

階層的地球流体スペクトルモデル集 SPMODEL

佐々木洋平 (京大・数学), 竹広真一 (京大・数理研)

with

地球流体電脳倶楽部
SPMODEL プロジェクト
dcmode1 プロジェクト
davis プロジェクト

2019年3月29日

本日のお題

数式を書くようにプログラミング!!

しかも

スペクトル法で!!

さらに

簡単に計算データの出力 & 描画を!!

例題: 一次元移流方程式

$$\frac{\partial \zeta}{\partial t} = -c \frac{\partial \zeta}{\partial x}$$

- ▶ 一次元周期境界条件: $\zeta(x + L, t) = \zeta(x, t)$
- ▶ 初期条件: $\zeta(x, t) = \zeta_0(x)$
- ▶ ちなみに解析解は $\zeta(x, t) = \zeta_0(x - ct)$
- ▶ サンプルプログラムその 1: `advect.f90`

まずは使ってみよう!

▶ コンパイル

```
% spmfrt -o advect.out advect.f90
```

⇒ `advect.out` が生成される

▶ 実行

```
% ./advect.out
```

⇒ `advect.nc` が出力される

▶ 結果表示

```
% gpllist advect.nc  
% gpview --anim t advect.nc@zeta  
% gpview --range 0:1.5e3 --anim t --Gaw advect.nc@zeta
```

用いているテクニックとライブラリ

- ▶ Fortran(≥ 90) の配列演算記法: 配列を返す関数

```
DO I=1,IM  
  A(I) = B(I) + C(I)  
ENDDO
```

⇒

```
A = B + C
```

```
DO I=1,IM  
  DATA(I) = EXP(-X(I)**2)  
ENDDO
```

⇒

```
DATA = EXP(-X**2)
```

- ▶ `spmodel library` (spml): 正/逆変換, 空間微分などを提供
- ▶ 結果出力: `gtool5 library`
- ▶ 結果表示: Dennou Ruby 製品群 (GPhys, `gpview` ...)

スペクトル法による数値計算

1. 境界条件を満たす関数系で展開

$$\zeta(x_j, t) = \sum_{k=-K}^K \tilde{\zeta}_k(t) e^{i \frac{2\pi k x_j}{L}}, \quad \tilde{\zeta}_k(t) = \frac{1}{J} \sum_{j=0}^{J-1} \zeta(x_j, t) e^{-i \frac{2\pi k x_j}{L}}$$

2. 方程式に代入すると常微分方程式になる

$$\frac{d\zeta_k}{dt} = -ci \frac{2\pi k}{L} \tilde{\zeta}_k$$

3. 適当な初期値から離散化して時間積分

$$\tilde{\zeta}_k(t + \Delta t) = \tilde{\zeta}_k(t) - ci \frac{2\pi k}{L} \tilde{\zeta}_k \Delta t$$

4. 実空間の変数に戻す

$$\zeta(x_j, t) = \sum_{k=-K}^K \tilde{\zeta}_k(t) e^{i \frac{2\pi k x_j}{L}},$$

数値計算の手順

1. 使用するモジュールの宣言

```
use ae_module
```

2. 変数宣言

```
real(kind(1.0D0)) :: g_Zeta(0:im-1) ! 格子点データ  
real(kind(1.0D0)) :: e_Zeta(-km:km) ! スペクトルデータ
```

3. スペクトル変換の初期化

```
call ae_Initial(im,km,xmin,xmax)
```

4. 初期値を与える

```
g_Zeta = U1*sech((g_X-X1)/sqrt(12/u1))**2 &  
        & + U2*sech((g_X-X2)/sqrt(12/u2))**2
```

5. スペクトルデータに変換

```
e_Zeta = e_g(g_Zeta)
```

6. スペクトルデータを時間積分

```
e_Zeta = e_Zeta + dt*( - c *e_Dx_e(e_Zeta))
```

7. 実空間データに戻して出力

```
g_Zeta = g_e(e_Zeta)
```

スペクトル計算のためのサブルーチン・関数を提供

▶ 初期化

```
subroutine ae_Initial(im, km, xmin, xmax)
```

▶ スペクトル正変換/逆変換

```
e_g(g_Data) ! 格子点 → スペクトルデータ
```

```
g_e(e_Data) ! スペクトルデータ → 格子点データ
```

▶ 微分計算

```
e_Dx_e(e_Data) ! X 微分
```

▶ 積分・平均演算

```
Int_g(g_Data) ! 全領域積分
```

```
Avr_g(g_Data) ! 全領域平均
```

▶ その中身は?

- ISPACK(石岡, 2015) を Fortran90 の関数でくるんだもの
- FFT 用の変換テーブル, 領域の大きさ → 微分計算に使用
- 座標変数の設定: `g_X`, `g_X_weight`
- 微分演算 → 波数のかけ算: $e_{Dx_e}(k) = -2\pi i \cdot (k/L) \cdot e_{Data}(-k)$

▶ 御利益: 数式の如くプログラムが書ける

$$f = \frac{\partial \zeta}{\partial x} \quad \Rightarrow \quad e_f = e_Dx_e(e_Zeta)$$

$$f = \frac{\partial^2 \zeta}{\partial x^2} \quad \Rightarrow \quad e_f = e_Dx_e(e_Dx_e(e_Zeta))$$

spml プログラミング書法

- ▶ 先頭の文字で変数の種類を区別:
- ▶ 実空間格子点データ: g_ で始める
- ▶ スペクトルデータ: e_ で始める
- ▶ spml の関数名の命名規則:
(出力の型)_(機能)_(入力の型)
- ▶ 縮約の如くプログラムを書けば型を間違えにくい

```
g_Data = g_e(e_Dx_e(e_g(g_Data1)))
```

練習問題 (1)

- ▶ 移流方程式に拡散項を付け加えてみよう

$$\frac{\partial \zeta}{\partial t} = -c \frac{\partial \zeta}{\partial x} + \frac{\partial^2 \zeta}{\partial x^2}$$

- ▶ Euler scheme だとこんな感じ..?

```
e_Zeta = e_Zeta &
      & + dt * ( - c * e_Dx_e(e_Zeta) &
      & + e_Dx_e(e_Dx_e(e_Zeta)) )
```

非線型項の取り扱い (変換法)

- ▶ 移流項が非線型 $\zeta \frac{\partial \zeta}{\partial x}$ である場合には
 スペクトル変数を一度実空間に戻してから積を取る
- ▶ spml の関数を用いると一行で書ける

```
g_e(e_Zeta) * g_e(e_Dx_e(e_Zeta))
```

練習問題 (2)

- ▶ `advect_lf.f90` を元に KdV 方程式のプログラムを書いてみよう

$$\frac{\partial \zeta}{\partial t} = -\zeta \frac{\partial \zeta}{\partial x} - \frac{\partial^3 \zeta}{\partial x^3}$$

- ▶ Leap frog scheme だとこんな感じ..?

```
e_Zeta = e_Zeta0 &
      & + 2 * dt * ( - e_g(g_e(e_Zeta1) * g_e(e_Dx_e(e_Zeta1))) &
      & - e_Dx_e(e_Dx_e(e_Dx_e(e_Zeta1))) ) &
e_Zeta0 = e_Zeta1 ; e_Zeta1 = e_Zeta
```

- ▶ 切断波数の取り方に注意
 - 変数の積から高波数成分が生成される
⇒ $J \sim 2K+1$ の格子点数では十分に表現できない
 - 2次の非線型項なら格子点数は $J \geq 3K+1$ に増やしておく

SPMODEL プログラミング

1. 領域と境界条件に適合したモジュールを選択
2. 支配方程式を形式的にスペクトル変換

$$\frac{\partial \tilde{\zeta}_m}{\partial t} = - \left[\zeta \frac{\partial \tilde{\zeta}}{\partial x} \right] - \left[\frac{\partial^3 \tilde{\zeta}_m}{\partial x^3} \right]$$

3. 適当なスキームで時間に関して差分化

$$\tilde{\zeta}_m^{\tau+1} = \tilde{\zeta}_m^\tau + \Delta t \times \left\{ - \left[\zeta \frac{\partial \tilde{\zeta}}{\partial x} \right] - \left[\frac{\partial^3 \tilde{\zeta}_m}{\partial x^3} \right] \right\}$$

4. 2,3 に従ってプログラムを書く

```
e_Zeta = e_Zeta &  
    & + dt * ( - e_g(g_e(e_Zeta) * g_e(e_Dx_e(e_Zeta))) &  
    & - e_Dx_e(e_Dx_e(e_Dx_e(e_Zeta))) )
```

もっとおすすめの方法

- ▶ サンプル・デモプログラムを元に改造
 - 計算したい領域形状に即したモジュールのサンプル
 - (たいてい) 拡散方程式用のプログラムが用意されている
 - 必要に応じて変数を追加, 演算を変更
 - 時間積分部分を改造
- ▶ サンプルの置き場所:
 - Web: <http://www.gfd-dennou.org/library/spmodel/>

一般的注意

- ▶ ライブラリ・サンプルプログラムは無保証
 - 必ず自分でテストしてみる
 - できれば解析解と比較してみる
- ▶ マニュアルを見よう
 - 気分でプログラムを書くのは危険
 - マニュアルで確認しつつ書くこと
spml のマニュアルを見てみよう!
モジュールと計算領域・境界条件に注目!
 - local: <http://localhost/spmodel/reference/>
 - Web: <http://www.gfd-dennou.org/library/spmodel/>
- ▶ できればソースも見てみよう

出力操作: gtool5 library

▶ モジュールの宣言

```
use gtool_history
```

▶ 出力ファイルの作成、次元の定義

```
call HistoryCreate(...
```

▶ 変数の定義

```
call HistoryAddVariable(...
```

▶ 出力

```
call HistoryPut(...
```

▶ 終了

```
call HistoryClose
```

練習問題 (3): 出力変数の追加

- ▶ `advect.f90` を使って $\frac{\partial \zeta}{\partial x}$ を x, t の 2次元データとして出力してみる
- ▶ 変数定義の追加

```
call HistoryAddVariable(                                &  
    varname='dzetadx', dims=('/x','t/'),               &  
    longname='derivative of displacement',&  
    units='1', xtype='double')
```

- ▶ 出力ルーチンの追加

```
call HistoryPut('dzetadx',g_e(e_Dx_e(e_Zeta)))
```

練習問題 (4): 出力変数の追加

- ▶ $\int_0^L \zeta^2 dx$ を t の 1次元データとして出力してみる
 - 変数定義の追加 (次元に注意)

```
call HistoryAddVariable(                                &  
    varname='z2int', dims=('/t/'),                      &  
    longname='Integral of square of zeta', &  
    units='1', xtype='double')
```

- 出力ルーチンの追加

```
call HistoryPut('z2int',Int_g(g_e(e_Zeta)*g_e(e_Zeta)))
```

- `gpprint` コマンドで数字を出力してみる

```
% gpprint advect.nc@z2int
```

例題: 2次元拡散方程式

- ▶ 2次元周期境界条件の下で解いてみる

$$\frac{\partial \zeta}{\partial t} = \nu \left(\frac{\partial^2 \zeta}{\partial x^2} + \frac{\partial^2 \zeta}{\partial y^2} \right)$$

- ▶ サンプルプログラム: `diffuse_2d.f90`

コンパイルして実行してみよう
ソースファイルを眺めてみよう

gpview の便利なオプション

▶ 範囲指定

```
% gpview --range 0:1 --anim t diffuse_2d.nc@zeta
```

▶ データ切出し

```
% gpview --anim t diffuse_2d.nc@zeta,x=0.5  
% gpview diffuse_2d.nc@zeta,x=0.5,y=0.5
```

▶ 平均操作

```
% gpview --mean x --anim t diffuse_2d.nc@zeta
```

▶ その他のオプション

```
% gpview --help  
% man gpview
```

その他の例題

- ▶ 2次元ベータ面: モドン
- ▶ 2次元チャネル領域: 熱対流問題
- ▶ 2次元球面領域: ロスビー波
- ...
- ▶ Web を見てみよう
 - Web:
 - <http://www.nagare.or.jp/mm/2006/spmodel/>
 - <http://www.gfd-dennou.org/library/spmodel/>

練習問題 (5)

- ▶ 移流方程式に変えてみよう

$$\frac{\partial \zeta}{\partial t} = -c_x \frac{\partial \zeta}{\partial x} - c_y \frac{\partial \zeta}{\partial y}$$

- ▶ Euler scheme だとこんな感じ?

```
ee_Zeta = ee_Zeta &  
    & + dt * ( - cx * ee_Dx_ee(ee_Zeta) &  
    &          - cy * ee_Dy_ee(ee_Zeta) )
```

- ▶ Leap frog scheme の方が安定性がよろしい

```
ee_Zeta = ee_Zeta0 &  
    & + dt * ( - cx * ee_Dx_ee(ee_Zeta1) &  
    &          - cy * ee_Dy_ee(ee_Zeta1) ) &  
ee_Zeta0 = ee_Zeta1; ee_Zeta1=ee_Zeta2
```

練習問題 (6)

- ▶ 線型 β 面順圧方程式に変えてみよう

$$\frac{\partial \zeta}{\partial t} = -\beta \frac{\partial \psi}{\partial x}, \quad \nabla^2 \psi = \zeta$$

- ▶ Euler scheme だとこんな感じ?

```
ee_Zeta = ee_Zeta + dt * ( - beta*ee_Dx_ee(ee_Psi) )  
ee_Psi = ee_LaplaInv_ee(ee_Zeta)
```

- ▶ (余力があれば) 非線型ベータ面順圧方程式に挑戦

$$\frac{\partial \zeta}{\partial t} = -J(\psi, \zeta) - \beta \frac{\partial \psi}{\partial x}, \quad \nabla^2 \psi = \zeta$$