水面と伝わる波：津波の数値シミュレーション

1．重力と水面を伝わる波

ここでは津波や川面の波のような水面を伝わる波をシミュレーションすることを考える。図のように水面に盛り上がりができたとする。このとき、盛り上がった部分の水中では、圧力が周りより高い。この圧力差によってこの部分の水が横に動かされるため、盛り上がりの部分が移動していく。このように、密度境界に働く重力が原因となって伝播する波を重力波とよぶ。

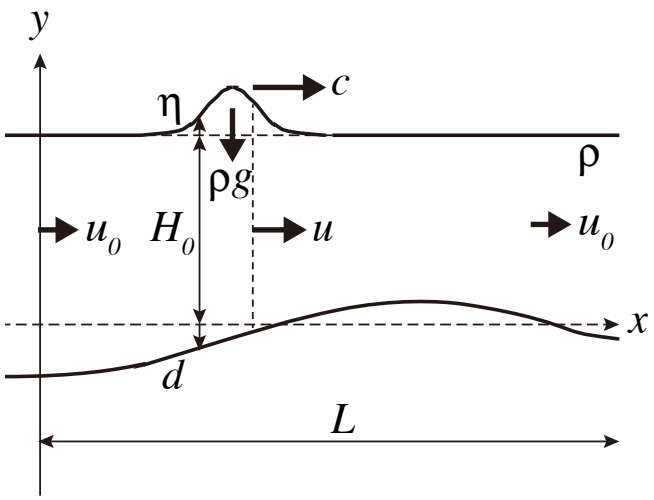

図 1：津波のモデル

ここでは長さ L、平均水深 H_0 の領域を考える。さらに、川のように水全体が一様な速度 u_0 で右へ流れているとする (この場合には、水深が一定 $d=0$ であるとする)。水が水平移動する流速を u とする。

1.1 浅水方程式

今、水平スケールに比べて垂直スケールの小さい流体を考える。この場合、流体の速度および圧力について次のように考えることができる。

(1) 流体の水平速度は深さに依存せず一定である。

(2) 圧力は静水圧であると考えることができる。

このような近似は浅水近似と呼ばれる。(2)により、盛り上がりによる圧力の増分が

$$P = \rho g \eta \tag{0}$$

であることに注意すると、粘性のない流体の連続の式(質量保存の式)および運動方程式は、浅水近似において

$$\frac{\partial \eta}{\partial t} + \nabla_h \cdot \left\{ \boldsymbol{u} \left(\eta + H_0 + d \right) \right\} = 0 \tag{1}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla_h \boldsymbol{u} = -g \nabla_h \eta \tag{2}$$

となる。ここで、ηは水面の垂直変位、H_0は平均水深、d は水深から平均水深を引いた値、$\boldsymbol{u}$ は水平速度である。なお、ナブラ演算子は水平成分のみを持ち、

$$\nabla_h = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, 0 \right)$$

である。

1次元のみを考えると、

$$\frac{\partial \eta}{\partial t} + \frac{\partial}{\partial x} \left\{ u \left(\eta + H_0 + d \right) \right\} = 0 \tag{3}$$

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + g \frac{\partial \eta}{\partial x} = 0 \tag{4}$$

となる。これらの式を解くと津波のような水面を伝播する波をシミュレートすることができる。水深一定でηがH_0に比べて小さいときには、２つの式は

$$\frac{\partial \eta}{\partial t} + H_0 \frac{\partial u}{\partial x} = 0 \tag{5}$$

$$\frac{\partial u}{\partial t} + g \frac{\partial \eta}{\partial x} = 0 \tag{6}$$

と近似することができる。(5)を t で、(6)を x で偏微分することにより、

$$\frac{\partial^2 \eta}{\partial t^2} = g H_0 \frac{\partial^2 \eta}{\partial x^2} \tag{7}$$

が得られる。この式は振幅ηが位相速度

$$c = \sqrt{g H_0} \tag{8}$$

で伝わる波を表す波動方程式となっている。

1.2 特性方程式

ここでは、この近似をせず、２つの式(3)(4)を数値的に解く方法を考える。今、

$$h = \eta + H_0 \tag{9}$$

とする。このとき、

$$\frac{\partial h}{\partial t} + \frac{\partial}{\partial x} \left\{ u \left(h + d \right) \right\} = 0 \tag{10}$$

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+g\frac{\partial h}{\partial x}=0 \tag{11}$$

が得られる。さらに変形すると、

$$\frac{\partial h}{\partial t}+u\frac{\partial h}{\partial x}+h\frac{\partial u}{\partial x}=-\frac{\partial}{\partial x}(ud) \tag{12}$$

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+g\frac{\partial h}{\partial x}=0 \tag{13}$$

となる。左辺の時間微分項以外を行列の式としてまとめると、

$$\begin{pmatrix} \frac{\partial h}{\partial t} \\ \frac{\partial u}{\partial t} \end{pmatrix}+\boldsymbol{A}\begin{pmatrix} \frac{\partial h}{\partial x} \\ \frac{\partial u}{\partial x} \end{pmatrix}=\begin{pmatrix} -\frac{\partial}{\partial x}(ud) \\ 0 \end{pmatrix} \tag{14}$$

であり、ここで行列 $\boldsymbol{A}$ は、

$$\boldsymbol{A}=\begin{pmatrix} u & h \\ g & u \end{pmatrix} \tag{15}$$

である。固有ベクトルから作られる行列を用いて対角化することを考えると、対角化するための行列 $\boldsymbol{P}$ と対角化した行列 $\boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P}$ は

$$\boldsymbol{P}=\begin{pmatrix} h & h \\ c & -c \end{pmatrix},\ \boldsymbol{P}^{-1}=\frac{1}{2}\begin{pmatrix} 1/h & 1/c \\ 1/h & -1/c \end{pmatrix},\ \boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P}=\begin{pmatrix} v & 0 \\ 0 & w \end{pmatrix} \tag{16}$$

と得られる(牛島, 2007)。ここで、

$$c=\sqrt{gh} \tag{17}$$

である。この時式(14)は、

$$\boldsymbol{P}^{-1}\begin{pmatrix} \frac{\partial h}{\partial t} \\ \frac{\partial u}{\partial t} \end{pmatrix}+\left(\boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P}\right)\boldsymbol{P}^{-1}\begin{pmatrix} \frac{\partial h}{\partial x} \\ \frac{\partial u}{\partial x} \end{pmatrix}=\boldsymbol{P}^{-1}\begin{pmatrix} -\frac{\partial}{\partial x}(ud) \\ 0 \end{pmatrix} \tag{18}$$

となる。(18)を c 倍すると、方程式(12)(13)は

$$\frac{\partial p}{\partial t}+v\frac{\partial p}{\partial x}=-\frac{c}{2h}\frac{\partial}{\partial x}(ud) \tag{19}$$

$$\frac{\partial q}{\partial t}+w\frac{\partial q}{\partial x}=-\frac{c}{2h}\frac{\partial}{\partial x}(ud) \tag{20}$$

という移流速度 v, w を持つ同じ形の２式となる。ここで、p, q は

$$p = c + \frac{1}{2}u \tag{21}$$

$$q = c - \frac{1}{2}u \tag{22}$$

である。また、v, w は特性速度とよばれる。それらは

$$v = u + c \tag{23}$$

$$w = u - c \tag{24}$$

であり、流速に左右へ伝わる波の速度 c, $-c$ を足したものである。p, q が方程式(19)(20)から求まると、u や h は

$$u = p - q \tag{25}$$

$$c = \frac{1}{2}(p + q) \tag{26}$$

$$h = \frac{c^2}{g} \tag{27}$$

から求められる。また、式(18)(19)の右辺は(24)(25)(26)を用いると

$$R = -\frac{g}{2c}\frac{\partial}{\partial x}(ud) = -\frac{g}{p+q}\frac{\partial}{\partial x}\left[(p-q)d\right] \tag{28}$$

となる。

2．数値解法

ここでは(19)(20)を数値的に解く方法を考える。

2.1 移流方程式の数値解

(19)(20)の右辺をゼロとおいた式は移流方程式

$$\frac{\partial f}{\partial t} + v\frac{\partial f}{\partial x} = 0 \tag{29}$$

である。移流方程式はある物理量 f が速度 v で移動してくることを表している。速度 v が一定の時は波動方程式と一致する。このことから、移流方程式を以下のように考えて解くことにする。

(1) $t \rightarrow t+\Delta t$ までは速度 $v(x,t)$は変化しない。

(2) (1)により、$t+\Delta t$ のとき x_i,での物理量 $f(x_i, t+\Delta t)$は $x=x_i-v\Delta t$ にあった量 $f(x_i-v\Delta t, t)$である。

(3) $v(x_i, t)>0$ とき、$x_i-v\Delta t$ での p の値はそれを挟む２つの格子点 x_i, と上流側の x_{i-1}

上の値 $f(x_i, t)$と $f(x_{i-1}, t)$から何らかの内挿により決定する ($v(x_i, t)<0$ のときは $f(x_{i+1}, t)$ と $f(x_i, t)$ から内挿する)。

内挿を直線補間とする場合は風上差分法と呼ばれる。風上差分法は本来ない拡散(数値拡散または人工拡散)を持つので、解がなまってしまう。そのため、ここでは内挿の方法として３次式を用いる CIP (Constrained Interpolation Profile) 法 (矢部・他, 2007) を使用する。

2.2 移流方程式と CIP 法

CIP 法では解 $f(x)$を３次式

$$f(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \tag{30}$$

で表す。a_i, b_i, c_i, d_iの４つの係数は２点 x_{i-1}, x_i において f の値と微分係数

$$G(x) = \frac{df}{dx} = 3a_i(x - x_i)^2 + 2b_i(x - x_i) + c_i \tag{31}$$

の連続から決定される。微分係数は全格子点上において連続であると仮定すると、連立１次方程式を解いて決めることができる (スプライン関数)。しかし、この方法では計算が大変である。CIP 法では、以下のように解くことで手間を省く。元の移流方程式(28)を x で微分すると、

$$\frac{\partial G}{\partial t} + v\frac{\partial G}{\partial x} = -G\frac{\partial v}{\partial x} \tag{32}$$

が得られる。この式は微分係数 G が移流し、格子点上の g_i が時間進行法で得られることを表している。このとき、a_i, b_i, c_i, d_iは以下のように表される。

$$a_i = \frac{G_i - G_{i-1}}{\Delta x} + \frac{2(f_i - f_{i-1})}{\Delta x^2} \tag{33}$$

$$b_i = \frac{3(f_{i-1} - f_i)}{\Delta x^3} - \frac{2G_i + G_{i-1}}{\Delta x^2} \tag{34}$$

$$c_i = G_i \tag{35}$$

$$d_i = f_i \tag{36}$$

$f(x_i, t+\Delta t)$と $G(x_i, t+\Delta t)$の値は(29)(30)式で$(x-x_i)$を$-v\Delta t$ と置くことにより得られる。この方法だと、微分係数の初期値が必要になるが、その値は中心差分などの方法で計算する。

方程式(19)(20)を解く場合には、移流項とそれ以外の部分に分け、

$$\frac{p^* - p^n}{\Delta t} = R \tag{37}$$

$$\frac{p^{n+1}-p^{*}}{\Delta t}+v\frac{\partial p}{\partial x}=0 \tag{38}$$

のように２段階で計算する。(38)を CIP 法の補間を用いて計算する。勾配の方程式は、(19)を x で微分した式

$$\frac{\partial G}{\partial t}+v\frac{\partial G}{\partial x}=-G\frac{\partial v}{\partial x}+\frac{\partial R}{\partial x} \tag{39}$$

である。もちろん、

$$G=\frac{\partial p}{\partial x} \tag{40}$$

である。右辺が 0 でないので、やはり移流項とそれ以外の項に分けて計算すれば良い。プログラムでは、

$$\frac{G^{*}-G^{n}}{\Delta t}=\frac{\partial}{\partial x}\left(\frac{p^{*}-p^{n}}{\Delta t}\right) \tag{41}$$

$$\frac{G'-G^{*}}{\Delta t}+v\frac{\partial G}{\partial x}=0 \tag{42}$$

$$\frac{G^{n+1}-G'}{\Delta t}=-G'\frac{\partial v}{\partial x} \tag{43}$$

のように３つに分けて計算している。ここで、(41)式の右辺は(37)式を R に代入したものである。CIP 法の補間を適用するのは移流項を計算する(42)式である。

参考文献

牛島　省：数値計算のための Fortran90/95 プログラミング入門，森北出版，2007.

矢部　孝・尾形洋一・滝沢研二：Java と CIP 法による CG シミュレーション，森北出版，2007.

３．数値シミュレーションの実行

ここでは(19)(20)を数値的に解く方法を考える。

3.1 モデルとパラメータの設定

ここまで、水面の重力波の基礎方程式について説明した。シミュレーションを行うには、モデルの設定を行わなければならい。計算を行うために必要な情報は以下の通りである。

(1) モデルの領域の大きさ

必要なのはモデルの水平サイズ L と水深である。今回、水深一定(d=0)の場合を考える

ので、平均水深 H_0 のみ考えれば良い。

(2) 流速

流れている水面の波を考えるときには、その流速 u_0 が必要である。ここでは、速度 u_0 を直接与えるのではなく、波の速度 c に対する比で与える。その速度比はフルード数(Froude number)とよばれ、

$$Fr = \frac{u_0}{\sqrt{gH_0}} \tag{44}$$

で表される。慣性力と重力の大きさの比を表している。フルード数は川のように流れる水面の波を考える場合は正の値、津波の場合は 0 とする。

(3) 初期に与える波の振幅: 初期条件

振幅を与えることで波を発生させる。初期振幅は計算領域の半分の所を中心に持つガウス分布型とする。関数

$$\eta = A_0 \exp\left[\frac{\left(x - L/2\right)^2}{\lambda_0^2}\right] \tag{45}$$

で表される。パラメータとして与えるのは振幅 A_0 と波長λ_0 である。

(4) 格子点数

空間座標を離散化して表したとき、計算を行う点を格子点と呼ぶ。波の波長の中に格子点が 10 個程度入るように全体の点数を決める。

(5) 時間間隔

時間進行を行うため計算する時間間隔Δt を与える。このΔt は格子点間で波が伝わる時間よりも小さくなければならない。つまり、1 タイムステップで波が伝わって良いのは隣の格子点までで、格子点を飛び越えて遠くまで波が伝わってはいけないということである。ここで、波が伝わる速さは $u+c$ である。格子点間距離と 1 タイムステップの間に波が伝わる長さの比をクーラン数と呼び、

$$cr = \frac{\left(u + c\right)\Delta t}{\Delta x} \tag{46}$$

で表される。Δx は格子点間隔

$$\Delta x = \frac{L}{n_x} \tag{47}$$

である。n_x は格子点間の数、つまり n_x に 1 足したものが格子点数である。cr が 1 以下になるようにΔt を設定しなければならない。プログラムでは与えた最大クーラン数よりクーラン数が大きくなると警告を出してプログラムの実行が止まるようになってい

る。

(6) 最大タイムステップ

時間進行は繰り返しで計算されるが、繰り返す数 N_{tmax} を指定する。このとき最終時間は時間間隔との積

$$t_{\max} = N_{t\max}\Delta t \tag{49}$$

である。

(7) ファイル出力の間隔

結果のファイル出力を行うタイムステップ間隔 I_{file} を指定する。これは、すべてのタイムステップで出力すると出力が多くなり過ぎるし、結果を把握するのに全部は必要ないからである。何タイムステップに 1 回出力するかを与える。初期値のファイルには番号 000 が付き、番号は出力するたびに 1 ずつ増える。このため、ファイルは水面高 h、速度 u、それぞれにつき、N_{file} +1 個(0 から N_{file} まで)できる。ここで、

$$N_{file} = N_{t\max}/I_{file}$$

である。

これらのモデルの設定のためのパラメータは `w_para.dat` の中で指定する。指定する値は下記の通りである。カギ括弧内は与える数値の単位である。

- 重力加速度: g [ms^{-2}]
- モデルの水平長: L [km]
- 平均水深: H_0 [m]
- フルード数: Fr
- 初期振幅: A_0 [m]
- 初期波長: λ_0 [m]
- 時間間隔: Δt [s]
- 最大タイムステップ数: N_{tmax}
- 最大クーラン数: cr
- 格子点間の数: n_x (n_x+1 が格子点数)
- 結果をファイルに出力するタイムステップ間隔数: I_{file}

3.2 シミュレーションの実行

今回のプログラムは偏微分方程式を高度な方法で解くために複雑で大規模なものとなっている。15 のプログラムファイル、パラメータを記述したファイル、コンパイルを自動的に行うための手続きを記述したファイルからなる。

(0) ファイルのダウンロードと解凍

ダウンロードするファイルは

```
http://home.hiroshima-u.ac.jp/nakakuki/Lectures/Computation.html
```

からリンクが張られている wave.tar というファイルである。ダウンロードしたファイルを適当な場所に移動させた後、ファイルを解凍(展開)する。解凍は以下のコマンドを入力する。

```
tar xvf wave.tar
```

ファイルを解凍すると、Wave というディレクトリができるので、

```
cd Wave
```

と入力してディレクトリ Wave の中に移動する。

(1) コンパイル

make と入力すると自動的にコンパイルされ、実行形式ファイル wave ができる。

(2) モデル設定の変更

モデルの設定の変更を行うには w_para.dat を変更する。

(3) 実行

./wave と入力する。

1つのタイムステップのデータが w_para.dat で指定したタイムステップおきに番号付きのファイルになって出力される。水面の高さ h のデータは h010 のように h で始まるファイルである。h000 は初期値、つまり最初に与えた振幅である。番号は 1 ずつ増える。tic.dat にはタイムステップ数 i、時間 t_i とクーラン数 c が出力される。

(4) 結果の可視化

gnuplot で

```
plot “h000” with line
replot “h004” with line
```

などと入力する。

以下は w_para.dat を変更せずに実行した結果を gnuplot で可視化したものである。深さ 10cm、長さ 4m の水路を左から右へ流れている水 (Fr=0.5)の表面上を伝わる波のシミュレーションとなっている。$c = 1.0$ [m s^{-1}], $u_0 = 0.5$ [m s^{-1}]である。図は 0, 0.1, 0.2, 0.4,…,2.0 秒での水面の高さを表している。

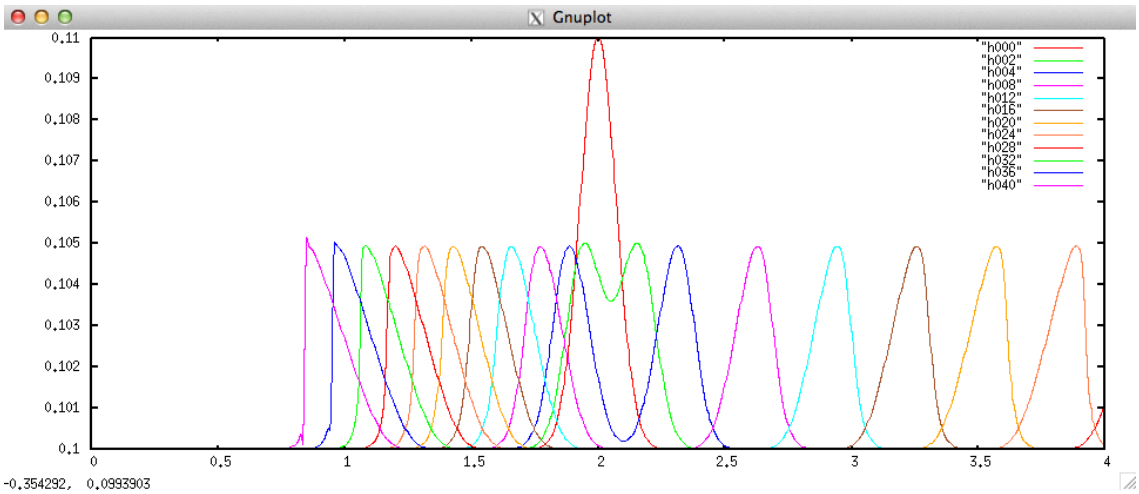
Gnuplot
0.11
0.109
0.108
0.107
0.106
0.105
0.104
0.103
0.102
0.101
0.1
0
0.5
1
1.5
2
2.5
3
3.5
4
"h000"
"h002"
"h004"
"h008"
"h012"
"h016"
"h020"
"h024"
"h028"
"h032"
"h036"
"h040"
-0.354292, 0.0993903

図２：計算結果

3.3 シミュレーション実行のヒント

1. プログラムのアーカイブファイル wave.tar をダウンロードし、シミュレーションの実行法に従って解凍(展開)せよ。ファイルは
   `http://home.hiroshima-u.ac.jp/nakakuki/others/wave.tar`
   にある。
2. プロクラムをコンパイルし、実行せよ。結果を図に表せ。
3. フルード数が 1 を超えるときにどのような現象が起きるのか、フルード数を変化させたシミュレーションを行い確認せよ。

図 3: 同じ場所に留まる川面の波

4. 津波のモデルに相応しい設定を考え、パラメータファイルを変更せよ。ここで、波長や振幅など物理パラメータだけでなく、格子点数や時間間隔の取り方にも注意せよ。津波の速度 c を考えて、クーラン数 cr が 0.1〜0.5 程度になるように時間間隔 Δt を設定せよ。
5. 津波のシミュレーションを実行し、水面高と流速のグラフを作成せよ。
6. 津波が(8)式の速度 c で伝播していることを確認せよ。

7. シンクに水を注いだとき、図ような輪がなぜ作られるのか３の結果から考察せよ。

図４: 水のリング

8. 波の進行方向の面が徐々に切り立ってくるのはなぜか、原因を考えよ。重力波の伝播速度を表す式(8)と式(17)を比べて考察せよ。
9. このシミュレーションでは波が垂直になってしまうと以降そのままの形で伝播する。実際の水面ではどのような現象が起きると考えられるか。

図５: 葛飾北斎・神奈川沖浪裏

図４・５画像引用元: Wikipedia

４．浅水方程式による水面の波のシミュレーションプログラム

CIP 法により、浅水方程式を解くプログラムである。プログラムファイル 15 個と合わせ、全部で 17 個のファイルからなる。これらをアーカイブしたファイルは、

```
http://home.hiroshima-u.ac.jp/nakakuki/others/wave.tar
```

からダウンロードできる。

注意：プログラムのコンパイルは次のような make コマンドで行うこと。

```
cd Wave
make
```

1. メインプログラム　wave_main.f90

配列変数の宣言、格子点の設定、時間進行の制御などを行う。配列の範囲は(-1:mx1)とした。-1, mx1 は境界の 1 つ外側の点で、0 と mx が境界上の点である。これは流入、流出の境界を扱うためである。

```
! ************************************************************************
! **  1-D wave propagation in the shallow water                         **
! **                                                                    **
! ************************************************************************

   program wave_main

      use m_param

      implicit none
      real(8), allocatable:: x(:),     d(:)
      real(8), allocatable:: u(:),     h(:)
      real(8), allocatable:: c(:),     v(:),     w(:)
      real(8), allocatable:: p(:),     q(:)
      real(8), allocatable:: pnew(:),  qnew(:)
      real(8), allocatable:: pstr(:),  qstr(:)
      real(8), allocatable:: px(:),    qx(:)
      real(8), allocatable:: pxnew(:), qxnew(:)
      real(8), allocatable:: pxstr(:), qxstr(:)
```

```
      real(8):: t
      integer:: itr, nf

!  ****  set model parameters  ****

      call set_param

!  ****  set grid  ****

      dx  = xL / dble( mx )
      mx1 = mx+1
!      write ( 6,* ) mx,mx1,dx

!  ****  allocate dimension variables  ****

      allocate ( x(-1:mx1), d(-1:mx1) )
      allocate ( u(-1:mx1), h(-1:mx1) )
      allocate ( c(-1:mx1), v(-1:mx1), w(-1:mx1) )
      allocate ( p(-1:mx1), q(-1:mx1) )
      allocate ( pnew(-1:mx1), qnew(-1:mx1) )
      allocate ( pstr(-1:mx1), qstr(-1:mx1) )
      allocate ( px(-1:mx1), qx(-1:mx1) )
      allocate ( pxnew(-1:mx1), qxnew(-1:mx1) )
      allocate ( pxstr(-1:mx1), qxstr(-1:mx1) )

!  ****  set bathymetry of ocean bottom  ****

      call set_bathy( d )

!  ****  set initial conditions  ****

      itr = 0
      nf  = 0
```

```
      t = 0.0d0

      call set_init( x, u, h )

!  ****  convert variables  *****

      call uh_to_cvwpq( u, h, c, v, w, p, q )

!  ****  write inital conditions to file  ****

      call uh_to_file( x, u, h, nf )
!      call pq_to_file( x, p, q, nf )

!  ****  Check Courant number  ****

      call chk_cour( v, w )

      open ( 18,file='tic.dat' )
      write( 18,* ) itr,t,cvw
      write( 6,* )  'main',itr,t,cvw

!  ****  set inital gradient values for CIP method  ****

      call set_igrad( p, px )
      call set_igrad( q, qx )

! ========================  time integration loop  ==================================

      do itr = 1,ntmax

        t = t + dt
        call propag( p, v, d, u, c, pnew, pstr, px, pxnew, pxstr )
        call propag( q, w, d, u, c, qnew, qstr, qx, qxnew, qxstr )
```

```
        call pq_to_cuhvw( p, q, c, u, h, v, w )

!  ****  write inital conditions to file  ****

        if ( mod( itr, ifskp )==0 ) then
          nf = nf + 1
          call uh_to_file( x, u, h, nf  )
!          call pq_to_file( x, p, q, nf )
        end if

        call chk_cour( v, w )
        write( 18,* ) itr,t,cvw
        write( 6,* )  'main',itr,t,cvw

      end do

! ========================  loop end  ==============================================

      close ( 10 )
      close ( 18 )

      stop
   end program wave_main
```

2. サブルーチンプログラム-1　w_set_param.f90

モデル設定のパラメータをファイル w_para.dat から読み込む

```
! ************************************************************************
! *   Set parameters                                                     *
! ************************************************************************

   subroutine set_param
```

```
      use m_param

      implicit none
      real(8):: xLk, wwk
      character(40) cpara

!  ****  Open parameter file  ****

      open ( 10,file='w_para.dat',status='old' )

!  ****  Read parameters from a file  ****

      read ( 10,* ) cpara         ! gravity accelaration
      read ( 10,* ) ga
      write( 6,* )  cpara,'=',ga

      read ( 10,* ) cpara         ! Length
      read ( 10,* ) xLk
      write( 6,* )  cpara,'=',xLk

      read ( 10,* ) cpara         ! Mean depth
      read ( 10,* ) H0
      write( 6,* )  cpara,'=',H0

      read ( 10,* ) cpara         ! fluid number
      read ( 10,* ) Fr
      write( 6,* )  cpara,'=',Fr

      read ( 10,* ) cpara         ! initial wave amplitude
      read ( 10,* ) wh
      write( 6,* )  cpara,'=',wh
```

```
      read ( 10,* ) cpara         ! initial wave width
      read ( 10,* ) wwk
      write( 6,* )  cpara,'=',wwk

      read ( 10,* ) cpara         ! time step intervals
      read ( 10,* ) dt
      write( 6,* )  cpara,'=',dt

      read ( 10,* ) cpara         ! maximum courant number
      read ( 10,* ) Cmax
      write( 6,* )  cpara,'=',Cmax

      read ( 10,* ) cpara         ! number of grid intervals
      read ( 10,* ) mx
      write( 6,* )  cpara,'=',mx

      read ( 10,* ) cpara         ! number of time steps
      read ( 10,* ) ntmax
      write( 6,* )  cpara,'=',ntmax

      read ( 10,* ) cpara         ! interval time steps for output files
      read ( 10,* ) ifskp
      write( 6,* )  cpara,'=',ifskp

!  ****  convert unit  ****

      xL = xLk * 1.0d3
      ww = wwk * 1.0d3

      return

   end subroutine set_param
```

3. サブルーチンプログラム-2　w_set_bathy.f90

海底地形を設定する。地形を変化させる場合はこのサブルーチンを変更する。平均水深からのずれを配列変数 d(:)に格納する。

```
! ************************************************************************
! *   Set bathymetry of the ocean bottom                                 *
! ************************************************************************

   subroutine set_bathy( d )

      use m_param

      implicit none
      real(8):: d(-1:mx1)
      real(8):: d0
      integer:: i

      d0 = 0.0d0

      do i = 0, mx1
        d(i) = d0
      end do

      return

   end subroutine set_bathy
```

4. サブルーチンプログラム-3　w_set_init.f90

速度と振幅の初期値を設定する。初期波形は正規分布の形とした。それぞれ u(:)および h(:)に格納する。今回のプログラムでは do 型の配列計算を積極的に用いた。

```
! ************************************************************************
!    Set inital conditions
```

```
! ************************************************************************

   subroutine set_init( x, u, h )

      use m_param

      implicit none
      real(8):: x(-1:mx1)
      real(8):: u(-1:mx1),  h(-1:mx1)
      integer:: i

      do i = -1, mx1
        x(i) = dx * dble(i)
!        write ( 6,* ) i,x(i)
      end do

      u(:) = Fr * sqrt( ga * H0 )
      h(:) = H0 + wh * exp( -( x(:) - 0.5d0 * xL )**2 / ww**2 )

      return

   end subroutine set_init
```

5. サブルーチンプログラム-4　w_uh_to_cvwpq.f90

u, h の値から、c, v, w, p, q を計算する。

```
! ************************************************************************
! *   calculate c, v, w, p, q from u, h                                  *
! ************************************************************************

   subroutine uh_to_cvwpq( u, h, c, v, w, p, q )

      use m_param
```

```
      implicit none
      real(8):: u(-1:mx1),  h(-1:mx1)
      real(8):: c(-1:mx1),  v(-1:mx1),  w(-1:mx1)
      real(8):: p(-1:mx1),  q(-1:mx1)
      integer:: i

      c(:) = sqrt( ga * h(:) )
      v(:) = u(:) + c(:)
      w(:) = u(:) - c(:)
      p(:) = c(:) + 0.5d0 * u(:)
      q(:) = c(:) - 0.5d0 * u(:)

      return

   end subroutine uh_to_cvwpq
```

6. サブルーチンプログラム-5　w_uh_file.f90

u(:), h(:)の値をファイルに出力する。ifskp ステップごとにこのサブルーチンが呼び出される。

```
! ***************************************************************************
! *   output u, h to files                                                  *
! ***************************************************************************

   subroutine uh_to_file( x, u, h, nf )

      use m_param

      implicit none
      real(8):: x(-1:mx1)
      real(8):: u(-1:mx1),  h(-1:mx1)
      integer:: i
      integer:: nf
```

```
      integer:: nfi1,  nfi2
      character(3) cnf
      character(4) ofile1, ofile2

!   open a new file for output

      nfi1 = 21
      nfi2 = 22

      write ( cnf, '(I3.3)' ) nf

      ofile1 = 'u'//cnf
      ofile2 = 'h'//cnf

      open( nfi1, file=ofile1 )
      open( nfi2, file=ofile2 )

!  write variables to file

      do i = 0, mx
        write ( nfi1,* ) x(i),u(i)
      end do
      do i = 0, mx
        write ( nfi2,* ) x(i),h(i)
      end do

      close( nfi1 )
      close( nfi2 )

      return

   end subroutine uh_to_file
```

7. サブルーチンプログラム-6　w_pq_file.f90

p(:), q(:)の値をファイルに出力する。プログラムのデバッグ用である。

```
! ************************************************************************
! *   output p, q to files                                               *
! ************************************************************************

   subroutine pq_to_file( x, p, q, nf )

      use m_param

      implicit none
      real(8):: x(-1:mx1)
      real(8):: p(-1:mx1),  q(-1:mx1)
      integer:: i
      integer:: nf
      integer:: nfi1,  nfi2
      character(3) cnf
      character(4) ofile1, ofile2

!   open a new file for output

      nfi1 = 25
      nfi2 = 26

      write ( cnf, '(I3.3)' ) nf

      ofile1 = 'p'//cnf
      ofile2 = 'q'//cnf

      open( nfi1, file=ofile1 )
      open( nfi2, file=ofile2 )
```

```
!  write variables to file

      do i = 0, mx
        write ( nfi1,* ) x(i),p(i)
      end do
      do i = 0, mx
        write ( nfi2,* ) x(i),q(i)
      end do

      close( nfi1 )
      close( nfi2 )

      return

   end subroutine pq_to_file
```

8. サブルーチンプログラム-7　w_chk_cour.f90

クーラン数 cr を計算する。Cvw>Cmax のときは dt を小さくするよう警告が出てプログラムの実行が止まる。

```
! ************************************************************************
! *   calculate Courant number                                           *
! ************************************************************************

   subroutine chk_cour( v, w )

      use m_param

      implicit none
      real(8):: v(-1:mx1),  w(-1:mx1)
      real(8):: cv, cw
      integer:: i
```

```
      cv = maxval( abs( v(:) ) * dt / dx )
      cw = maxval( abs( w(:) ) * dt / dx )
      Cvw = max( cv, cw )

      if ( Cvw > Cmax ) then
        write(6,*) ' C > Cmax: reduce dt '
        stop
      end if

      return

   end subroutine chk_cour
```

9. サブルーチンプログラム-8　w_igrad.f90

勾配 g の初期値を f の中心差分を用いて求める。

```
! ************************************************************************
! *   calculate initial value of gradient                                *
! ************************************************************************

   subroutine set_igrad( f, g )

      use m_param

      implicit none
      real(8):: f(-1:mx1)
      real(8):: g(-1:mx1)
      integer:: i

      do i = 0,mx
        g(i) = 0.5d0 * ( f(i+1) - f(i-1) ) / dx
!        write ( 6,* ) i,g(i)
      end do
```

```
      g(-1)  = 0.0d0
      g(mx1) = 0.0d0

   end subroutine set_igrad
```

10. サブルーチンプログラム-9　w_propag.f90

浅水方程式を時間積分する。移流項はサブルーチン advec_1D で処理される。境界外側の点は set_bound で扱われる。

```
! ************************************************************************
! *   time integration for wave propagation                              *
! ************************************************************************

   subroutine propag( f, vel, d, u, c, fnew, fstr, g, gnew, gstr )

      use m_param

      implicit none
      real(8):: f(-1:mx1)
      real(8):: vel(-1:mx1)
      real(8):: d(-1:mx1),    u(-1:mx1)
      real(8):: c(-1:mx1)
      real(8):: fnew(-1:mx1), fstr(-1:mx1)
      real(8):: g(-1:mx1)
      real(8):: gnew(-1:mx1), gstr(-1:mx1)
      integer:: i, iB

!  ****  non-advection term (bathymetric effect)  ****

      do i = 0,mx
        fstr(i) = f(i) - dt * 0.5d0 * ga / c(i) *  0.5d0 / dx *             &
                                ( u(i+1) * d(i+1) - u(i-1) * d(i-1) )
```

```
      end do

!  **** boundary condition  ****

      iB = -1
      if ( vel(iB) < 0.0d0 ) then
        fstr(iB) = f(iB) - dt * 0.5d0 * ga / c(iB) / dx *                  &
                           ( u(iB+1) * d(iB+1) - u(iB) * d(iB) )
       else
        fstr(iB) = f(iB)
      end if

      iB = mx1
      if ( vel(iB) > 0.0d0 ) then
        fstr(iB) = f(iB) - dt * 0.5d0 * ga / c(iB) / dx *                  &
                           ( u(iB) * d(iB) - u(iB-1) * d(iB-1) )
       else
        fstr(iB) = f(iB)
      end if

!  ****  non-advection term for gradient equation  ****

      do i = 1,mx
        gstr(i) = g(i) + 0.5d0 / dx *                                      &
                       ( fstr(i+1) - f(i+1) - (fstr(i-1) - f(i-1) ) )
      end do

!  ****  Boundary condition  ****

      iB = -1
      if ( vel(iB) < 0.0d0 ) then
        gstr(iB) = g(iB) + ( fstr(iB+1) - f(iB+1) - (fstr(iB) - f(iB) ) ) / dx
       else
```

```
        gstr(iB) = g(iB)
      end if
      iB = mx1
      if ( vel(iB) < 0.0d0 ) then
        gstr(iB) = g(iB) + ( fstr(iB) - f(iB) - (fstr(iB-1) - f(iB-1) ) ) / dx
       else
        gstr(iB) = g(iB)
      end if

!  ****  advection term  ****

      call advec_1D ( fstr, gstr, vel, fnew, gnew )
      call set_bound( fstr, gstr, vel, fnew, gnew )

!  ****  change old and new time-step values  ****

      f(:) = fnew(:)
      g(:) = gnew(:)

      return

   end subroutine propag
```

11. サブルーチンプログラム-10　w_pq_to_cuhvw.f90

p, q の値から、c, v, w, u, h を計算する。配列変数の代入文に do 型の並びを用いている。

```
! ************************************************************************
! *   calculate c, v, w, u, h from p, q                                  *
! ************************************************************************

   subroutine pq_to_cuhvw( p, q, c, u, h, v, w )

      use m_param
```

```
      implicit none
      real(8):: p(-1:mx1),  q(-1:mx1)
      real(8):: u(-1:mx1),  h(-1:mx1)
      real(8):: c(-1:mx1),  v(-1:mx1),  w(-1:mx1)

      c(:) = 0.5d0 * ( p(:) + q(:) )
      u(:) = p(:) - q(:)
      h(:) = c(:)**2 / ga
      v(:) = u(:) + c(:)
      w(:) = u(:) - c(:)

      return

   end subroutine pq_to_cuhvw
```

12. サブルーチンプログラム-11　w_advec1d.f90

方程式の移流項が計算される。CIP 法のサブルーチンが呼び出される。iup は上流側の点を表す。CIP 法のコードは矢部・他 (2007)を参考にした。

```
! ***************************************************************************
! *   calculate advection term                                              *
! ***************************************************************************

      subroutine advec_1D( f, g, vel, fnew, gnew )

      use m_param

      implicit none
      real(8):: f(-1:mx1), g(-1:mx1)
      real(8):: vel(-1:mx1)
      real(8):: fnew(-1:mx1), gnew(-1:mx1)
      real(8):: fi, fiup, gi, giup, fn1, gn1
```

```
      real(8):: XX, DD, gzi
      integer:: i
      integer:: ixsgn, iup

!  ****  calculate advection using a CIP method  ****

      do i = 0,mx

        ixsgn=1
        XX = vel(i)
        if ( XX < 0.0d0 ) ixsgn = -1
        iup = i - ixsgn

        DD  = -dble( ixsgn ) * dx

        fi   = f(i)
        fiup = f(iup)
        gi   = g(i)
        giup = g(iup)

        gzi = -vel(i) * dt

        call CIP_1d( fi, fiup, gi, giup, DD, gzi, fn1, gn1 )

        fnew(i) = fn1
        gnew(i) = gn1 - dt * gn1 * 0.5d0 * ( vel(i+1) - vel(i-1) ) / dx

      end do

      return

   end subroutine advec_1D
```

13. サブルーチンプログラム-12　w_set_bound.f90

境界の外側の点が計算される。外側の点が下流側にある場合は移流してくる値を CIP 法により計算し、上流側にある場合は前の時間の値としている。前者の場合、移流項以外の微分値は片側差分により求めている。

```
! ************************************************************************
! *   set boundary conditions consistent with CIP method                 *
! ************************************************************************

      subroutine set_bound( f, g, vel, fnew, gnew )

      use m_param

      implicit none
      real(8):: f(-1:mx1), g(-1:mx1)
      real(8):: vel(-1:mx1)
      real(8):: fnew(-1:mx1), gnew(-1:mx1)
      real(8):: fi, fiup, gi, giup, fn1, gn1
      real(8):: XX, DD, gzi
      integer:: i
      integer:: ixsgn, iup

!   ****  Left boundary  ****

      i = -1
!      write (6,* ) vel(i)
      if ( vel(i) < 0.0d0 ) then      ! flow out (advection from inside)

        ixsgn=-1
        iup = 0

        DD  = -dble( ixsgn ) * dx
```

```
        fi   = f(i)
        fiup = f(iup)
        gi   = g(i)
        giup = g(iup)

        gzi = -vel(i) * dt

        call CIP_1d( fi, fiup, gi, giup, DD, gzi, fn1, gn1 )

        fnew(i) = fn1
        gnew(i) = gn1 - dt * gn1 * ( vel(i+1) - vel(i) ) / dx
!        write (6,*) 'Bound-L_out',i,fnew(i),gnew(i)

       else       ! flow in (do nothing)

        fnew(i) = f(i)
        gnew(i) = g(i)
!        write (6,*) 'Bound-L_in ',i,fnew(i),gnew(i)

      end if

!   ****  Right boundary  ****

      i = mx1
!      write (6,* ) vel(i)
      if ( vel(i) > 0.0d0 ) then   ! flow out (advection from inside)

        ixsgn=1
        iup = mx

        DD  = -dble( ixsgn ) * dx

        fi   = f(i)
```

```
         fiup = f(iup)
         gi   = g(i)
         giup = g(iup)

         gzi = -vel(i) * dt

         call CIP_1d( fi, fiup, gi, giup, DD, gzi, fn1, gn1 )

         fnew(i) = fn1
         gnew(i) = gn1 - dt * gn1 * ( vel(i) - vel(i-1) ) / dx
!         write (6,*) 'Bound-R_out',i,fnew(i),gnew(i)

        else   ! flow in (do nothing)

         fnew(i) = f(i)
         gnew(i) = g(i)
!         write (6,*) 'Bound-R_in ',i,fnew(i),gnew(i)

       end if

!       write (6,*) 'Bound_v',fnew(-1),gnew(-1),fnew(mx1),gnew(mx1)

       return

   end subroutine set_bound
```

14. サブルーチンプログラム-13　w_cip1d.f90

CIP 法により移流してくる点における f, g の値を求め、これを$(x_i, t+\Delta t)$での値とする。$\xi = -v\Delta t$ は移流点までの距離である。プログラム中では gzi とした。DD は格子点間隔 dx に方向による符号をつけたものである。

```
! ************************************************************************
! *   CIP method for 1-D problems                                        *
```

```
! ************************************************************************

      subroutine CIP_1D( fi, fiup, gi, giup, DD, gzi, fn1, gn1 )

      use m_param

      implicit none
      real(8):: fi, fiup, gi, giup
      real(8):: fn1, gn1
      real(8):: DD, gzi
      real(8):: a, b, c, d
      integer:: i

!      write ( 6,* ) 'gzi =',gzi

!  ****  Calculate coefficients of the spline  ****

      a = ( gi + giup ) / DD**2 + 2.0d0 * ( fi - fiup ) / DD**3
      b =  3.0d0 * ( fiup - fi ) / DD**2 - ( 2.0d0 * gi + giup ) / DD
      c = gi
      d = fi

!  ****  pickup values at gzi ****

      fn1 = a * gzi**3 + b * gzi**2 + c * gzi + d
      gn1 = 3.0d0 * a * gzi**2 + 2.0d0 * b * gzi + c

      return

   end subroutine CIP_1D
```

15. サブルーチンプログラム (モジュール)-14　w_mod_param.f90

プログラム全体で使う変数を宣言するモジュールである。これらの変数はグローバル変数として扱われる。今回は配列変数をグローバル変数として扱わず、整合配列を用いている。このため、モジュール内では配列宣言をしていない。

```
! ************************************************************************
! *   type definitions for variables                                     *
! ************************************************************************
   module m_param

      implicit none
      real(8):: ga
      real(8):: xL, Fr, H0
      real(8):: wh, ww
      real(8):: dx
      real(8):: dt, Cmax
      real(8):: Cvw
      integer:: mx, mx1
      integer:: ntmax
      integer:: ifskp

   end module m_param
```

16. 入力データファイル　w_para.dat

物理パラメータを与えるデータである。流速については、速度ではなく Fr を与えるようになっている。長さ・水深の単位がそれぞれ[km]・[m]であることに注意しよう。

流しの水などの計算の例

```
'Gravity acceleration (m/s^2)'
10.0d0
'Length of the model (km)'
4.0d-3
```

```
'Mean depth (m)'
10.0d-2
'Froude number (non-dim.)'
0.5d0
'Initial wave amplitude (m)'
1.0d-2
'Initial wave width (km)'
0.1d-3
'Time step intervals (sec)'
1.0d-3
'Maximum Courant number (non-dim.)'
0.9d0
'Number of grid intervals'
800
'Number of time steps'
2000
'Interval time steps for output files'
50
```

津波の計算用の例

```
'Gravity acceleration (m/s^2)'
10.0d0
'Length of the model (km)'
4.0d3
'Mean depth (m)'
1.0d3
'Froude number (non-dim.)'
0.0d0
'Initial wave amplitude (m)'
20.0d0
'Initial wave width (km)'
40.0d0
```

```
'Time step intervals (sec)'
1.0d0
'Maximum Courant number (non-dim.)'
0.9d0
'Number of grid intervals'
4000
'Number of time steps'
16000
'Interval time steps for output files'
400
```

17. コンパイルの手続き　Makefile

コンパイルの手続きを記述したファイルである。Makefike というファイルがディレクトリ内にあると、make というコマンドを入力することによって自動的にコンパイルすることができるようになる。Makefile の中には使用するコンパイラやプログラムファイル、コンパイルする順番が記載されている。make clean と入力すると、オブジェクトファイル w_*.o や実行形式ファイル wave が消去される。

```
FC = f95
LD = f95

.SUFFIXES : .f90

TARGET = wave

SRC = w_mod_param.f90 \
          wave_main.f90 \
          w_advec1d.f90 \
          w_chk_cour.f90 \
          w_cip1d.f90 \
          w_igrad.f90 \
          w_pq_file.f90 \
          w_pq_to_cuhvw.f90 \
```

```
          w_propag.f90 \
          w_set_bathy.f90 \
          w_set_bound.f90 \
          w_set_init.f90 \
          w_set_param.f90 \
          w_uh_file.f90 \
          w_uh_to_cvwpq.f90

OBJ = w_mod_param.o \
          wave_main.o \
          w_advec1d.o \
          w_chk_cour.o \
          w_cip1d.o \
          w_igrad.o \
          w_pq_file.o \
          w_pq_to_cuhvw.o \
          w_propag.o \
          w_set_bathy.o \
          w_set_bound.o \
          w_set_init.o \
          w_set_param.o \
          w_uh_file.o \
          w_uh_to_cvwpq.o

MOD_FILES = m_param.mod

all: $(TARGET)

${TARGET} : ${OBJ}
          ${LD} -o $@ ${OBJ}

.f90.o:
          ${FC} -c $<
```

```
clean:
	rm -f ${TARGET} ${OBJ} ${MOD_FILES}
```

春惜しむ頃