# 階層的地球流体スペクトルモデル集
# SPMODEL

竹広真一（京大数理研）佐々木洋平（北大理）
with
地球流体電脳倶楽部
SPMODEL プロジェクト
dcmodel プロジェクト
Davis プロジェクト

2010年3月9日

# 本日のお題

**数式を書くようにプログラミングする**

しかも

スペクトル法の計算で?!

＋

**計算データの出力とお絵かき**

# 例題：1次元移流方程式

- 1次元周期境界条件の下で解いてみる

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

- サンプルプログラム：advect.f90, advect_lf.f90

- 解析解
  初期条件 $\zeta(x, t=0) = \zeta_0(x)$ に対して

$$\zeta(x, t) = \zeta_0(x - ct)$$

# まずは使ってみよう!

- コンパイル
  $ spmfrt -o advect.out advect.f90
  → advect.out ができる
- 実行
  $ ./advect.out → advect.nc ができる
- 結果表示
  $ gplist advect.nc (変数のリスト)
  $ gpview --anim t advect.nc@zeta (アニメ)
  $ gpview --range 0:1500 --anim t --Gaw advect.nc@zeta
  $ gave advect.nc

# 用いているテクニックとライブラリ

- Fortran90：配列計算機能

```
DO I=0,IM-1
   A(I) = B(I)+C(I)              →   A=B+C
ENDDO
DO I=0,IM-1
   DATA(I) = EXP(-X(I)**2)   →   DATA = EXP(-X**2)
ENDDO
```

- Fortran90：配列を返す関数を作れる

  →spmodel library (spml) ：正/逆変換, 空間微分など

- 結果出力： gtool5(gt4f90io)ライブラリ
- 結果表示： Dennou-Ruby 製品 (gpview, gave など)

# スペクトル法による数値計算(離散化)

1. 境界条件を満たす関数系で展開

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

2. 方程式に代入すると常微分方程式になる

$$\frac{d\tilde{\zeta}_k}{dt}=-\frac{2\pi ik}{L}c\tilde{\zeta}_k$$

3. 適当な初期値から離散化して時間積分

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

4. 実空間の変数に戻す

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

# 数値計算の手順

1. 使用するモジュールの宣言
```
use ae_module
```
2. 変数を宣言
```
real(8) :: g_Zeta(0:im-1)     ！格子データ
real(8) :: 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 + ...
```
5. スペクトルデータへ変換
```
e_Zeta = e_g(g_Zeta)
```
6. スペクトルで時間積分
```
e_Zeta = e_Zeta + dt*( -c*e_Dx_e(e_Zeta) )
```
7. 実空間データへ戻す（出力時）
```
g_Zeta = g_e(e_Zeta)
```

# spml/ae_module

- 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)
  - スペクトル正逆変換

    e_g(g_Data)　！格子データ→スペクトルデータ
    g_e(e_Data)　！スペクトルデータ→格子データ
  - 微分計算

    e_Dx_e(e_Data) ！x微分
  - 積分・平均計算

    Int_g(g_Data)　　！全領域積分
    Avr_g(g_Data)　 ！全領域平均

# spml/ae_module

- その中身は…
  - ISPACKをFortran90の関数でくるんだもの
    FFT 用の変換テーブルを記憶
    領域の大きさを記憶→微分計算に使用
  - 座標変数の設定　(g_X, g_X_weight)
  - 微分計算→波数をかけているだけ
    e_Dx_e(k) =- 2*pi*k/L*e_Data(-k)
- ご利益：数式のごとくプログラムを書ける

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

$$f = \frac{\partial^2 \zeta}{\partial x^2} \longrightarrow$$ 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)

- 縮約のごとくプログラムを書けば型を間違えない

  g_Data2=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 スキームだとこんな感じ

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

# 非線形項の取り扱い（変換法）

- 移流項が非線形項 $\zeta\dfrac{\partial\zeta}{\partial x}$ である場合には

  →スペクトル変数をいちど実空間に戻してから積をとる（変換法）

- spml の関数を用いると一発で書ける

  e_g( 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かいてみよう $$\frac{\partial\zeta}{\partial t} = -\zeta\frac{\partial\zeta}{\partial x} - \frac{\partial^3\zeta}{\partial x^3}$$
- Leapfrog スキームだとこんな感じ

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

- 切断波数のとり方に注意
  - 変数の積→高波数成分の生成
    → J ～ 2K+1 の格子点数では十分に表せない
  - 2 次の非線形項→格子点数を J>3K+1 にふやしておく

# 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
   [1次元周期境界条件]

2. 支配方程式を形式的にスペクトル変換する

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

3. 適当なスキームで時間に関して差分化

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

4. 2と3にしたがってプログラムを書き下す

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

方程式の形そのままにプログラムできる!

# 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プログラムが用意されている
  - 必要に応じて変数を追加・変更
  - 時間積分部分を改造
- サンプルの場所
  - Desktop/Tutorial/dcmodel/spmodel/SPMODEL_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こと
  - 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！

    Desktop/Tutorial/dcmodel/spmodel/SPMODEL_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(gt4f90io)

- モジュールの宣言
  use gtool_history　(use gt4_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)

- gtool library のサブルーチンを使って出力の練習
  - 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( &                    ! 変数定義
    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)

- gtool library のサブルーチンを使って出力の練習

  - $\int_0^L \zeta^2 dx$ 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( &                                    ! 変数定義
     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)))
```
  - 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

# 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

  - ローカル：
    Desktop/Tutorial/dcmodel/spmodel/NagareMultimedia
    Desktop/Tutorial/dcmodel/spmodel/SPMODEL_WEB
  - ネット上：
    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 スキームだとこんな感じ

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

- Leap frog スキームの方が安定性がよろしい

```
ee_Zeta2 = 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）

- 線形 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 スキームだとこんな感じ

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

- （余力があれば）非線形ベータ面方程式に挑戦

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