Manual Tutorial para la API DVS-Central

Grupo de Geofísica Matemática y Computacional

Departamento de Recursos Naturales

Instituto de Geofísica

Universidad Nacional Autónoma de México

31 de julio de 2018

Resumen

Este documento es un un manual tutorial de una API para el método Derived Vector Space (DVS, [1, 2]), a la que llamamos DVS-Central, desarrollado por el Grupo de Geofísica Matemática y Computacional (GGMC). Se explica el uso del programa DVS-Central con el propósito de facilitar el uso del programa.

Coordinadores

- Dra. Graciela Herrera Zamarrón
- Dr. Ismael Herrera Revilla

Desarrolladores

- Dr. Iván Contreras Trejo
- Dr. Guillermo de J. Hernández García
- Dr. Norberto C. Vera Guzmán

Asesor de diseño del código

- Dr. Luis Miguel de la Cruz Salas

Índice

Índice de figuras

1. Restricciones y condiciones de uso

El DVS-Central está diseñado para resolver en paralelo sistemas de ecuaciones lineales que contengan matrices simétricas y positivas definidas que provengan de la aplicación de un método de discretización que satisfaga la condición de que siempre que un par de nodos (p,q) no estén en un mismo subdominio, la entrada correspondiente de la matriz $M_{p,q}$ será cero (en términos de las definiciones que se introducirán posteriormente, siempre que la multiplicidad de un par de nodos (p,q) sea igual a cero, la entrada correspondiente de la matriz $M_{p,q}$ también será cero). Esto lo satisfacen, por ejemplo, los métodos de diferencias finitas, elemento finito y volumen finito.

2. Problema bien planteado.

Considere un problema bien planteado, definido por la ecuación diferencial parcial (o un sistema de tales ecuaciones)

$$\mathscr{L}u = f \tag{2.1}$$

y condiciones de frontera adecuadas. Para tratarlo de la manera tradicional (en serie), se introduciría una malla (”la malla fina”) y se aplicaría un método estándar de discretización para obtener una versión discreta de la misma:

$$\underline{\underline{M}}\,\underline{U} = \underline{F} \tag{2.2}$$

Cuando se usan métodos de descomposición de dominio no superpuestos, como es el caso del método DVS, es necesario introducir una segunda malla (la malla gruesa), que de hecho constituye la descomposición del dominio.

3. Método de discretización en paralelo.

La implementación del método de discretización es completamente independiente del DVS-Central. Cualquier código que implemente un método de discretización en secuencial puede ser adaptado para usar el DVS-central. En este documento se muestran los pasos a seguir para usar el DVS-Central con un código en secuencial que implenta un método de discretización desarrollado en Fortran, así como las modificaciones necesarias a dicho código.

3.1. Adaptación al código secuencial para su ejecución en paralelo

Para implementar en paralelo el código secuencial que aplica el método de discretización se utiliza MPI. En el código para la discretización de la EDP se inicia una topología virtual de MPI con los siguientes parámetros :

1. La malla gruesa, indicando cuantas particiones habrá en cada eje coordenado. Estos datos se almacenan en un arreglo unidimensional.
2. El número de procesadores que se usarán para el procesamiento en paralelo, este número debe de coincidir con el producto de las particiones en cada eje coordenado.
3. Las especificaciones para que sea una topología cartesiana.

Lo anterior se hace insertando el siguiente código al inicio del programa secuencial.

```
integer,save,pointer,dimension(:) :: mesh

CM2------Inicializando MPI
      call MPI_INIT(ierr)
      call MPI_COMM_SIZE(MPI_COMM_WORLD,sizep,ierr)
      call MPI_COMM_RANK(MPI_COMM_WORLD,rank,ierr)
CM2a-----Inicializando variables para topología virtual
      dims(1)=4  !Filas de la topología virtual
      dims(2)=4   !Columnas de topología virtual
      period(1)=.false.
      period(2)=.false.
      reorder=.true.
CM2b----Construyendo topología virtual
      call MPI_CART_CREATE(MPI_COMM_WORLD,2,dims, &
           period,reorder,comm,ierr)
      call MPI_CART_COORDS(comm,rank,2,coord,ierr)
```

3.2. Adecuación del método de discretización en el marco del DVS-Central

La implementación del método de discretización numérico para la solución de una EDP en particular requiere ser adecuado a las especificaciones del DVS. El código debe de considerar ciertos cálculos para la construcción del sistema de ecuaciones lineales para que el DVS-Central pueda procesar adecuadamente la solución numérica.

El método de discretización se compone de dos procesos, uno que discretiza el dominio y otro que discretiza el operador diferencial. El programador debe de identificar en su código qué sección del mismo implementa cada uno de esos procesos y realizar las adecuaciones correspondientes. A continuación se describen las acciones requeridas.

3.2.1. Discretización de un dominio

Construya un conjunto finito de nodos. Para hacer esta discretización se especifican las particiones en cada eje coordenado y se numeran los puntos que quedan en las intersecciones de esas coordenadas en un orden de izquierda a derecha en una dimensión, de abajo hacia arriba para la segunda dimensión y de ser necesario, de adelante hacia atrás para la tercera dimensión. esos puntos se les refiere como nodos.

3.2.2. Clasificación de nodos

Identifique cada uno de los nodos dependiendo de su ubicación geométrica en vértices, aristas o interiores. Con esa identificación se determina la multiplicidad de los nodos y se clasifican en el esquema dual primal. La multiplicidad se calcula para cada nodo $m(p)$ y para parejas de nodos $s(p,q)$.

3.2.3. Método de discretización del operador diferencial

Modifique el código que implementa el método numérico para que opere en cada subdominio de tal forma que cada entrada de la matriz local se calcule mediante la siguiente fórmula

$$\underline{\underline{M}}^{\alpha}_{pq} = \frac{\underline{\underline{M}}_{pq}}{s(p,q)} \tag{3.1}$$

La α se refiere al número de subdominio, mientras que p y q son los nodos originales, por lo tanto, lo que indica la fórmula es que la entrada (p,q) de la matriz local $\underline{\underline{M}}^{\alpha}$ correspondiente a los nodos derivados (p,α) y (q,α) es la entrada correspondiente de la matriz original dividida entre la multiplicidad del par de nodos originales.

El cálculo de la multiplicidad en una geometría ortogonal en dos dimensiones se hace con el siguiente procedimiento

$$m(p) = \begin{cases} 1 & nodo\ interior \\ 2 & nodo\ arista \\ 4 & nodo\ vértice \end{cases} \tag{3.2}$$

La matriz se almacenan en un arreglo unidimensional por lo que su longitud será igual a la suma del número de columnas distintas de cero en cada renglón. El lado derecho del sistema está dado por la fórmula

$$\frac{\underline{F}_{(p,\alpha)}}{m(p)} \tag{3.3}$$

donde el valor de la entrada (p,α)del vector $\underline{F}$ tiene que ser dividido entre la multiplicidad del nodo original p

El cálculo de la multiplicidad de una pareja de nodos (p,q) está dado como

$$s(p,q) = min\{m(p), m(q)\} \tag{3.4}$$

El sistema de ecuaciones se almacena en arreglos unidimensionales que son declarados como apuntadores para que al momento de enviarlos a un programa externo haya compatibilidad en los tipos de datos. La información que requiere almacenar el DVS para resolver el sistema de ecuaciones es la siguiente:

1. Un arreglo de enteros int *incg_ con el número de nodo que tiene un grado de libertad.
2. Un arreglo unidimensional de enteros int *l_ren que indica cuántas columnas distintas de cero hay en cada renglón con grados de libertad.
3. Un arreglo unidimensional de enteros int *l_col con los números de columnas distintos de cero para cada renglón con grados de libertad.
4. Un arreglo unidimensional de doble precisión con las entradas de la matriz que son distintas de cero para cada renglón con grados de libertad.
5. Un arreglo unidimensional de doble precisión double * l_rhs con el lado derecho del sistema de ecuaciones.

3.3. Diseño del programa para DVS-Central que acopla con el exterior

3.3.1. Acomplamiento

Se cuenta con un código que invoca al DVS-Central y le envía los parámetros para resolver el sistema de ecuaciones en paralelo y reciba en una variable adecuada los resultados generados por el propio DVS-Central.

Ese código se invoca desde el programa de Fortran con la instrucción

El DVS-Central se llama desde el programa de Fortran y requiere de los siguientes parámetros para resolver el sistema de ecuaciones en paralelo.

```
extern "C"{

 void mainc_(int *d, int *mesh, int *comm, int *ierr,int *len,int *incg_,
             int *l_row, int *l_col ,double *l_mtx, double * l_rhs,
             double * u_sol ){
           }
   }
```

Donde los parámetros corresponden a

```
int *d     es la dimesión del dominio
int *mesh  contiene las particiones en la malla gruesa y en la malla fina
int *comm  es el comunicador actual
int *ierr  es el código de error. en caso de existir alguno
int *len   es el arrego con los nodos incógnitaen el subdominio
int *l_ren indica cuántas columnas distintas de cero hay en cada renglón len
int *incg_ es el arreglo con los nodos incógnita que
           y su número de nodo en la numeración local del subdominio
int *l_row
int *l_col
double *l_mtx, es el arreglo que contiene las entradas de la matriz.
           Se almacena por renglón y por columna.
           Se van recorriendo las columnas de cada renglón con el contador k
           y se van acumulando en count_
double * l_rhs
double * u_sol
incog[i]-1 . Se le resta uno por la diferencia entre las
           numeraciones de enteros entre Fortran y C
l_matrix[count_+ k] l_col[count_+k])-1 es el arreglo que contiene el número
           de incógnita con la que está relacionado cada renglón, así como
           l_matrix contiene el valor de la entrada,
           l_col contiene el número de columna
```

3.4. Acciones del DVS-Central para preparar su ejecución

Agregar los parámetros para la clasificación de los nodos en duales y primales.

Subprograma que intercambia la información entre la Interfaz y el DVS-Central . Dicha información son las estructuras de datos que contienen al sistema de ecuaciones lineales.

Subprograma que llama al DVS-Central.

Ejecuta el código que resuelve numéricamente en paralelo utilizando el DVS-BDDC.

En el DVS-Central se tienen que instanciar las clases de subdominio y del método de discretización para enviarles lo parámetros correspondientes al sistema de ecuaciones lineales y la clasificación de nodos.

El DVS central tiene las siguientes clases y métodos para llevar a cabo lo anterior.

- Clasificación de los nodos en las caras del subdominio
- Clasificación de nodos duales y primales
- Copia del sistema de ecuaciones lineales en sus propias estructuras de datos

3.5. Preparación de programas ejecutables

Una vez que se tienen las adecuaciones al código secuencial la siguiente acción es agregar las líneas de código que llaman al DVS-Central, preparar los parámetros que se le envian al DVS-Central y modificar el archivo bash que al ejecutarse realiza la compilación de ambos códigos generando un solo programa ejecutable.

3.6. Compilación

Debido a que el código fuente está desarrollado en dos lenguajes de programación distintos es necesario rener instalados los compiladores mpif90 para los códigos en Fortran y gcc para los códigos en C++. También se requiere tener disponibles las bibliotecas

- lmpi_cxx
- lstdc++

para llevar a cabo el linkeado entre los códigos .

Directorio de archivos

Los códigos del DVS-Central tienen que estar en el directorio ParallelApplication el cual a su vez estará en el sistema de archivos donde se encuentra el programa fuente del código que aplica el método numérico de discretización.

Makefile y compilado de códigos fuente

La compilación se hace usando un archivo make que proporcionamos con la distribución del DVS-Central el cual requiere ser modificado de la siguiente manera:

1. El nombre del archivo fuente en Fortran.
2. El nombre del codigo objeto .o

Para generar el programa ejecutable en la línea de comandos se ejecuta el programa make modificado para la compilación usando MPI.

3.7. Ejecución

Para ejecutar el programa se le da la siguiente instrucción en la línea de comandos

```
bsub -m g1 -oo nombre-archivo-salida
           -eo archivo-error -q q_hpc
           -n N mpirun ./nombre_executable
```

3.8. Salida de datos

Una vez que el DVS-Central obtuvo la solución numérica en sus estructuras de datos devuelve el control al programa que lo invocó. La solución numérica quedará almacenada en la variable

```
u_sol
```

que es un arreglo de doubles y al cuál se tiene acceso desde el programa externo una vez que el DVS termine su ejecución. Este arreglo es de una longitud igual al número de nodos en el subdominio (tanto nodos con incógnitas como con valores conocidos). Por lo que la solución numérica, que sólo se calcula en los nodos con incógnitas) hay que copiarla en el orden adecuado para ello se utiliza la numeración local. Las entradas de ese arreglo correspondientes a los nodos que no son incógnitas se les asigna el valor cero.

El usuario externo requiere extraer la solución usando la información en la matriz del sistema de cuántas y cuáles columnas por renglón son distintas de cero.

La concatenación de la solución global y el mapeo a la numeración de la malla fina de todo el subdominio la realizará el usuario en el programa externo.

4. Aplicación del método DVS aun problema de Poisson

Sea $\Omega \subset R^2$ una región rectangular cualquiera y consideremos la siguiente ecuación diferencial

$$-\nabla^2 u = f, \quad \text{en} \quad \Omega \tag{4.1}$$

con una condición de frontera tipo Dirichlet

$$u = \hat{u}, \quad \text{sobre} \quad \partial\Omega. \tag{4.2}$$

Utilizando el método de diferencias finitas centrales, la versión discreta de (4.1) queda expresada como:

$$u_{i-1,j} - u_{i+1,j} + 4u_{i,j} - u_{i.j-1} - u_{i,j+1} = h^2 f_{i,j} \tag{4.3}$$

Como dominio Ω, consideramos el rectángulo con dimensiones $(LX)(LY)$, y una condición de frontera tipo Dirichlet $u = \hat{u}$ sobre $\partial\Omega$.

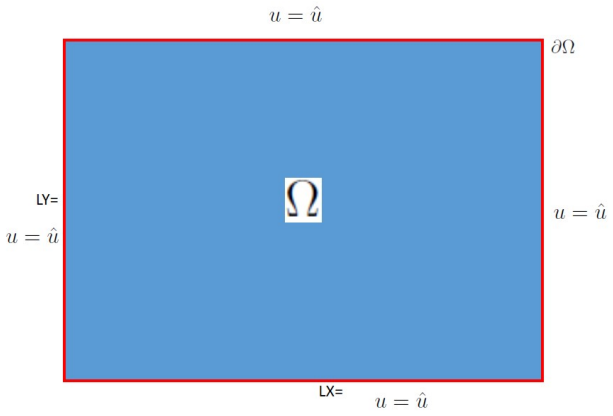


Figura 1: Dominio para resolver un problema de Laplace con CF-Dirichlet

4.1. Cálculo e identificación de subdominios por tipo

Los datos que son necesarios para descomponer un dominio Ω en subdominios Ω_e, son las siguientes.

- LX = Longitud del dominio en la dirección X
- (NSX,NSY) = (Núm.Interv.DX, Núm.Interv.DY)
- (NX,NY) = (Núm.Interv.DX.Por.Subd., Núm.Intev.DY.Por.subd.)
- LY se calcula debido a que estamos considerando que $h = H$=longitud.de.malla, es la misma en la dirección x y en la dirección y.
- LY = NSY*NY*H
- H = HX/NX
- HX = LX/NSX

Entonces, dados LX, NSX, NSY, NX, NY, se calculan NS, LY.

$$\begin{aligned} NS &= (NSX) \times (NSY), \\ LY &= NSY * NY * H. \end{aligned} \tag{4.4}$$

donde

$$\begin{aligned} H &= HX/NX, \\ HX &= LX/NSX. \end{aligned}$$

Como estamos considerando que: $h = H = \triangle x = \triangle y$, debe haber alguna restricción para que se cumpla esta condición.

La condición en este caso es que LY se calcule conforme a (4.4).

Se tienen $NSX \times NSY$ subdominios que se pueden identificar de la siguiente manera.

Para el caso que nos ocupa, cualquier subdomnio pertenece a uno de los siguientes 9 tipos de subdominios.

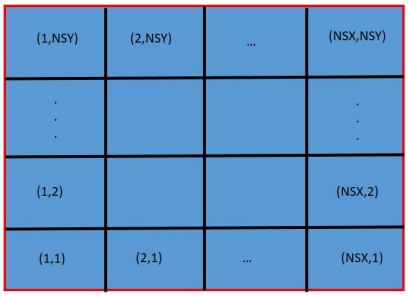


Figura 2: Identificación de subdominios considerando NSX=NSY=4

Figura 3: Identificación de subdominios considerando 9 diferentes tipos

4.2. Numeración de nodos por subdominio

Cada subdominio tiene dimensiones $(HX) \times (HY)$, con

$$HX = \frac{LX}{NSX}, \quad HY = \frac{LY}{NSY} \tag{4.5}$$

Como estamos considerando que $H = h = \triangle x = \triangle y$ en el estencil de diferencias

finitas, requerimos que:

$$h = \frac{HX}{NX} = \frac{HY}{NY} \tag{4.6}$$

siendo NX, NY el número de segmentos en la dirección x y en la dirección y respectivamente.

Nótese que estamos utilizando indistintamente $(h = H, nx = NX, ny = NY)$.

La forma de numerar los nodos en cada subdominio es a través de la función

$$nn(i,j) = i + (j-1)(nx+1) \tag{4.7}$$

con

$$i = 1,2,...,(nx+1), \quad \text{en la dirección x,}$$
$$j = 1,2,...,(ny+1), \quad \text{en la dirección y.}$$

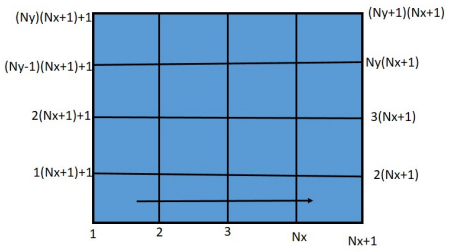


Figura 4: Forma de mallar y numerar un subdominio

4.3. Multiplicidad de los nodos por subdominio

Cada subdominio tiene $(n_x+1)(n_y+1)$ nodos.

Cuando se hace la descomposición de dominio, cada uno de los subdominios queda incluido en uno de 9 diferentes tipos de subdominios, que mostramos a continuación.

De acuerdo con los diferentes tipos de subdominios mostrados antes, la multiplicidad $s(nn)$ de un nodo *nn* y un subdominio *ns* puede expresarse como:

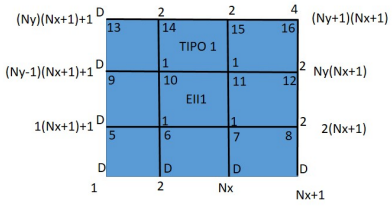
Para subdominios del tipo 1 se tiene

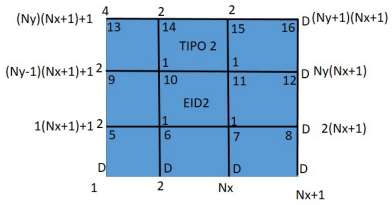
$$\begin{aligned}
s(nn(i,1),1) &= D \quad \text{para} \quad i = 1,...,(n_x+1) \\
s(nn(1,j),1) &= D \quad \text{para} \quad j = 2,...,(n_y+1) \\
s(nn(i,j),1) &= 1 \quad \text{para} \quad i = 2,3,...,n_x, \quad j = 2,3,...,n_y, \\
s(nn(i,n_y+1),1) &= 2 \quad \text{para} \quad i = 2,3,...,n_x \\
s(nn(n_x+1,j),1) &= 2 \quad \text{para} \quad j = 2,3,...,n_y \\
s(nn(n_x+1,n_y+1),1) &= 4
\end{aligned}$$

Para subdominios del tipo 2 se tiene

$$\begin{aligned}
s(nn(i,1),2) &= D \quad \text{para} \quad i = 1,...,(n_x+1) \\
s(nn(n_x+1,j),2) &= D \quad \text{para} \quad j = 2,...,(n_y+1) \\
s(nn(i,j),2) &= 1 \quad \text{para} \quad i = 2,3,...,n_x, \quad j = 2,3,...,n_y, \\
s(nn(1,j),2) &= 2 \quad \text{para} \quad j = 2,3,...,n_y \\
s(nn(i,n_y+1),2) &= 2 \quad \text{para} \quad i = 2,3,...,n_x \\
s(nn(1,n_y+1),2) &= 4
\end{aligned}$$

Para subdominios del tipo 3 se tiene





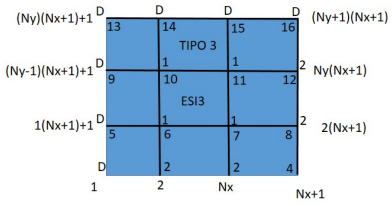


Figura 5: Multiplicidad para subdominios tipo 1, 2 y 3

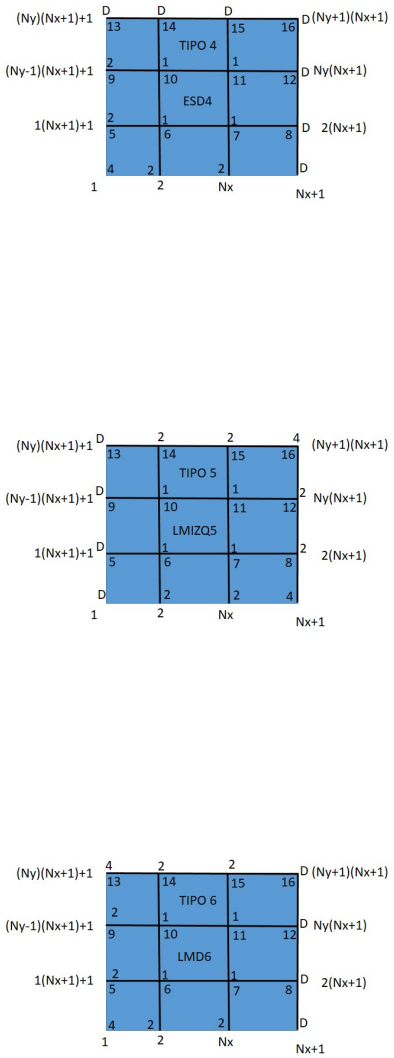
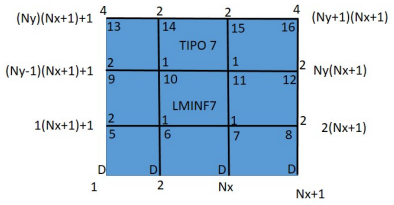
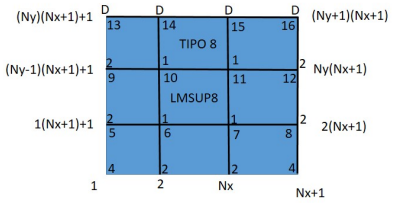


Figura 6: Multiplicidad para subdominios tipo 4, 5 y 6





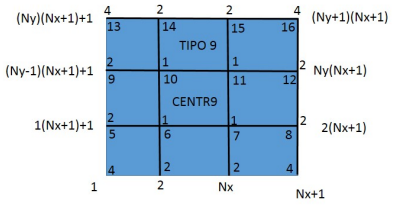


Figura 7: Multiplicidad para subdominios tipo 7, 8 y 9

$$
\begin{aligned}
s(nn(1,j),3) &= D \quad \text{para} \quad j = 1,...,(n_y+1) \\
s(nn(i,n_y+1),3) &= D \quad \text{para} \quad i = 2,...,(n_x+1) \\
s(nn(i,j),3) &= 1 \quad \text{para} \quad i = 2,3,...,n_x, \quad j = 2,3,...,n_y, \\
s(nn(i,1),3) &= 2 \quad \text{para} \quad i = 2,3,...,n_x \\
s(nn(n_x+1,n_j),3) &= 2 \quad \text{para} \quad j = 2,3,...,n_y \\
s(nn(n_x+1,1),3) &= 4
\end{aligned}
$$

Para subdominios del tipo 4 se tiene

$$
\begin{aligned}
s(nn(1,n_y+1),4) &= D \quad \text{para} \quad i = 1,...,(n_x+1) \\
s(nn(n_x+1,j),4) &= D \quad \text{para} \quad j = 1,...,n_y) \\
s(nn(i,j),4) &= 1 \quad \text{para} \quad i = 2,3,...,n_x, \quad j = 2,3,...,n_y, \\
s(nn(1,j),4) &= 2 \quad \text{para} \quad j = 2,3,...,n_y \\
s(nn(i,1),4) &= 2 \quad \text{para} \quad i = 2,3,...,n_x \\
s(nn(1,1),4) &= 4
\end{aligned}
$$

Para subdominios del tipo 5 se tiene

$$
\begin{aligned}
s(nn(1,j),5) &= D \quad \text{para} \quad j = 1,...,(n_y+1) \\
s(nn(i,j),5) &= 1 \quad \text{para} \quad i = 2,3,...,n_x, \quad j = 2,3,...,n_y, \\
s(nn(i,1),5) &= 2 \quad \text{para} \quad i = 2,...,n_x \\
s(nn(n_x+1,j),5) &= 2 \quad \text{para} \quad j = 2,3,...,n_y \\
s(nn(i,n_y+1),5) &= 2 \quad \text{para} \quad i = 2,3,...,n_x \\
s(nn(n_x+1,n_y+1),5) &= 4
\end{aligned}
$$

Para subdominios del tipo 6 se tiene

$$
\begin{aligned}
s(nn(n_x+1,j),6) &= D \quad \text{para} \quad j = 1,...,(n_y+1) \\
s(nn(i,j),6) &= 1 \quad \text{para} \quad i = 2,3,...,n_x, \quad j = 2,3,...,n_y, \\
s(nn(i,1),6) &= 2 \quad \text{para} \quad i = 2,...,n_x \\
s(nn(i,n_y+1),6) &= 2 \quad \text{para} \quad i = 2,3,...,n_x \\
s(nn(1,j),6) &= 2 \quad \text{para} \quad j = 2,3,...,n_y \\
s(nn(1,n_y+1),6) &= 4
\end{aligned}
$$

Para subdominios del tipo 7 se tiene

$$
\begin{aligned}
s(nn(i,1),7) &= D \quad \text{para} \quad i = 1,...,(n_x+1) \\
s(nn(i,j),7) &= 1 \quad \text{para} \quad i = 2,3,...,n_x, \quad j = 2,3,...,n_y, \\
s(nn(n_x+1,j),7) &= 2 \quad \text{para} \quad j = 2,...,n_y \\
s(nn(i,n_y+1),7) &= 2 \quad \text{para} \quad i = 2,3,...,n_x \\
s(nn(1,j),7) &= 2 \quad \text{para} \quad j = 2,3,...,n_y \\
s(nn(1,n_y+1),7) &= 4 \\
s(nn(n_x+1,n_y+1),7) &= 4
\end{aligned}
$$

Para subdominios del tipo 8 se tiene

$$
\begin{aligned}
s(nn(i,n_y+1),8) &= D \quad \text{para} \quad i = 1,...,(n_x+1) \\
s(nn(i,j),8) &= 1 \quad \text{para} \quad i = 2,3,...,n_x, \quad j = 2,3,...,n_y, \\
s(nn(i,1),8) &= 2 \quad \text{para} \quad i = 2,...,n_x \\
s(nn(n_x+1,j),8) &= 2 \quad \text{para} \quad j = 2,3,...,n_y \\
s(nn(1,j),8) &= 2 \quad \text{para} \quad j = 2,3,...,n_y \\
s(nn(1,1),8) &= 4 \\
s(nn(n_x+1,1),8) &= 4
\end{aligned}
$$

Para subdominios del tipo 9 se tiene

$$
\begin{aligned}
s(nn(i,j),9) &= 1 && \text{para} \quad i = 2,3,...,n_x, \quad j = 2,3,...,n_y, \\
s(nn(i,1),9) &= 2 && \text{para} \quad i = 1,...,n_x \\
s(nn(n_x+1,j),9) &= 2 && \text{para} \quad j = 2,...,n_y \\
s(nn(i,n_y+1),9) &= 2 && \text{para} \quad i = 2,3,...,n_x \\
s(nn(1,j),9) &= 2 && \text{para} \quad j = 2,3,...,n_y \\
s(nn(1,1),9) &= 4 \\
s(nn(n_x+1,1),9) &= 4 \\
s(nn(n_x+1,n_y+1),9) &= 4 \\
s(nn(1,n_y+1),9) &= 4
\end{aligned}
$$

Todas las multiplicidades de un subdominio quedan almacenadas en un vector de dimension $(n_x+1)(n_y+1)$.

4.4. Construcción de las matrices locales

Con base en la multiplicidad de los nodos mostrada arriba, las matrices locales por subdomnio se pueden construir de la siguiente manera.

Si $s(nn) = D$ No se aplica el estencil y se tiene un renglón vacío,
Si $s(nn) = 1,2,4$ Se aplica el estencil considerando solo nodos del subdominio.

Los nodos del estencil que sean dato se pasan al vector del lado derecho.

Siguiendo esta metodología, se desarrolló un programa $dvs.f90$ para construir cada una de las matrices locales. Consideremos el siguiente dominio ya descompuesto

La matríz local asociada al primer subdominio queda como

Este es el estencil

$$-u_{i-1,j} - u_{i+1,j} + 4u_{i,j} - u_{i,j-1} - u_{i.j+1} = h^2 f_{i,j}$$

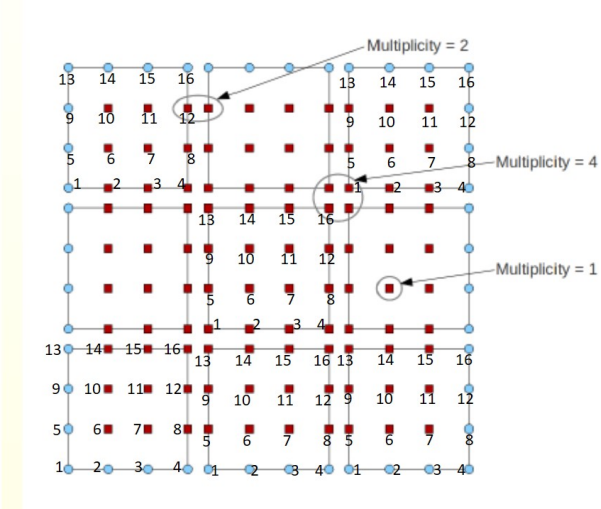


Figura 8: Dominio descompuesto en 9 subdominios

- 3.00 3.00 3 3 3 3
- LX,LY,NSX,NSY,NX,NY
- TIPOS DE SUBDOM JY IX 1 1 0 0
- TIPOS DE SUBDOM JY IX 2 7 0 1
- TIPOS DE SUBDOM JY IX 3 2 0 2
- TIPOS DE SUBDOM JY IX 4 5 1 0
- TIPOS DE SUBDOM JY IX 5 9 1 1
- TIPOS DE SUBDOM JY IX 6 6 1 2
- TIPOS DE SUBDOM JY IX 7 3 2 0
- TIPOS DE SUBDOM JY IX 8 8 2 1
- TIPOS DE SUBDOM JY IX 9 4 2 2

- SUBDOMINIO 1 TIPO 1 EII1
- 1 0 0.000 0.000
- 2 0 0.000 0.000
- 3 0 0.000 0.000
- 4 0 0.000 0.000
- 5 0 0.000 0.000
- 6 1 0.000 0.000
- 7 1 0.000 0.000
- 8 2 0.000 0.000
- 9 0 0.000 0.000
- 10 1 0.000 0.000
- 11 1 0.000 0.000
- 12 2 0.000 0.000
- 13 0 0.000 0.000
- 14 2 0.000 0.000
- 15 2 0.000 0.000
- 16 4 0.000 0.000
- NN(I,J),S(NN(I,J),1),HHX,HHY

$$\underline{\underline{M}}^{\gamma} = \left(M_{pq}^{\gamma}\right)$$
$$M_{pq}^{\gamma} = \frac{M_{pq}}{s(p,q)}\delta_{pq}^{\gamma}$$

$$s(p,q) = \begin{cases} 1 \text{ si } m(p,q) = 0 \\ m(p,q) \text{ en cualquier otro caso} \end{cases}$$

$$\delta_{pq}^{\gamma} = \begin{cases} 1 \text{ si } p,q \in \overline{\Omega}_{\gamma} \\ 0 \text{ en cualquier otro caso} \end{cases}$$

4.5. Conclusiones

El programa dvsp.f90 genera renglón por renglón, las entradas de las matrices locales asociadas a la ecuación de Laplace en dos dimensiones, considerando nueve tipos diferentes de subdominios. Las matrices locales se obtienen en base a tres diferentes vectores NR,NC y VM. Las dimensiones de estos vectores son:

- dim[NR] = (NX+1)*(NY+1)+1
- dim[NC] = 5*(NX+1)*(NY+1)
- dim[VM] = 5*(NX+1)*(NY+1)

Los valores de *NX* y *NY* son definidos al principio del programa.

Dentro del programa dvsp.f90 y para propósitos de acoplamiento con el DVS_Central, la multiplicidad de cada nodo en cada subdominio se guarda en el arreglo "*MUL*". Los valores de este arreglo pueden ser: $\{0,1,2,4\}$ y quedan identificados de la siguiente manera.

- MUL(i)=0 indica que i es un nodo de frontera externa.
- MUL(i)=1 indica que i es un nodo interior.
- MUL(i)=2 indica que i es un nodo dual.
- MUL(i)=4 indica que i es un nodo primal.

Se definieron 5 nuevos arreglos para propósitos de acoplamiento con el DVS_central:

- incognitas(i), cuyo valor es un nodo incógnita del subdominio.
- lrow(i), identificando un renglón (o nodo) incógnita de la matríz local.
- lcol(j), indicando las entradas diferentes de cero que tiene el renglón lrow(i).
- lmatrix(j), indicando los valores de las entradas de la matriz que son diferentes de cero.
- lrhs(i), indicando las entradas del vector del lado derecho que son diferentes de cero.

5. Aplicación del método DVS a MODFLOW

En esta sección describimos cómo adaptamos el código de MODFLOW en base a los requerimientos del *DVS-Central*, su aplicación a un problema específico y presentamos los resultados de métricas de eficiencia de corridas con diferentes números de procesadores.

5.1. Proceso de paralelización de MODFLOW

El diseño del software, *MODFLOW-DVS*, se hizo teniendo en cuenta que MODFLOW hace una llamada, mediante una interfaz, a *DVS-Central*. Por tanto, se diseñó el software analizando los siguientes puntos:

1. **Preprocesamiento:** Se plantea un problema a resolver con MODFLOW, con el cual obtendremos un sistema de ecuaciones derivado de aplicar diferencias finitas a la ecuación diferencial parcial de flujo con condiciones de frontera y condiciones iniciales que corresponden al problema. Con *MODFLOW-DVS* construimos sistemas de ecuaciones locales que cumplan con las condiciones del problema y del método DVS. *MODFLOW-DVS* dará como resultado de este proceso una matriz y un vector con el lado derecho del sistema de ecuaciones local (RHS, por sus siglas en inglés) por cada subdominio, es decir, por procesador.

2. **Interfaz:** La interfaz se diseña como una clase de C++ que pueda ser usada por *MODFLOW-DVS* donde éste mande tres arreglos como parámetros:

   - Matriz local: Matriz correspondiente a cada procesador.
   - RHS local: Lado derecho del sistema de ecuaciones local.
   - Columnas locales: Se refiere a las columnas de las matrices locales que tienen entradas distintas de cero.

3. **Método de solución del sistema de ecuaciones en paralelo:** Las clases desarrolladas en *DVS-Central* son usadas desde MODFLOW para resolver el sistema global con los sistemas de ecuaciones locales mediante los parámetros enviados. Se obtiene una solución en el espacio derivado.

4. **Solución en MODFLOW:** Se mapea la solución del espacio derivado al espacio original y se aplican los procesos de balance de MODFLOW para que el usuario acceda a los resultados como los produce el MODFLOW original.

En la figura 9 se muestra el diagrama de flujo que sigue *MODFLOW-DVS* para resolver cualquier problema que se introduzca. Describimos cada paso de la Figura 9 aplicado al problema que se presenta de la siguiente manera:

- **Inicia MPI y topologías virtuales:** Se preparan todos los procesadores y se crean topologías virtuales desde *MODFLOW-DVS*
- **Reservar memoria y leer:** En este proceso el procesador maestro es el encargado de leer los archivos con toda la información básica del problema y enviarla a todos los procesadores.
- **Periodo de Esfuerzos:** En el problema que se presenta solo se usa un periodo de esfuerzo.
- **Leer y preparar:** El procesador maestro lee toda la información relacionada con el periodo de esfuerzo correspodiente y la envía a todos los procesadores.
- **Avanza en el tiempo:** Como en el problema que se presenta es estacionario y no se requieren condiciones iniciales, en cada uno A QUÉ SE REFIERE CADA UNO se avanza en el tiempo teniendo. Esto sólo se hace una vez.
- **Formulación:** Se construye la matriz y lado derecho locales de cada procesador con la información de los arreglos en memoria local.
- **Interfaz:** ME PARECE CONVENIENTE QUE PONGAS EL PROCESO DE LA INTERFAZ C++ EXPLÍCITAMENTE
- **Aproximación:** Se incluye *DVS-Central* como método de solución de las matrices locales. FALTARÍA EXPLICAR QUE UNA PARTE ES EL DVS-BDDC Y OTRA ES ?? PARA QUE NOS SIRVA PARA EXAPLICAR A LO QUE SE MIDEN LOS TIEMPOS DESPUÉS.
- **Control de salida:** El procesador maestro recibe los resultados, integra la solución global y los escribe en el archivo de salida.

- **Balance:** Se hace el balance global de masa con los resultados obtenidos.
- **Salida:** El procesador maestro escribe el balance de masa en el archivo de salida.
- **Deallocate memory:** Se desaloja memoria en todos los procesadores.

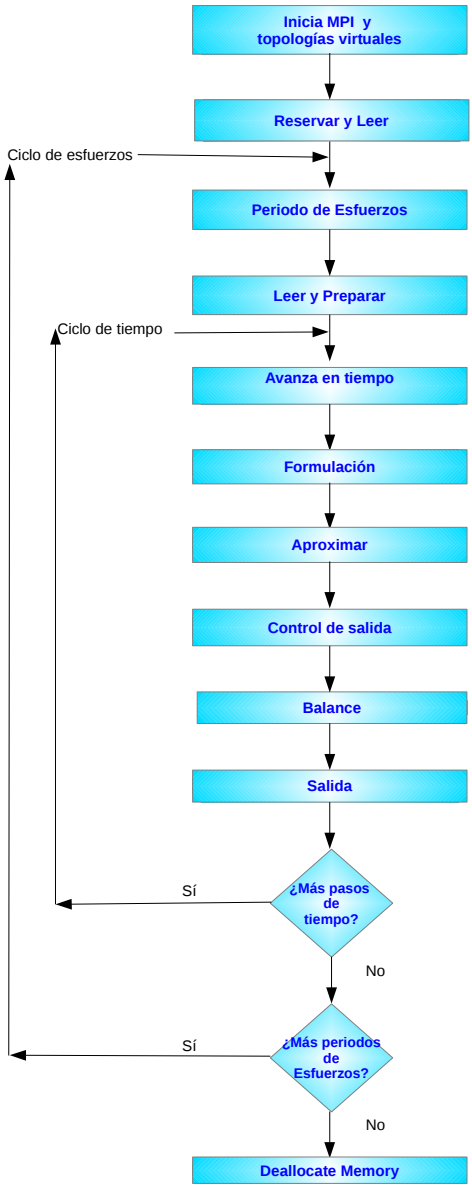


Figura 9: Diagrama de flujo de *MODFLOW-DVS*

5.2. Descripción del problema

A continuación se describe el problema con el que se demuestra el uso del *MODFLOW-DVS*. El problema considera un acuífero cuyo flujo es estacionario y está descrito por la Ec. (5.1).

$$\frac{\partial}{\partial x}\left(K_{xx}\frac{\partial h}{\partial x}\right)+\frac{\partial}{\partial y}\left(K_{yy}\frac{\partial h}{\partial y}\right)=0, \tag{5.1}$$

El dominio está en 2D y es cuadrado, como el que se muestra el la figura (10). Las condiciones de frontera son de tipo Dirichlet (o de carga asignada) en las fronteras superior e inferior y de tipo Neumann, sin flujo, en los dos restantes, como se muestra en la Fig. (10).

El dominio se discretiza en 4705 filas y 4705 columnas con cada celda de $1m^2$, éstas constituyen la malla fina. Por otro lado, $K_{xx} = K_{yy}$, son variables espacialmente y su distribución se puede observar en la fig. (11). Por lo tanto, si el problema se resolviera en serie, utilizando diferencias finitas, se tendría que resolver la Ec. (5.2):

$$\begin{aligned} CR_{i,j-1/2,k}(h_{i,j-1,k}-h_{i,j,k}) &+ CR_{i,j+1/2,k}(h_{i,j+1,k}-h_{i,j,k})+ \\ CC_{i-1/2,j,k}(h_{i-1,j,k}-h_{i,j,k}) &+ CC_{i+1/2,j,k}(h_{i+1,j,k}-h_{i,j,k})=0, \\ i=2,\ldots,4704, j &= 2,\ldots,4704, k=1, \end{aligned} \tag{5.2}$$

donde $CR_{i,j}$ y $CC_{i,j}$ son coeficientes que dependen la conductividad hidráulica. Por otro lado, las condiciones de frontera cumplen con

$$h_{i,j,k}=100 \tag{5.3}$$
$$i=4705, j=1,\ldots,4705, k=1$$
$$h_{i,j,k}=0 \tag{5.4}$$
$$i=0, j=1,\ldots,4705, k=1$$
$$CR_{i,j+1/2,k}h_{i,j+1,k}-CR_{i,j,k}h_{i,j,k}=0 \tag{5.5}$$
$$i=1,\ldots,4705, j=1,4705, k=1,$$
$$\tag{5.6}$$

como se muestra en la Fig. (10).

La discretización en la Fig. (10) genera 22137025 celdas, sin embargo, debido a las condiciones que se muestran en las Ecs. (5.3) y (5.4) donde se tiene carga asignada en ciertas celdas, el número de incógnitas para el sistema (5.2) es de 22127615. Estas incógnitas incluyen las celdas que tiene asignadas condiciones de Neumann.

El problema consiste entonces en resolver un sistema del tipo $\underline{\underline{A}}\underline{x} = \underline{b}$ donde $\underline{\underline{A}}$ sería una matriz de 22127615^2 y $\underline{x}, \underline{b}$ vectores de 22127615 entradas. Como un sistema algebraico a resolver, sería necesario invertir la matriz $\underline{\underline{A}}$. Es importante mencionar que por el método de discretización que se usa, $\underline{\underline{A}}$ es una matriz de cinco bandas para este caso.

Para resolver el problema utilizando DVS, se determina en cuántos subdominios se va descomponer el dominio original (la malla gruesa). En este ejemplo el número total de celdas de la malla fina se dejó constante para todas las pruebas, por lo que la condición que se debe cumplir depende de los siguientes parámetros: si *nx* es el número de particiones de la malla fina en el eje *x* (número de celdas menos una), *ny* es el número de particiones de la malla fina en el eje *y*, *mx* es el número de particiones de la malla gruesa en el eje x y *my* es el número de particiones de la malla gruesa en el eje y, entonces se debe cumplir que

$$mod(nx, mx) = mod(ny, my) = 1, \tag{5.7}$$

donde

$$mod(x, y) = x - y[\frac{x}{y}], \quad \forall x, y \in Z, \tag{5.8}$$

donde $[\frac{x}{y}]$ es la parte entera del cociente $\frac{x}{y}$. La condición (5.7) determina la lista del número de procesadores con los que se pueden hacer las pruebas.

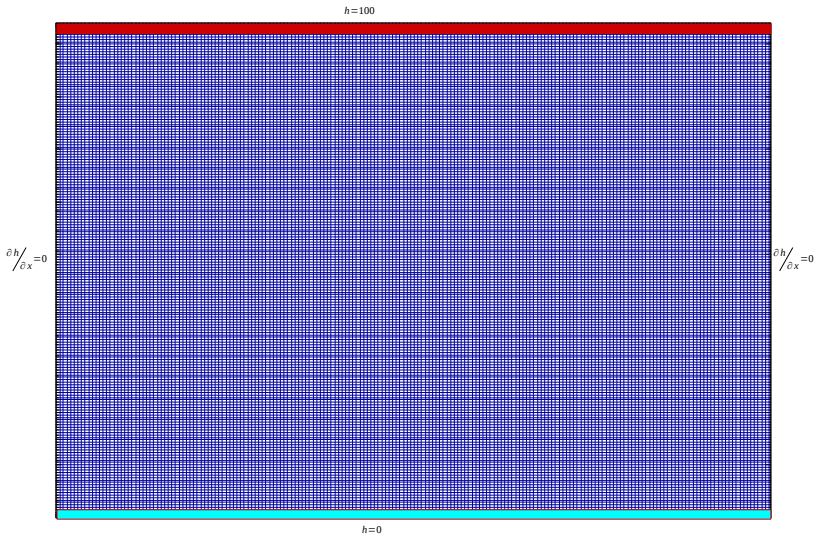


Figura 10: Dominio y condiciones de frontera del problema.

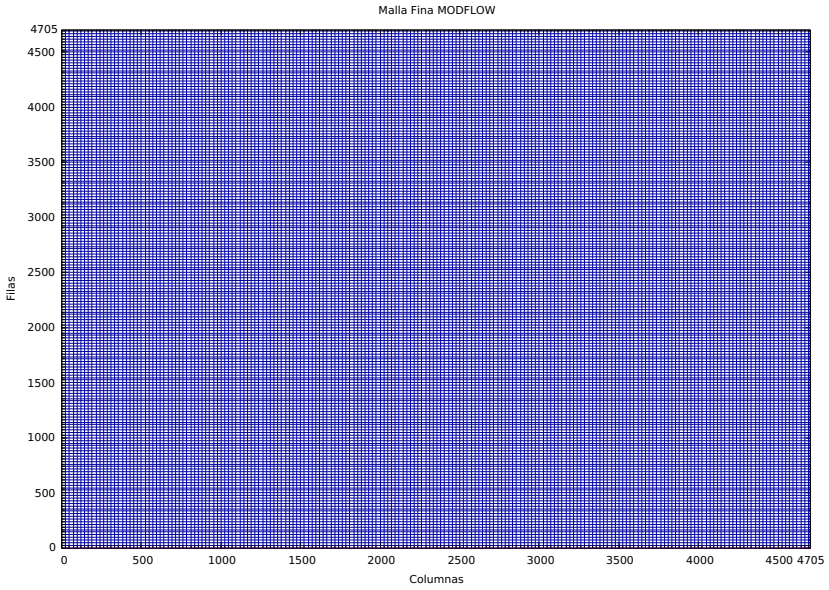


Figura 11: Malla fina y conductividad hidráulica (K) en el dominio.

5.3. Resultados

La Tabla 1 muestra la lista del número de procesadores con los se hicieron las pruebas y los tiempos de ejecución de diferentes etapas del código MODFLOW-DVS.

Proc	Part	Preproc	Matrices	Solver DVS	DVS-BDDC	Total
144	12	4.72	2.59E-02	1097.51	234.49	1104.05
196	14	8.815	1.54E-02	578.25	146.61	588.09
256	16	27.97	1.15E-02	341.18	89.78	369.76
441	21	10.24	7.45E-03	138.90	49.29	149.75
576	24	9.61	6.65E-03	86.46	33.63	96.83
784	28	4.36	5.47E-03	55.95	27.58	60.58
1024	32	5.73	4.94E-03	37.75	20.75	43.69
1764	42	5.28	3.90E-03	16.04	9.92	21.45
2401	49	5.87	4.58E-03	8.98	5.88	14.96
3136	56	7.52	3.50E-03	8.77	6.37	16.39

Cuadro 1: Tiempos obtenidos en las distintas etapas de MODFLOW-DVS, medidos en segundos, para un problema con 22127615 grados de libertad. Las primeras dos columnas muestran el número total de procesadores utilizados y el número de particiones de la malla gruesa por eje, respectivamente. Las siguientes cinco columnas muestran los tiempos en segundos de cada etapa del proceso.

La Tabla 1 muestra los tiempos que tardaron las pruebas al correr las etapas de *preprocesamiento*, *matrices*, *solver* y *DVS-BDDC*. El *Preprocesamiento* incluye todos los procesos preparatorios, explicados en la sección 5.1, para que se formen las matrices locales (*Inicia MPI y topologías virtuales*, *Reservar memoria y leer*, *Periodo de Esfuerzos*, *Leer y preparar* y *Avanza en el tiempo*); la etapa *Matrices* incluye la construcción de las matrices locales en cada procesador (conforme a lo explicado en la sección 5.1, es el proceso *Formulación*); la etapa *Solver* incluye desde la interfaz en C++ hasta la resolución de las matrices locales (*Interfaz* Y *Aproximar*, conforme a las sección 5.1); la etapa *DVS-BDDC* incluye exclusivamente el algoritmo *BDDC* (un subproceso dentro de *Aproximar*). Finalmente, la columna *Total* describe el tiempo total que tarda MODFLOW-DVS sin tomar en cuenta el postprocesamiento (¿DE *Balance* EN ADELANTE?) que se hace en el

código original de MODFLOW.
Los resultados de la Tabla 1 se muestran gráficamente en la Figura 12. La gráfica muestra que gran parte del tiempo total es debido al tiempo que se tarda el método incluido en MODFLOW como *solver*. Se puede decir que el código es escalable hasta los 1024 procesadores debido a que después ya no se obtienen grandes ahorros en el tiempo de procesamiento.
En el caso del *Preprocesamiento* se esperaba que el tiempo aumentara conforme el número de procesadores, sin embargo, en las primeras pruebas sube el tiempo pero a partir de 441 comienza a bajar y después a subir en los 2401 procesadores. Esto puede deberse a la distribución de los procesadores que se usan en las distintas pruebas. A pesar de esto, el tiempo de preprocesamiento es muy bajo a comparación del tiempo total del programa.

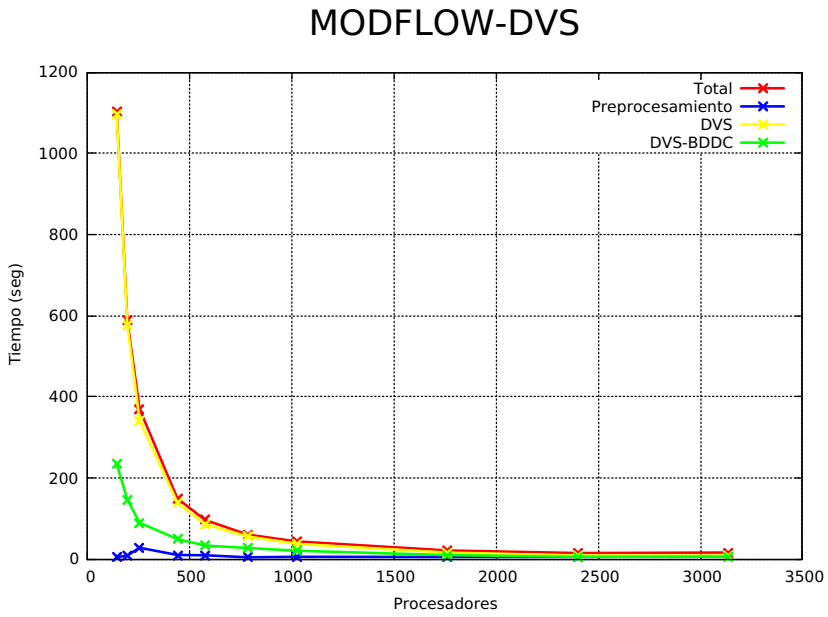


Figura 12: Tiempos obtenidos para distintos procesadores

Para el tiempo que se tarda en formar las matrices locales, es decir, la columna de *Matrices* el tiempo va disminuyendo debido a que el número de coeficientes que se tienen que calcular por procesador va disminuyendo conforme se usan más procesadores. En la columna *Solver DVS* el tiempo va descendiendo debido a que al ir resolviendo localmente, las matrices se van haciendo más pequeñas lo cual

reduce el tiempo de preprocesamiento, en parte también por lo que se tiene en la columna *DVS-BDDC*. Finalmente, el tiempo *Total* se va reduciendo hasta que se llega a los 3136 procesadores. Se puede ver que el tiempo de preprocesamiento sube y también el tiempo en *DVS-BDDC* debido a que son las únicas etapas donde existe comunicación entre procesadores. Esto indica que con ese número de procesadores las comunicaciones comienzan a influir más en los resultados de tiempo.

No fue posible ejecutar el problema que estamos resolviendo en un procesador debido a que se requería mucha memoria por lo que la primer prueba se hace en 144 procesadores como se puede ver en la Tabla 1. Se deben calcular la aceleración y la eficiencia con respecto a 144 procesadores. Sea la aceleración relativa dada por

$$S_p^{p'} = \frac{T_{p'}}{T_p}, \tag{5.9}$$

donde $S_p^{p'}$ es la aceleración relativa a p' procesadores con p procesadores; $T_{p'}$ es el tiempo de procesamiento en p' procesadores que va ser la referencia en el cálculo de la aceleración y T_p es el tiempo de procesamiento en p procesadores. Por otro lado la eficiencia relativa se define como

$$E_p^{p'} = \frac{p'}{p} S_p^{p'} = \frac{p'}{p} \frac{T_{p'}}{T_p}, \tag{5.10}$$

donde $E_p^{p'}$ es la eficiencia con p procesadores relativa a p'.

Los resultados de la Tabla 2 se muestran gráficamente en las Figs. 13 y 14 . Se puede ver que la aceleración relativa está por encima de la ideal y comienza a bajar para 3136 procesadores. Por otro lado, para la eficiencia, como se había visto, la ideal es 100 %, sin embargo, las eficiencias están por encima de la ideal alcanzando casi 450 %.

Procesadores	$S_p^{p'}$	$E_p^{p'}$
144	1	1
196	1.87	1.37
256	2.98	1.67
441	7.37	2.40
576	11.40	2.85
784	18.22	3.34
1024	25.26	3.55
1764	51.45	4.20
2401	73.75	4.42
3136	67.32	3.09

Cuadro 2: Aceleración y eficiencias relativas para cada una de las pruebas con distintos procesadores con respecto a 144 procesadores.

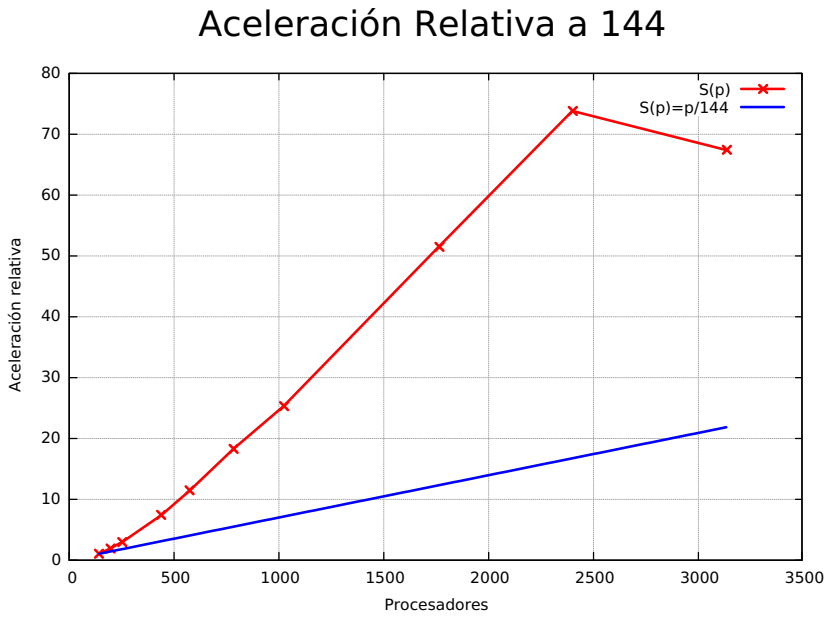


Figura 13: Aceleración relativa a 144 procesadores. La línea roja muestra los resultados de MODFLOW-DVS y la línea azul es la aceleración relativa ideal con respecto a 144 procesadores.

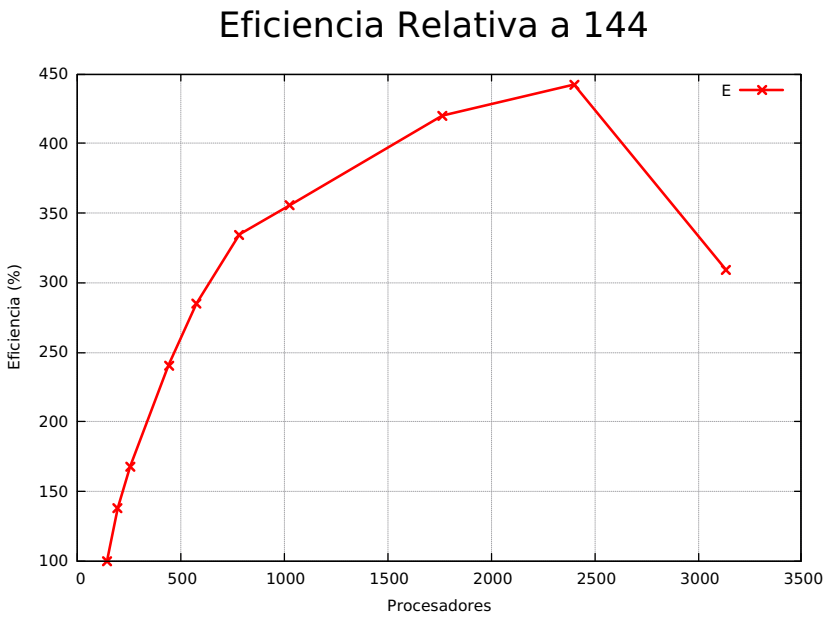


Figura 14: Eficiencia relativa a 144 procesadores. Muestra una eficiencia de casi 450 % para 2401 procesadores.

Referencias

[1] Herrera, Ismael and de la Cruz, Luis M. and Rosas-Medina, Alberto, *Nonoverlapping Discretization Methods for Partial Differential Equations*, Numerical Methods for Partial Differential Equations, 30-5: 1427–1454, 2014.

[2] Herrera, I. and I. Contreras, *An Innovative Tool for Effectively Applying Highly Parallelized Hardware. To Problems of Elasticity* , Geofísica internacional (2016) 55-1: 363-386

[3] Herrera, I. and G.F. Pinder, *Mathematical Modelling in Science and Engineering: An axiomatic approach* , John Wiley, 243p., 2012.