SIMULACIÓN NUMÉRICA POR ORDENADOR EN ASTROFÍSICA
(CURSO DE DOCTORADO 2003–04)

PRÁCTICA 3. *Aspectos básicos de los métodos en diferencias finitas aplicados a los sistemas hiperbólicos de leyes de conservación. Métodos de alta resolución de captura de choques*

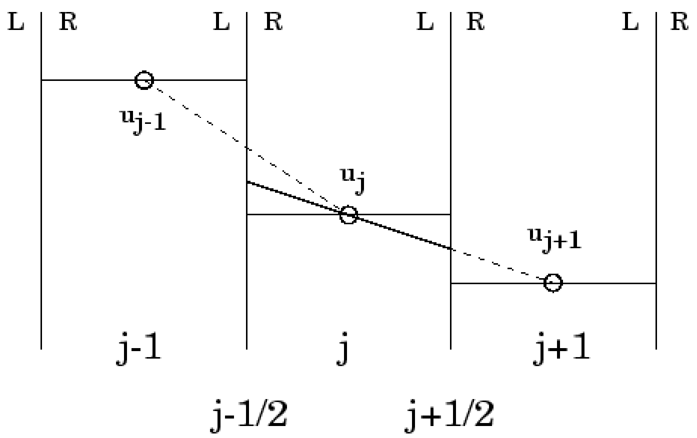
1.– Introducción

- Los métodos de alta resolución de captura de choques tienen forma conservativa.
- Son algoritmos de alto orden que tienen variación total estable. La variación total de una solución en $t = t^n$, $\mathrm{TV}(u^n)$, se define como

$$\mathrm{TV}(u^n) = \sum_{j=0}^{+\infty} |u^n_{j+1} - u^n_j| \,. \tag{1}$$

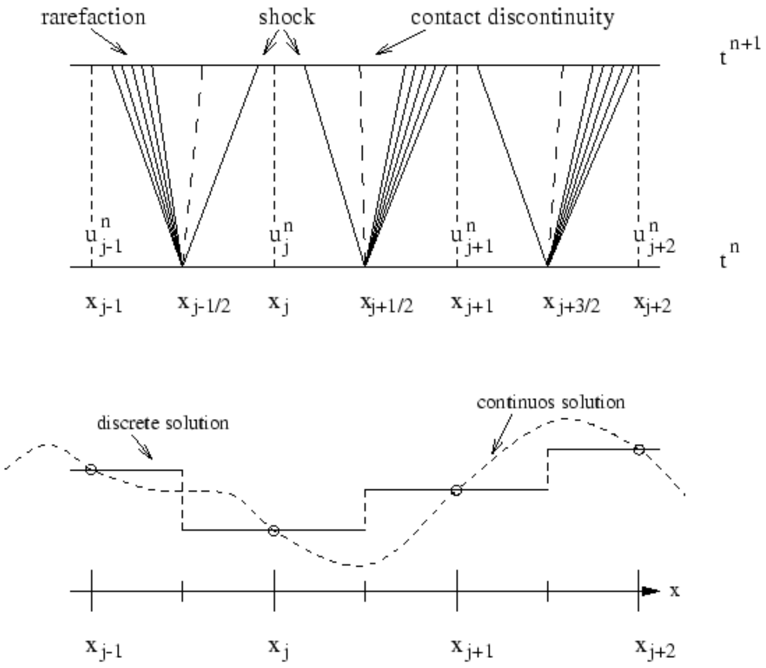
Un esquema numérico se dice que es TV-estable si $\mathrm{TV}(u^n)$ está acotada para cualquier Δt en cualquier instante y para cualquier dato inicial. El interés en los métodos TV-estables radica en el hecho de que para esquemas numéricos en forma conservativa con flujos numéricos consistentes, la estabilidad TV es una condición suficiente de convergencia.

El alto orden se consigue mediante funciones polinómicas monótonas conservativas que interpolan las soluciones aproximadas o los flujos numéricos en las celdas numéricas. Algoritmos de reconstrucción bien conocidos son el MINMOD (segundo orden), PPM (tercer orden) o los métodos ENO (que mantienen la variación total acotada).



Reconstrucción MINMOD: en cada celda numérica se efectúa una interpolación lineal con la menor de las pendientes con las celdas contiguas.

- La disipación numérica necesaria en las proximidades de los choques se consigue mediante la resolución de problemas de Riemann exactos o aproximados entre las celdas numéricas contiguas (métodos tipo Godunov) o mediante la adición de términos disipativos conservativos a métodos en diferencias finitas estándar.



Godunov's scheme: local solutions of Riemann problems

- La precisión de alto orden en el tiempo se consigue mediante métodos semidiscretos tipo Runge-Kutta, que permiten preservar la propiedad TVD del esquema completo de forma natural.

- Referencias:

R.J. LeVeque, *Numerical Methods for Conservation Laws*, Birkhäuser, 1992. Parte II

E.F. Toro, *Riemann solvers and Numerical Methods for Fluid Dynamics*, Springer, 1997. Capítulo 4

C.-W. Shu, S. Osher, *Efficient Implementation of Essentially Non-Oscillatory Shock-Capturing Schemes*, JCP, **77**, 439 (1988)

2.– Considera el programa `bugod.f` que integra la ecuación de Burgers (ver práctica 2) usando el método de Godunov (primer orden).

0. Haz un diagrama de flujo del programa

1. Algoritmo de segundo orden espacial. Programa una subrutina que

   – Calcule las pendientes entre los valores de `U` de celdas contiguas.

```
        DO I = 0, CELLS+2
           S(I) = (U(I) - U(I-1))/DX
        END DO
```

   – Construya valores de `U` a ambos extremos de cada celda numérica, `UM`, `UP`, usando las pendientes anteriores convenientemente monotonizadas para preservar la condición TVD.

   Limitador de pendiente MINMOD:

```
        DO I = 0, CELLS+1
           IF ((S(I).EQ.0.0).OR.(S(I+1).EQ.0.0)) THEN
              SL(I) = 0.0
           ELSE
              SL(I) = 0.5*(S(I)/ABS(S(I)) + S(I+1)/ABS(S(I+1)))
     &                   *AMIN1(ABS(S(I)),ABS(S(I+1)))
           END IF
           UM(I) = U(I) - SL(I)*DX/2.0
           UP(I) = U(I) + SL(I)*DX/2.0
        END DO
```

   Limitador de pendiente SUPERBEE:

```
        DO I = 0, CELLS+1
           IF ((S(I).EQ.0.0).OR.(S(I+1).EQ.0.0)) THEN
              SL(I) = 0.0
            ELSE IF (ABS(S(I)).GE.ABS(S(I+1))) THEN
              SL(I) =  0.5*(S(I)/ABS(S(I)) + S(I+1)/ABS(S(I+1)))
     &                      *AMIN1(ABS(S(I)),ABS(2*S(I+1)))
           ELSE
              SL(I) =  0.5*(S(I)/ABS(S(I)) + S(I+1)/ABS(S(I+1)))
                            *AMIN1(ABS(2*S(I)),ABS(S(I+1)))
           END IF
           UM(I) = U(I) - SL(I)*DX/2.0
           UP(I) = U(I) + SL(I)*DX/2.0
        END DO
```

Limitador de pendiente VANLEER:

```
      DO I = 0, CELLS+1
         IF ((S(I).EQ.0.0).AND.(S(I+1).EQ.0.0)) THEN
            SL(I) = 0.0
         ELSE
            SL(I) = 2.D0*S(I)*S(I+1)/(S(I) + S(I+1))
         END IF
         UM(I) = U(I) - SL(I)*DX/2.D0
         UP(I) = U(I) + SL(I)*DX/2.D0
      END DO
```

– Inserta la subrutina que acabas de programar (`minmod.f`, `superbee.f`, `vanleer.f`) en `bugod.f`, junto con las declaraciones de variables, actualizaciones de los bloques `COMMON`, redefiniciones de dimensiones y modificaciones de las condiciones de contorno necesarias.

– Modifica la subrutina `fluxes.f` para usar los valores de `U` reconstruidos (`UM`, `UP`) en la definición de los problemas de Riemann locales:

```
      DO 10 I = 0, CELLS
         UL = UP(I)
         UR = UM(I+1)
         ...
   10 CONTINUE
```

2. Algoritmo de segundo orden temporal. El algoritmo básico del avance temporal es el del esquema explícito conservativo de primer orden (ver práctica 2)

$$u_j^{n+1} = u_j^n - \frac{\Delta t}{\Delta x}(\hat{f}_{j+1/2} - \hat{f}_{j-1/2}) \quad (\equiv u_j^n + \Delta t L(u^n)).$$

Implementa el siguiente algoritmo de Runge-Kutta en el avance temporal de `bugod.f` para conseguir segundo orden de precisión temporal:

$$u_j^{n+1/2} = u_j^n + \Delta t L(u^n)$$
$$u_j^{n+1} = \frac{1}{2}u_j^n + \frac{1}{2}u_j^{n+1/2} + \frac{1}{2}\Delta t L(u^{n+1/2})$$

3. Utiliza el programa para integrar la ecuación de Burgers con dato inicial $u_0(x) = \sin(2\pi x)$ en el intervalo $[0,1]$. Puedes usar `gnuplot` (ver práctica 1) para visualizar los resultados.

3.– Resumen del programa `bugod.f`

`program bugod`: Resuelve la ecuación de Burgers sin viscosidad usando el esquema de Godunov de primer orden, explícito, con el resolvedor de Riemann exacto.

`subroutine reader`: lee los parámetros iniciales del fichero `bugod.ini`.

`subroutine initia`: especifica el dato inicial.

`subroutine bconci`: establece las condiciones de contorno.

`subroutine cflcon`: calcula el paso de tiempo de la siguiente iteración temporal de acuerdo con la condición de Courant.

`subroutine fluxes`: calcula los flujos numéricos a partir de la solución exacta de los problemas de Riemann en cada interfase.

`subroutine riemann`: resuelve el problema de Riemann para la ecuación de Burgers de forma exacta.

`subroutine update`: obtiene la solución al final del paso temporal usando el algoritmo conservativo explícito de primer orden.

`subroutine output`: guarda la solución en el fichero `bugod.out`.