Manual de usuario de MICSc

Este manual contiene las instrucciones básicas para la instalación, ejecución y creación de nuevos casos de uso, así como una descripción de la estructura jerárquica de los ficheros de salida producidos en formato SILO.

Instalación y configuración

Para instalar MICSc, una vez se ha obtenido el paquete, se debe descomprimir y entrar en el directorio creado.

```
$ tar xvzf micsc.tar.gz
$ cd micsc
```

Posteriormente, para poder realizar la configuración del sistema, el requisito mínimo es disponer de una instalación de PETSc y definir las variables de entorno `PETSC_DIR` y `PETSC_ARCH` si no están ya definidas.

```
$ PETSC_DIR=<directorio_petsc>; export PETSC_DIR
$ PETSC_ARCH=<arquitectura_petsc>; export PETSC_ARCH
```

En este punto, MICSc proporciona la opción de realizar una instalación que haga uso de SILO, en cuyo caso este último debe estar también instalado en la máquina. Las opciones que admite el *script* de configuración para incluir SILO son:

`--with-silo`: Configura la instalación buscando las librerías y cabeceras de SILO en los directorios por defecto (i.e. `/usr/local`, `/opt`, ...).

`--with-silo-dir=<silo_dir>`: Configura la instalación de MICSc buscando las librerías de SILO en `<silo_dir>/lib` y las cabeceras en `<silo_dir>/include`.

`--with-silo-flags=<silo_flags>`: Configura la instalación especificando exactamente los parámetros que se deben utilizar en el comando de compilación (por ejemplo, `-lsiloh5 -lhdf5 -L<dir> -I<dir>` ...).

De esta manera, es posible realizar la configuración con soporte para SILO:

```
$ ./configure --with-silo-dir=<silo_dir>
```

o bien realizar una instalación sin soporte para SILO, con PETSc como única dependencia.

```
$ ./configure
```

Llegados a este punto, es imprescindible definir la variable de entorno `MICSC_DIR` (en caso de no haberse definido ya anteriormente) para poder proceder con la compilación.

```
$ MICSC_DIR=$PWD; export MICSC_DIR
$ make
```

Esta compilación creará un directorio `$PETSC_ARCH` dentro del directorio de MICSc (`$MICSC_DIR`) que contendrá la librería compilada así como cualquier otro fichero específico de la configuración realizada. En este punto se puede comprobar la corrección de la instalación mediante

```
$ make test
```

Es importante destacar que estos tests hacen uso MPI, por lo que el testeo en máquinas con sistemas de colas para este tipo de ejecuciones puede ser problemático. Además, cabe destacar que es posible compilar los casos básicos proporcionados con la distribución de MICSc con

```
$ make examples
```

Los binarios generados, junto con el resto de ficheros necesarios para la ejecución de estos casos se pueden encontrar en `$MICSC_DIR/cases`, en un subdirectorio distinto para cada uno de los casos.

Ejecución

MICSc proporciona diferentes casos de uso sobre los que se pueden realizar diferentes pruebas. Para ello, accedemos al directorio de un caso como, por ejemplo, el del canal periódico, lo compilamos (si no lo hemos compilado ya antes con `make examples`) y lo ejecutamos.

```
$ cd $MICSC_DIR/cases/periodchannel
$ make
$ mpirun -np <p> ./main <args>
```

Las opciones que admiten los casos generados con MICSc se pueden obtener con el argumento `-help`. Al margen de las opciones proporcionadas por PETSc para vectores, matrices, *solvers*, precondicionadores y demás, las opciones de MICSc son las siguientes (`<n>` determina un entero, `r` un real, `e` un enumerado y `f` un fichero):

- Opciones de inicialización

  `-grid_file <f>` : Fichero de malla.

  `-rod_file <f>` : Fichero con los parámetros por defecto del caso.

- Opciones de monitorización

  `-log_level <e>` : Nivel de monitorización {0, 1, 2}

  `-out_ievery <n>` : Número de iteraciones exteriores a realizar para escribir resultados. -1 para no escribir resultados intermedios.

  `-out_idtevery <n>` : Número de pasos temporales a realizar para escribir resultados. -1 para no escribir resultados intermedios.

  `-start_instant_means <n>` : Paso de tiempo en el que se debe empezar el cálculo de medias de valores instantáneos. -1 para no realizar dicho cálculo.

  `-start_variance_means <n>` : Paso de tiempo en el que se debe empezar el cálculo de medias de las varianzas. -1 para no realizar dicho cálculo.

  `-inifl` : Determina si se debe escribir el resultado inicial.

  `-finfl` : Determina si se debe escribir el resultado final.

  `-print_props` : Determina si se deben escribir resultados de propiedades. Actualmente la única propiedad que admite la escritura de resultados es la viscosidad LES.

- Opciones de resolución del sistema de ecuaciones

  `-max_outer_its <n>` : Número máximo de iteraciones exteriores.

  `-min_outer_its <n>` : Número mínimo de iteraciones exteriores.

  `-res_max <r>` : Norma máxima del residuo normalizado.

  `-res_max <r>` : Norma máxima del vector corrección normalizado.

  `-k_restart` : Determina si se debe reiniciar una ejecución anterior a partir de un fichero de resultados binario `restart.bin`.

  `-init_it <n>` : Número de la primera iteración.

  `-k_wrap <e>` : Periodicidad del dominio del problema {`NONPERIODIC`, `XPERIODIC`, `YPERIODIC`, `XYPERIODIC`, `XYZPERIODIC`, `XZPERIODIC`, `YZPERIODIC`, `ZPERIODIC`, `XYZGHOSTED`}

`-n_ghost <n>` : Número de nodos vecinos.

`-gravity <r>` : Fuerza de la gravedad. 0 para ignorarla.

`-n_order_trans <e>` : Esquema temporal {`steady`, `eul1`, `eul2`, `adm2`, `adm3` }

`-deltat <r>` : Paso de tiempo.

`-tinit <r>` : Instante de tiempo inicial.

`-tfinal <r>` : Instante de tiempo final.

- Opciones de variables del problema

`-uref <r>` : Valor constante para la entrada de flujo en la dirección X.

`-mu <r>` : Valor de μ (viscosidad dinámica).

`-cs0 <r>` : Constante de Smagorinsky para el modelo LES.

`-dpdx <r>` : Valor del flujo de masa $\partial p/\partial x$

`-tref <r>` : Valor de referencia para la temperatura.

`-rho <r>` : Valor de ρ (densidad).

`-ndim <n>` : Número de dimensiones.

`-ncoupvar <n>` : Número de variables acopladas.

`-cont_eq <e>` : Ecuación de continuidad {`poisson`, `simple`, `simplec`, `simpler` }

`-inlet <e>` : Función para el flujo de entrada {`u`, `f`, `rho` }

`-les` : Determina si se debe utilizar el modelo LES.

`-kemod` : Determina si se debe utilizar el modelo Kϵ.

`-var_t` : Determina si se debe considerar la temperatura como una variable a resolver.

Implementación de nuevos casos

Al margen de los casos base proporcionados, MICSc permite la creación de nuevos casos. Para ello, en primer lugar se debe crear un nuevo directorio para el caso

```
$ cd $MICSC_DIR/cases
$ mkdir <caso>
```

Este directorio generalmente suele contener (al margen de otros ficheros que puedan ser necesarios para el caso particular) cuatro ficheros principales:

- `grd.inp`: Contiene la malla y las condiciones de contorno tal y como se especifica en el siguiente apartado.

- `rod.inp`: Contiene las opciones por defecto del caso. Se proporciona por comodidad para que no sea necesario especificarlas siempre por línea de comandos. Cabe destacar que las opciones que luego se puedan especificar en el momento de la ejecución sobreescriben las de este fichero.
- `makefile`: Fichero para la compilación del caso. Se puede copiar de cualquier otro caso.
- `main.c`: Código fuente para la ejecución del caso. Aparte de las funciones propias del caso, consta de una función `main` en la que se puede hacer uso de las siguientes llamadas a la librería de MICSc:

`MICScFinalize():`

Libera toda la memoria utilizada por la aplicación y llama a la rutina de finalización de los objetos de PETSc. En el código, debe ser la última llamada.

`MICScInitialize(PetscInt argc, char** argv[]):`

Se encarga de realizar la inicialización de PETSc, así como de leer los ficheros de entrada y reservar e inicializar todos los valores y estructuras de datos comunes a todos los casos. Debe ser la primera llamada del programa y los parámetros de entrada deben de ser punteros a los parámetros de entrada de la función `main`.

`MICScSetProp(St_Prop *prop, St_Patch *patch, ktypeProp propType, PetscReal cons, PetscErrorCode(funct*)(St_Prop *prop), ktypeIp interpType, PetscReal linRelax, PetscTruth calcMeans):`

Modifica los valores por defecto de una propiedad. El primer parámetro es el puntero a la propiedad a modificar y el resto son la región de la malla sobre la que se aplica (`patch`), el tipo de propiedad (`propType`), el valor de la propiedad si es constante (`cons`), la función de evaluación si es variable (`funct`), el tipo de interpolación (`interpType`) y la relajación lineal (`linRelax`). Además, es posible especificar si se desean calcular medias para esta propiedad mediante `calcMeans`.

`MICScSolve():`

Resuelve el sistema de ecuaciones. Es el núcleo principal de MICSc y la llamada que consume prácticamente todo el tiempo de ejecución de la simulación.

`MICScVarSetGamma(PetscInt index, ktypeProp gammaType, ktypeIp gammaInterpType, PetscReal gammaConst, PetscErrorCode (gammaFunct*) (St_Prop *prop)):`

Sustituye la propiedad del coeficiente de difusión por una nueva con los valores especificados como parámetro. El primer parámetro es el índice de la variable; es posible utilizar las macros `U`, `V`, `W`, `P`, etc para este parámetro. Esto es aplicable para todas las funciones de nombre `MICScVarSet*`.

`MICScVarSetInitValues(PetscInt index, ktypeProp initValuesType, ktypeIp initValuesInterpType, PetscReal initValuesConst, PetscErrorCode (initValuesFunct*)(St_Prop *prop)):`

Se define de manera análoga a `MICScVarSetGamma` para los valores iniciales de la variable en cada celda.

`MICScVarSetInterpolation(PetscInt index, ktypeIp interpType):`

Modifica el tipo de interpolación de la variable.

`MICScVarSetLimits(PetscInt index, PetscReal minValue, PetscReal maxValue):`

Modifica los valores mínimo y máximo de la variable.

`MICScVarSetReference(PetscInt index, PetscReal resRef, PetscReal corrRef):`

Modifica los valores de referencia de la variable.

`MICScVarSetRelaxation(PetscInt index, PetscReal linRelax, PetscReal fdtRelax):`

Modifica los valores de relajación de la variable.

`MICScSetInletFunction(PetscErrorCode (*function)(PetscInt*, PetscReal*)):`

Establece la función que se debe utilizar para calcular el flujo de entrada, en caso de que se defina tal condición de contorno.

Formato de entrada de la malla

Cada caso debe disponer de un fichero que defina tanto la malla como las condiciones de contorno. Este fichero se estructura de la siguiente manera:

```
<{0,1}>
<num_divisiones_eje_X>
<num_divisiones_eje_Y>
<num_divisiones_eje_Z>
<lista_coords_eje_X>
<lista_coords_eje_Y>
```

```
<lista_coords_eje_Z>
<lista_condiciones_de_contorno>
```

El primer valor representa si la malla es rectangular (0) o cilíndrica (1). Aunque actualmente sólo se soportan mallas rectangulares, también es posible que en un futuro se implementen mallas cilíndricas. Las siguientes tres líneas contienen un único número entero por línea, que define el número de divisiones para cada uno de los ejes. Posteriormente deben aparecer tres líneas con una lista de coordenadas para cada uno de los ejes. El tamaño de la lista se deberá corresponder con el número definido en las líneas anteriores y deberán estar formados por números reales separados por espacios. En este punto es importante destacar que se asume la coordenada $[0, 0, 0]$ como origen de la malla y no es necesario incluirla en este fichero. Por último, aparece una lista de condiciones de contorno donde cada una de éstas ocupa una línea distinta con el siguiente formato:

```
<x> <nx> <y> <ny> <z> <nz> <vecino> <tipo> <porosidad>
```

Los primeros 6 parámetros son de tipo entero y especifican la región de la malla sobre la que se define la condición de contorno ($[x, y, z] - [x + nx, y + ny, z + nz]$); el siguiente parámetro especifica sobre qué cara de las celdas se define la condición ({*north, south, east, west, high, low, none*}); el tipo define la condición de contorno ({*wall, moving_wall, inlet, outflow, outpress_newman, outpress_extrap, symmetry, fix*}); y la porosidad es un número real en el rango [-1, 1].

Formato de salida SILO

Por último, en caso de haber configurado la instalación de MICSc con SILO, los ficheros producidos podrán abrirse en diversos visores, tales como VisIt. Sin embargo, es necesario conocer la estructura jerárquica que presentan los ficheros SILO generados por MICSc. Dicha estructura se puede encontrar en la Figura 1. En esta figura, los elementos marcados en negrita son hojas (variables y malla), mientras que los elementos en cursiva son directorios.

Así, los resultados se dividen en iniciales (`InitialResults`), intermedios (`IntermediateResults`), finales (`FinalResults`) y de paso temporal (`TimeStepResults`). Cabe destacar que los resultados de paso temporal se escriben al obtener la convergencia (o alcanzar el número máximo de iteraciones), mientras que los resultados intermedios se refieren a aquellos resultados producidos en una determinada iteración, a mitad de un paso temporal. Dentro de cada uno de estos directorios habrá subdirectorios para el paso de tiempo (`time_<tiempo>`) o la iteración (`field_<it>`) cuando corresponda. Por último, los directorios más alejados de la raíz con-

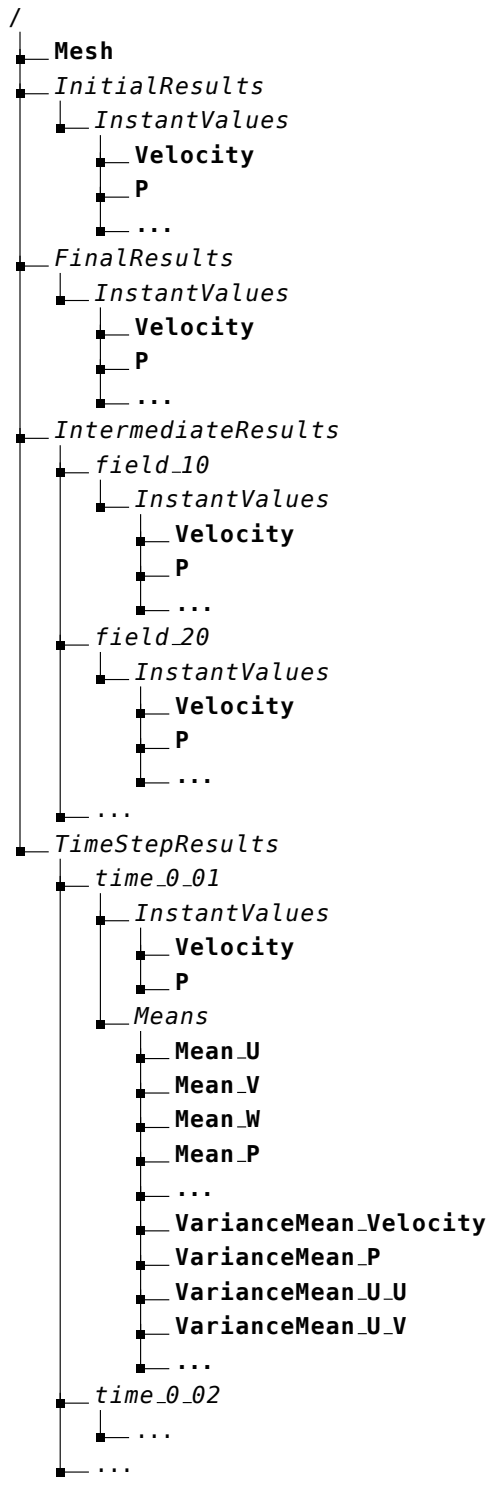

Figura 1: Estructura jerárquica de un fichero SILO generado por MICSc.

tendrán valores instantáneos para las variables (`InstantValues`) o las medias cuando corresponda (`Means`). Es importante notar que para las medias `Mean_<var>` indica la media de los valores instantáneos de la variable `var`, `VarianceMean_<var>` indica la media de la varianza de la variable `var` y `VarianceMean_<var1>_<var2>` indica la media de la covarianza entre las variables `var1` y `var2`.