ができていればOK！



（おまけ）

SP成果研究における処理ではpx1,px2は自動判別

余力がある人はやってみよう

コンティニューム自動決定アルゴリズム

①仮のコンティニュームに対して各pxにおいて大小関係を調べる

②反射率がコンティニュームより高いpxで、コンティニュームを再決定

③これを収束するまで繰り返す