Bearclaw 2D Euler equations

1 Theory

1.1 Conservation laws of gas dynamics

The 2D Euler equations

$$\boldsymbol{q}_t + \boldsymbol{f}(\boldsymbol{q})_x + \boldsymbol{g}(\boldsymbol{q})_y = 0, \tag{1}$$

$$\boldsymbol{q} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix} = \begin{pmatrix} \rho \\ l \\ m \\ \varepsilon \end{pmatrix}, \boldsymbol{f}(\boldsymbol{q}) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ \rho u H \end{pmatrix}, \boldsymbol{g}(\boldsymbol{q}) = \begin{pmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ \rho v H \end{pmatrix}, \tag{2}$$

express conservation of mass, momentum, total energy in an inviscid, compressible fluid. For a perfect gas the constitutive relations relating thermodynamic variables are

$$e = c_v T = \frac{RT}{\gamma - 1}, E = e + \frac{u^2 + v^2}{2}, \tag{3}$$

$$h = e + \frac{p}{\rho}, H = h + \frac{u^2 + v^2}{2} \Rightarrow H = E + \frac{p}{\rho}, \tag{4}$$

$$p = \rho R T = \rho(\gamma - 1)\left(E - \frac{u^2 + v^2}{2}\right), \tag{5}$$

$$H = \gamma E - (\gamma - 1)\frac{u^2 + v^2}{2}. \tag{6}$$

Physical quantity	Notation	SI units	Physical quantity	Notation	SI units
Mass density	ρ	kg/m^3	Pressure	p	N/m^2
Temperature	T	K	Internal energy	e	J/kg=m^2/s^2
Specific heat, constant volume	c_v	J/kg/K	Adiabatic constant	γ	-
Perfect gas constant	R	J/kg/K	Total energy	E	J/kg=m^2/s^2
Enthalpy	h	J/kg=m^2/s^2	Total enthalpy	H	J/kg=m^2/s^2
x-momentum	$l = \rho u$	kg/m^2/s	y-momentum	$m = \rho v$	kg/m^2/s

Table 1. Notations

The fluxes $(\boldsymbol{f}, \boldsymbol{g})$ expressed in the conservative variables $\boldsymbol{q} = (\rho, l, m, \varepsilon)$ are

○

$$\boldsymbol{f}(\boldsymbol{q}) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ \rho u H \end{pmatrix} = \begin{pmatrix} l \\ \dfrac{l^2}{\rho} + (\gamma - 1)\left(\varepsilon - \dfrac{l^2 + m^2}{2\rho}\right) \\ \dfrac{l m}{\rho} \\ \dfrac{l}{\rho}\left(\gamma\varepsilon - (\gamma - 1)\dfrac{l^2 + m^2}{2\rho}\right) \end{pmatrix}, \tag{7}$$

$$\boldsymbol{g}(\boldsymbol{q}) = \begin{pmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ \rho v H \end{pmatrix} = \begin{pmatrix} m \\ \dfrac{l m}{\rho} \\ \dfrac{m^2}{\rho} + (\gamma - 1)\left(\varepsilon - \dfrac{l^2 + m^2}{2\rho}\right) \\ \dfrac{m}{\rho}\left(\gamma\varepsilon - (\gamma - 1)\dfrac{l^2 + m^2}{2\rho}\right) \end{pmatrix}. \tag{8}$$

1.2 Eigenstructure

1.2.1 Flux Jacobians

- The flux Jacobians are defined as

$$\mathbf{A} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}, \mathbf{B} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{q}}, \tag{9}$$

and can be concisely re-expressed in the primitive variables (ρ, u, v, p) and sound speed $c^2 = \gamma p / \rho$.

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ \frac{1}{2}((\gamma-3)\,u^2 + (\gamma-1)\,v^2) & -(\gamma-3)\,u & v - \gamma v & \gamma - 1 \\ -u\,v & v & u & 0 \\ \frac{u\,((\gamma^2 - 3\,\gamma + 2)\,(u^2+v^2) - 2\,c^2)}{2\,(\gamma-1)} & \frac{c^2}{\gamma-1} + \frac{1}{2}((3-2\,\gamma)\,u^2 + v^2) & -(\gamma-1)\,u\,v & \gamma u \end{pmatrix} \tag{10}$$

$$\mathbf{B} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ -u\,v & v & u & 0 \\ \frac{1}{2}((\gamma-1)\,u^2 + (\gamma-3)\,v^2) & u - \gamma u & -(\gamma-3)\,v & \gamma - 1 \\ \frac{v\,((\gamma^2 - 3\,\gamma + 2)\,(u^2+v^2) - 2\,c^2)}{2\,(\gamma-1)} & -(\gamma-1)\,u\,v & \frac{c^2}{\gamma-1} + \frac{1}{2}(u^2 + (3-2\,\gamma)\,v^2) & \gamma v \end{pmatrix} \tag{11}$$

```
In[4]:= ToPrimitiveVariables={l->rho u, m->rho v, epsilon->p/(gamma-1) + rho (u^2+v^2)/2}
```

$$\left\{l \to \rho\,u, m \to \rho\,v, \epsilon \to \frac{p}{\gamma-1} + \frac{1}{2}\,\rho\,(u^2+v^2)\right\}$$

```
In[5]:= ToSoundSpeed = p->rho c^2/gamma
```

$$p \to \frac{c^2\,\rho}{\gamma}$$

```
In[6]:= A=Simplify[Grad[f,q] /. ToPrimitiveVariables /. ToSoundSpeed]
```

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ \frac{1}{2}((\gamma-3)\,u^2 + (\gamma-1)\,v^2) & -(\gamma-3)\,u & v - \gamma v & \gamma - 1 \\ -u\,v & v & u & 0 \\ \frac{u\,((\gamma^2 - 3\,\gamma + 2)\,(u^2+v^2) - 2\,c^2)}{2\,(\gamma-1)} & \frac{c^2}{\gamma-1} + \frac{1}{2}((3-2\,\gamma)\,u^2 + v^2) & -(\gamma-1)\,u\,v & \gamma u \end{pmatrix}$$

```
In[7]:= B=Simplify[Grad[g,q] /. ToPrimitiveVariables /. ToSoundSpeed]
```

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ -u\,v & v & u & 0 \\ \frac{1}{2}((\gamma-1)\,u^2 + (\gamma-3)\,v^2) & u - \langle\text{gammau}\rangle & -(\gamma-3)\,v & \gamma - 1 \\ \frac{v\,((\gamma^2 - 3\,\gamma + 2)\,(u^2+v^2) - 2\,c^2)}{2\,(\gamma-1)} & -(\gamma-1)\,u\,v & \frac{c^2}{\gamma-1} + \frac{1}{2}(u^2 + (3-2\,\gamma)\,v^2) & \gamma v \end{pmatrix}$$

```
In[8]:=
```

1.2.2 Eigendecomposition of two-dimensional flux Jacobians

- The solution of $\mathbf{A}\,\mathbf{X} = \mathbf{X}\boldsymbol{\Lambda}^x$ is

$$\mathbf{X} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ u - c & 0 & u & c + u \\ v & 1 & v & v \\ H - c\,u & v & \frac{1}{2}(u^2+v^2) & H + c\,u \end{pmatrix}, \boldsymbol{\Lambda}^x = \begin{pmatrix} u - c & 0 & 0 & 0 \\ 0 & u & 0 & 0 \\ 0 & 0 & u & 0 \\ 0 & 0 & 0 & c + u \end{pmatrix}.$$

The first, last eigenvectors correspond to eigenvalues $u \pm c$ and describe backward, forward acoustic modes. There is a repeated u eigenvalue, and the corresponding eigenvectors can be expressed such they correspond to propagation of shear and entropy waves. Note that the quantities arising in the eigenvector matrix are expressed in terms of $\boldsymbol{x}(\boldsymbol{q}) = (c, u, v, H)$, a transformation of the conservative variables $\boldsymbol{q}$.

```
In[23]:= lx = Simplify[Eigenvalues[A],gamma>1 && c>0]
```

$$\{c + u, u - c, u, u\}$$

```
In[26]:= X=Simplify[Eigenvectors[A] /. gamma -> 1 + c^2/(H-(u^2+v^2)/2)];
         X=Simplify[Table[X[[i]]/X[[i,1]],{i,1,4}]];
         Transpose[X]
```

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ c+u & u-c & u & u \\ v & v & 0 & \frac{v^2-u^2}{2\,v} \\ H+c\,u & H-c\,u & \frac{1}{2}(u^2-v^2) & 0 \end{pmatrix}$$

```
In[27]:= x3=Simplify[X[[3]]-X[[4]]]; x3 = x3/x3[[3]]
```

$\{0,0,1,v\}$

```
In[28]:= x4=Simplify[(u^2+v^2)/(u^2-v^2) X[[3]]+a X[[4]] /. {a -> 1 - (u^2+v^2)/(u^2-v^2)}]
```

$\left\{1,u,v,\frac{1}{2}(u^2+v^2)\right\}$

```
In[29]:= X={X[[2]],x3,x4,X[[1]]};
         Transpose[X]
```

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ u-c & 0 & u & c+u \\ v & 1 & v & v \\ H-c\,u & v & \frac{1}{2}(u^2+v^2) & H+c\,u \end{pmatrix}$$

```
In[13]:= lambdax = {lx[[2]],lx[[3]],lx[[4]],lx[[1]]}
```

$\{u-c,u,u,c+u\}$

```
In[14]:= Table[Simplify[A.X[[i]] - lambdax[[i]] X[[i]] /. gamma -> 1 + c^2/(H-(u^2+v^2)/2)],{i,1,4}]
```

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

```
In[15]:= DiagonalMatrix[lambdax]
```

$$\begin{pmatrix} u-c & 0 & 0 & 0 \\ 0 & u & 0 & 0 \\ 0 & 0 & u & 0 \\ 0 & 0 & 0 & c+u \end{pmatrix}$$

```
In[16]:=
```

$\mathbf{B\,Y=Y\Lambda}^y$.

- The solution of $\mathbf{B\,Y=Y\Lambda}^y$ is

$$\mathbf{Y}=\begin{pmatrix} 1 & 0 & 1 & 1 \\ u & 1 & u & u \\ v-c & 0 & v & c+v \\ H-c\,v & u & \frac{1}{2}(u^2+v^2) & H+c\,v \end{pmatrix}, \mathbf{\Lambda}^y=\begin{pmatrix} v-c & 0 & 0 & 0 \\ 0 & v & 0 & 0 \\ 0 & 0 & v & 0 \\ 0 & 0 & 0 & c+v \end{pmatrix}.$$

```
In[30]:= ly = Simplify[Eigenvalues[B],gamma>1 && c>0]
```

$\{c+v,v-c,v,v\}$

```
In[31]:= Y=Simplify[Eigenvectors[B] /. gamma -> 1 + c^2/(H-(u^2+v^2)/2)];
         Y=Simplify[{Y[[1]]/Y[[1,1]],Y[[2]]/Y[[2,1]],Y[[3]]/Y[[3,2]],Y[[4]]/Y[[4,1]]}];
         Transpose[Y]
```

$$\begin{pmatrix} 1 & 1 & 0 & 1 \\ u & u & 1 & \frac{u^2-v^2}{2\,u} \\ c+v & v-c & 0 & v \\ H+c\,v & H-c\,v & u & 0 \end{pmatrix}$$

```
In[32]:= y4=Simplify[(u^2+v^2)/(2u) Y[[3]] +  Y[[4]]]
```

$\left\{1,u,v,\frac{1}{2}(u^2+v^2)\right\}$

```
In[33]:= Y={Y[[2]],Y[[3]],y4,Y[[1]]};
         Transpose[Y]
```

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ u & 1 & u & u \\ v-c & 0 & v & c+v \\ H-c\,v & u & \frac{1}{2}(u^2+v^2) & H+c\,v \end{pmatrix}$$

```
In[34]:= lambday = {ly[[2]],ly[[3]],ly[[4]],ly[[1]]}
```

$\{v-c, v, v, c+v\}$

```
In[35]:= Table[Simplify[B.Y[[i]] - lambday[[i]] Y[[i]] /. gamma -> 1 + c^2/(H-(u^2+v^2)/2)],{i,1,4}]
```

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

```
In[36]:= DiagonalMatrix[lambday]
```

$$\begin{pmatrix} v-c & 0 & 0 & 0 \\ 0 & v & 0 & 0 \\ 0 & 0 & v & 0 \\ 0 & 0 & 0 & c+v \end{pmatrix}$$

```
In[37]:=
```

1.2.3 Roe average

Applying dimensional splitting to the quasi-linear form of (1),

$$\boldsymbol{q}_t + \mathbf{A}(\boldsymbol{q})\boldsymbol{q}_x + \mathbf{B}(\boldsymbol{q})\boldsymbol{q}_y = 0, \tag{12}$$

leads to the problem

$$\boldsymbol{q}_t + \mathbf{A}(\boldsymbol{q})\boldsymbol{q}_x = 0, \boldsymbol{q}(t,x) = \boldsymbol{R}(\boldsymbol{Q}), \tag{13}$$

with $\boldsymbol{R}(\boldsymbol{Q})$ the reconstruction of $\boldsymbol{q}(t,x)$ from grid data $\boldsymbol{Q}$. For a piecewise constant reconstruction $\boldsymbol{R}_0$, the Riemann problem at interface $x_i = i h$ is stated as

$$\boldsymbol{q}_t + \mathbf{A}(\boldsymbol{q})\boldsymbol{q}_x = 0, \tag{14}$$

$$\boldsymbol{q}(t,x) = \boldsymbol{R}_0(\boldsymbol{Q}) = \begin{cases} \boldsymbol{Q}_{i-1} & x < x_i \\ \boldsymbol{Q}_i & x > x_i \end{cases}. \tag{15}$$

Problem (14-15) can be solved exactly for a constant Jacobian matrix $\mathbf{A}(\bar{\boldsymbol{q}}_i)$. The state $\bar{\boldsymbol{q}}_i(\boldsymbol{Q}_{i-1}, \boldsymbol{Q}_i)$ at which the Jacobian is evaluated should satisfy the conditions:

1. Discrete conservation

$$\delta\boldsymbol{f}_i = \boldsymbol{f}(\boldsymbol{Q}_i) - \boldsymbol{f}(\boldsymbol{Q}_{i-1}) = \mathbf{A}(\bar{\boldsymbol{q}}_i)(\boldsymbol{Q}_i - \boldsymbol{Q}_{i-1}) = \mathbf{A}(\bar{\boldsymbol{q}}_i)\,\delta\boldsymbol{Q}_i \tag{16}$$

2. Consistency

$$\mathbf{A}(\bar{\boldsymbol{q}}_i) \to \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}(\bar{\boldsymbol{q}}_i), \text{as } \boldsymbol{Q}_{i-1} \to \boldsymbol{Q}_i \tag{17}$$

3. Hyperbolicity, solution of the eigenproblem $\mathbf{A}(\bar{\boldsymbol{q}}_i)\mathbf{X}_i = \mathbf{X}_i\boldsymbol{\Lambda}_i$ gives
   a) Real eigenvalues, $\boldsymbol{\Lambda}_i \in \mathbb{R}^{m\times m}$, $\boldsymbol{q} \in \mathbb{R}^m$, $m=4$
   b) A complete system of eigenvectors, $\exists \mathbf{X}_i^{-1}$.

- An average state that satisfies the above conditions can be determined analytically for the Euler equations by re-expressing both $\boldsymbol{q}$ and $\boldsymbol{f}(\boldsymbol{q})$ in terms of the vector $\boldsymbol{z}$

$$\boldsymbol{z} = \begin{pmatrix} \sqrt{\rho} \\ \sqrt{\rho}\,u \\ \sqrt{\rho}\,v \\ (\varepsilon + p)/\sqrt{\rho} \end{pmatrix}, \boldsymbol{q}(\boldsymbol{z}) = \begin{pmatrix} z_1^2 \\ z_1\,z_2 \\ z_1\,z_3 \\ \frac{1}{\gamma}z_1 z_4 + \frac{\gamma-1}{2\gamma}(z_2^2 + z_3^2) \end{pmatrix}, \boldsymbol{f}(\boldsymbol{z}) = \begin{pmatrix} z_1\,z_2 \\ \frac{\gamma+1}{2\gamma}z_2^2 + \frac{\gamma-1}{\gamma}\left(z_1 z_4 - \frac{z_2^2}{2}\right) \\ z_2\,z_3 \\ z_2\,z_4 \end{pmatrix}.$$

```
In[37]:= sr=Sqrt[rho]
```

$$\sqrt{\rho}$$

```
In[38]:= z={sr, sr u, sr v, (epsilon+p)/sr}
```

$$\left\{\sqrt{\rho},\sqrt{\rho}\,u,\sqrt{\rho}\,v,\frac{\epsilon+p}{\sqrt{\rho}}\right\}$$

```
In[39]:= qP = q /. ToPrimitiveVariables
```

$$\left\{\rho,\rho\,u,\rho\,v,\frac{p}{\gamma-1}+\frac{1}{2}\,\rho\,(u^2+v^2)\right\}$$

```
In[40]:= zP = z /. ToPrimitiveVariables
```

$$\left\{\sqrt{\rho},\sqrt{\rho}\,u,\sqrt{\rho}\,v,\frac{\frac{p}{\gamma-1}+p+\frac{1}{2}\,\rho\,(u^2+v^2)}{\sqrt{\rho}}\right\}$$

```
In[41]:= qz = {z[[1]]^2, z[[1]] z[[2]], z[[1]] z[[3]],
                   z[[1]] z[[4]]/gamma + (gamma-1)/(2 gamma) (z[[2]]^2+z[[3]]^2)};
         qPz = Simplify[qz /. ToPrimitiveVariables]
```

$$\left\{\rho,\rho\,u,\rho\,v,\frac{p}{\gamma-1}+\frac{1}{2}\,\rho\,(u^2+v^2)\right\}$$

```
In[42]:= fP = Simplify[f /. ToPrimitiveVariables]
```

$$\left\{\rho\,u,p+\rho\,u^2,\rho\,u\,v,\frac{u\,(2\,\gamma\,p+\gamma\,\rho\,(u^2+v^2)-\rho\,(u^2+v^2))}{2\,(\gamma-1)}\right\}$$

```
In[43]:= fz={z[[1]] z[[2]],
              (gamma+1)/(2 gamma) z[[2]]^2 + (gamma-1)/gamma (z[[1]] z[[4]] - z[[3]]^2/2),
              z[[2]] z[[3]], z[[2]] z[[4]]};
         fPz = Simplify[fz /. ToPrimitiveVariables]
```

$$\left\{\rho\,u,p+\rho\,u^2,\rho\,u\,v,u\left(\frac{\gamma\,p}{\gamma-1}+\frac{1}{2}\,\rho\,(u^2+v^2)\right)\right\}$$

```
In[44]:= Simplify[fP - fPz]
```

$$\{0,0,0,0\}$$

```
In[45]:=
```

- An important observation is that both $\boldsymbol{q}(\boldsymbol{z})$ and $\boldsymbol{f}(\boldsymbol{z})$ are quadratic in $\boldsymbol{z}$, such that

$$\mathrm{d}\boldsymbol{q}(\boldsymbol{z})=\frac{\partial\boldsymbol{q}}{\partial\boldsymbol{z}}\mathrm{d}\boldsymbol{z}=\begin{pmatrix} 2z_1 & 0 & 0 & 0\\ z_2 & z_1 & 0 & 0\\ z_3 & 0 & z_1 & 0\\ \frac{1}{\gamma}z_4 & \frac{\gamma-1}{\gamma}z_2 & \frac{\gamma-1}{\gamma}z_3 & \frac{1}{\gamma}z_1\end{pmatrix}\begin{pmatrix}\mathrm{d}z_1\\ \mathrm{d}z_2\\ \mathrm{d}z_3\\ \mathrm{d}z_4\end{pmatrix}=\mathbf{L}(\boldsymbol{z})\,\mathrm{d}\boldsymbol{z},$$

$$\mathrm{d}\boldsymbol{f}(\boldsymbol{z})=\frac{\partial\boldsymbol{f}}{\partial\boldsymbol{z}}\mathrm{d}\boldsymbol{z}=\begin{pmatrix} z_2 & z_1 & 0 & 0\\ \frac{\gamma-1}{\gamma}z_4 & \frac{\gamma+1}{\gamma}z_2 & \frac{\gamma-1}{\gamma}z_3 & \frac{\gamma-1}{\gamma}z_1\\ 0 & z_3 & z_2 & 0\\ 0 & z_4 & 0 & z_2\end{pmatrix}\begin{pmatrix}\mathrm{d}z_1\\ \mathrm{d}z_2\\ \mathrm{d}z_3\\ \mathrm{d}z_4\end{pmatrix}=\mathbf{M}(\boldsymbol{z})\,\mathrm{d}\boldsymbol{z},$$

with $\mathbf{L}(\boldsymbol{z}),\mathbf{M}(\boldsymbol{z})$ linear in $\boldsymbol{z}$.

```
In[45]:= qz = {z1^2, z1 z2, z1 z3,
                   z1 z4/gamma + (gamma-1)/(2 gamma) (z2^2+z3^2)};
         L=Simplify[Grad[qz,{z1,z2,z3,z4}]]
```

$$\begin{pmatrix} 2\mathrm{z1} & 0 & 0 & 0\\ \mathrm{z2} & \mathrm{z1} & 0 & 0\\ \mathrm{z3} & 0 & \mathrm{z1} & 0\\ \frac{\mathrm{z4}}{\gamma} & \frac{(\gamma-1)\mathrm{z2}}{\gamma} & \frac{(\gamma-1)\mathrm{z3}}{\gamma} & \frac{\mathrm{z1}}{\gamma}\end{pmatrix}$$

```
In[46]:= fz={z1 z2,
              (gamma+1)/(2 gamma) z2^2 + (gamma-1)/gamma (z1 z4 - z3^2/2),
              z2 z3, z2 z4};
         M=Simplify[Grad[fz,{z1,z2,z3,z4}]]
```

$$\begin{pmatrix} \mathrm{z2} & \mathrm{z1} & 0 & 0 \\ \frac{(\gamma-1)\mathrm{z4}}{\gamma} & \left(1+\frac{1}{\gamma}\right)\mathrm{z2} & \left(\frac{1}{\gamma}-1\right)\mathrm{z3} & \frac{(\gamma-1)\mathrm{z1}}{\gamma} \\ 0 & \mathrm{z3} & \mathrm{z2} & 0 \\ 0 & \mathrm{z4} & 0 & \mathrm{z2} \end{pmatrix}$$

In[47]:=

Integration between the Riemann problem states gives

$$\delta \boldsymbol{Q}_i = \int_{\boldsymbol{Q}_{i-1}}^{\boldsymbol{Q}_i} \mathrm{d}\boldsymbol{q}(\boldsymbol{z}) = \int_{\boldsymbol{z}_{i-1}}^{\boldsymbol{z}_i} \mathbf{L}(\boldsymbol{z})\mathrm{d}\boldsymbol{z} = \mathbf{L}(\bar{z}_i)\delta\boldsymbol{Z}_i, \delta\boldsymbol{f}_i = \int_{\boldsymbol{f}(\boldsymbol{Q}_{i-1})}^{\boldsymbol{f}(\boldsymbol{Q}_i)} \mathrm{d}\boldsymbol{f}(\boldsymbol{z}) = \int_{\boldsymbol{z}_{i-1}}^{\boldsymbol{z}_i} \mathbf{M}(\boldsymbol{z})\mathrm{d}\boldsymbol{z} = \mathbf{M}(\bar{z}_i)\delta\boldsymbol{Z}_i,$$

with

$$\bar{\boldsymbol{Z}}_i = \frac{1}{2}(\boldsymbol{Z}_{i-1} + \boldsymbol{Z}_i),$$

since $\boldsymbol{L}(\boldsymbol{z}), \boldsymbol{M}(\boldsymbol{z})$ are linear. It results that

$$\delta\boldsymbol{f}_i = \mathbf{M}(\bar{\boldsymbol{Z}}_i)\delta\boldsymbol{Z}_i = \mathbf{M}(\bar{\boldsymbol{Z}}_i)\,\mathbf{L}^{-1}(\bar{\boldsymbol{Z}}_i)\delta\boldsymbol{Q}_i = \mathbf{A}(\boldsymbol{q}(\bar{\boldsymbol{Z}}_i))\,\delta\boldsymbol{Q}_i,$$

and the eigenvalue matrices evaluated at state $\bar{\boldsymbol{z}}_i$ satisfy the conditions of discrete conservation, consistency, and hyperbolicity.

2 Implementation

Define the BEARCLAW `problem` module.

2.1 Global definitions

A module is constructed with global definitions.

```
MODULE problem
  USE NodeInfoDef
  IMPLICIT NONE
  SAVE
  PRIVATE
  PUBLIC setprob,afterrun,qinit,b4step,setaux,src,physflux,problemBC, &
         afterstep,afterfixup,problemIO,problemBadCFL,problemErrFlag
  REAL (KIND=qPrec) :: gamma,gamma1,xQ,yQ
  REAL (KIND=qPrec), DIMENSION(4) :: pQ,rhoQ,uQ,vQ
  LOGICAL efix
  INTEGER, PARAMETER :: OK=0
  REAL (KIND=qPrec), PARAMETER :: zero=0.,half=0.5d0,one=1.d0,two=2.d0
  CONTAINS
```

Definition of problem module

Uses Bearclaw Info structure
All variables have to be declared
Save variables between calls
Internal variables cannot be seen
Public entry points

Define global variables

2.2 Problem definition: `setprob`

```
    SUBROUTINE setprob
      INTEGER i
      OPEN(UNIT=7,FILE='setprob.data',STATUS='old',FORM='formatted')
      READ(7,*) efix
      READ(7,*) gamma
      gamma1 = gamma - 1.d0
      READ(7,*) xQ,yQ
      DO i=1,4
        READ(7,*) pQ(i),rhoQ(i),uQ(i),vQ(i)
      END DO
      CLOSE(7)
    END SUBROUTINE setprob
```

Input parameters defining the problem from the `setprob.data` file.

2.3 Problem definition: `afterrun`

```
    SUBROUTINE afterrun
    END SUBROUTINE afterrun
```

Actions after run completion

2.4 Problem definition: `problemBC`

Define problem-specific BCs

```fortran
  SUBROUTINE problemBC(Info)
    TYPE (NodeInfo) :: Info
  END SUBROUTINE problemBC
```

2.5 Problem definition: `qinit`

Define initial condition

```fortran
  SUBROUTINE qinit(Info)
    TYPE (NodeInfo) :: Info
    INTEGER i,j,iq
    REAL (KIND=xPrec) :: x,y
    REAL (KIND=qPrec), POINTER, DIMENSION (:,:,:,:,:) :: q
```

Shorter variable name

```fortran
    q=>Info%q
    x=Info%Xlower(1)+Info%dX(1)/2.0
    DO i=1,Info%mX(1)
      y=Info%Xlower(2)+Info%dX(2)/2.0
      DO j=1,Info%mX(2)
        IF (x >= xQ .AND. y >= yQ) iq = 1
        IF (x <  xQ .AND. y >= yQ) iq = 2
        IF (x <  xQ .AND. y <  yQ) iq = 3
        IF (x >= xQ .AND. y <  yQ) iq = 4
        q(i,j,1,1,1) = rhoQ(iq)
        q(i,j,1,1,2) = rhoQ(iq)*uQ(iq)
        q(i,j,1,1,3) = rhoQ(iq)*vQ(iq)
        q(i,j,1,1,4) = pQ(iq)/gamma1 + &
                       half*rhoQ(iq)*(uQ(iq)**2 + vQ(iq)**2)
        y=y+Info%dX(2)
      END DO
      x=x+Info%dX(1)
    END DO
  END SUBROUTINE qinit
```

2.6 Problem definition: `b4step`

Actions before each time step

```fortran
  SUBROUTINE b4step(Info)
    TYPE (NodeInfo) :: Info
  END SUBROUTINE b4step
```

2.7 Problem definition: `afterstep`

Actions after each time step

```fortran
  SUBROUTINE afterstep(Info)
    TYPE (NodeInfo) :: Info
  END SUBROUTINE afterstep
```

2.8 Problem definition: `afterfixup`

Actions after coarse grid update from fine grid values

```fortran
  SUBROUTINE afterfixup(Info)
    TYPE (NodeInfo) :: Info
  END SUBROUTINE afterfixup
```

2.9 Problem definition: `problemIO`

Application-specific I/O

```
SUBROUTINE problemIO(nframe,tnow,IOrequest,Info,qmax,qmin)
  INTEGER :: nframe,IOrequest; REAL (KIND=qPrec) :: tnow
  TYPE (NodeInfo), OPTIONAL :: Info
  REAL (KIND=qPrec), OPTIONAL, DIMENSION(:) :: qmax,qmin
  INTEGER, PARAMETER :: UserBeforeGridIO=-1, UserAfterGridIO=-2
  INTEGER, PARAMETER :: UserIOMinMax=0,UserBeforeOutput=1, &
                        UserAfterOutput=2
END SUBROUTINE problemIO
```

2.10 Problem definition: **setaux**

Define initial auxilliary variables

```
SUBROUTINE setaux(Info)
  TYPE (NodeInfo) :: Info
END SUBROUTINE setaux
```

2.11 Problem definition: **src**

Define source term

```
SUBROUTINE src(Info)
  TYPE (NodeInfo) :: Info
END SUBROUTINE src
```

2.12 Problem definition: **problemBadCFL**

Actions if CFL >1

```
SUBROUTINE problemBadCFL(Info)
  TYPE (NodeInfo) :: Info
END SUBROUTINE problemBadCFL
```

2.13 Problem definition: **problemErrFlag**

Application-specific refinement criteria

```
SUBROUTINE problemErrFlag(Info,CoarseInfo)
  TYPE (NodeInfo) :: Info        ! Current grid
  TYPE (NodeInfo) :: CoarseInfo  ! Coarsened version of current grid
  Info%ErrorFlags=1
END SUBROUTINE problemErrFlag
```

2.14 Problem definition: **physflux**

2.14.1 Variable declarations

Physical fluxes, solve Riemann prob.

```
SUBROUTINE physflux(ixy,indx,irequest,Info,q,f,s,A)
  INTEGER ixy
  INTEGER indx(MaxDims)
  INTEGER irequest
  TYPE (NodeInfo) :: Info
  REAL (KIND=qPrec), POINTER, DIMENSION (:,:) :: q,f,s
  REAL (KIND=qPrec), POINTER, DIMENSION (:,:,:) :: A

  INTEGER, PARAMETER :: Mmq=4,Mmbc=8,Mmx=10000, &
                        imn=1-Mmbc,imx=Mmx+Mmbc
  INTEGER mx,mbc,mQ,iQ,mWaves,iW,mu,mv
  INTEGER, POINTER, DIMENSION(:) :: iC,iE,iL,iR
  REAL (KIND=qPrec), POINTER, DIMENSION (:,:) :: Apdq,Amdq,speed
  REAL (KIND=qPrec), POINTER, DIMENSION (:,:) :: Asdq,BmAsdq,BpAsdq
  REAL (KIND=qPrec), POINTER, DIMENSION (:,:,:) :: wave
  REAL (KIND=qPrec), DIMENSION (imn:imx,Mmq), SAVE :: dQ,dW
  REAL (KIND=qPrec), DIMENSION (imn:imx), SAVE :: p,pR
  REAL (KIND=qPrec), DIMENSION (imn:imx,Mmq), SAVE :: z,zR
  REAL (KIND=qPrec), DIMENSION (imn:imx,Mmq+1), SAVE, TARGET :: xR
  REAL (KIND=qPrec), POINTER, DIMENSION (:) :: cR,uR,vR,HR,KR
```

Direction along which to solve
Indices of current slice
Request code
Current grid Info structure
One-dimensional slice quantities

Local variables:
max q components, BC cells, cells
min, max cell index ranges
indices
index arrays
normal fluctuations
transverse fluctuations
waves
conservative, characteristic jumps
cell-center, Roe average pressure
cell-centered $\boldsymbol{z}$, average $\bar{\boldsymbol{z}}$
Roe-average $\bar{\boldsymbol{x}} = (\bar{c}, \bar{u}, \bar{v}, \bar{H}, \bar{K})$
$(\bar{c}, \bar{u}, \bar{v}, \bar{H}, \bar{K})$

2.14.2 Variable definitions

```
mx=Info%mXnow; mbc=Info%mbc; mQ=Info%NrVars; mWaves=Info%mwaves
IF ((mx>Mmx) .OR. (mbc>Mmbc) .OR. (mq>Mmq)) THEN
  PRINT *,'Increase buffer sizes in physflux'; STOP
ENDIF
iE=>Info%I1D        ! Interior faces: 2-mbc <= iEdge <= mx+mbc
iC=>Info%I1Dcells   ! Cell centers:   1-mbc <= iCell <= mx+mbc
iL=>Info%I1Dleft    ! Left faces:     1-mbc <= iLft  <= mx+mbc-1
iR=>Info%I1Dright   ! Right faces:    2-mbc <= iRgt  <= mx+mbc
Apdq=>Info%Apdq; Amdq=>Info%Amdq
speed=>Info%speed; wave=>Info%wave
Asdq=>Info%Asdq; BmAsdq=>Info%BmAsdq; BpAsdq=>Info%BpAsdq
SELECT CASE (ixy)
  CASE(1)
    mu=2; mv=3
  CASE(2)
    mu=3; mv=2
END SELECT
```

Get information from Info structure
Check whether current 1D slice fits in pre-allocated buffers

Associate local pointers to components of the Info structure

1D-slice fluctuations $(\mathcal{A}\Delta q)^{\pm}$
1D-slice λ,$\mathcal{W}$
1D-slice $(\mathcal{A}_s\Delta q)$, $(\mathcal{B}\mathcal{A}_s\Delta q)^{\pm}$

Select components depending on direction of this slice

2.14.3 Request responses

```
SELECT CASE (irequest)
```

Identify request

Initialize Determine the Roe-average state at edges by computing:

1. $\boldsymbol{z}_i$ at cell centers

$$\boldsymbol{z}_i = \begin{pmatrix} \sqrt{\rho} \\ \sqrt{\rho}\, u \\ \sqrt{\rho}\, v \\ (\varepsilon + p)/\sqrt{\rho} \end{pmatrix}_i = \begin{pmatrix} \sqrt{\rho} \\ \sqrt{\rho}\, u \\ \sqrt{\rho}\, v \\ \sqrt{\rho}\, H \end{pmatrix}_i,$$

2. $\bar{\boldsymbol{z}}_i = \frac{1}{2}(\boldsymbol{z}_{i-1} + \boldsymbol{z}_i)$ at cell edges
3. the Roe-averaged quantities $\bar{\boldsymbol{x}}_i = (\bar{c}_i, \bar{u}_i, \bar{v}_i, \bar{H}_i, \bar{K}_i)$ arising in the eigenvector matrices $\mathbf{X}, \mathbf{Y}$ using the relations

$$z_4 = \sqrt{\rho}\left(E + \frac{p}{\rho}\right) = \sqrt{\rho}\, H = \sqrt{\rho}(h + K) = \sqrt{\rho}\left(\frac{c^2}{\gamma - 1} + K\right), K = \frac{u^2 + v^2}{2}$$

```
  CASE (Initialize)
    z(iC,1) = SQRT(q(iC,1))
    z(iC,2)= q(iC,2)/z(iC,1); z(iC,3)= q(iC,3)/z(iC,1)
    p(iC)=gamma1*(q(iC,4)-half*(q(iC,2)**2+q(iC,3)**2)/q(iC,1))
    z(iC,4) = (q(iC,4)+p(iC))/z(iC,1)
    zR(iE,1:mQ) = half*(z(iL,1:mQ)+z(iR,1:mQ))
    xR(iE,2) = zR(iE,2)/zR(iE,1); xR(iE,3) = zR(iE,3)/zR(iE,1)
    xR(iE,4) = zR(iE,4)/zR(iE,1)
    xR(iE,5) = half*(xR(iE,2)**2+xR(iE,3)**2)
    xR(iE,1) = gamma1*(xR(iE,4)-xR(iE,5))
    IF (MINVAL(xR(iE,1))<zero) THEN
      PRINT *,'Error:Negative Roe-average sound speed in physflux'
      STOP
    ENDIF
    xR(iE,1) = SQRT(xR(iE,1))
```

Initialize computations on this 1D slice

$z_{1i} = \sqrt{\rho_i}$
$(z_{2i}, z_{3i}) = \sqrt{\rho_i}\,(u_i, v_i) = (l_i, m_i)/\sqrt{\rho_i}$
$p_i = (\gamma - 1)\big(\varepsilon_i - \frac{1}{2}(l_i^2 + m_i^2)/\rho_i\big)$
$z_{4i} = (\varepsilon_i + p_i)/\sqrt{\rho_i}$
$\bar{\boldsymbol{z}}_i = \frac{1}{2}(\boldsymbol{z}_{i-1} + \boldsymbol{z}_i)$
$(\bar{x}_2, \bar{x}_3) = (\bar{u}, \bar{v}) = (\bar{z}_2, \bar{z}_3)/\bar{z}_1$
$\bar{x}_4 = \bar{H} = \bar{z}_4/\bar{z}_1$
$\bar{K} = \frac{1}{2}(\bar{u}^2 + \bar{v}^2)$
$\bar{c}^2 = (\gamma - 1)(\bar{H} - \bar{K})$

$c = \sqrt{c^2}$

RequestNormalWaves Define the Roe-average eigenmodes at edges

$$\mathbf{X} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ u - c & 0 & u & u + c \\ v & 1 & v & v \\ H - c\,u & v & K & H + c\,u \end{pmatrix}, \mathbf{Y} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ u & 1 & u & u \\ v - c & 0 & v & v + c \\ H - c\,v & u & K & H + c\,v \end{pmatrix},$$

with $\boldsymbol{x} = (c, u, v, H)$, and solve the normal Riemann problem at cell edges.

Permute 1D slice components

$i_{\text{xy}} = 1$ $\;i_{\text{xy}} = 2$

```
    CASE (RequestNormalWaves)
      ! ixy=1: mu=2, mv=3; ixy=2: mu=3, mv=2
      cR=>xR(:,1); uR=>xR(:,mu); vR=>xR(:,mv)
      HR=>xR(:,4); KR=>xR(:,5)
```

Backward acoustic wave

$$\begin{array}{ll} \lambda_1 = u - c & \lambda_1 = v - c \\ (\mathbf{X}_1)_1 = 1 & (\mathbf{Y}_1)_1 = 1 \\ (\mathbf{X}_1)_2 = u - c & (\mathbf{Y}_1)_2 = u \\ (\mathbf{X}_1)_3 = v & (\mathbf{Y}_1)_3 = v - c \\ (\mathbf{X}_1)_4 = H - cu & (\mathbf{Y}_1)_4 = H - cv \end{array}$$

```
      speed(iE  ,1) = uR(iE) - cR(iE)
      wave(iE,1 ,1) = one
      wave(iE,mu,1) = speed(iE,1)
      wave(iE,mv,1) = vR(iE)
      wave(iE,4 ,1) = HR(iE) - cR(iE)*uR(iE)
```

Shear wave

$$\begin{array}{ll} \lambda_2 = u & \lambda_2 = v \\ (\mathbf{X}_2)_1 = 0 & (\mathbf{Y}_2)_1 = 0 \\ (\mathbf{X}_2)_2 = 0 & (\mathbf{Y}_2)_2 = 1 \\ (\mathbf{X}_2)_3 = 1 & (\mathbf{Y}_2)_3 = 0 \\ (\mathbf{X}_2)_4 = v & (\mathbf{Y}_2)_4 = u \end{array}$$

```
      speed(iE  ,2) = uR(iE)
      wave(iE,1 ,2) = zero
      wave(iE,mu,2) = zero
      wave(iE,mv,2) = one
      wave(iE,4 ,2) = vR(iE)
```

Entropy wave

$$\begin{array}{ll} \lambda_3 = u & \lambda_3 = v \\ (\mathbf{X}_3)_1 = 1 & (\mathbf{Y}_3)_1 = 1 \\ (\mathbf{X}_3)_2 = u & (\mathbf{Y}_3)_2 = u \\ (\mathbf{X}_3)_3 = v & (\mathbf{Y}_3)_3 = v \\ (\mathbf{X}_3)_4 = K & (\mathbf{Y}_3)_4 = K \end{array}$$

```
      speed(iE  ,3) = uR(iE)
      wave(iE,1 ,3) = one
      wave(iE,mu,3) = uR(iE)
      wave(iE,mv,3) = vR(iE)
      wave(iE,4 ,3) = KR(iE)
```

Forward acoustic wave

$$\begin{array}{ll} \lambda_1 = u + c & \lambda_1 = v + c \\ (\mathbf{X}_4)_1 = 1 & (\mathbf{Y}_4)_1 = 1 \\ (\mathbf{X}_4)_2 = u + c & (\mathbf{Y}_4)_2 = u \\ (\mathbf{X}_4)_3 = v & (\mathbf{Y}_4)_3 = v + c \\ (\mathbf{X}_4)_4 = H + cu & (\mathbf{Y}_4)_4 = H + cv \end{array}$$

```
      speed(iE  ,4) = uR(iE) + cR(iE)
      wave(iE,1 ,4) = one
      wave(iE,mu,4) = speed(iE,4)
      wave(iE,mv,4) = vR(iE)
      wave(iE,4 ,4) = HR(iE) + cR(iE)*vR(iE)
```

Decompose the jump $\delta \boldsymbol{Q}$ at each edge between two cells onto the Roe-average eigenbasis.

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ u - c & 0 & u & c + u \\ v & 1 & v & v \\ H - c\,u & v & \frac{1}{2}(u^2 + v^2) & H + c\,u \end{pmatrix} \delta \boldsymbol{w}_x = \delta \boldsymbol{q}, \begin{pmatrix} 1 & 0 & 1 & 1 \\ u & 1 & u & u \\ v - c & 0 & v & c + v \\ H - c\,v & u & \frac{1}{2}(u^2 + v^2) & H + c\,v \end{pmatrix} \delta \boldsymbol{w}_y = \delta \boldsymbol{q},$$

- Solution of $\mathbf{X}\,\delta \boldsymbol{w}_x = \delta \boldsymbol{q}$,

$$\delta \boldsymbol{w}_x = \begin{pmatrix} \frac{\gamma - 1}{2c^2}\left(\frac{u^2 + v^2}{2}\,\delta Q_1 - u\,\delta Q_2 - v\,\delta Q_3 + \delta Q_4\right) - \frac{1}{2c}(\delta Q_2 - u\,\delta Q_1) \\ \delta Q_3 - v\,\delta Q_1 \\ \delta Q_1 - \frac{\gamma - 1}{c^2}\left(\frac{u^2 + v^2}{2}\,\delta Q_1 - u\,\delta Q_2 - v\,\delta Q_3 + \delta Q_4\right) \\ \frac{\gamma - 1}{2c^2}\left(\frac{u^2 + v^2}{2}\,\delta Q_1 - u\,\delta Q_2 - v\,\delta Q_3 + \delta Q_4\right) + \frac{1}{2c}(\delta Q_2 - u\,\delta Q_1) \end{pmatrix} = \begin{pmatrix} A - B \\ \delta Q_3 - v\,\delta Q_1 \\ \delta Q_1 - 2A \\ A + B \end{pmatrix},$$

with notation $A = \frac{\gamma - 1}{2c^2}\left(\frac{u^2 + v^2}{2}\,\delta Q_1 - u\,\delta Q_2 - v\,\delta Q_3 + \delta Q_4\right)$, $B = \frac{1}{2c}(\delta Q_2 - u\,\delta Q_1)$.

- Solution of $\mathbf{Y}\,\delta \boldsymbol{w}_y = \delta \boldsymbol{q}$,

$$\delta \boldsymbol{w}_y = \begin{pmatrix} \frac{\gamma - 1}{2c^2}\left(\frac{u^2 + v^2}{2}\,\delta Q_1 - u\,\delta Q_2 - v\,\delta Q_3 + \delta Q_4\right) - \frac{1}{2c}(\delta Q_3 - v\,\delta Q_1) \\ \delta Q_2 - u\,\delta Q_1 \\ \delta Q_1 - \frac{\gamma - 1}{c^2}\left(\frac{u^2 + v^2}{2}\,\delta Q_1 - u\,\delta Q_2 - v\,\delta Q_3 + \delta Q_4\right) \\ \frac{\gamma - 1}{2c^2}\left(\frac{u^2 + v^2}{2}\,\delta Q_1 - u\,\delta Q_2 - v\,\delta Q_3 + \delta Q_4\right) + \frac{1}{2c}(\delta Q_3 - v\,\delta Q_1) \end{pmatrix} = \begin{pmatrix} A - B \\ \delta Q_2 - u\,\delta Q_1 \\ \delta Q_1 - 2A \\ A + B \end{pmatrix},$$

with notation $A = \frac{\gamma-1}{2c^2}\left(\frac{u^2+v^2}{2}\delta Q_1 - u\delta Q_2 - v\delta Q_3 + \delta Q_4\right), B = \frac{1}{2c}(\delta Q_3 - v\delta Q_1)$.

```
dQ(iE,1:mQ) = Q(iR,1:mQ) - Q(iL,1:mQ)
dW(iE,3) = (half*gamma1/cR(iE)**2) * &
          (KR(iE)*dQ(iE,1)-uR(iE)*dQ(iE,mu)-vR(iE)*dQ(iE,mv)+dQ(iE,4))
dW(iE,4) = (dQ(iE,mu)-uR(iE)*dQ(iE,1))/(2*cR(iE))
dW(iE,1) = dW(iE,3) + dW(iE,4)
dW(iE,4) = dW(iE,3) - dW(iE,1)
dW(iE,2) = dQ(iE,mv)-vR(iE)*dQ(iE,1)
dW(iE,3) = dQ(iE,1) - two*dW(iE,3)
```

Jump in $\boldsymbol{Q}$ at interfaces
Store A in $(\delta\boldsymbol{w})_3$

Store B in $(\delta\boldsymbol{w})_4$
$(\delta\boldsymbol{w})_1 = A - B$
$(\delta\boldsymbol{w})_4 = A + B$
$(\delta\boldsymbol{w})_2 = \delta Q_3 - v\delta Q_1$
$(\delta\boldsymbol{w})_3 = \delta Q_1 - 2A$

```
Apdq=0.; Amdq=0.
DO iW=1,mWaves
  DO iQ=1,mQ
    WHERE (speed(iE,iW)<0)
      Amdq(iE,iQ)=Amdq(iE,iQ)-speed(iE,iW)*dW(iE,iW)*wave(iE,iQ,iW)
    ELSEWHERE
      Apdq(iE,iQ)=Apdq(iE,iQ)+speed(iE,iW)*dW(iE,iW)*wave(iE,iQ,iW)
    END WHERE
  END DO
END DO
```

Define $(\mathcal{A}\Delta q)^+, (\mathcal{A}\Delta q)^-$
Initialize

RequestTransverseWaves Define the transverse Roe-averaged eigenmodes at cell edges.

```
CASE (RequestTransverseWaves)
  ! ixy=1: mu=2, mv=3; ixy=2: mu=3, mv=2
  cR=>xR(:,1); uR=>xR(:,mu); vR=>xR(:,mv)
  HR=>xR(:,4); KR=>xR(:,5)
```

Permute 1D slice components
$i_{xy} = 1$ $i_{xy} = 2$

```
  speed(iE  ,1) = vR(iE) - cR(iE)
  wave(iE,1 ,1) = one
  wave(iE,mu,1) = uR(iE)
  wave(iE,mv,1) = speed(iE,1)
  wave(iE,4 ,1) = HR(iE) - cR(iE)*vR(iE)
```

Backward acoustic wave

$\lambda_1 = v - c$	$\lambda_1 = u - c$
$(\mathbf{Y}_1)_1 = 1$	$(\mathbf{X}_1)_1 = 1$
$(\mathbf{Y}_1)_2 = u$	$(\mathbf{X}_1)_2 = u - c$
$(\mathbf{Y}_1)_3 = v - c$	$(\mathbf{X}_1)_3 = v$
$(\mathbf{Y}_1)_4 = H - cv$	$(\mathbf{X}_1)_4 = H - cu$

```
  speed(iE  ,2) = vR(iE)
  wave(iE,1 ,2) = zero
  wave(iE,mu,2) = one
  wave(iE,mv,2) = zero
  wave(iE,4 ,2) = uR(iE)
```

Shear wave

$\lambda_2 = v$	$\lambda_2 = u$
$(\mathbf{Y}_2)_1 = 0$	$(\mathbf{X}_2)_1 = 0$
$(\mathbf{Y}_2)_2 = 1$	$(\mathbf{X}_2)_2 = 0$
$(\mathbf{Y}_2)_3 = 0$	$(\mathbf{X}_2)_3 = 1$
$(\mathbf{Y}_2)_4 = u$	$(\mathbf{X}_2)_4 = v$

```
  speed(iE  ,3) = vR(iE)
  wave(iE,1 ,3) = one
  wave(iE,mu,3) = uR(iE)
  wave(iE,mv,3) = vR(iE)
  wave(iE,4 ,3) = KR(iE)
```

Entropy wave

$\lambda_3 = v$	$\lambda_3 = u$
$(\mathbf{Y}_3)_1 = 1$	$(\mathbf{X}_3)_1 = 1$
$(\mathbf{Y}_3)_2 = u$	$(\mathbf{X}_3)_2 = u$
$(\mathbf{Y}_3)_3 = v$	$(\mathbf{X}_3)_3 = v$
$(\mathbf{Y}_3)_4 = K$	$(\mathbf{X}_3)_4 = K$

```
  speed(iE  ,4) = vR(iE) + cR(iE)
  wave(iE,1 ,4) = one
  wave(iE,mu,4) = uR(iE)
  wave(iE,mv,4) = speed(iE,4)
  wave(iE,4 ,4) = HR(iE) + cR(iE)*vR(iE)
```

Backward acoustic wave

$\lambda_1 = v + c$	$\lambda_1 = u + c$
$(\mathbf{Y}_4)_1 = 1$	$(\mathbf{X}_4)_1 = 1$
$(\mathbf{Y}_4)_2 = u$	$(\mathbf{X}_4)_2 = u + c$
$(\mathbf{Y}_4)_3 = v + c$	$(\mathbf{X}_4)_3 = v$
$(\mathbf{Y}_4)_4 = H + cv$	$(\mathbf{X}_4)_4 = H + cu$

Decompose the normal fluctuations $(\mathcal{A}_s\Delta q)$ along the transverse eigenmodes, and compute $(\mathcal{B}\mathcal{A}_s\Delta q)^{\pm}$

```
          dQ(iE,1:mQ) = Asdq(iE,1:mQ)
          dW(iE,3) = (half*gamma1/cR(iE)**2) * &
                     (KR(iE)*dQ(iE,1)-uR(iE)*dQ(iE,mu)-vR(iE)*dQ(iE,mv)+dQ(iE,4))
          dW(iE,4) = (dQ(iE,mv)-vR(iE)*dQ(iE,1))/(2*cR(iE))
          dW(iE,1) = dW(iE,3) + dW(iE,4)
          dW(iE,4) = dW(iE,3) - dW(iE,1)
          dW(iE,2) = dQ(iE,mv)-uR(iE)*dQ(iE,1)
          dW(iE,3) = dQ(iE,1) - two*dW(iE,3)
```

Jump in $\boldsymbol{Q}$ at interfaces
Store A in $(\delta\boldsymbol{w})_3$

Store B in $(\delta\boldsymbol{w})_4$
$(\delta\boldsymbol{w})_1 = A - B$
$(\delta\boldsymbol{w})_4 = A + B$
$(\delta\boldsymbol{w})_2 = \delta Q_3 - v\delta Q_1$
$(\delta\boldsymbol{w})_3 = \delta Q_1 - 2A$

```
          BpAsdq=0.; BmAsdq=0.
          DO iW=1,mWaves
            DO iQ=1,mQ
              WHERE (speed(iE,iW)<0)
                BmAsdq(iE,iQ)=BmAsdq(iE,iQ)-speed(iE,iW)*dW(iE,iW)*wave(iE,iQ,iW)
              ELSEWHERE
                BpAsdq(iE,iQ)=BpAsdq(iE,iQ)+speed(iE,iW)*dW(iE,iW)*wave(iE,iQ,iW)
              END WHERE
            END DO
          END DO
```

$(\mathcal{B}\mathcal{A}_s\Delta q)^+, (\mathcal{B}\mathcal{A}_s\Delta q)^-$
Initialize

2.15 Close problem module

```
      END SELECT
    END SUBROUTINE physflux
END MODULE problem
```

3 Simulations

3.1 Data files

- bear.data - (update)

```
!==================================================================================
! BEARCLAW bear.data input file. Global parameters valid for all root-level grids.
! Application: 2D Euler quadrants problem. (c) Sorin Mitran 2008, mitran@unc.edu
!==================================================================================
!:RunFlags:!    | Variable        | Description
!==================================================================================
F     0         Restart, Frame    Resume from checkpoint data dump
F               LevelEqSets       Solve different equations  on grid levels
F               LevelMethods      Apply different algorithms on grid levels
F               SaveAtFixedTimes  F=maintain CFL, T=save data at desired times
F               MaintainAuxArrays Treat aux similarly to q in MPI runs
F               InitialAMRonly    Generate initial AMR structure and stop
T               OutputStyleParams Outputstyle line contains additional formatting
!==================================================================================
!:RunParameters:!
!==================================================================================
1               nRootGrids        Number of root-level grids
3               MaxLevels         Maximum number of grid refinement levels
2 2 2 2 2       CoarsenRatio      ... of child grid to obtain parent spacing
4               MinimumGridPoints ... along one dimension
1               TimeStepMethod    0 fixed dt, 1 variable dt
0.d0            t0                initial time (if not Restart)
1.00d0          tfinal            final time
4     0.5       MaxCFLRetry, rCFL Try reducing CFL by this ratio this many times
3               OutputStyle       1 AMRCLAW, 2 TECPLOT, 3 HDF, 9 GnuPlot, 11 VTK
100             OutputFrames      Number of data checkpoints
T T T T T T     OutputLevel       Level output flag
!==================================================================================
```

- grid.data - (update)

```
!==================================================================================
! BEARCLAW grid.data input file. Parameters specific to root-level grid.
!==================================================================================
!:GridParameters:! Variable        Description
!==================================================================================
2                  nDim            Grid spatial dimensions
4                  MaxLevel        Max grid refinement levels for this grid
100                mx              Cells in x direction
100                my              Cells in y direction
1 100              mGlobal(1)      Global index extents of this grid (x-direction)
1 100              mGlobal(2)      Global index extents of this grid (y-direction)
0.0d0              xlower          Left edge of computational domain
1.0d0              xupper          Right edge of computational domain
0.0d0              ylower          Bottom edge of computational domain
1.0d0              yupper          Top edge of computational domain
2                  mbc             Number of ghost cells at each boundary
1                  mthbc(1)        Left   boundary condition code
1                  mthbc(2)        Right  boundary condition code
1                  mthbc(3)        Bottom boundary condition code
1                  mthbc(4)        Top    boundary condition code
0.45d-2            dtv(1)          Initial time step (constant dt TimeStepMethod=0)
1.0d99             dtv(2)          Max allowable time step
1.00d0             cflv(1)         Max allowable Courant number
0.88d0             cflv(2)         Desired Courant number
1.0                cflv(3)         Time step relaxation parameter
!==================================================================================
!:MultiphysicsParameters:! - one value if LevelEqSets==F else (>=MaxLevel) values
!==================================================================================
! NrVars           = Number of primary field variables
4
! Output style parameters
0 1 1 0
! nEquationSet     = Equation set for these fields
1
! maux             = Number of auxilliary fields
0
!==================================================================================
!:GridRefinementParameters:! - (>=MaxLevel) values for each parameter
!==================================================================================
! qTolerance       = Field variable tolerances that trigger refinement
1.0e-2 1.0e-2 1.0e-2 1.0e-2 1.0e-2 1.0e-2
! xTolerance       = Spatial tolerances that trigger refinement
5.0e-2 5.0e-2 5.0e-2 5.0e-2 5.0e-2 5.0e-2
! iBuffer          = Size of buffer arround area flagged for refinement
2  2  2  2  2  2
! DesiredFillRatios= New subgrids should have this percentage of flagged cells
0.85 0.85 0.85 0.85 0.85 0.85
! InterpOpt        = Interpolation method used to obtain child data from parent
! (0=minmod, 1=constant, 2=centered, 3=left, 4=right, 5=piecewise)
1  1  1  1  1  1
! ErrorFlagOpt     = Error flag method
! (0=child, 1=parent, 2=apriori 3=user)
0  0  0  0  0  0
!==================================================================================
!:NumericalSchemeParameters:! one value if LevelMethods==F else (>=MaxLevel) values
!==================================================================================
0                  method(1)   = (reserved)
2                  method(2)   = convergence order
2                  method(3)   = transverse convergence order
0                  method(4)   = verbosity of wavebear output
```

```
0                  method(5)   = source term splitting
0                  method(6)   = 0 split q differences, 1 split flux differences
0                  method(7)   = radius of slab around current 1D array of cells

4                  mwaves      = number of waves in each Riemann solution
3 3 3 3            mthlim(mw)  = limiter for each wave  (mw=1,mwaves)
!==================================================================================
!:UserRootLevelParameters:!
!==================================================================================
! (none for this application)
!==================================================================================
```

3.2 Results

```
   Shell session inside TeXmacs pid = 22857
Shell] make clean > /dev/null; make outclean > /dev/null; make distclean > /dev/null
Shell] exo-open --launch TerminalEmulator make &
Shell] exo-open --launch TerminalEmulator xbear &
Shell] exo-open --launch TerminalEmulator make anim.gif SCRIPT=quadrants &
   [2] 8802
   [1]   Done                    exo-open --launch TerminalEmulator make anim.gif
Shell] animate anim.gif & > /dev/null
   [1] 9348
Shell] make frames

Shell]
```

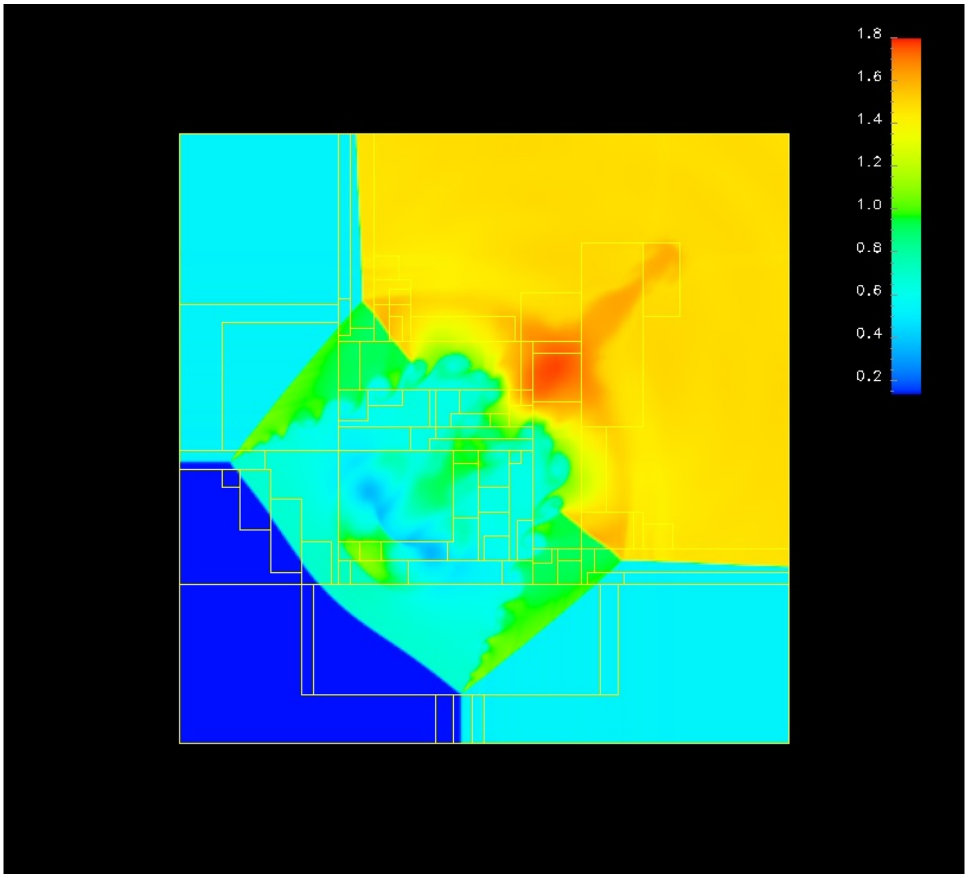

Figure 1. Quadrants solution.