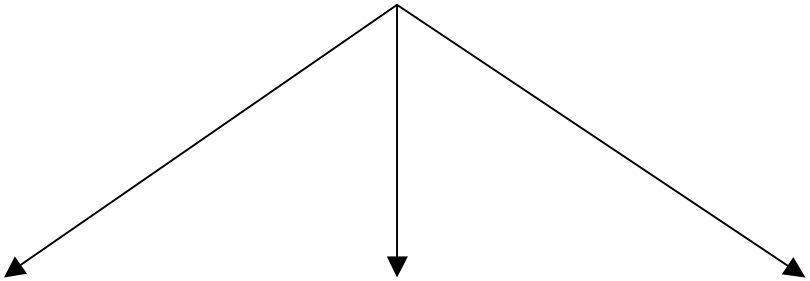
Simulación numérica de altas prestaciones

Sergio Hoyas

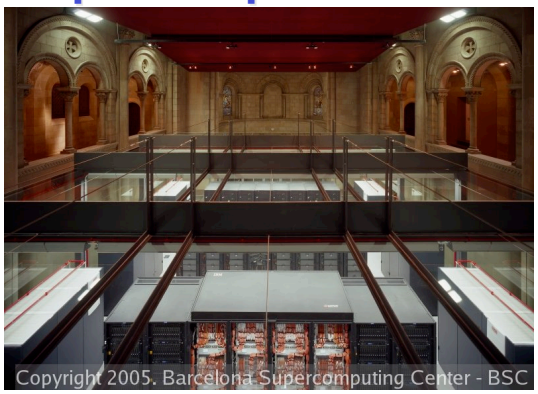
Departamento de informática, Universidad de Valencia
Computational fluid dynamics lab, UPM

Arquitecturas actuales de cálculo

Arquitecturas actuales



Pc y Servidores ⇄ **Grid computing** ⇄ **Clusters y supercomputadores**



www.pic.es

www.bsc.es

Arquitecturas actuales de cálculo

Servidores

Objetivos Pruebas, problemas pequeños, post – proceso.

Ventajas: Precio, fácil administración.

Inconvenientes Tamaño problema limitado, Espacio y refrigeración.

Evolución Más procesadores, cálculo/memoria >>1.

Recomendaciones:

- Hardware: Mejor Intel que AMD. El principal motivo el compilador de Intel.
- Sistema operativo: Ubuntu o Debian. Nunca Suse.

Arquitecturas actuales de cálculo

Grid y Supercomputadores

Objetivos Barridos parámetricos, grandes problemas.

Ventajas: Administración, grandes recursos, herramientas.

Inconvenientes Acceso a recursos, administrativas.

Evolución

- Se prevee un aumento significativo del ratio de procesadores por nodo.
- Máquinas de 2000 procesadores son ya comunes.

Software numérico

Lenguajes de alto nivel:

Matlab Fortran C

Software numérico

Lenguajes de alto nivel:

Matlab Fortran C

Ventajas:

Intuitivo, fácil de programar y debuggear
Imprescindible para la visualización de resultados
Gran cantidad de bibliotecas y rutinas de fácil acceso

Inconvenientes:

Interpretado (no compilado) -> lento
Es fácil de programar, pero difícil de programar bien.
No tiene todavía implementaciones paralelas.
La implementación GNU (Octave) está lejos de Matlab

Software numérico

Lenguajes de alto nivel:

Matlab **Fortran** C

Ventajas:

Potente y versatil.
Gran cantidad de bibliotecas y rutinas, sobre todo básicas.
Varias implementaciones paralelas: MPI, OpenMP.
La mayoría de los compiladores son gratuitos bajo Linux.

Inconvenientes:

No hay herramientas de visualización de resultados.
Los debbugeadores son poco intuitivos. No existen en paralelo
El linkado con librerías y optimización puede ser difícil

Software numérico

Lenguajes de alto nivel:

Matlab Fortran C

Ventajas:

El más potente de los tres. Se puede hacer cualquier cosa.
Admite varias paralelizaciones: MPI, OpenMP y Posix
Enorme cantidad de recursos de libre acceso en la red

Inconvenientes:

Muy díficil de programar bien para un no-informático.
Gran cantidad de herramientas no intuitivas.

Claves

Legilibidad

Los códigos deben estar llenos de comentarios

¡Velocidad vs Legibilidad!

Nombre de rutinas y variables: significativo y consecuente

Reglas generales

Basicas:

- Orden: columnas, filas.
- **Matlab:**
  - Ordenes vectoriales.
  - Predimensionalización
- **Fortran:**
  - Trabajar punto a punto.
  - Implicit none

Avanzadas:

- Memoria cache
- Minimización de llamadas a funciones

Herramientas

Profilers

- **Matlab:**

`Profile on; My_code; profile viewer.`

`Compilador: mcc -m My_code`

- **Fortran**

`ifort -pg my_code.f90 ; ./a.out; gprof gmon.out a.out`

Bibliotecas

- **FFT:**

`FFTW www.fftw.org`

- **Álgebra lineal básica**

`BLAS (Basic Linear Algebra Subprograms)` http://www.netlib.org/blas/

- **Álgebra lineal**

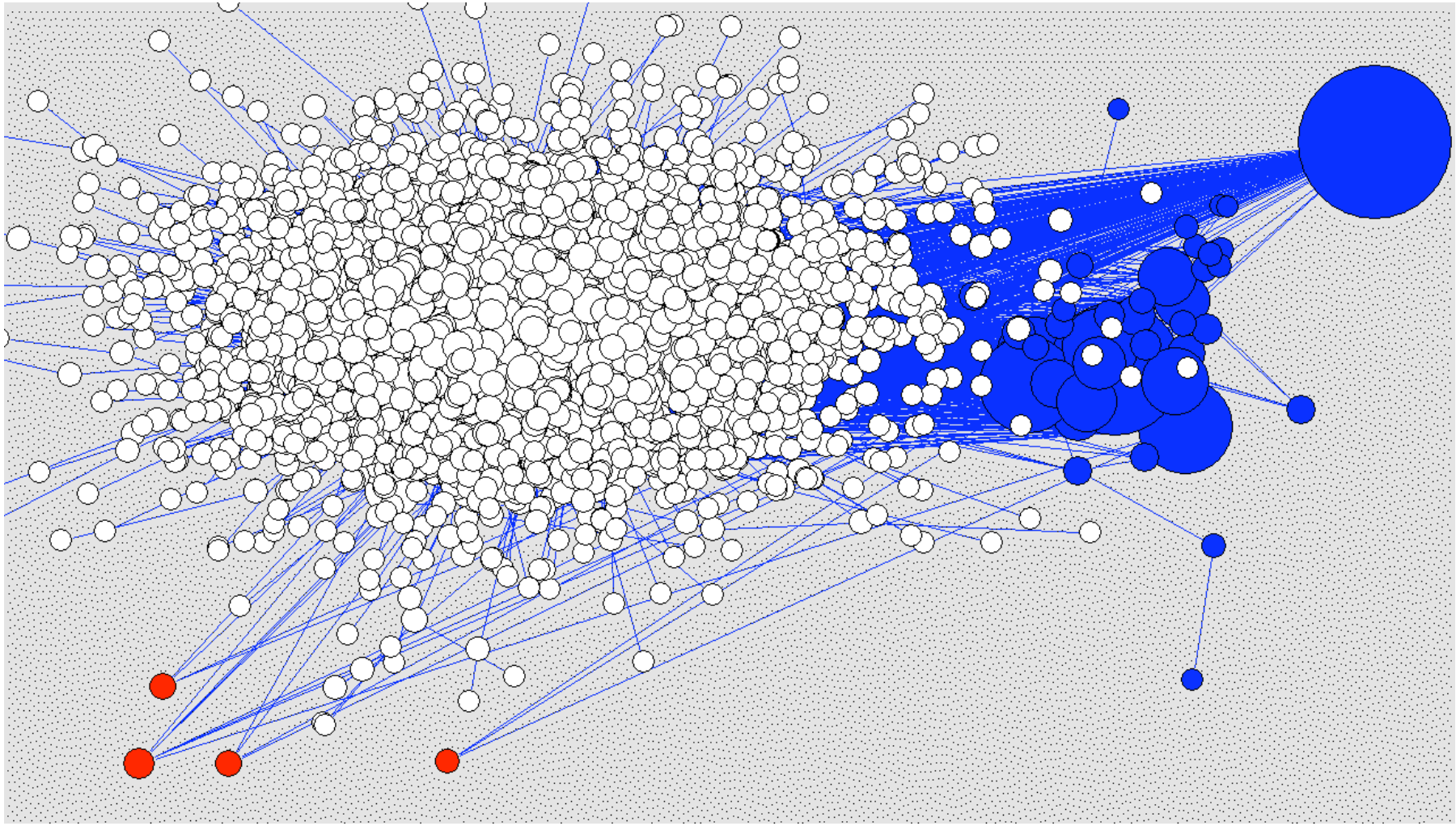
`Lapack (Linear Algebra PACKage)` http://www.netlib.org/lapack/

- **Varios**

`Numerical Recipes`

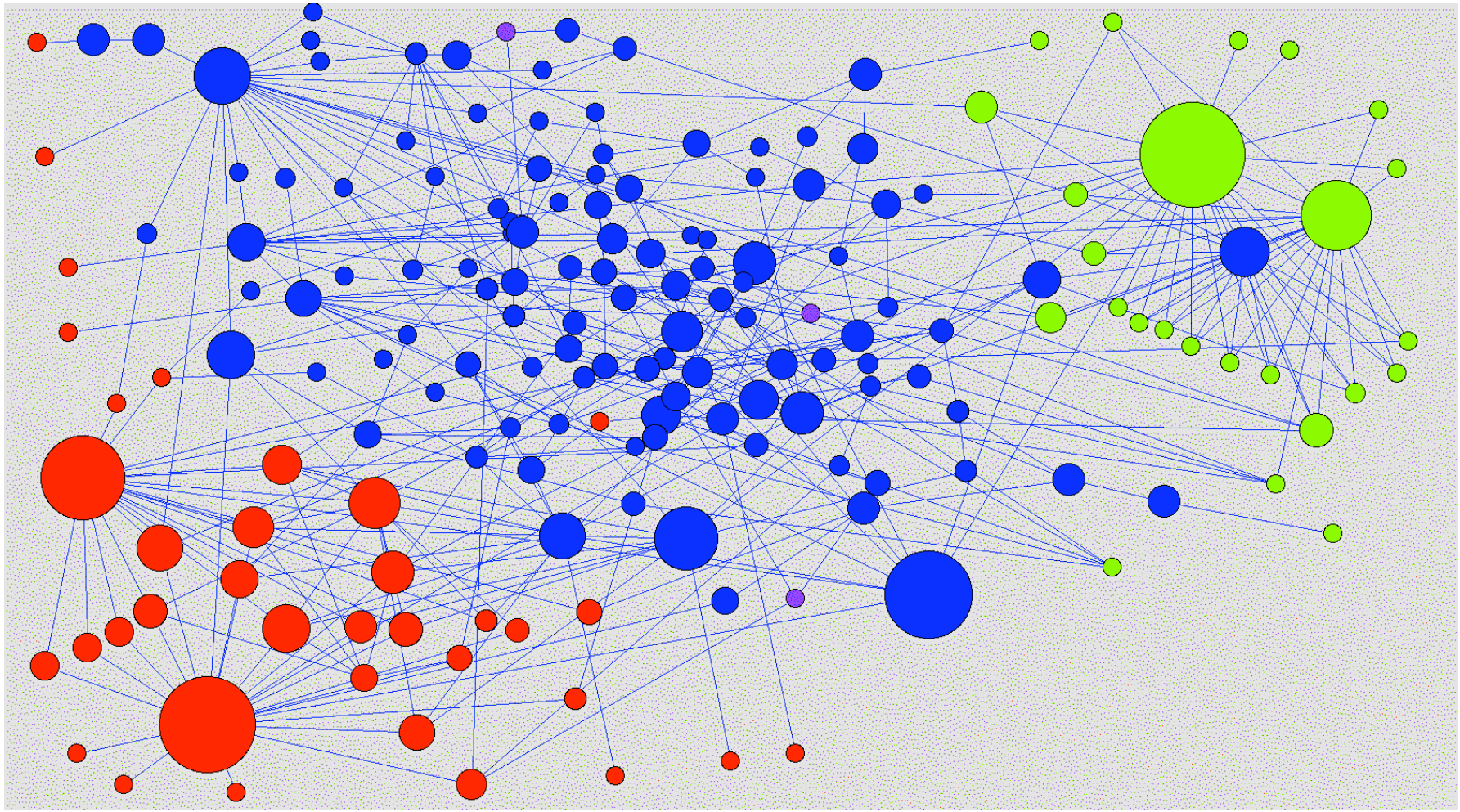
Teoría de grafos

Grafo mínimo entre grupos de proteinas



Teoría de grafos

Grafo mínimo entre grupos de proteinas



Teoría de grafos

Implementaciones

Algoritmos: Dijkstra (distancia), Breadth-first (caminos), Eliminación (H&M)

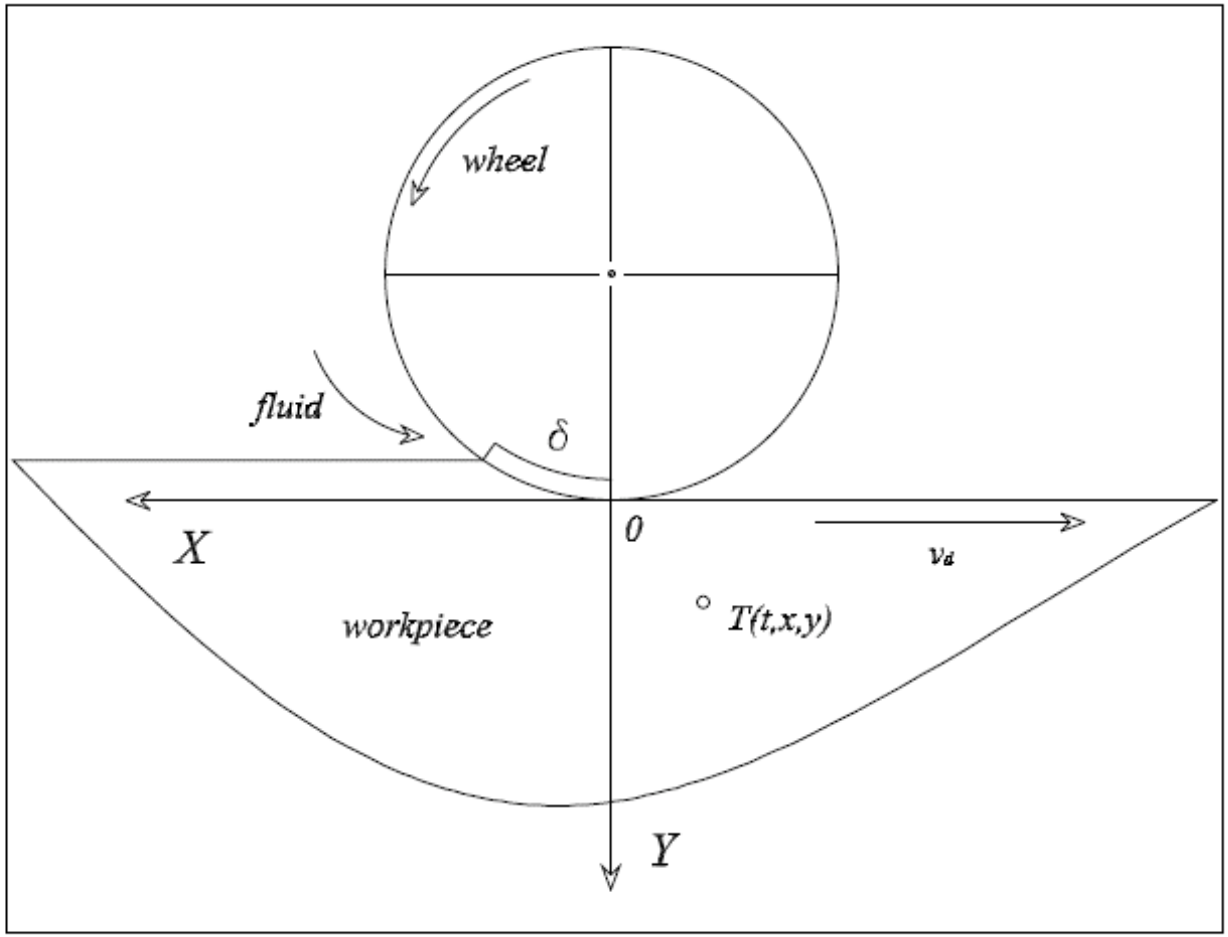
Matlab: Tiempos: Cálculo de caminos 1mn. Eliminación 2mn
Primeros ejemplos y problemas pequeños.
Principal problema: requerimientos de memoria.

Fortran: Tiempos: Cálculo de caminos 5-6s. Eliminación 2.7s
Rapidez permite análisis más detallados.
Paralización: OpenMP para aumentar el rendimiento de memoria.
MPI para aumentar el número de procesadores. (Grid)
MPI para repartir datos entre procesadores. (Cluster)

Hoyas & Marín, 2007

Grinding

Transmisión del calor



Ecuaciones

Ecuaciones y solución general

$$\left.\begin{aligned}
&\partial_t T(t,x,y) = a\left(\partial_{xx}T(t,x,y) + \partial_{yy}T(t,x,y)\right) - v_d\partial_x T(t,x,y),\\
&\lambda\partial_y T(t,x,0) = b(t,x)T(t,x,0) + d(t,x),\ -\infty < x < \infty,\ t \geq 0,\\
&T(0,x,y) = 0,\ -\infty < x < \infty,\ y \geq 0,
\end{aligned}\right\}$$

$$T(t,x,y) = \frac{1}{4\pi}\int_0^t \left[\int_{-\infty}^{+\infty} s^{-1} e^{-\frac{(x'-x-v_d s)^2+y^2}{4as}}\right.$$

$$\left.\times\left(\left[\left(\frac{y}{2as} - \frac{b(t-s,x')}{\lambda}\right)T(t-s,x',0) - \frac{d(t-s,x')}{\lambda}\right]\right)dx'\right]ds,$$

Kuratov, Perez et al 06

Ecuaciones

Aproximación en la superficie

$$T^{(0)}(t,x,0) = \frac{qa}{2\lambda\sqrt{\pi}} \int_0^t \frac{f_p(t-s)}{2\sqrt{as}} \left(\operatorname{erf}\left(\frac{\delta - x - v_d s}{2\sqrt{as}} \right) + \operatorname{erf}\left(\frac{x + v_d s}{2\sqrt{as}} \right) \right) ds.$$

Idea del algoritmo:

Dividir el intervalo $(0,t_{end})$ en N partes equiespaciadas
Usar un segundo mallado para cada subintervalo con (m+1) puntos
Integral de $(0,t_1)$ -> m+1 puntos, $(0,t_2)$ -> 2m+1 puntos.
Con la información necesaria para hallar el valor en t_{end} calculamos el valor

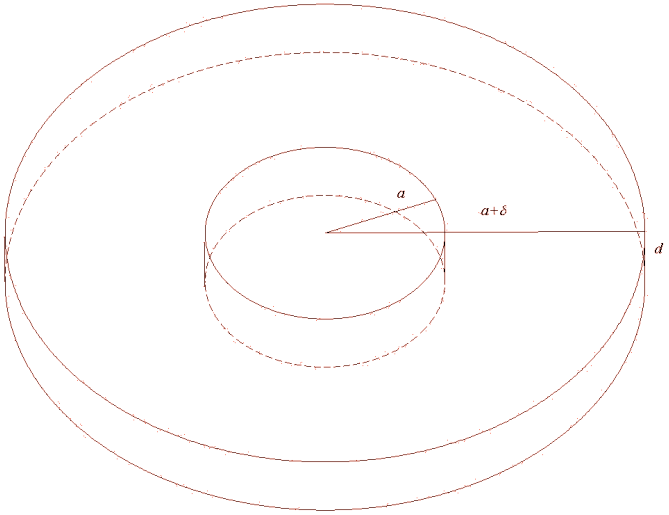
en N puntos

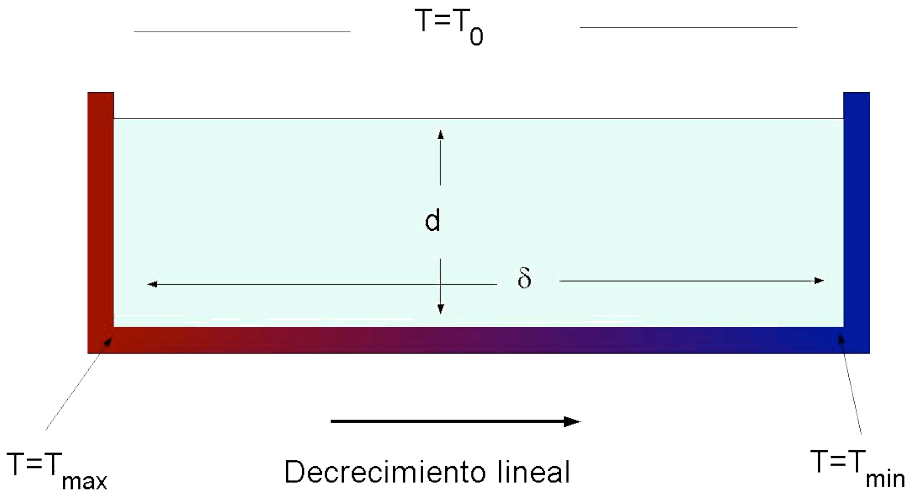
Claves del código:

Escrito para Matlab. Operaciones poco naturales
Parte más cara: Evaluación de las funciones de error

Pérez, Hoyas et al, 07

Cilindro

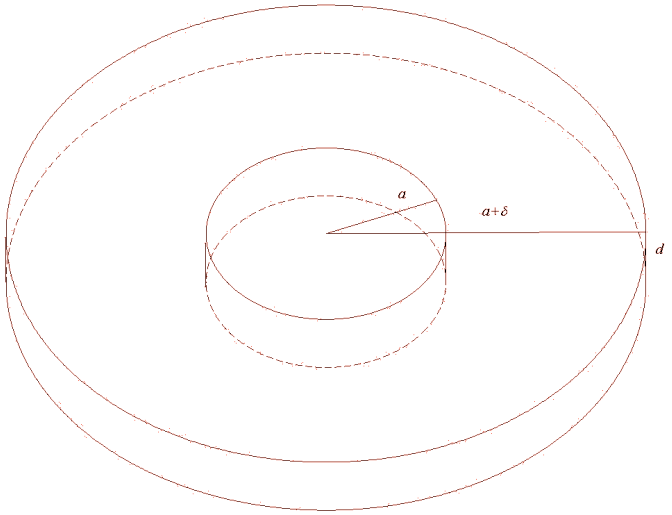


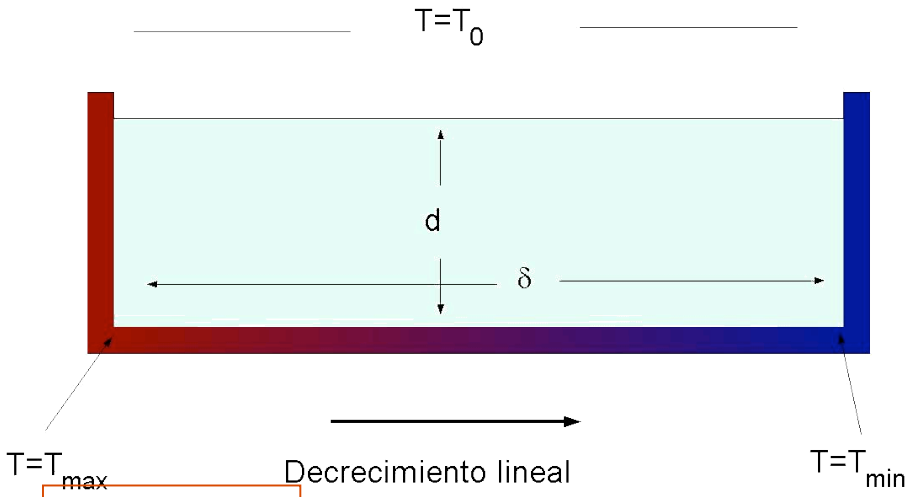


$$
\begin{aligned}
-\Delta_B u_r + G^2 u_r + A\partial_r p &= 0, \\
-\Delta_B u_z + 2\partial_z p - R\Theta &= -b, \\
-\Delta_B \Theta + u_r A\partial_r \Theta + 2u_z \partial_z \Theta &= 0, \\
G u_r + A\partial_r u_r + 2\partial_z u_z &= 0,
\end{aligned}
$$

Hoyas, Herrero & Mancho, 02

Cilindro





$$
\begin{aligned}
-\Delta_B u_r + G^2 u_r + A\partial_r p &= 0, \\
-\Delta_B u_z + 2\partial_z p - R\Theta &= -b, \\
-\Delta_B \Theta + u_r A\partial_r \Theta + 2u_z \partial_z \Theta &= 0, \\
G u_r + A\partial_r u_r + 2\partial_z u_z &= 0,
\end{aligned}
$$

Discretización

Chebyshev-Gauss-Lobatto

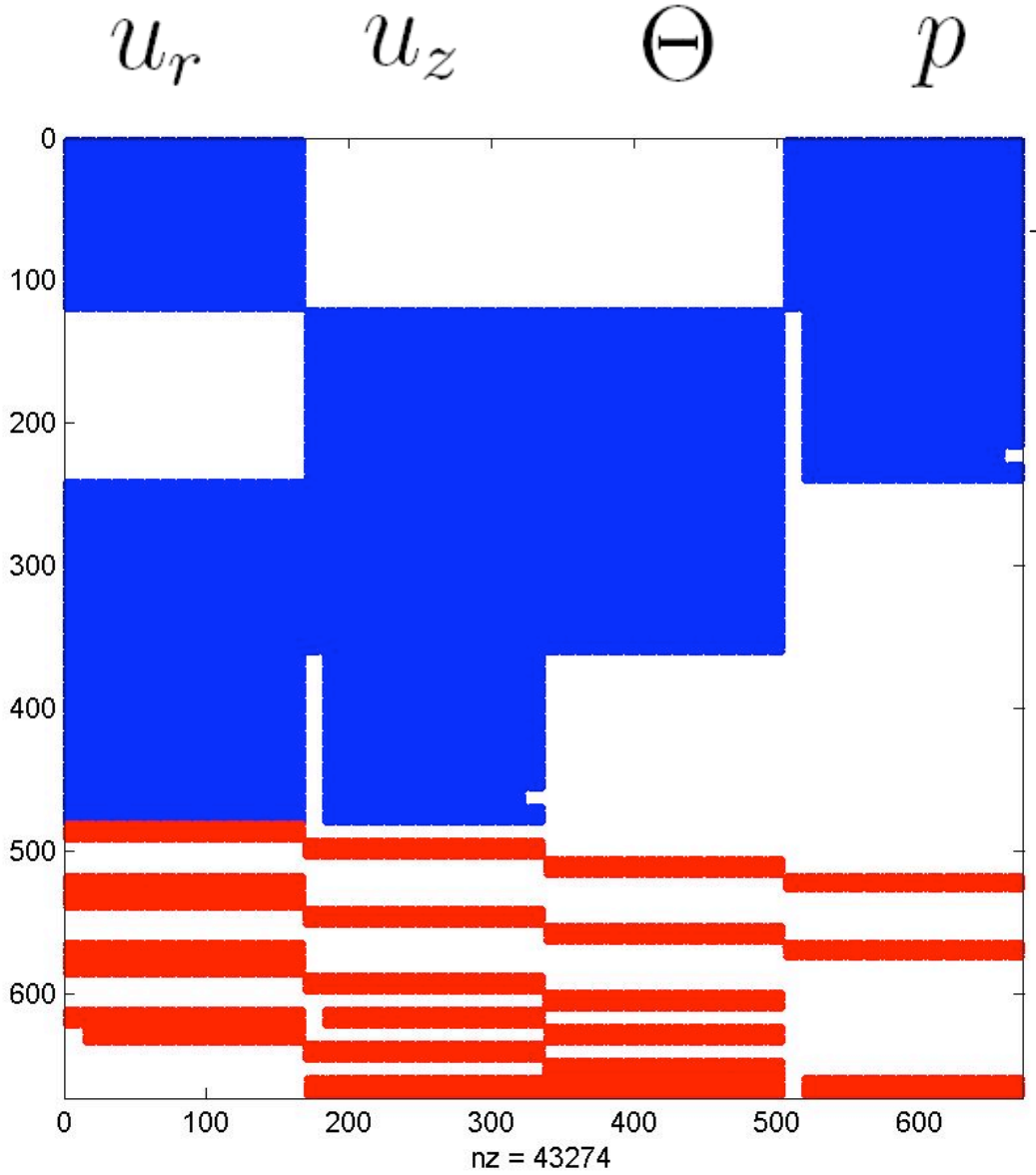
$$u_r(r,z) \simeq \sum_{n=0}^{N}\sum_{m=0}^{M} a_{nm} T_n(r)\, T_m(z), \quad \Theta(r,z) \simeq \sum_{n=0}^{N}\sum_{m=0}^{M} c_{nm} T_n(r)\, T_m(z),$$

$$u_z(r,z) \simeq \sum_{n=0}^{N}\sum_{m=0}^{M} b_{nm} T_n(r)\, T_m(z), \quad p(r,z) \simeq \sum_{n=0}^{N}\sum_{m=0}^{M} d_{nm} T_n(r)\, T_m(z),$$

$$r_i = -\cos\frac{\pi i}{N}, \quad z_j = -\cos\frac{\pi j}{M}.$$

$$Ax = B,\ x = (a_{n,m}, b_{n,m}, c_{n,m}, d_{n,m})$$

Cilindro



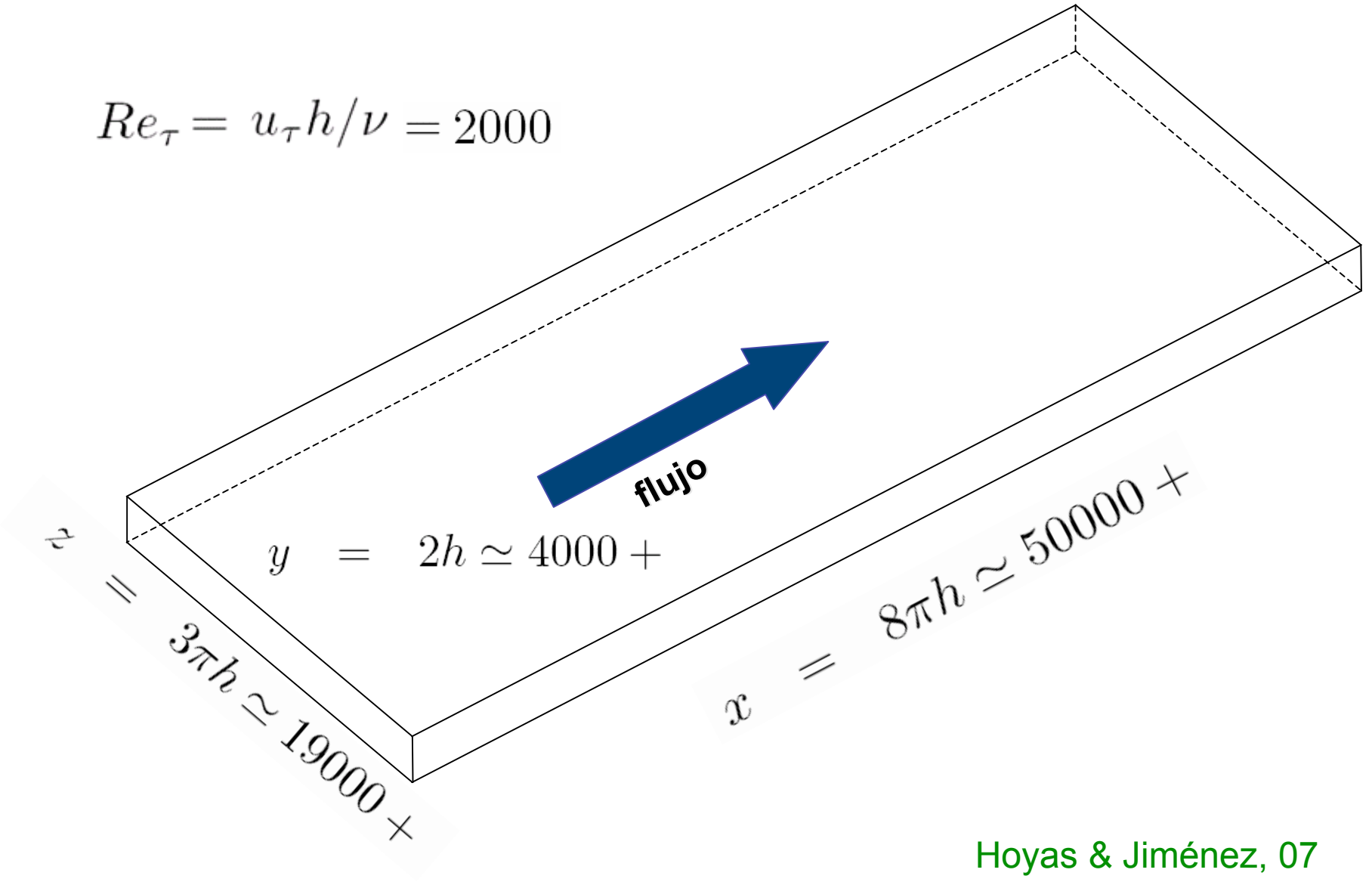
$r_i \; z_j$

Ecuaciones

Condiciones de contorno

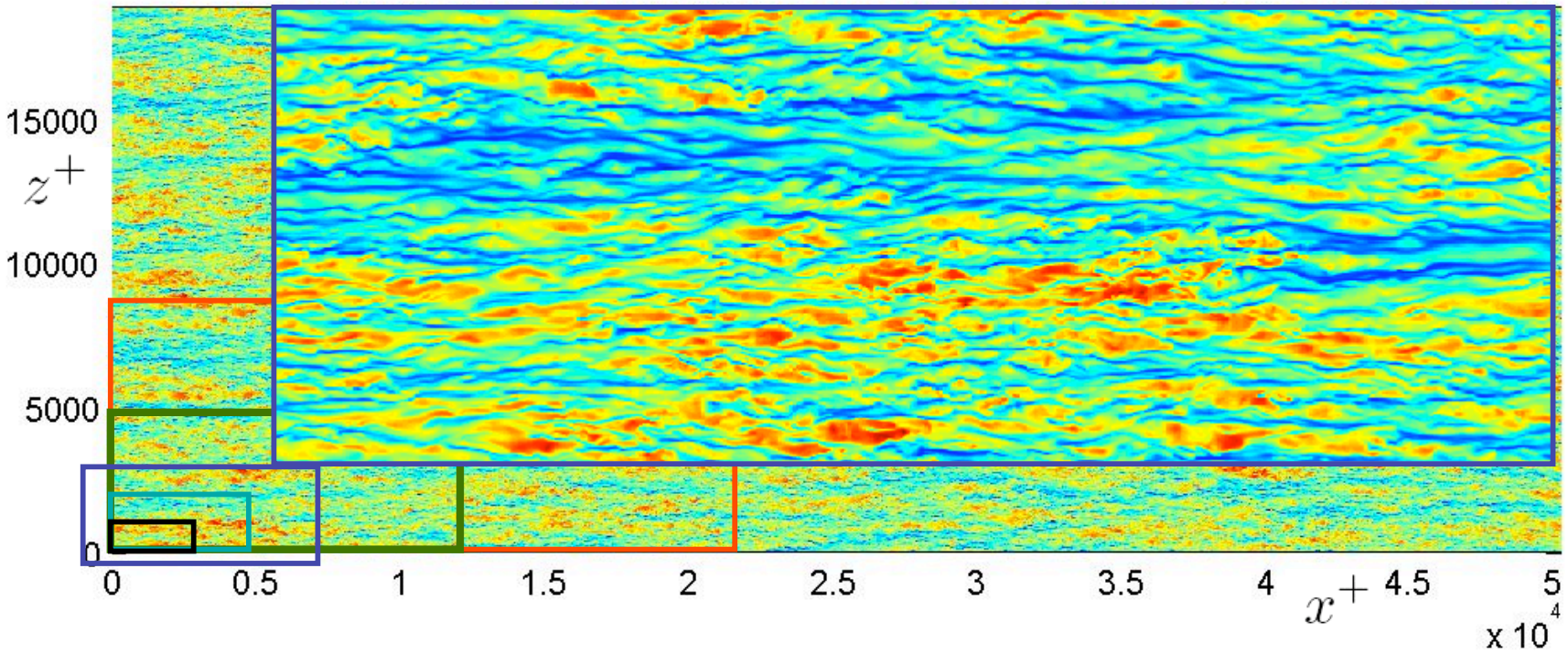
Resolución: Slash

Dominio



Hoyas & Jiménez, 07

Simulaciones anteriores



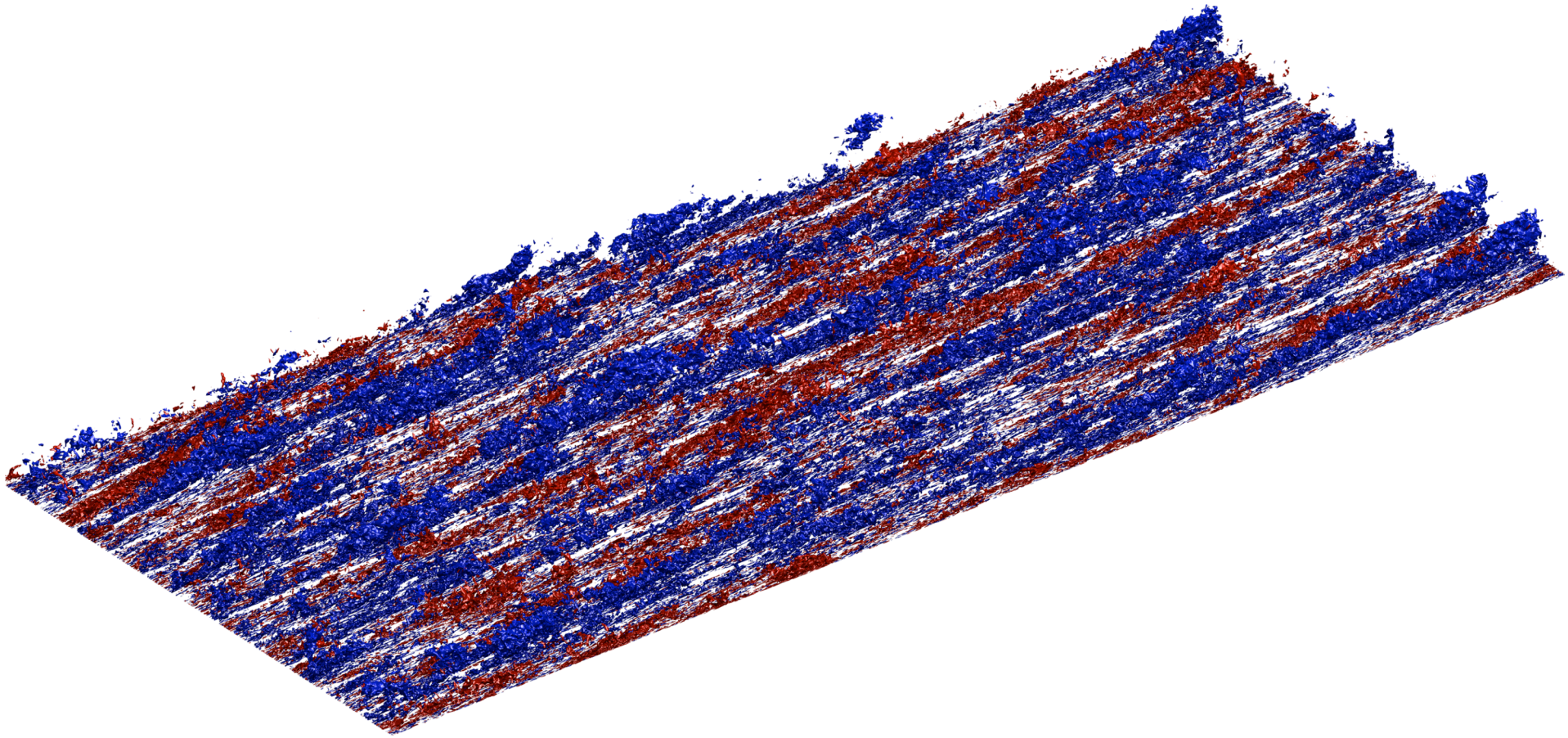
Kim, Moin and Moser, 1987, 180 (Cray XMP, NASA Ames)

Del Álamo and Jiménez, 2003 (Kadesh, CEPBA) 180

Del Álamo and Jiménez, 2003 (Kadesh, CEPBA) 550

Del Álamo, Moser, Jiménez and Zandonade, 2004 (Blue Horizon) 950

¡Estructuras complejas!



Simulaciones anteriores

Mallado del canal

	# Puntos (Fis.)	# Puntos (Fourier, R.)	# Puntos (Fourier, C.)
N_x	$6144 = 2^{11} \times 3$	4096	2048
N_y	633		
N_z	$4608 = 2^9 \times 9$		3072

Espacio físico	Espacio de Fourier	
$\Delta x^+ \simeq 8.18$	$\Delta x^+ \simeq 12.27$	$\min(\Delta y^+) \simeq 0.32$
$\Delta z^+ \simeq 4.1$	$\Delta z^+ \simeq 6.13$	$\max(\Delta y^+) \simeq 8.88$

Memoria total 400GB (simple precision).

Ecuaciones de Navier-Stokes

$$
\begin{aligned}
\frac{\partial u_i}{\partial t} &= -\frac{\partial p}{\partial x_i} + H_i + \frac{1}{Re}\nabla^2 u_i, \\
\frac{\partial u_i}{\partial x_i} &= 0, \\
\\
\vec{u} &= (u_1, u_2, u_3) = (u, v, w) \\
\vec{\omega} &= \vec{\nabla} \times \vec{u} = (\omega_1, \omega_2, \omega_3) = (\omega_x, \omega_y, \omega_z) \\
\vec{H} &= (\vec{u} \times \vec{\omega}) - \frac{1}{2}\nabla\left(\vec{u} \cdot \vec{u}\right) = (H_1, H_2, H_3)
\end{aligned}
$$

Forma Velocidad-Vorticidad

$$\frac{\partial}{\partial t}\phi = h_v + \frac{1}{Re}\nabla^2\phi \qquad \phi = \nabla^2 v$$

$$\frac{\partial}{\partial t}\omega_y = h_g + \frac{1}{Re}\nabla^2\omega_y$$

70-80% del tiempo

99% de las comunicaciones

$$h_g = \frac{\partial H_1}{\partial z} - \frac{\partial H_3}{\partial x}$$

$$h_v = -\frac{\partial}{\partial y}\left(\frac{\partial H_1}{\partial x} + \frac{\partial H_3}{\partial z}\right) + \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right)H_2$$

Discretización en x y z

Discretización

Espacial: Fourier en x y z, diferencias finitas compactas en z

$$\varphi(x,y,z) = \sum_{\alpha'=0}^{N_x-1} \sum_{\beta'=0}^{N_z-1} \varphi_{\alpha',\beta'}(y) e^{ic_\alpha \alpha' x} e^{ic_\beta \beta' z}$$

con

$$c_\alpha = 2\pi L_y / L_X \qquad c_\beta = 2\pi L_y / L_z$$

tomando

$$k_x = c_\alpha \alpha' \qquad k_z = c_\beta \beta'$$

$$\varphi(x,y,z) = \sum_{kx} \sum_{kz} \varphi_{k_x,k_z}(y)\, e^{(k_x x + k_y y)i}$$

Ecuaciones en el espacio de Fourier

$$
\begin{aligned}
\partial_t \hat{\omega}_y &= \hat{h}_g - \frac{1}{Re}\left(k_x^2 + k_z^2 - \partial_{yy}\right)\hat{\omega}_y \\
\hat{\omega}_y\left(\pm 1\right) &= 0
\end{aligned}
$$

$$
\begin{aligned}
\partial_t \hat{\phi} &= \hat{h}_v + \frac{1}{Re}\nabla^2 \hat{\phi} \\
\nabla^2 \hat{v} &= \hat{\phi}, \\
\partial_n \hat{v}\left(\pm 1\right) &= 0, \hat{v}\left(\pm 1\right) = 0
\end{aligned}
$$

Condición de Neumann para v

$$\partial_y v\,(\pm 1) = 0 \quad \longrightarrow \quad v = v_p + c_1 v_1 + c_2 v_2,$$

$$\partial_t \hat{\phi}_1 - \frac{1}{Re}\left(k_x^2 + k_z^2 - \partial_{yy}\right)\hat{\phi}_1 = 0, \qquad \partial_t \hat{\phi}_p = \hat{h}_v + \frac{1}{Re}\left(k_x^2 + k_z^2 - \partial_{yy}\right)\hat{\phi},$$

$$\nabla^2 \hat{v}_1 = \hat{\phi}_1, \qquad \nabla^2 \hat{v}_p = \hat{\phi}_p,$$

$$\hat{v}_1\,(\pm 1) = 0,\ \phi_1\,(1) = 0, \phi_1\,(-1) = 1. \qquad \hat{v}_p\,(\pm 1) = 0,\ \phi_p\,(\pm 1) = 0$$

~~$$\partial_t \hat{\phi}_2 - \frac{1}{Re}\left(k_x^2 + k_z^2 - \partial_{yy}\right)\hat{\phi}_2 = 0,$$~~

~~$$\hat{v}_2\,(\pm 1) = 0,\ \phi_2\,(1) = 0\ \phi_2\,(-1) = 1,$$~~

~~$$\nabla^2 \hat{v}_2 = \hat{\phi}_2.$$~~

$$v_2(y) = v_1(-y)$$

Discretización

Método: Runge-Kutta de tercer orden (Spalart et al, 1991)

$$\partial_t u = L(u) + N(u)$$

$$\begin{aligned}
u^1 &= u^0 + \Delta t \left[L\left(\alpha_1 u^0 + \beta_1 u^1\right) + \gamma_1 N^0 \right] \\
u^2 &= u^1 + \Delta t \left[L\left(\alpha_2 u^1 + \beta_2 u^2\right) + \gamma_2 N^1 + \xi_1 N^0 \right] \\
u^3 &= u^2 + \Delta t \left[L\left(\alpha_3 u^2 + \beta_3 u^3\right) + \gamma_3 N^2 + \xi_2 N^1 \right]
\end{aligned}$$

$$R_i = \frac{Re}{\beta_i \Delta t}, \quad Rk_i = \left(R^i + k_x^2 + k_z^2\right),$$

118 e6 ecuaciones/paso

$$\begin{aligned}
\left(\partial_{yy} - Rk_1\right) u^1 &= -R_1 \left[u^0 + \Delta t \left(\alpha_1 L u^0 + \gamma_1 N^0 \right) \right], \\
\left(\partial_{yy} - Rk_2\right) u^2 &= -R_2 \left[u^1 + \Delta t \left(\alpha_2 L u^1 + \gamma_2 N^1 + \xi_1 N^0 \right) \right], \\
\left(\partial_{yy} - Rk_3\right) u^3 &= -R_3 \left[u^2 + \Delta t \left(\alpha_3 L u^2 + \gamma_3 N^2 + \xi_1 N^1 \right) \right],
\end{aligned}$$

Discretización - y

Normal: Diferencias finitas compactas (Lele, 1991)

Primera derivad: malla de 7 puntos. Mapeada a la original

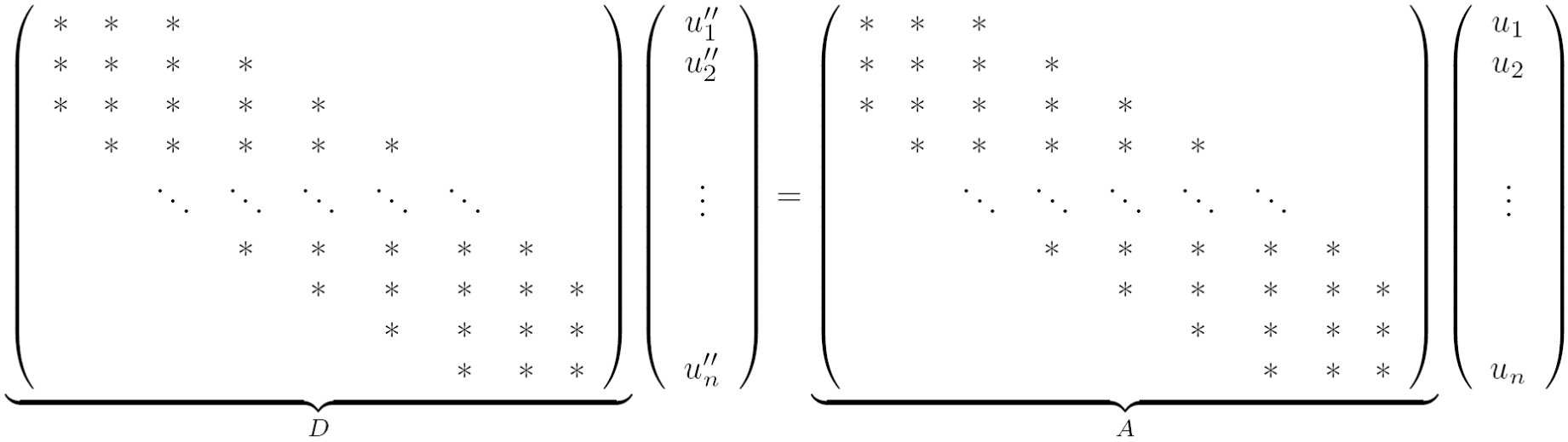
$$D_j + \sum_{m=1}^{M} b_m (D_{j+m} + D_{j-m}) = \frac{1}{h} \sum_{n=1}^{N} a_n (u_{j+n} - u_{j-n}) \qquad N=M=7$$

Segunda derivada: malla real

$$D_j^{(2)} + \sum_{m=1}^{M} b_m (D_{j+m}^{(2)} + D_{j-m}^{(2)}) = \frac{a_0 u_j + \sum_{n=1}^{N} a_n (u_{j+n} + u_{j-n})}{h^2} \qquad N=M=5$$

Sistemas: métodos LU, sin pivotaje, adaptadas de “Numerical Recipes”

Discretización - y



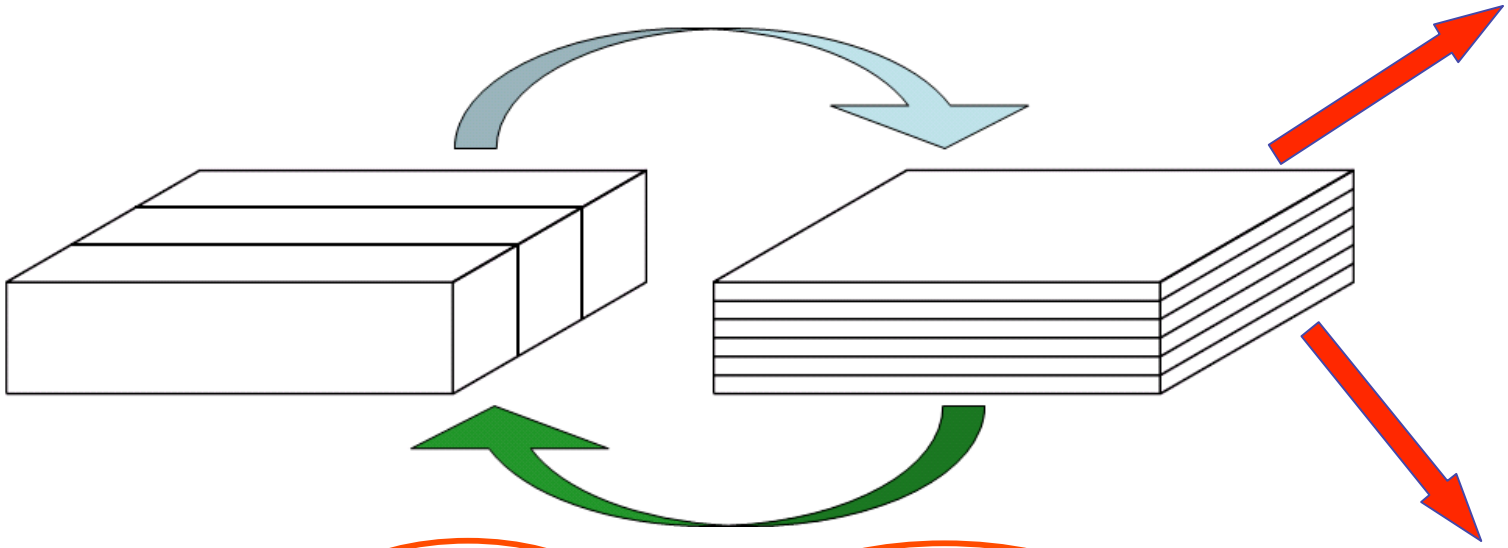
$$
\begin{aligned}
u^{n+1} - K\partial_{yy}u^{n+1} &= f(u^n) \\
Du^{n+1} - K\left(D\partial_{yy}u^{n+1}\right) &= Df(u^n) \\
Du^{n+1} - KAu^{n+1} &= Df(u^n) \\
\boxed{\left(D - KA\right)u^{n+1} &= Df(u^n)}
\end{aligned}
$$

Sistemas: métodos LU, sin pivotaje, adaptadas de “Numerical Recipes”

Esquema clásico de paralelización

$$\vec{H} = \vec{F}\left(v, \partial_y v_y, \phi, \omega_y, \partial_y \omega_y\right)$$

Solo podemos usar N_y procs



$$h_v = -\frac{\partial}{\partial y}\left(\frac{\partial H_1}{\partial x} + \frac{\partial H_3}{\partial z}\right) + \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right) H_2$$

$$h_g = \frac{\partial H_1}{\partial z} - \frac{\partial H_3}{\partial x}$$

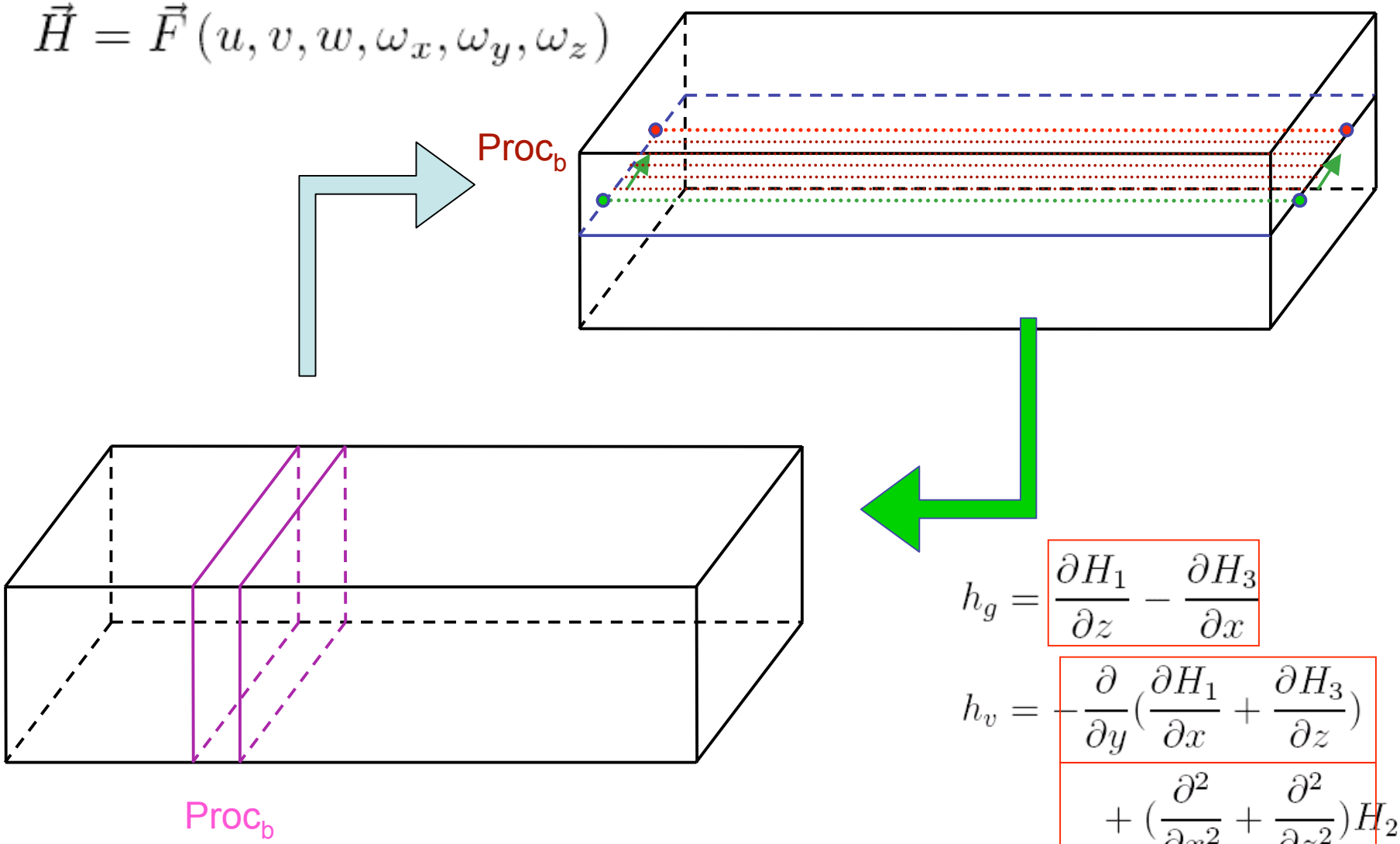
Problema dealiasing 2D

$$Re_\tau = 2000 \Rightarrow 650MB$$

$$Re_\tau = 4000 \Rightarrow 2.4GB$$

Paralelización en planos-líneas

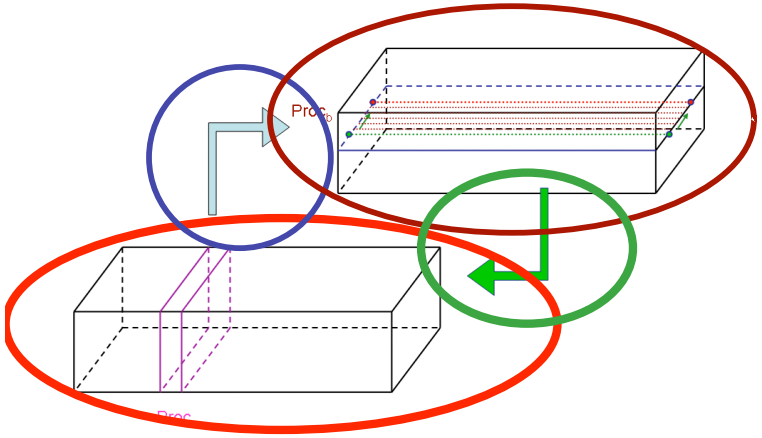
$$\vec{H} = \vec{F}\,(u, v, w, \omega_x, \omega_y, \omega_z)$$



$$h_g = \frac{\partial H_1}{\partial z} - \frac{\partial H_3}{\partial x}$$

$$h_v = -\frac{\partial}{\partial y}\left(\frac{\partial H_1}{\partial x} + \frac{\partial H_3}{\partial z}\right) + \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right) H_2$$

Paralelización en lineas planos: esquema



1.- 10%

2.- 40% (133MB)

3.- 10%

4.- 20% (66 MB)

5.- 20%

Primera parte

1.- Calculamos vel. y vort. (F-P-F)
2.- Transformamos z al espacio físico

Segunda parte

Movemos de yz a líneas en x

Tercera parte

1.- Transformamos x a físico
2.- Cálculo de la helicidad
3.- Transfomamos la helicidad a Fourier

Cuarta parte

Movemos de líneas en x a yz

Quinta parte

1.- Transformamos a (F-P-F)
2.- Calculo del RHS de la ecuación
3.- Resolvemos los sistemas
4.- Avanzamos en tiempo

!DNS son muy caras!

	Procesador	Total
Memoria	**0.2GB**	**400GB**
Pasos	***125.000***	***125.000***
Tiempo por cada paso del Runge-Kutta	**40s**	**40s**
CPU-hours totales	***2800h***	***6e6h (1.3e6)***
Horas humanas totales	**4 months**	**4months**
Transferencia de datos entre procesadores	***0.6GB***	***1.2PB***
Total de datos transmitidos	**73PB**	**145EB**
Base de datos obtenidad	***25TB***	***25TB***
Flops conseguidos	**50GF**	**3TF**
Flops totales	***18.3PF***	***3.6EF***

¡Gracias!

Fluid Dynamics Lab

Escuela de Aeronáutica, UPM

http://torroja.dmt.upm.es