Conservation Laws & Applications

Lecture IV: Short CLAWPACK demo

James A. Rossmanith

Department of Mathematics
University of Wisconsin – Madison

July 1st, 2010

Outline

Outline

Some basic information

- Main website:

  `http://www.clawpack.org`

- Authors:
  - R.J. LeVeque, U. Washington (Originator)
  - M. Berger, NYU (AMR)
  - J.O. Langseth, Norwegian Defence Res. Estab. (3D code)
  - D. Calhoun, Commissariat l'Energie Atomique (3D AMR)

- Code: `Fortran 77`/`Fortran 95` routines, `Python` wrappers

Basic features

- Solve hyperbolic balance laws of the form:

$$q_{,t} + \nabla \cdot \mathbf{F}\left(q,\, t,\, \mathbf{x}\right) = \psi\left(q,\, t,\, \mathbf{x}\right)$$

- Time-dependent problem in 1, 2, or 3 spatial dimensions
- Cartesian or logically Cartesian grids
- Adaptive mesh refinement in 2D and 3D
- Second-order accuracy with TVD wave limiters
- MPI parallel implementation
- Dynamically chosen time-steps
- Boundary conditions: extrapolation, periodic, solid wall
- Applications: advection, acoustics, Burgers, shallow water, Euler
- Easy to switch out ICs, BCs, Riemann solvers, & limiters
- MATLAB plotting routines

Outline

Getting started

- System requirements:
  1. Need a fortran compiler (`http://www.gnu.org`)
  2. Need MATLAB for plotting (latest CLAWPACK also supports `Python`)
- Download code from:

  `http://www.clawpack.org`
- Set environment variable in `.bashrc`

  `export CLAW=/Users/someuser/claw`
- Modify MATLAB path:

  `path(path,'/Users/someuser/claw/matlab')`

Input data file

claw2ez.data

```
125          mx           = cells in x direction
125          my           = cells in y direction

3            nout         = number of output times to print results
1            outstyle     = style of specifying output times
1.50d0       tfinal       = final time

0.1d0        dtv(1)       = initial dt (used in all steps if method(1)=0)
1.0d99       dtv(2)       = max allowable dt
1.0d0        cflv(1)      = max allowable Courant number
0.7d0        cflv(2)      = desired Courant number
500          nv(1)        = max number of time steps per call to claw2

1            method(1)    = 1 for variable dt,   = 0 for fixed dt
2            method(2)    = order
2            method(3)    = transverse order
1            method(4)    = verbosity of output
0            method(5)    = source term splitting
0            method(6)    = mcapa
0            method(7)    = maux (should agree with parameter in driver)
```

Input data file

claw2ez.data

```
3              meqn          = number of equations in hyperbolic system
3              mwaves        = number of waves in each Riemann solution
4 4 4          mthlim(mw)    = limiter for each wave  (mw=1,mwaves)

 0.d0          t0            = initial time
-2.5d0         xlower        = left edge of computational domain
 2.5d0         xupper        = right edge of computational domain
-2.5d0         ylower        = bottom edge of computational domain
 2.5d0         yupper        = top edge of computational domain

2              mbc           = number of ghost cells at each boundary
3              mthbc(1)      = type of boundary conditions at left
1              mthbc(2)      = type of boundary conditions at right
3              mthbc(3)      = type of boundary conditions at bottom
1              mthbc(4)      = type of boundary conditions at top
```

Basic functions

`Fortran` files:

- `qinit.f` – initial condition routine
- `setprob.f` – read-in any extra parameters from `setprob.data`
- `setaux.f` – auxiliary variables
- `../rp/rpn2.f` – normal Riemann solver
- `../rp/rpt2.f` – transverse Riemann solver

MATLAB **files:**

- `setplot2.m` – plotting options
- `beforeframe.m` – actions to carry out before plotting current frame
- `afterframe.m` – actions to carry out after plotting current frame

Just do it:

- Compile: `make` (Unix comand line)
- Run: `xclaw` (Unix comand line)
- Plot: `plotclaw2` (MATLAB comand line)

Outline

Shallow water with bottom topography

$$\begin{bmatrix} h \\ hu \\ hv \end{bmatrix}_{,t} + \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix}_{,x} + \begin{bmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix}_{,y} = \begin{bmatrix} 0 \\ -ghb_{,x} \\ -ghb_{,y} \end{bmatrix}$$

- Flux Jacobian: (assume $\|\mathbf{n}\| = 1, \quad \text{let } c := \sqrt{gh}$)

$$A(q, \mathbf{n}) = \begin{bmatrix} 0 & n^1 & n^2 \\ n^1 gh - u(\mathbf{u}\cdot\mathbf{n}) & n^1 u + \mathbf{n}\cdot\mathbf{u} & n^2 u \\ n^2 gh - v(\mathbf{u}\cdot\mathbf{n}) & n^1 v & n^2 v + \mathbf{n}\cdot\mathbf{u} \end{bmatrix}$$

- Eigenvalues:

$$\lambda \quad = \quad \mathbf{u}\cdot\mathbf{n} - c, \quad \mathbf{u}\cdot\mathbf{n}, \quad \mathbf{u}\cdot\mathbf{n} + c$$

- Eigenvectors:

$$R = \begin{bmatrix} 1 & 0 & 1 \\ u - n^1 c & -n^2 & u + n^1 c \\ v - n^2 c & n^1 & v + n^2 c \end{bmatrix}, \quad L = \frac{1}{2c}\begin{bmatrix} c + \mathbf{u}\cdot\mathbf{n} & -n^1 & -n^2 \\ 2c\left(n^2 u - n^1 v\right) & -2c\,n^2 & 2c\,n^1 \\ c - \mathbf{u}\cdot\mathbf{n} & n^1 & n^2 \end{bmatrix}$$

Shallow water with bottom topography

Normal Riemann solver in the x-direction

1. Define Roe-average state:

$$\hat{h} = \frac{1}{2}(h_r + h_\ell), \quad \hat{u} = \frac{\sqrt{h_r}\,u_r + \sqrt{h_\ell}\,u_\ell}{\sqrt{h_r} + \sqrt{h_\ell}}, \quad \hat{v} = \frac{\sqrt{h_r}\,v_r + \sqrt{h_\ell}\,v_\ell}{\sqrt{h_r} + \sqrt{h_\ell}}$$

2. Source term:

$$\hat{\psi} = \begin{bmatrix} 0, & -g\hat{h}\dfrac{(b_r - b_\ell)}{\Delta x}, & 0 \end{bmatrix}^T$$

3. Decompose flux differences and determine $\mathcal{Z}$-waves:

$$\beta^{(p)} = \ell_x^{(p)} \cdot \left(f_r - f_\ell - \Delta x\,\hat{\psi}\right) \qquad \text{and} \qquad \mathcal{Z}^{(p)} = \beta^{(p)} r_x^{(p)}$$

4. Define wave speeds:

$$s^{(p)} = \begin{bmatrix} \hat{u} - c, & \hat{u}, & \hat{u} + c \end{bmatrix}^T$$

5. Determine fluctuations:

$$\mathcal{A}^- \Delta Q = \sum_{p:s^{(p)}<0} \mathcal{Z}^{(p)} + \frac{1}{2} \sum_{p:s^{(p)}=0} \mathcal{Z}^{(p)}, \quad \mathcal{A}^+ \Delta Q = \sum_{p:s^{(p)}>0} \mathcal{Z}^{(p)} + \frac{1}{2} \sum_{p:s^{(p)}=0} \mathcal{Z}^{(p)}$$

Shallow water with bottom topography

Transverse Riemann solver in the x-direction

1. Transverse velocities:

$$s^{(p)} = \begin{bmatrix} \hat{v} - c, & \hat{v}, & \hat{v} + c \end{bmatrix}^T$$

2. Decompose left/right fluctuations:

$$\beta^{(p),-} = \ell_y^{(p)} \cdot \mathcal{A}^- \Delta Q \qquad \text{and} \qquad \beta^{(p),+} = \ell_y^{(p)} \cdot \mathcal{A}^+ \Delta Q$$
$$\mathcal{Z}^{(p),-} = \beta^{(p),-} r_y^{(p)} \qquad \text{and} \qquad \mathcal{Z}^{(p),+} = \beta^{(p),+} r_y^{(p)}$$

3. Compute up/down fluctuations:

$$\mathcal{B}^+ \mathcal{A}^+ dq = \sum_{p:s^{(p)}>0} \mathcal{Z}^{(p),+} \qquad \text{and} \qquad \mathcal{B}^- \mathcal{A}^+ dq = \sum_{p:s^{(p)}<0} \mathcal{Z}^{(p),+}$$
$$\mathcal{B}^+ \mathcal{A}^- dq = \sum_{p:s^{(p)}>0} \mathcal{Z}^{(p),-} \qquad \text{and} \qquad \mathcal{B}^- \mathcal{A}^- dq = \sum_{p:s^{(p)}<0} \mathcal{Z}^{(p),-}$$

Shallow water with bottom topography

Example

Example:

$$\Omega = [0,1] \times [0,1], \qquad \text{non-reflecting BCs on } \partial\Omega$$

$$b(x,y) = \frac{1}{2}\exp\left(-10\,(2x-1)^2 - 20\,(2y-1)^2\right)$$

$$h(0,x,y) + b(x,y) = \begin{cases} 1.02 & \text{if} \quad |x-0.1| \le 0.2 \\ 1 & \text{otherwise} \end{cases}$$

$$u(0,x,y) = 0$$

$$v(0,x,y) = 0$$

Files to edit:

- `claw2ez.data`
- `qinit.f`
- `rp/rpn2.f`
- `rp/rpt2.f`

Shallow water with bottom topography

qinit.f

```
c     =====================================================
      subroutine qinit(maxmx,maxmy,meqn,mbc,mx,my,xlower,ylower,
     &     dx,dy,q,maux,aux)
c     =====================================================
c
      implicit none
      integer maxmx,maxmy,meqn,mbc,mx,my,maux
      double precision dx,dy,xlower,ylower
      double precision   q(1-mbc:maxmx+mbc, 1-mbc:maxmy+mbc, meqn)
      double precision aux(1-mbc:maxmx+mbc, 1-mbc:maxmy+mbc, maux)
      double precision xc,yc,bottom
      integer i,j

      do i=1,mx
         xc = xlower + (i-0.5d0)*dx
         do j=1,my
            yc = ylower + (j-0.5d0)*dy

            bottom =  0.5d0*dexp(-10.d0*(2.d0*xc-1.d0)**2
     &           -20.d0*(2.d0*yc-1.d0)**2)
```

Shallow water with bottom topography

qinit.f

```
        bottom =  0.5d0*dexp(-10.d0*(2.d0*xc-1.d0)**2
&            -20.d0*(2.d0*yc-1.d0)**2)

        aux(i,j,1) = bottom

        if (dabs(xc-0.1d0).le.0.05d0) then
           q(i,j,1) = 1.02d0 - bottom
        else
           q(i,j,1) = 1.00d0 - bottom
        endif

        q(i,j,2) = 0.d0
        q(i,j,3) = 0.d0

     enddo
  enddo

  return
  end
```

Shallow water with bottom topography

rpn2sw.f

```
c     =====================================================
      subroutine rpn2(ixy,maxm,meqn,mwaves,mbc,mx,ql,qr,auxl,auxr,
     & fwave,s,amdq,apdq)
c     =====================================================
c
      implicit double precision (a-h,o-z)
      dimension fwave(1-mbc:maxm+mbc, meqn, mwaves)
      dimension     s(1-mbc:maxm+mbc, mwaves)
      dimension    ql(1-mbc:maxm+mbc, meqn)
      dimension    qr(1-mbc:maxm+mbc, meqn)
      dimension  auxl(1-mbc:maxm+mbc, *)
      dimension  auxr(1-mbc:maxm+mbc, *)
      dimension  apdq(1-mbc:maxm+mbc, meqn)
      dimension  amdq(1-mbc:maxm+mbc, meqn)
      double precision g
      common /param/  g    !# gravitational parameter
      double precision hl,ul,vl,hr,ur,vr,h,u,v,c
      double precision n1,n2,unl,unr,un
      double precision Rmat(3,3)
      double precision Lmat(3,3)
      double precision df(3),beta(3)
```

Shallow water with bottom topography

rpn2sw.f

```
      if (ixy.eq.1) then
         n1 = 1.0
         n2 = 0.0
      else
         n1 = 0.0
         n2 = 1.0
      endif

c     # loop over all cells in this slice
      do i=2-mbc,mx+mbc

c        # Left states
         hl = qr(i-1,1)
         ul = qr(i-1,2)/hl
         vl = qr(i-1,3)/hl
         unl = ul*n1 + vl*n2
         bl = auxr(i-1,1)

c        # Right states
         hr = ql(i,1)
         ur = ql(i,2)/hr
         vr = ql(i,3)/hr
         unr = ur*n1 + vr*n2
         br = auxl(i,1)
```

Shallow water with bottom topography

rpn2sw.f

```
c         # Average states
          h = 0.5*(hl+hr)
          u = (dsqrt(hl)*ul + dsqrt(hr)*ur)/(dsqrt(hl)+dsqrt(hr))
          v = (dsqrt(hl)*vl + dsqrt(hr)*vr)/(dsqrt(hl)+dsqrt(hr))
          c = dsqrt(g*h)
          un = n1*u + n2*v

c         # Flux differences
          df(1) = (hr*unr) - (hl*unl)
          df(2) = (hr*ur*unr + 0.5*g*hr*hr*n1)
     &          - (hl*ul*unl + 0.5*g*hl*hl*n1)
     &          + g*h*(br-bl)*n1
          df(3) = (hr*vr*unr + 0.5*g*hr*hr*n2)
     &          - (hl*vl*unl + 0.5*g*hl*hl*n2)
     &          + g*h*(br-bl)*n2

c         # Right eigenvectors
          Rmat(1,1) =  1.0
          Rmat(2,1) =  u - n1*c
          Rmat(3,1) =  v - n2*c

          Rmat(1,2) =  0.0
          Rmat(2,2) = -n2
          Rmat(3,2) =  n1

          Rmat(1,3) =  1.0
          Rmat(2,3) =  u + n1*c
          Rmat(3,3) =  v + n2*c
```

Shallow water with bottom topography

rpn2sw.f

```
c       # Left eigenvectors
        Lmat(1,1) =  (c+un)/(2.0*c)
        Lmat(1,2) = -n1/(2.0*c)
        Lmat(1,3) = -n2/(2.0*c)

        Lmat(2,1) =  n2*u-n1*v
        Lmat(2,2) = -n2
        Lmat(2,3) =  n1

        Lmat(3,1) =  (c-un)/(2.0*c)
        Lmat(3,2) =  n1/(2.0*c)
        Lmat(3,3) =  n2/(2.0*c)

c       # Beta's
        do m=1,meqn
           beta(m) = 0.0
           do k=1,meqn
              beta(m) = beta(m) + Lmat(m,k)*df(k)
           enddo
        enddo

c       # Z-waves
        do m=1,meqn
           do k=1,meqn
              fwave(i,m,k) = beta(k)*Rmat(m,k)
           enddo
        enddo
```

Shallow water with bottom topography

rpn2sw.f

```
c         # Wave speeds
          s(i,1) = un - c
          s(i,2) = un
          s(i,3) = un + c

c         # Fluctuations
          do m=1,meqn
             amdq(i,m) = 0.0
             apdq(i,m) = 0.0
             do k=1,meqn
                if (s(i,k)<1.0e-14) then
                   amdq(i,m) = amdq(i,m) + fwave(i,m,k)
                elseif (s(i,k)>1.0e-14) then
                   apdq(i,m) = apdq(i,m) + fwave(i,m,k)
                else
                   amdq(i,m) = amdq(i,m) + 0.5*fwave(i,m,k)
                   apdq(i,m) = apdq(i,m) + 0.5*fwave(i,m,k)
                endif
             enddo
          enddo

       enddo

       return
       end
```

Shallow water with bottom topography

rpt2sw.f

```
c     =====================================================
      subroutine rpt2(ixy,maxm,meqn,mwaves,mbc,mx,
     &                ql,qr,aux1,aux2,aux3,
     & ilr,asdq,bmasdq,bpasdq)
c     =====================================================
      implicit double precision (a-h,o-z)
      dimension     ql(1-mbc:maxm+mbc, meqn)
      dimension     qr(1-mbc:maxm+mbc, meqn)
      dimension   asdq(1-mbc:maxm+mbc, meqn)
      dimension bmasdq(1-mbc:maxm+mbc, meqn)
      dimension bpasdq(1-mbc:maxm+mbc, meqn)
      common /param/  g    !# gravitational parameter
c
      double precision hl,ul,vl,hr,ur,vr,h,u,v,c
      double precision n1,n2,unl,unr,un
      double precision Rmat(3,3)
      double precision Lmat(3,3)
      double precision df(3),beta(3),sb(3),fwaveb(3,3)

      if (ixy.eq.1) then
         n1 = 0.0
         n2 = 1.0
      else
         n1 = 1.0
         n2 = 0.0
      endif
```

Shallow water with bottom topography

rpt2sw.f

```
c     # loop over all cells in this slice
      do i=2-mbc,mx+mbc

c        # Left states
         hl = qr(i-1,1)
         ul = qr(i-1,2)/hl
         vl = qr(i-1,3)/hl
         unl = ul*n1 + vl*n2

c        # Right states
         hr = ql(i,1)
         ur = ql(i,2)/hr
         vr = ql(i,3)/hr
         unr = ur*n1 + vr*n2

c        # Average states
         h = 0.5*(hl+hr)
         u = (dsqrt(hl)*ul + dsqrt(hr)*ur)/(dsqrt(hl)+dsqrt(hr))
         v = (dsqrt(hl)*vl + dsqrt(hr)*vr)/(dsqrt(hl)+dsqrt(hr))
         c = dsqrt(g*h)
         un = n1*u + n2*v

c        # Flux differences
         df(1) = asdq(i,1)
         df(2) = asdq(i,2)
         df(3) = asdq(i,3)
```

Shallow water with bottom topography

rpt2sw.f

```
c       # Right eigenvectors
        Rmat(1,1) =  1.0
        Rmat(2,1) =  u - n1*c
        Rmat(3,1) =  v - n2*c

        Rmat(1,2) =  0.0
        Rmat(2,2) = -n2
        Rmat(3,2) =  n1

        Rmat(1,3) =  1.0
        Rmat(2,3) =  u + n1*c
        Rmat(3,3) =  v + n2*c

c       # Left eigenvectors
        Lmat(1,1) =  (c+un)/(2.0*c)
        Lmat(1,2) = -n1/(2.0*c)
        Lmat(1,3) = -n2/(2.0*c)

        Lmat(2,1) =  n2*u-n1*v
        Lmat(2,2) = -n2
        Lmat(2,3) =  n1

        Lmat(3,1) =  (c-un)/(2.0*c)
        Lmat(3,2) =  n1/(2.0*c)
        Lmat(3,3) =  n2/(2.0*c)
```

Shallow water with bottom topography

rpt2sw.f

```
c         # Beta's
          do m=1,meqn
             beta(m) = 0.0
             do k=1,meqn
                beta(m) = beta(m) + Lmat(m,k)*df(k)
             enddo
          enddo

c         # Z-waves
          do m=1,meqn
             do k=1,meqn
                fwaveb(m,k) = beta(k)*Rmat(m,k)
             enddo
          enddo

c         # Wave speeds
          sb(1) = un - c
          sb(2) = un
          sb(3) = un + c
```

Shallow water with bottom topography

rpt2sw.f

```
c         # Fluctuations
          do me=1,meqn
             bmasdq(i,me) = 0.d0
             bpasdq(i,me) = 0.d0
             do mw=1,mwaves

                sr = dmax1(sb(mw), 0.d0)
                sl = dmin1(sb(mw), 0.d0)

                bmasdq(i,me) = bmasdq(i,me) + sl*fwaveb(me,mw)
                bpasdq(i,me) = bpasdq(i,me) + sr*fwaveb(me,mw)

             enddo
          enddo

       enddo

       return
       end
```

Shallow water with bottom topography

DEMO

Outline

Extensions of CLAWPACK

Extensions:

- `CLAWMAN` – for solving problems on curved manifolds
- `TsunamiClaw` – shallow water wave tsunami modeling
- `ChomboClaw` – `AMRCLAW` in `Chombo` framework (LBL)
- `WENOCLAW` – High order accurate wave-propagation algorithms
- `sphereCLAW` – Multi-block code for cubed sphere grids
- `MHDCLAW` – Constrained transport algorithm for MHD

Related software:

- `AMROC` – object oriented `C++` AMR (R. Deiterding)
- `BEARCLAW` – `Fortran 95`, different AMR, cut-cells (S. Mitran)
- `ZPLCLAW` – `ZPL` parallel language (Y. Hu)
- `DoGPack` – object oriented `C++` DG-FEM (J. Rossmanith)