Tutorial: Modflow 6 + Flopy

Luis Miguel de la Cruz Salas

2026-07-05

Table of contents

Introducción

Cuadernos interactivos para MODFLOW 6 + Flopy

1 MODFLOW 6: introducción.

1.1 Resumen.

El propósito de este documento es describir la arquitectura de MODFLOW 6, sus componentes y paquetes, así como los elementos principales que se usan en una simulación de flujo con el modelo GWF. Esta descripción se basa en la documentación de MODFLOW 6 y solo se explican los conceptos principales para entender la forma en que interactúan todas las componentes y paquetes durante una simulación.

2 Introducción.

MODFLOW 6 utiliza la Programación Orientada a Objetos (POO) para admitir una amplia gama de capacidades y para integrar las funcionalidades de todas las versiones y variantes de MODFLOW anteriores.

Esta última versión está centrada en la solución numérica de uno o más modelos hidrológicos numéricos que pueden conectarse e interactuar entre ellos. Cada modelo es un objeto de una clase que encapsula datos y métodos. Se pueden crear múltiples objetos (que representan modelos hidrológicos) en una sola simulación.

Además, se utilizan los conceptos de abstracción y herencia para gestionar y ocultar la complejidad de los modelos, y para permitir que otros autores agreguen nuevos modelos de procesos hidrológicos, de forma sistemática y eficiente, sin interferir con otros desarrollos.

Este enfoque ha dado como resultado componentes de simulación genéricos, denominados **Modelos Numéricos**, **Intercambios Numéricos** y **Soluciones Numéricas**, diseñados para funcionar conjuntamente y abordar muchos de los requisitos de diseño de esta nueva versión.

3 Componentes.

En MODFLOW 6, una simulación consiste de una única ejecución, que puede incluir múltiples modelos de flujo (y de transporte). Estos modelos son configurados y controlados por componentes y paquetes.

Una componente es un término general para describir una parte de MODFLOW 6, que puede ser un módulo, un objeto, una subrutina o un conjunto de estos. La figura 1 muestra las componentes principales que pueden ser usadas en una simulación de MODFLOW 6.

Figura 1. Diagrama de componentes de MODFLOW 6. Tomada de [1].

Una simulación en MODFLOW 6 típicamente se basa en cuatro componentes, como se explica en [1]: * **Models**. Un modelo numérico resuelve un proceso hidrológico; por ejemplo, el *GWF model*, resuelve numéricamente la ecuación de flujo subterráneo usando el método CVFD (*Control-Volume Finite-Difference*). * **Exchange**. Un intercambio numérico facilita la comunicación entre dos modelos; por ejemplo, un intercambio GWF-GWF, permite que algunas celdas de un modelo GWF estén hidráulicamente conectadas con celdas del otro modelo GWF. * **Solutions**. Una solución numérica resuelve uno o más modelos hidrológicos y utiliza métodos iterativos para resolver sistemas lineales y no-lineales. * **Timing**. Este módulo controla el paso de tiempo y determina el fin de una simulación.

4 Simulación de flujo con GWF.

En la figura 2 se muestra una representación de una simulación en MODFLOW 6 que contiene un solo modelo de GWF.

Figura 2. Diagrama que muestra las componentes que se usan para la simulación de un solo modelo GWF. Las componentes son: (1) una simulación (que es el programa principal), (2) una componente para gestionar el tiempo, (3) una componente para gestionar la solución numérica y (4) un único modelo GWF. Figura tomada de [2].

4.1 La simulación.

Una simulación consiste de una ejecución que puede incluir varios modelos (de flujo, de transporte, de intercambios, numéricos, etcétera). La simulación es la componente de mayor nivel y está controlada por el programa principal. La siguiente figura, tomada de [1], esquematiza el flujo de una simulación.

El programa principal ejecuta varios procedimientos, conocidos como ***primary procedures*** en la secuencia apropiada para realizar y controlar la simulación.

- CR: Crea objetos de tipo *model*, *package*, *exchange* y/o *solution*.
- DF: Define algunos atributos de los objetos, por ejemplo el tamaño de algunos arreglos.
- AR: Genera los arreglos. También lee información que será constante durante toda la simulación.
- TU: Incrementa las variables del tiempo y calcula la longitudes del paso de tiempo.
- RP: Lee la información de los archivos de entrada, conforme se necesite, para actualizar los estreses hidrológicos u otras entradas que varían con el tiempo.
- CA: Actualiza las variables dependientes. Para soluciones numéricas, utiliza métodos iterativos para resolver sistemas de ecuaciones no lineales.
- OT: Escribe los resultados de la simulación en archivos para cada paso de tiempo o como sea requerido.
- FP: Escribe mensajes de terminación y cierra todos los archivos.
- DA: Desaloja toda la memoria.

Como puede observarse TU, RP, CA y OT se ejecutan varias veces dentro del ciclo temporal. Cada paso en el diagrama representa ejecuciones para todos los *models*, *exchanges* y *solutions* que forman parte de la simulación.

4.2 La discretización temporal.

Una simulación se divide en intervalos de tiempo, conocidos como periodos de estrés, **durante los cuales los datos de entrada de todos los estreses externos permanecen constantes**. Los periodos de estrés se dividen a su vez en pasos de tiempo, como se muestra en la siguiente figura (véase **Timing Module** en [1], pp 10-12):

Esta componente implementa y ejecuta las operaciones del procedimiento primario *Time Update* (TU).

La tabla siguiente describe algunos de los parámetros que se usan en este módulo:

Variable	Tipo	Significado
`PERLEN`	double	es la longitud del periodo de estrés
`NSTP`	integer	es el número de pasos de tiempo en un periodo de estrés
`TSMULT`	double	es el multiplicador para calcular la longitud de pasos de tiempo sucesivos

Para calcular la longitud del primer paso de tiempo en un periodo de estrés se hace lo siguiente:

- Para TSMULT = 1:

$$\Delta t_1 = \frac{\text{PERLEN}}{\text{NSTP}}$$

- Para TSMULT $\neq$ 1:

$$\Delta t_1 = \text{PERLEN}\frac{\text{TSMULT} - 1}{\text{TSMULT}^{\text{nstp}} - 1}$$

La longitud de cada paso de tiempo sucesivo es calculada multiplicando la longitud del paso de tiempo previo por `TSMULT`.

$$\Delta t = \Delta t_{old}\text{TSMULT}$$

4.2.1 Ejemplo 1. Periodo de estrés.

Considérese un modelo de flujo GWF que usa los paquetes de *River* y *Well*, en donde el tiempo de simulación es de 90 días, con pasos de tiempo de un día.

- Si el nivel del río o el bombeo de los pozos solo cambian cada 30 días, se pueden usar 3 períodos de estrés, cada uno de 30 días.
- De igual manera, si el bombeo o el nivel del río varían cada 3 días, se usarían 30 períodos de estrés de 3 días.
- Cuando comienza un nuevo período de estrés, se deben redefinir todos los datos de estrés; sin embargo, la mayoría de los paquetes podrían reutilizar los datos del período de estrés anterior.
- En este ejemplo, se podrían reutilizar los datos del río en un nuevo período de estrés cuando solo cambian las tasas de bombeo mientras que el nivel del río permanece igual.

NOTA. Los períodos de estrés se implementan solo por conveniencia. Los paquetes que definen estreses dependientes del tiempo leen los datos de entrada en cada período de estrés. Los períodos de estrés facilitan la gestión de datos de entrada que se mantienen constantes por múltiples intervalos de tiempo. Sin embargo, no son inusuales las situaciones en las que surge la necesidad de cambiar los datos de estrés en cada paso de tiempo. En este caso, cada período de estrés debería constar de un solo paso de tiempo; como alternativa, se puede utilizar la funcionalidad de series temporales (*Time Series*) que está integrada en MODFLOW 6, véase **Utilities - Time Series** en [1], pp 27-28.

4.3 La solución numérica.

La componente *Numerical Solution* es una parte clave del simulador, ya que se encarga de configurar el solucionador que se usará para resolver las ecuaciones de uno o más modelos y de los intercambios que los conectan.

En términos simples en esta componente se definen:

- el tipo de solucionador numérico que se usará (ej. Newton-Raphson, métodos iterativos lineales (CG o BiCGSTAB),
- los criterios de convergencia (número máximo de iteraciones, tolerancias, etc.),
- la asociación entre modelos y la solución numérica (por ejemplo, un modelo de flujo puede estar vinculado con una o varias soluciones numéricas en caso de que haya múltiples modelos o dominios acoplados).

El modelo numérico de GWF formula un sistema lineal de ecuaciones de la forma:

$$\mathbf{Ax} = \mathbf{b}$$

donde **A** es la matriz de coeficientes, **x** es el vector de variables dependientes (la carga hidráulica, por ejemplo) y **b** es el lado derecho del sistema con valores conocidos (ej. las condiciones iniciales y de frontera).

- En la figura de la izquierda, tomada de [1], se muestra un ejemplo ilustrativo de una matriz para tres modelos numéricos, con sus respectivos intercambios que los conectan entre sí, que se resuelve por un solo componente de tipo *Numerical Solution*.
- La componente *Numerical Solution* resuelve el sistema de ecuaciones para uno o más modelos usando métodos iterativos.
- Estos métodos son capaces de manejar matrices dispersas, simétricas y no simétricas, de tal manera que es posible resolver problemas en mallas no estructuradas, formulaciones tipo Newton-Raphson, flujo de agua anisotrópico, transporte de solutos dispersivo, entre otros.

Jerarquía de clases de los solucionadores numéricos.

- *BaseSolutionType* es la clase base general para cualquier tipo de solución en MODFLOW 6; de esta superclase deben heredar todas las clases para calcular las soluciones numéricas.
- *NumericalSolutionType* Es una especialización de *BaseSolutionType*, que está orientada exclusivamente a métodos numéricos iterativos. Está diseñada para resolver uno o más modelos numéricos.
- *IterativeModelSolutionType* (IMS) es una implementación concreta de la clase *Numerical-SolutionType* y se encarga de resolver uno o varios modelos acoplados. En una simulación se crean objetos de esta clase para calcular la solución numérica.
- Los procedimientos primarios de la clase *BaseSolutionType* se muestran en la figura de la derecha (tomada de [1]).
- Observése que el programa principal ejecuta procedimientos específicos de *BaseSolution-Type*.
- Obsérvese también que en este caso no se ejecuta el procedimiento TU, pues esta es tarea del *TimingModule*.
- En el procedimiento CA es donde se implementan los métodos iterativos para resolver el sistema de ecuaciones generado por los modelos e intercambios que se agreguen a la simulación.

- Cuando el sistema es no lineal, se resuelve una forma linealizada del sistema de manera iterativa usando métodos iterativos y precondicionados.
- El solucionador descrito en [3] ha sido extendido en MODFLOW 6 para incluir los aceleradores lineales CG, BiCGSTAB e ILU.
- La naturaleza no lineal de los sistemas de ecuaciones, se gestiona con los métodos *backtracking*, *pseudo-transient continuation*, *under-relaxation methods* y *Newton Dampening*.

Las siguientes figuras muestran diagramas de flujo de los métodos lineales y no lineales implementados en el procedimiento CA.

4.4 Los modelos.

Jerarquía de clases de los modelos.

- Todos los modelos deben ser subclases de la clase principal *BaseModelType*.
- Un modelo numérico está definido por la clase *NumericalModelType* y es un tipo especial de un modelo diseñado para colaborar con la clase *NumericalSolutionType*.
- El modelo GWF está definido en la clase *GwfModelType* que es una subclase de la clase *NumericalModelType*.

En MODFLOW 6 un modelo es una componente principal de la simulación y representa un proceso hidrológico (flujo subterráneo, flujo laminar o turbulento en conductos, flujo superficial, transporte de calor o de soluto, entre otros).

- Los procedimientos que se ejecutan de *BaseModelType* , desde el programa principal, son los que se muestran en la figura de la izquierda (tomada de [1]).
- Estos métodos o procedimientos, generalmente están vacíos (son abstractos en términos de la POO) y pueden ser implementados por métodos que están definidos en las subclases de *BaseModelType* (sobrecarga).
- Obsérvese que en este caso no se ejecutan los procedimientos TU ni CA, pues ellos están definidos en otras componentes de la simulación.
- Esta construcción hace más fácil la adaptación y modificación de nuevos modelos.

4.4.1 Modelos numéricos.

En la figura se muestran los procedimientos primarios que se ejecutan de las clases *NumericalModelType* y *NumericalSolutionType*.

- CR: Crea el modelo numérico y los paquetes requeridos.
- DF: Define el modelo numérico leyendo la información del tamaño del modelo.
- AC: Adiciona las conexiones del modelo a Numerical Solution reservando espacios dentro de la matriz de coeficientes.
- MC: Crea un arreglo de índices que mapea las conexiones del modelo dentro del sistema de ecuaciones.
- AR: Asigna memoria para los arreglos del modelo y lee información que será constante durante toda la simulación.
- RP: Lee información del modelo de archivos, conforme es requerida, para actualizar los estreses hidrológicos u otras entradas que varían con el tiempo.
- AD: Avanza el modelo al siguiente paso en el tiempo, típicamente almacenando el valor anterior de las variables dependientes.
- CF: Calcula y actualiza los coeficientes que dependen de los resultados de la última iteración.
- FC: Calcula y agrega los términos del modelo a la matriz de coeficientes y al RHS de Numerical Solution.
- NR: Calcula y agrega los términos del método de Newton-Raphson del modelo a la matriz de coeficientes y al RHS de Numerical Solution.
- CC: Realiza una revisión de la convergencia en las variables dependientes que no son parte del modelo numérico.
- ND: Ajusta los valores de las variables dependientes, lo que puede mejorar la convergencia para modelos que usan la formulación de Newton-Raphson.
- BD: Calcula el balance con base en la solución actualizada para la variable dependiente.
- OT: Escribe los resultados de la simulación en archivos para cada paso de tiempo o como sea requerido.
- FP: Escribe mensajes de terminación y cierra todos los archivos.
- DA: Desaloja toda la memoria.

4.4.2 Groundwater Flow Model (GWF)

El modelo GWF es un tipo específico de un modelo numérico y está definido en la clase *GwfModelType* la cual está diseñada para colaborar con la clase *IterativeModelSolutionType*. La figura muestra un diagrama de flujo donde se muestran los procedimientos primarios que son ejecutados de ambas clases, veáse [4].

- CR: crea un objeto del modelo GWF con el nombre especificado y es almacenado en una lista de modelos. También crea objetos de todos los paquetes asociados a la simulación, además asigna valores a algunos parámetros del modelo.

- DF: abre archivos, lee la información acerca del tamaño del modelo, la conectividad de las celdas de la malla y crea los objetos de los paquetes que definen las condiciones de frontera.
- AC: informa al solucionador IMS sobre el número total de celdas del modelo GWF y cómo están conectadas entre ellas; esta información es usada por IMS para configurar el tamaño y patrón de conectividad de la matriz del sistema de ecuaciones.
- MC: este procedimiento se ejecuta después de que el patrón de conectividad ha sido determinado para IMS. Se crea un arreglo que mapea la posición de las celdas del modelo GWF con sus conecciones dentro del sistema de ecuaciones. Esta información es usada por GWF que son parte de los procedimientos que agregan términos al sistema de ecuaciones.
- AR: se determinan algunos parámetros hidrológicas y se reserva memoria para los componentes de la simulación que no fueron definidas por el procedimiento DF. También se leen los datos que se mantienen constantes de un periodo de estrés a otro. Estos datos incluyen: parte de la información de las fronteras, cargas hidráulicas iniciales y propiedades hidráulicas del acuífero. Se realizan algunos cálculos preliminares para preparar los datos para un procesamiento posterior.
- RP: se ejecuta al inicio de cada paso de tiempo, pero solamente lee y procesa bloques de información si es el primer paso de tiempo del periodo de estress. Esta información incluye razones de bombeo y áreas de recarga, por ejemplo.
- AD: inicializa la carga, realiza realiza sustitución de series de tiempo y ejecuta los procedimientos de los paquetes de fronteras individuales. También realiza otros procesamientos que deben ser efectuados al principio de cada paso de tiempo.
- CF: calcula términos que serán requeridos en otros procedimientos. También rehumedece las celdas, si es necesario, y vuelve inactivas las celdas secas.
- FC: adiciona los coeficients a la matriz A y al vector de lado derecho b.
- NR: adiciona los términos de Newton-Raphson a la matriz A y al vector de lado derecho b. Solo se invoca cuando el método de Newton-Raphson es usado por el modelo GWF.
- CC: se realiza una verificación de la convergencia de variables que no son parte de la solución numérica, particularmente sobre paquetes avanzados que trabajan sobre la frontera.
- ND: se ejecuta el procedimiento Newton Dampening para amortiguar cambios de cargar grandes e irreales. Específicamente, si la carga simulada de una celda está por debajo del límite inferior del modelo, esta se ajusta hacia arriba, hacia el límite inferior del modelo.
- BD: calcula los términos del balance de flujo para el modelo GWF y los paquetes de frontera.
- FP: escribe la salida final del modelo y realiza algunas tareas de procesamiento final.
- DA: Desaloja toda la memoria utilizada por el modelo.

5 Paquetes.

En la sección anterior se describió que el modelo GWF opera ejecutando una secuencia de procedimientos primarios, y estos son fundamentales desde el punto de vista de un programa de cómputo. No obstante, los usuarios prefieren pensar en las caracterísiticas de un programa en términos de sus capacidades para resolver problemas hidrológicos.

Para este propósito, el modelo GWF se divide en paquetes. Un paquete es la parte del modelo que se ocupa de un único aspecto de la simulación. Por ejemplo los paquetes *River* y *Well* se encargan de todos los aspectos de los ríos y de los pozos, respectivamente.

La siguiente figura, tomada de [4], muestra un ejemplo de configuración de GWF con varios paquetes.

Existen tres tipos de paquetes hidrológicos y un paquete de salida. La tabla siguiente, tomada de [4], muestra la lista completa de paquetes para GWF.

- **Hydrologic/Internal**. Paquetes para simular el flujo entre celdas adyacentes o para manejar cambios de almacenamiento para todas las celdas del modelo. En general se usan para calcular los términos necesarios para resolver la ecuación de flujo de agua subterránea para cada celda del modelo o para almacenar la información necesaria para calcular estos términos.
  - **DIS**, **DISV**, **DISU**. Discretización espacial. Paquete para calcular o gestionar áreas y volúmenes de cada celda de la malla, así como las propiedades geométricas de las conexiones entre ellas.
  - **IC** (Initial Conditions). Condiciones iniciales. Paquete para leer los valores iniciales de la carga hidráulica para el modelo GWF.
  - **NPF** (Node-Property Flow). Propiedades de flujo de los nodos. Paquete para calcular la conductancia hidráulica entre celdas adyacentes, gestiona la humectación y el secado de las celdas y calcula el flujo entre celdas adyacentes.
  - **HFB** (Horizontal Flow Barrier). Barrera de flujo horizontal. Paquete complementario que, en conjunto con el paquete NPF, permite modificar las conductancias para simular una barrera horizontal entre nodos adyacentes.
  - **GNC** (Ghost Node Correction). Corrección de nodos fantasma. Paquete complementario que, en conjunto con el paquete NPF, permite mejorar la precisión de los cálculos de flujo para algunos tipos de malla.
  - **STO** (Storage). Almacenamiento. Paquete para calcular el cambio en el volumen de agua que ocurre durante un paso de tiempo.

- **Hydrologic/Stress**. Paquetes que formulan los coeficientes que describen flujos externos o en la frontera del dominio, como ríos, pozos y recarga. Por ejemplo, el paquete *River* calcula los coeficientes que describen el flujo entre una celda y un río superficial. Los paquetes de este tipo son siete: **CHD**, **WEL**, **RCH**, **RIV**, **GHB**, **DRN** y **EVT**. Estos paquetes proveen la funcionalidad que ha estado presente en versiones previas de MODFLOW.
- **Hydrologic/Advanced Stress**. Paquetes para simular estreses más avanzados. Estos paquetes incluyen **MAW**, **SFR**, **UZF** y **MVR**.
- **Output**. Paquetes que no entran en la categoría de paquetes hidrológicos:
  - **OBS** (Observation). Paquete para la observación e impresión de la salida.
  - **OC** (Output Control). Paquete para el guardado de los resultados en archivos.

6 Referencias

[1] Hughes, J.D., Langevin, C.D., and Banta, E.R., 2017, Documentation for the MODFLOW 6 framework: U.S. Geological Survey Techniques and Methods, book 6, chap. A57, 40 p. https://doi.org/10.3133/tm6A57.

[2] Langevin, C. D., Hughes, J. D., Provost, A. M., Russcher, M. J., & Panday, S. (2023). MODFLOW as a configurable Multi-Model Hydrologic Simulator. Ground Water. https://doi.org/10.1111/gwat.13351.

[3] Hughes, J.D., and White, J.T., 2013, Use of general purpose graphics processing units with MODFLOW: Groundwater, v. 51, no. 6, p. 833–846, accessed June 27, 2017. https://doi.org/10.1111/gwat.12004.

[4] Langevin, C.D., Hughes, J.D., Provost, A.M., Banta, E.R., Niswonger, R.G., and Panday, Sorab, 2017, Documentation for the MODFLOW 6 Groundwater Flow (GWF) Model: U.S. Geological Survey Techniques and Methods, book 6, chap. A55, 197 p., accessed August 4, 2017. https://doi.org/10.3133/tm6A55.

7 MODFLOW 6: inicializando una simulación.

7.1 Resumen.

El propósito de este documento es describir cómo se realiza una simulación de flujo usando el modelo GWF de MODFLOW 6. Esta descripción combina los conceptos descritos en la documentación de MODFLOW 6 y en el software ***flopy*** (repositorio, documentación). El software flopy permite simplificar la generación de los archivos de entrada para una simulación a través de Python, además de realizar la ejecución de la simulación y el post-procesamiento de la salida. No se hace una descripción detallada, sino que solo se explican los conceptos principales y se relacionan con los archivos de entrada requeridos por MODFLOW 6 y con los objetos de flopy. Se hace una configuración de una simulación agregando las componentes *Timing module*, *Numerical Solution* (IMS) y un *GWF model*, usando parámetros reducidos.

7.2 Ejemplo. Las tres componentes de una simulación con flopy.

Una simulación simple con GWF requiere de las componentes que se muestran en la siguiente figura:

Figura 1. Diagrama que muestra las componentes que se usan para la simulación de un solo modelo GWF. Las componentes son: (1) una simulación (que es el programa principal), (2) una componente para gestionar el tiempo, (3) una componente para gestionar la solución numérica y (4) un único modelo GWF. Figura tomada de [1].

Para definir las tres componentes mostradas en la figura anterior hacemos uso de la biblioteca *flopy*. Realizaremos este proceso para un ejemplo simple, cuya única utilidad es explicar el proceso (sin resolver un problema de flujo).

7.2.1 Paso 1. Inicialización de la simulación.

La simulación se inicia creando un objeto de la clase `flopy.mf6.MFSimulation` para cargar, construir y salvar los archivos de una simulación de MODFLOW 6. Se debe crear un objeto

de este tipo antes que cualquier otro objeto de las componentes o paquetes de MODFLOW 6 que se vayan a usar en la simulación.

A continuación creamos el objeto `o_sim` de la clase `flopy.mf6.MFSimulation` como sigue:

```
import os
import flopy

# --- Ruta a los ejecutables de MODFLOW 6. Opciones de SO's: linux, macos, macosarm, windows
#exe_name = "C:\\Users\\luiggi\\Documents\\GitSites\\xmf6\\mf6\\windows\\mf6"
exe_name = "../../../mf6/macosarm/mf6"
```

```
# Creación del objeto 'sim' para la simulación. Es el que controla todo.
o_sim = flopy.mf6.MFSimulation(
    sim_name = "flow",
    exe_name = exe_name,
    sim_ws   = "sandbox1"
)
```

- Al ejecutar la celda de código anterior, se crea la carpeta `sandbox1` en donde se almacenarán todos los archivos, de entrada y de salida, que genera MODFLOW 6. Este es el espacio de trabajo (*workspace*) que por ahora estará vacío.
- El nombre de la simulación será `flow` y este nombre será usado para generar archivos de entrada y salida, con las extensiones correspondientes.
- Una vez creado un objeto de esta clase, los demás objetos que contribuyen a la simulación se deben enlazar al objeto de la simulación `o_sim`.
- Con la instrucción `print(o_sim)` es posible imprimir información de los atributos del objeto `o_sim`, como se hace en la siguiente celda:

```
print(o_sim)
```

```
sim_name = flow
sim_path = /Users/luiggi/GitSites/xmf6/src/examples/00_MF6/sandbox1
exe_name = ../../../mf6/macosarm/mf6

####################
Package mfsim.nam
####################

package_name = mfsim.nam
filename = mfsim.nam
```

```
package_type = nam
model_or_simulation_package = simulation
simulation_name = flow
```

- También, con el método `write_simulation()` es posible escribir en archivos la información que usará MODFLOW 6 como entrada para realizar la simulación:

```
o_sim.write_simulation()
```

```
writing simulation...
  writing simulation name file...
  writing simulation tdis package...

AttributeError: 'NoneType' object has no attribute 'write'
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[4], line 1
----> 1 o_sim.write_simulation()
File ~/Library/Python/3.12/lib/python/site-packages/flopy/mf6/mfsimbase.py:1711, in MFSimulat
   1706 if (
   1707     self.simulation_data.verbosity_level.value
   1708     >= VerbosityLevel.normal.value
   1709 ):
   1710     print("  writing simulation tdis package...")
-> 1711 self._tdis_file.write(ext_file_action=ext_file_action)
   1713 # write solution files
   1714 for solution_file in self._solution_files.values():
AttributeError: 'NoneType' object has no attribute 'write'
```

- Observa que se obtiene un error (`AttributeError`) y esto es debido a que se requiere de agregar la componente para la gestión del tiempo de la simulación. Esto lo haremos en el siguiente paso.
- Se genera el archivo `mfsim.nam`, el cual es necesario para ejecutar la simulación. Este archivo se almacena en la carpeta del espacio de trabajo, en este caso en `sandbox1` y por ahora no contiene mucha información:

```
# File generated by Flopy version 3.9.2 on 05/09/2025 at 14:31:30.
BEGIN options
END options

BEGIN exchanges
END exchanges
```

- Este archivo se actualizará conforme se agreguen componentes a la simulación.

7.2.2 Paso 2. Discretización temporal.

Es necesario definir los parámetros para gestionar el tiempo de la simulación. En este caso usaremos:

- Unidades: `DAYS`
- Periodos de estrés: `NPER` $= 1$
- Datos del periodo de estrés: `(PERLEN, NSTP, TSMULT )` $\$ = (1.0, 1, 1.0)\$$

Ejecutamos la siguiente celda:

```
o_tdis = flopy.mf6.ModflowTdis(
    simulation = o_sim,
    time_units = "DAYS",
    nper = 1,
    perioddata = [(1.0, 1, 1.0)]
)
```

- En la celda anterior se construye el objeto `o_tdis` que es de tipo `flopy.mf6.ModflowTdis`.
- El primer parámetro es el objeto `o_sim`, es decir la simulación. De esta manera la simulación conoce los parámetros de la discretización del tiempo.
- El parámetro `perioddata` es una lista que contiene tuplas, cada una de ellas con los datos `(PERLEN, NSTP, TSMULT)` para cada periodo de estrés.
- También es posible usar la instrucción `print()` para imprimir información de los atributos de este objeto, veamos:

```
print(o_tdis)
```

```
package_name = flow.tdis
filename = flow.tdis
package_type = tdis
model_or_simulation_package = simulation
simulation_name = flow

Block options
--------------------
time_units
{internal}
(days)
```

```
Block dimensions
--------------------
nper
{internal}
(1)


Block perioddata
--------------------
perioddata
{internal}
([(1., 1, 1.)])
```

- Obsérva que se usa el nombre de archivo `filename = flow.tdis` para almacenar la información del tiempo, esto es porque el nombre de la simulación es `flow`.
- Para generar este archivo hacemos lo siguiente:

```
o_sim.write_simulation()
```

```
writing simulation...
  writing simulation name file...
  writing simulation tdis package...
```

- La instrucción anterior actualiza el archivo `mfsim.nam`. Este archivo contendrá información de los objetos (componentes) que contribuyen a la simulación. Después de ejecutar la celda anterior el contenido de este archivo debe ser similar a lo siguiente:

```
# File generated by Flopy version 3.9.2 on 04/20/2025 at 12:02:33.
BEGIN options
END options

BEGIN timing
  TDIS6  flow.tdis
END timing

BEGIN exchanges
END exchanges
```

- Nota que solo se ha incluido la información del módulo de tiempo. Esta información será actualizada más adelante.

- Se genera también el archivo `flow.tdis` con la información para la discretización temporal. El contenido de este archivo es como sigue:

```
# File generated by Flopy version 3.9.2 on 04/20/2025 at 12:02:33.
BEGIN options
  TIME_UNITS  days
END options

BEGIN dimensions
  NPER  1
END dimensions

BEGIN perioddata
       1.00000000  1       1.00000000
END perioddata
```

7.2.3 Paso 3. Solución numérica.

Para cada modelo se requiere un objeto que calcule la solución numérica.

En la celda siguiente se define un objeto de la clase `flopy.mf6.ModflowIms`, se imprime la información y se escriben los archivos correspondientes.

```
o_ims = flopy.mf6.ModflowIms(simulation = o_sim)
print(o_ims)
o_sim.write_simulation()
```

```
package_name = ims_-1
filename = flow.ims
package_type = ims
model_or_simulation_package = simulation
simulation_name = flow


writing simulation...
  writing simulation name file...
  writing simulation tdis package...
  writing solution package ims_-1...
```

- El objeto `o_ims` representa a la solución numérica; el único parámetro que se usó en este ejemplo fue el objeto de la simulación para ligar la solución numérica con el modelo que se va a resolver.

- Se pueden agregar muchos más parámetros para configurar los métodos iterativos de solución.
- En este caso la información se guarda en el archivo `flow.ims` y su contenido es mínimo debido a que no usan más parámetros:

```
# File generated by Flopy version 3.9.2 on 04/20/2025 at 12:18:32.
BEGIN options
END options
```

7.2.4 Paso 4. Modelo GWF.

Ahora vamos a agregar un modelo numérico a la simulación. Para ello creamos un objeto de la clase `flopy.mf6.ModflowGwf`, imprimimos sus parámetros en pantalla y generamos los archivos correspondientes como sigue:

```
o_gwf = flopy.mf6.ModflowGwf(
    simulation = o_sim,
    modelname = "flow",
    model_nam_file = "flow.nam",
)

print(o_gwf)
o_sim.write_simulation()
```

```
name = flow
model_type = gwf6
version = mf6
model_relative_path = .


writing simulation...
  writing simulation name file...
  writing simulation tdis package...
  writing solution package ims_-1...
  writing model flow...
    writing model name file...
```

- Observa que el primer parámetro también es el objeto de la simulación `o_sim`.
- Adicionalmente se agrega el nombre del modelo, que en este caso es igual al de la simulación, y el nombre del archivo donde se guardará la información del modelo GWF.

- El código de la celda anterior crea el archivo `flow.nam` que por ahora tiene información reducida:

```
# File generated by Flopy version 3.9.2 on 04/20/2025 at 12:34:26.
BEGIN options
END options
```

Posteriormente, este archivo contendrá la información de los paquetes que se requieran para realizar la simulación.

- La instrucción `o_sim.write_simulation()` actualiza también el archivo `mfsim.nam` con la información del solucionador y del modelo GWF:

```
# File generated by Flopy version 3.9.2 on 04/20/2025 at 12:25:36.
BEGIN options
END options

BEGIN timing
  TDIS6  flow.tdis
END timing

BEGIN models
  gwf6  flow.nam  flow
END models

BEGIN exchanges
END exchanges

BEGIN solutiongroup  1
  ims6  flow.ims  flow
END solutiongroup  1
```

- Observa que ahora este archivo contiene la información de las tres componentes necesarias para iniciar una simulación: Timing, Models y SolutionGroup. En este ejemplo no se agregan intercambios, pues se trata de un solo modelo.

Más información acerca de los archivos de entrada y salida de MODFLOW 6 se puede encontrar en [2].

En este ejemplo se ha construido el esquema requerido para iniciar una simulación de flujo con GWF de MODFLOW 6 usando las herramientas de flopy. Para que la simulación tenga más sentido, se requiere de agregar paquetes al modelo GWF, lo cual se hará en la notebook 02_MF6_GWF_paq.ipynb.

8 Referencias

[1] Langevin, C. D., Hughes, J. D., Provost, A. M., Russcher, M. J., & Panday, S. (2023). MODFLOW as a configurable Multi-Model Hydrologic Simulator. Ground Water. https://doi.org/10.1111/gwat.13351.

[2] MODFLOW 6 – Description of Input and Output. Version mf6.4.4—February 13, 2024. U.S. Department of the Interior. U.S. Geological Survey.. Archivo: mf6io.pdf de la documentación de MODFLOW 6 que se puede obtener de https://github.com/MODFLOW-ORG/modflow6/releases.

9 MODFLOW 6: modelo GWF.

9.1 Resumen.

En este documento se realiza una simulación de flujo en dos dimensiones explicando paso a paso el proceso y describiendo los paquetes del modelo GWF que son usados. Se utilizan las herramientas de flopy para facilitar el pre y postprocesamiento de la información requerida por el simulador.

10 Introducción.

El modelo GWF (*Groundwater Flow*) de MODFLOW 6 es un componente principal que simula el flujo del agua subterránea en medios porosos, como acuíferos.

Este modelo resuelve la ecuación de flujo en tres dimensiones, permitiendo representar cómo el agua se mueve bajo tierra en función de las propiedades del subsuelo, las condiciones de frontera y las fuentes o sumideros, como recarga, pozos y ríos.

El modelo GWF se define dentro de una simulación general y puede conectarse con otros modelos, como el de transporte de solutos (GWT) o modelos de flujo de superficie. Esto permite construir simulaciones integradas con diferentes procesos acoplados.

Un modelo GWF puede ser configurado por varios paquetes. Un paquete es la parte del modelo que se ocupa de un único aspecto de la simulación.

11 Ejemplo: Simulación de flujo en 2D.

En este ejemplo se simula una distribución de carga hidráulica en un dominio rectangular de 15 m $\times$ 10 m con una sola capa con espesor de 1 m. La carga va de 10 m en el lado izquierdo a 5 m en lado derecho, formando un gradiente horizontal de flujo. Los lados superior e inferior del dominio son impermeables. No hay fuentes ni sumideros. El valor de la conductividad hidráulica es constante $k = 1.0$.

11.1 Paso 1. Configuración inicial.

- Haremos la configuración inicial de la simulación de manera similar al ejemplo de la notebook 01_MF6_GWF_init.ipynb.
- En este caso agregaremos algunas opciones adicionales al solucionador IMS para ver el efecto que tiene en los archivos de entrada, particularmente en `flow.ims`, véase **Iterative Model Solution**, pp 225-231, en [2].
- El espacio de trabajo donde se escriben todos los archivos será la carpeta `sandbox2`.
- La simulación así como el modelo de flujo tendrán el nombre `flow`.
- Todo lo anterior se realiza en la siguiente celda de código:

```
import flopy
import numpy as np
import matplotlib.pyplot as plt
import os

# Biblioteca con algunas herramientas adicionales que se puede instalar como sigue:
#pip install git+https://github.com/luiggix/xmf6
import xmf6

# --- Ruta a los ejecutables de MODFLOW 6. Opciones de SO's: linux, macos, macosarm, windows
exe_name = "C:\\Users\\luiggi\\Documents\\GitSites\\xmf6\\mf6\\windows\\mf6"
#exe_name = "../../../mf6/macosarm/mf6"

# --- Configuración inicial ---
sim_name = "flow"
sim_ws = "sandbox2"
```

```
o_sim = flopy.mf6.MFSimulation(
    sim_name = sim_name,
    exe_name = exe_name,
    sim_ws   = sim_ws
)

# --- Tiempos de simulación ---
o_tdis = flopy.mf6.ModflowTdis(
    simulation = o_sim,
    time_units = "DAYS",
    nper = 1,
    perioddata = [(1.0, 1, 1.0)],
)

# --- Solucionador numérico (IMS) ---
o_ims = flopy.mf6.ModflowIms(
    simulation          = o_sim,
    print_option        = "SUMMARY",
    complexity          = "SIMPLE",
    outer_dvclose       = 1e-4,
    outer_maximum       = 50,
    under_relaxation    = "NONE",
    linear_acceleration = "CG",
    inner_maximum       = 30,
    inner_dvclose       = 1e-6,
)

# --- Modelo GWF ---
o_gwf = flopy.mf6.ModflowGwf(
    simulation = o_sim,
    modelname = sim_name,
    model_nam_file = f"{sim_name}.nam",
    save_flows = True # Almacena los flujos, particularmente el budget
)

# --- Escribimos los archivos ---
o_sim.write_simulation()
```

```
writing simulation...
  writing simulation name file...
  writing simulation tdis package...
```

```
writing solution package ims_-1...
writing model flow...
  writing model name file...
```

Observación.

Revisa los archivos generados en la carpeta `sandbox2`, particularmente el archivo `flow.ims`.

11.2 Paquetes.

Para generar esta simulación se requiere de agregar paquetes al modelo GWF como se muestra en la siguiente figura, tomada de [1]:

11.3 Paso 2. Discretización espacial.

El dominio espacial de una simulación en MODFLOW 6 se divide en una rejilla de celdas y provee de tres paquetes diferentes (véase **Chapter 3. Spatial Discretization** en [1]):

- **DIS**. Paquete para discretización estructurada.
- **DISV**. Paquete para discretización por vértices.
- **DISU**. Paquete para discretización NO estructurada.

Solo se puede usar uno de los paquetes de discretización en una simulación con GWF.

La información almacenada en los paquetes de discretización es usada durante la simulación por GWF para realizar varias acciones, por ejemplo: * calcular la conductividad hidráulica entre celdas conectadas, * calcular los volúmenes de celda para los cálculos de almacenamiento, * convertir los flujos de recarga y evapotranspiración, por ejemplo de las dimensiones de L/T a L^3/T.

Los paquetes de discretización: * contienen información acerca de las celdas, como las elevaciones superior (`TOP`) e inferior (`BOTTOM`) de la celda y las dimensiones o áreas de la celda (en vista de planta). * definen la conectividad de las celdas para describir cómo están conectadas entre sí las celdas del modelo y las propiedades geométricas de las conexiones. * En el paquete **DIS**, la conectividad de la celda se describe implícitamente por la estructura de la malla, las dimensiones de la celda en las direcciones de renglones y columnas, y las elevaciones de la parte superior e inferior de la celda.

La siguiente figura, tomada de [1], muestra un ejemplo de una malla regular en tres dimensiones en donde se definen columnas, renglones y capas.

- En este ejemplo tenemos 9 columnas (`NROW`), 5 renglones (`NCOL`) y 5 capas (`NLAY`).

- El tamaño horizontal de las celdas está definido por Δr_i y Δc_i (variables `DELR`$_i$ y `DELC`$_i$).
- Observa que Δr_i se mide en la dirección de los renglones (en el eje x en un sistema Cartesiano), es decir, mide el tamaño de la columna i-ésima.
- Por otro lado, Δc_i se mide en la dirección de las columnas (en el eje y), por lo que mide el tamaño del renglón i-ésimo
- Las capas se numeran empezando por la capa superior y hacia abajo.
- Las elevaciones de las celdas se especifican en la variable `BOTM` (indica la parte inferior de cada capa).
- La elevación de la parte superior de la primera capa se debe especificar en la variable `TOP`.
- La información de las elevaciones se usa para calcular el espesor de las celdas.

Parámetros a usar.

- Para obtener el modelo del dominio rectangular requerido, una capa de 15 m $\times$ 10 m con un espesor de 1 m, usaremos los parámetros `nrow`, `ncol`, `nlay`, `delr` y `delc`.
- Para obtener la longitud correcta del dominio se debe cumplir que:
  - `nrow` * `delc` $ = 15$,
  - `ncol` * `delr` $ = 10$.
- Los parámetros `top` y `botm` deben ser tales que `top - botm` $= 1$.
- En la siguiente celda se definen estos parámetros para generar la malla del dominio:

```
# --- Discretización espacial ---
nlay = 1
nrow = 10
ncol = 15
delr = 1.0
delc = 1.0
top  = 0.0
botm = -1.0

o_dis = flopy.mf6.ModflowGwfdis(
    model = o_gwf,
    length_units = "METERS",
    nlay = nlay,
    nrow = nrow,
    ncol = ncol,
    delr = delr,
    delc = delc,
```

```
    top  = top,
    botm = botm,
)

print(o_dis)
o_sim.write_simulation()
```

```
package_name = dis
filename = flow.dis
package_type = dis
model_or_simulation_package = model
model_name = flow

Block options
---------------------
length_units
{internal}
(meters)


Block dimensions
---------------------
nlay
{internal}
(1)

nrow
{internal}
(10)

ncol
{internal}
(15)


Block griddata
---------------------
delr
{constant 1.0}

delc
{constant 1.0}
```

```
top
{constant 0.0}

botm
{constant -1.0}
```

```
writing simulation...
  writing simulation name file...
  writing simulation tdis package...
  writing solution package ims_-1...
  writing model flow...
    writing model name file...
    writing package dis...
```

Explicación.

- En la celda anterior se construye el objeto `o_dis` de la clase `flopy.mf6.ModflowGwfdis` para generar la información de la discretización espacial. Esta clase construye una discretiación de tipo DIS.
- El primer parámetro en esta construcción es el objeto del modelo al que esta discretización pertenece, en este caso `o_gwf`. Posteriormente se agregan todos los parámetros de la discretización.
- La celda anterior genera el archivo `flow.dis` con la información correspondiente a la discretización espacial, revisa este archivo en la carpeta `sandbox2`.
- Adicionalmente, el archivo `flow.nam` es modificado agregando la información del paquete DIS, deberías ver un bloque como el siguiente en ese archivo:

```
BEGIN packages
  DIS6  flow.dis  dis
END packages
```

Flopy contiene herramientas para obtener la información de la discretización espacial y hacer una gráfica del dominio. Por ejemplo:

- `o_gwf.modelgrid`: regresa un objeto que contiene toda la información de la discretización.
- `o_gwf.modelgrid.extent`: extensión del dominio (xmin, xmax, ymin, ymax).
- `o_gwf.modelgrid.nrow`: número de renglones.
- `o_gwf.modelgrid.ncol`: número de columnas.

- `o_gwf.modelgrid.nlay`: número de capas.
- `o_gwf.modelgrid.delr`: tamaño de las celdas en dirección de los renglones (coord. x, tamaño de las columas).
- `o_gwf.modelgrid.delc`: tamaño de las celdas en dirección de las columnas (coord. y, tamaño de los renglones).
- `o_gwf.modelgrid.delz`: tamaño de las celdas en dirección vertical (coord. z, tamaño de las capas).
- `o_gwf.modelgrid.top`: parte superior de la primera capa.
- `o_gwf.modelgrid.botm`: parte inferior de todas las celdas.
- `o_gwf.modelgrid.xyzcellcenters`: centros de las celdas de la malla

Veamos como podemos obtener y usar esta información.

```
# Obtenemos el objeto con la información de la discretización espacial
grid = o_gwf.modelgrid

grid.extent
print("Extensión del dominio (xmin, xmax, ymin, ymax)")
print(grid.extent[0], grid.extent[1], grid.extent[2], grid.extent[3])

print("\nNúmero de renglones, columnas y capas")
print(grid.nrow, grid.ncol, grid.nlay)
```

```
Extensión del dominio (xmin, xmax, ymin, ymax)
0.0 15.0 0.0 10.0

Número de renglones, columnas y capas
10 15 1
```

```
print(len(grid.delr), grid.delr)
# grid.delr es un numpy.ndarray 1D de tamaño ncol
# grid.delc es un numpy.ndarray 1D de tamaño nrow
# grid.delz es un numpy.ndarray 3D de tamaño nlay x ncol x nrow
# Usamos la función xmf6.info_array() para imprimir las propiedades de arreglos de numpy
print("\ngrid.delr =", grid.delr[0])
xmf6.info_array(grid.delr)
print("\ngrid.delc =", grid.delc[0])
xmf6.info_array(grid.delc)
print("\ngrid.delz =", grid.delz[0][0][0])
xmf6.info_array(grid.delz)
```

```
15 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

```
grid.delr = 1.0
 tipo  : <class 'numpy.ndarray'>
 dtype : float64
 dim   : 1
 shape : (15,)
 size(bytes) : 8
 size(elements) : 15

grid.delc = 1.0
 tipo  : <class 'numpy.ndarray'>
 dtype : float64
 dim   : 1
 shape : (10,)
 size(bytes) : 8
 size(elements) : 10

grid.delz = 1.0
 tipo  : <class 'numpy.ndarray'>
 dtype : float64
 dim   : 3
 shape : (1, 10, 15)
 size(bytes) : 8
 size(elements) : 150
```

```
# grid.top es un numpy.ndarray 2D de tamaño nrow x ncol
# grid.botm es un numpy.ndarray 3D de tamaño nlay x nrow x ncol
print("grid.top =", grid.top[0][0])
xmf6.info_array(grid.top)
print("\ngrid.botm =", grid.botm[0][0][0])
xmf6.info_array(grid.botm)
```

```
grid.top = 0.0
 tipo  : <class 'numpy.ndarray'>
 dtype : float64
 dim   : 2
 shape : (10, 15)
 size(bytes) : 8
 size(elements) : 150

grid.botm = -1.0
 tipo  : <class 'numpy.ndarray'>
```

```
 dtype : float64
 dim   : 3
 shape : (1, 10, 15)
 size(bytes) : 8
 size(elements) : 150
```

```
# Dos maneras de calcular la longitud del dominio en las direcciones de los ejes son:
print("(1) Lx = ", grid.extent[1] - grid.extent[0])
print("    Ly = ", grid.extent[3] - grid.extent[2])
print("(2) Lx = ", grid.ncol * grid.delr[0])
print("    Ly = ", grid.nrow * grid.delc[0])
```

```
(1) Lx =  15.0
    Ly =  10.0
(2) Lx =  15.0
    Ly =  10.0
```

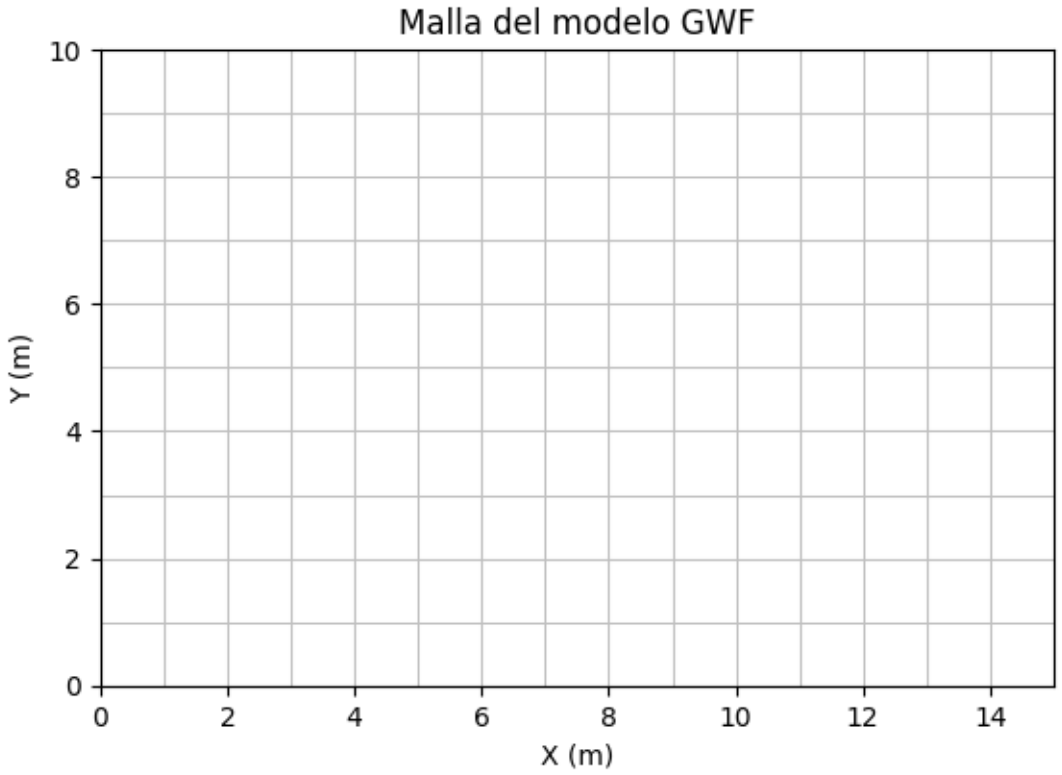
```
# grid.xyzcellcenters es una lista que contiene un numpy.ndarray de nrow x ncol
print("grid.xyzcellcenters[0]")
xmf6.info_array(grid.xyzcellcenters[0])
```

```
grid.xyzcellcenters[0]
 tipo  : <class 'numpy.ndarray'>
 dtype : float64
 dim   : 2
 shape : (10, 15)
 size(bytes) : 8
 size(elements) : 150
```

Flopy contiene herramientas para visualización. Por ejemplo podemos crear un objeto de la clase `flopy.plot.PlotMapView` para graficar la malla:

```
# Crear objeto de vista en planta
mapview = flopy.plot.PlotMapView(model=o_gwf)

# Dibujar la malla
mapview.plot_grid(linewidths=0.75, color="silver")

plt.title("Malla del modelo GWF")
plt.xlabel("X (m)")
plt.ylabel("Y (m)")
```

```
plt.gca().set_aspect('equal')  # Para mantener proporciones
plt.show()
```



11.4 Paso 3. Condiciones iniciales.

El paquete encargado de definir las condiciones iniciales es IC (*Initial Conditions*). Véase **Initial Conditions (IC) Package**, pp 43 en [2], para más información.

Se utiliza para asignar los valores iniciales de la carga hidráulica en todas las celdas activas del modelo al comenzar una simulación, especialmente importante cuando se trabaja con modelos transitorios.

Se aplica al inicio del primer periodo de estrés y es obligatorio en el modelo GWF.

En simulaciones estacionarias, los valores iniciales no afectan los resultados finales, pero aún así son requeridos por MODFLOW 6.

El modelo no podrá ejecutarse correctamente si no se proporciona este paquete.

Parámetros a usar.

- Usaremos la clase `flopy.mf6.ModflowGwfic` para crear el objeto `o_ic`.
- El primer parámetro es el modelo `o_gwf` al que se le desean agregar las condiciones iniciales.
- El parámetro `strt` indica el valor de la condición inicial para la carga hidráulica. Puede ser un número escalar (mismo valor en todo el dominio) o un arreglo 2D o 3D para especificar la carga en cada celda.

En la siguiente celda se genera una condición inicial de carga hidráulica igual a 10:

```
o_ic = flopy.mf6.ModflowGwfic(
    model = o_gwf,
    strt=10.0
)

print(o_ic)
o_sim.write_simulation()
```

```
package_name = ic
filename = flow.ic
package_type = ic
model_or_simulation_package = model
model_name = flow

Block griddata
--------------------
strt
{constant 10.0}
```

```
writing simulation...
  writing simulation name file...
  writing simulation tdis package...
  writing solution package ims_-1...
  writing model flow...
    writing model name file...
    writing package dis...
    writing package ic...
```

Explicación.

- La celda anterior genera el archivo `flow.ic` con la información para las condiciones iniciales.

- Adicionalmente se agrega la información de este paquete en el archivo `flow.nam`. Deberías obtener un bloque en ese archivo como el que sigue:

```
BEGIN packages
  DIS6  flow.dis  dis
  IC6  flow.ic  ic
END packages
```

TIP. Si se tiene un modelo con topografía variable y se desea usar la elevación superior como condición inicial, se podría hacer algo como:

```
strt = top.copy()  # usar la elevación como nivel inicial
ic = flopy.mf6.ModflowGwfic(gwf, strt=strt)
```

11.5 Paso 4. Condiciones de frontera.

EL paquete CHD tienen un funcionamiento diferente al de los otros paquetes hidrológicos de MODFLOW 6; no adiciona cambios en el sistema de ecuaciones como lo hacen otros paquetes. Lo que hace CHD es simplemente asignar un código interno indicando que:

1. la carga varía con el tiempo (celda con carga variable),
2. la carga es constante (celda con carga constante),
3. no hay flujo hacia dentro o hacia afuera de la celda (celda inactiva).

Este código es usado por la componente de solución numérica para determinar cómo debe manejar las ecuaciones.

El paquete CHD lee los datos del archivo con extensión `.chd` en donde se indica la lista de celdas con el valor correspondiente y lee la información al inicio de cada periodo de estrés, lo que hace posible cambiar el valor de la carga en dichas celdas durante la simulación.

Con este paquete se pueden imponer condiciones de frontera de tipo Dirichlet para la carga hidráulica.

Véase **Constant-Head (CHD) Package**, pp 76 en [2], para más información.

Preparación de los datos.

- Para definir las condiciones $h = 10$ en $x = 0$ y $h = 5$ en $x = 15$, primero crearemos una lista de índices y valores como sigue:

```
chd_data = []
for row in range(nrow):
    chd_data.append([(0, row, 0), 10.0])       # Condición en la pared izquierda
    chd_data.append([(0, row, ncol - 1), 5.0]) # Condición en la pared derecha

# Revisamos el resultado
for c in chd_data:
    print(c)
```

```
[(0, 0, 0), 10.0]
[(0, 0, 14), 5.0]
[(0, 1, 0), 10.0]
[(0, 1, 14), 5.0]
[(0, 2, 0), 10.0]
[(0, 2, 14), 5.0]
[(0, 3, 0), 10.0]
[(0, 3, 14), 5.0]
[(0, 4, 0), 10.0]
[(0, 4, 14), 5.0]
[(0, 5, 0), 10.0]
[(0, 5, 14), 5.0]
[(0, 6, 0), 10.0]
[(0, 6, 14), 5.0]
[(0, 7, 0), 10.0]
[(0, 7, 14), 5.0]
[(0, 8, 0), 10.0]
[(0, 8, 14), 5.0]
[(0, 9, 0), 10.0]
[(0, 9, 14), 5.0]
```

Parámetros a usar.

- Ahora creamos un objeto de la clase `flopy.mf6.ModflowGwfchd`.
- El primer parámetro será el modelo `o_gwf` al que se le impondrán las condiciones iniciales.
- En este ejemplo usamos el parámetro `stress_period_data` para definir las condiciones de frontera, que requiere una lista de la forma: `[cellid, head, aux, boundname]` donde:
  - `cellid` es el identificador de la celda.
    - Para una malla de tipo DIS el CELLID está formado por tuplas (layer, row, column).
    - Se usan arreglos que inician en 0.

        * Flopy ajustará el valor de los índices para ejecutar MODFLOW 6 en donde los índices inician en 1.
    - `head` es el valor de la carga hidráulica en la frontera.
    - los valores `aux` y `boundname` son opcionales y no se usarán por ahora.

- La lista `chd_data` creada anteriormente se usará como argumento de `stress_period_data`, veamos como:

```
o_chd = flopy.mf6.ModflowGwfchd(
    model = o_gwf,
    stress_period_data=chd_data,
)

print(o_chd)
o_sim.write_simulation()
```

```
package_name = chd_0
filename = flow.chd
package_type = chd
model_or_simulation_package = model
model_name = flow

Block period
--------------------
stress_period_data
{internal}
(    cellid_layer  cellid_row  cellid_column  head
0               0           0              0  10.0
1               0           0             14   5.0
2               0           1              0  10.0
3               0           1             14   5.0
4               0           2              0  10.0
5               0           2             14   5.0
6               0           3              0  10.0
7               0           3             14   5.0
8               0           4              0  10.0
9               0           4             14   5.0
10              0           5              0  10.0
11              0           5             14   5.0
12              0           6              0  10.0
13              0           6             14   5.0
14              0           7              0  10.0
```

```
15              0           7              14   5.0
16              0           8               0  10.0
17              0           8              14   5.0
18              0           9               0  10.0
19              0           9              14   5.0)



writing simulation...
  writing simulation name file...
  writing simulation tdis package...
  writing solution package ims_-1...
  writing model flow...
    writing model name file...
    writing package dis...
    writing package ic...
    writing package chd_0...
INFORMATION: maxbound in ('gwf6', 'chd', 'dimensions') changed to 20 based on size of stress
```

Observación.

El resultado se escribirá en el archivo `flow.chd` que contendrá la lista de celdas con el valor correspondiente.

11.6 Paso 5. Propiedades hidráulicas.

El paquete NPF (*Node Property Flow*) calcula la conductancia hidráulica, tanto vertical como horizontal, entre celdas adyancentes.

Este paquete calcula los coeficientes $C_{n,m}$ que aparecen en la siguiente forma de la ecuación CVFD:

$$\sum_{m \in \eta_n} C_{n,m}(h_m - h_n) - P_n h_n + Q_n = SS_n A_n \Delta v_n \frac{h_n - HOLD_n}{t - t_{old}}$$

NPF soporta celdas confinadas y convertibles. Para determinar el tipo de celda se usa el parámetro *ICELLTYPE*.

También, usando NPF se define la conductividad hidráulica k.

Para más información véase [1] **Chapter 4. Internal Flow Packages**, pp 4-1, y **Node Property Flow (NPF) Package**, pp 49 en [2].

Parámetros a usar.

- En este caso la conductividad hidráulica es constante $k = 1.0$.
- Usamos un objeto de la clase `flopy.mf6.ModflowGwfnpf` para definir el valor de k.
- También agregamos el parámetro `save_specific_discharge = True` para almacenar la descarga específica q.

```
o_npf = flopy.mf6.ModflowGwfnpf(
    model = o_gwf,
    save_specific_discharge = True,
    k = 1.0, # conductividad hidráulica (constante)
)

print(o_npf)
o_sim.write_simulation()
```

```
package_name = npf
filename = flow.npf
package_type = npf
model_or_simulation_package = model
model_name = flow

Block options
---------------------
save_specific_discharge
{internal}
(True)


Block griddata
---------------------
icelltype
{constant 0}

k
{constant 1.0}



writing simulation...
  writing simulation name file...
  writing simulation tdis package...
```

```
  writing solution package ims_-1...
  writing model flow...
    writing model name file...
    writing package dis...
    writing package ic...
    writing package chd_0...
    writing package npf...
```

Observación.

El código anterior genera el archivo `flow.npf` con el siguiente contenido:

```
# File generated by Flopy version 3.9.2 on 04/26/2025 at 11:32:43.
BEGIN options
  SAVE_SPECIFIC_DISCHARGE
END options

BEGIN griddata
  icelltype
    CONSTANT  0
  k
    CONSTANT       1.00000000
END griddata
```

11.7 Paso 6. Configuración de la salida.

EL paquete OC (*Output Control*) determina cómo y cuándo se imprimirá o almacenará la información de salida de la simulación carga hidráulica (head) y del *budget*.

Esta información se puede enviar a: * la pantalla de salida cuando se ejectua MODFLOW 6, * los archivos de listado `.lst`, * archivos en formato binario.

Para más información véase **Output Control (OC) Option**, pp 44 en [2].

Parámetros a usar.

- Para configurar la salida usamos un objeto de la clase `flopy.mf6.ModflowGwfoc` indicando:
  - El modelo GWF,
  - los archivos para almacenar el *budget* (`flow.bud`) y la carga hidráulica (`flow.hds`),
  - la información que se almacenará en archivos binarios usando el parámetro `saverecord`,

- la información que se enviará al archivo de listado (`flow.lst`) usando el parámetro `printrecord`.

```
o_oc = flopy.mf6.ModflowGwfoc(
    model = o_gwf,
    budget_filerecord = f"{sim_name}.bud",
    head_filerecord = f"{sim_name}.hds",
    saverecord = [("HEAD", "ALL"), ("BUDGET", "ALL")],
    printrecord = [("HEAD", "ALL")]
)

print(o_oc)
o_sim.write_simulation()
```

```
package_name = oc
filename = flow.oc
package_type = oc
model_or_simulation_package = model
model_name = flow

Block options
---------------------
budget_filerecord
{internal}
([('flow.bud',)])

head_filerecord
{internal}
([('flow.hds',)])


Block period
---------------------
saverecord
{internal}
([('HEAD', 'ALL', None) ('BUDGET', 'ALL', None)])

printrecord
{internal}
([('HEAD', 'ALL', None)])
```

```
writing simulation...
  writing simulation name file...
  writing simulation tdis package...
  writing solution package ims_-1...
  writing model flow...
    writing model name file...
    writing package dis...
    writing package ic...
    writing package chd_0...
    writing package npf...
    writing package oc...
```

Observación.

La celda anterior genera el archivo `flow.oc` con los datos para la salida como sigue:

```
# File generated by Flopy version 3.9.2 on 04/26/2025 at 11:36:03.
BEGIN options
  BUDGET  FILEOUT  flow.bud
  HEAD  FILEOUT  flow.hds
END options

BEGIN period  1
  SAVE  HEAD  ALL
  SAVE  BUDGET  ALL
  PRINT  HEAD  ALL
END period  1
```

11.8 Paso 7. Ejecución de la simulación.

En este punto tenemos con toda la información de entrada en los siguientes archivos:

- `flow.tdis`: datos para la discretización temporal.
- `flow.ims`: datos para la solución numérica.
- `flow.dis`: datos para la discretización espacial.
- `flow.ic`: datos para las condiciones iniciales.
- `flow.chd`: datos para las condiciones de frontera.
- `flow.npf`: datos para los flujos entre celdas.
- `flow.oc`: datos para la salida de los resultados.
- `flow.nam`: datos para el modelo GWF, informa de los paquetes que se usan.

- `mfsim.nam`: datos de la simulación completa, informa de las componentes que se usan.

Podemos ejecutar la simulación como sigue:

```
o_sim.run_simulation(silent=False)
```

```
FloPy is using the following executable to run the model: ..\..\..\..\mf6\windows\mf6.exe
                                    MODFLOW 6
                U.S. GEOLOGICAL SURVEY MODULAR HYDROLOGIC MODEL
                                VERSION 6.6.1 02/10/2025

   MODFLOW 6 compiled Feb 10 2025 17:37:25 with Intel(R) Fortran Intel(R) 64
   Compiler Classic for applications running on Intel(R) 64, Version 2021.7.0
                               Build 20220726_000000

This software has been approved for release by the U.S. Geological
Survey (USGS). Although the software has been subjected to rigorous
review, the USGS reserves the right to update the software as needed
pursuant to further analysis and review. No warranty, expressed or
implied, is made by the USGS or the U.S. Government as to the
functionality of the software and related material nor shall the
fact of release constitute any such warranty. Furthermore, the
software is released on condition that neither the USGS nor the U.S.
Government shall be held liable for any damages resulting from its
authorized or unauthorized use. Also refer to the USGS Water
Resources Software User Rights Notice for complete use, copyright,
and distribution information.


 MODFLOW runs in SEQUENTIAL mode

 Run start date and time (yyyy/mm/dd hh:mm:ss): 2025/05/18 13:17:59

 Writing simulation list file: mfsim.lst
 Using Simulation name file: mfsim.nam

    Solving:  Stress period:     1    Time step:     1

 Run end date and time (yyyy/mm/dd hh:mm:ss): 2025/05/18 13:17:59
 Elapsed run time:  0.202 Seconds

 Normal termination of simulation.
```

```
(True, [])
```

Observaciones.

La ejecución de la simulación genera los siguientes archivos:

- `flow.dis.grb`: información de la malla con todos los datos de las coordenadas, conectividad y otros (binario).
- `flow.bud`: datos del *budget* (binario).
- `flow.hds`: datos de la carga hidráulica (binario).
- `flow.lst`: archivo de listado con información de los resultados del modelo GWF.
- `mfsim.lst`: archivo de listado con información general de la simulación.

11.9 Paso 8. Recuperación y visualización de los resultados.

11.9.1 Recuperación de la carga hidráulica.

Para analizar los resultados generados por la simulación almacenados en los archivos descritos antes usaremos las herramientas que ofrece Flopy.

Objetos, funciones y archivos a usar.

- Primero recuperamos la carga hidráulica almacenada en el archivo `flow.hds` usando un objeto de la clase `flopy.utils.HeadFile`.
- Posteriormente, con la función `get_data` obtenemos un arreglo de numpy con la información correspondiente:

```
# construimos el nombre con la ruta y extensión correcta
headfile = os.path.join(f"{sim_ws}", f"{sim_name}.hds")

# Objeto para obtener y manipular arreglos o series de tiempo de arreglos con datos
# de una o más celdas con información del archivo binario de la carga hidráulica.
hds = flopy.utils.HeadFile(headfile)

# Solo se obtiene la capa 0, cuando no se pone ningún valor en el parámetro mlay se obtienen
head = hds.get_data(mflay=0)

# Imprimimos la información
print("head array")
xmf6.info_array(head)
print("\nHEAD:\n", head)
```

```
head array
 tipo  : <class 'numpy.ndarray'>
 dtype : float64
 dim   : 2
 shape : (10, 15)
 size(bytes) : 8
 size(elements) : 150

HEAD:
 [[10.          9.64285714  9.28571428  8.92857142  8.57142856  8.21428571
   7.85714285  7.5         7.14285714  6.78571427  6.42857141  6.07142855
   5.71428569  5.35714284  5.        ]
 [10.          9.64285714  9.28571428  8.92857142  8.57142856  8.2142857
   7.85714286  7.50000001  7.14285715  6.78571428  6.42857141  6.07142855
   5.71428569  5.35714285  5.        ]
 [10.          9.64285714  9.28571428  8.92857143  8.57142857  8.21428571
   7.85714285  7.5         7.14285715  6.78571428  6.42857142  6.07142856
   5.71428571  5.35714286  5.        ]
 [10.          9.64285714  9.28571429  8.92857143  8.57142858  8.21428572
   7.85714286  7.5         7.14285715  6.78571429  6.42857143  6.07142858
   5.71428572  5.35714286  5.        ]
 [10.          9.64285715  9.28571429  8.92857143  8.57142858  8.21428572
   7.85714287  7.5         7.14285715  6.7857143   6.42857145  6.07142859
   5.71428573  5.35714287  5.        ]
 [10.          9.64285715  9.28571429  8.92857144  8.57142858  8.21428572
   7.85714287  7.50000001  7.14285715  6.7857143   6.42857145  6.07142859
   5.71428573  5.35714287  5.        ]
 [10.          9.64285715  9.28571429  8.92857143  8.57142857  8.21428571
   7.85714286  7.50000001  7.14285715  6.78571429  6.42857144  6.07142859
   5.71428572  5.35714286  5.        ]
 [10.          9.64285714  9.28571429  8.92857143  8.57142857  8.21428571
   7.85714285  7.5         7.14285714  6.78571428  6.42857142  6.07142858
   5.71428572  5.35714285  5.        ]
 [10.          9.64285714  9.28571428  8.92857142  8.57142857  8.21428571
   7.85714284  7.49999998  7.14285712  6.78571427  6.42857141  6.07142856
   5.71428572  5.35714286  5.        ]
 [10.          9.64285714  9.28571428  8.92857142  8.57142857  8.21428571
   7.85714284  7.49999998  7.14285712  6.78571426  6.42857141  6.07142855
   5.7142857   5.35714285  5.        ]]
```

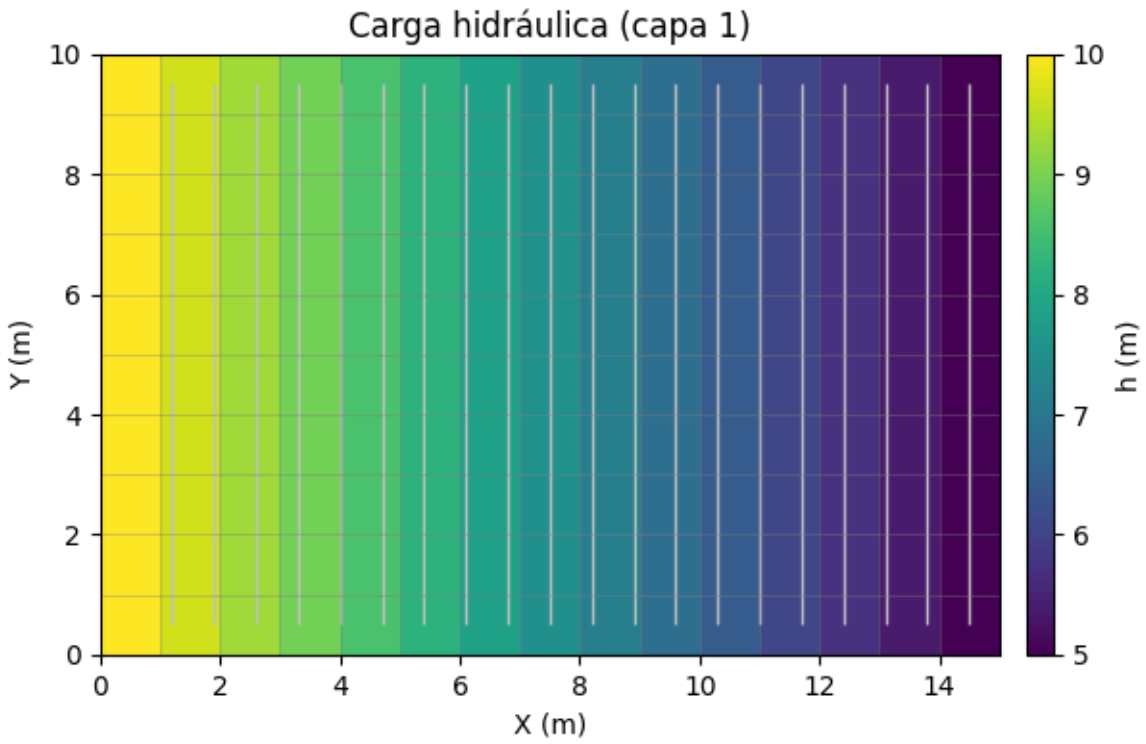
11.9.2 Visualización de la carga hidráulica.

Flopy también contiene herramientas para visualizar los resultados. Se combina con la biblioteca `matplotlib` para generar gráficas XY, mapas de color, contornos, vectores, entre otros.

Objetos y funciones a usar.

- Para visualizar estos resultados usamos un objeto de la clase `flopy.plot.PlotMapView` con la cual se crea un mapa del modelo y delega la funcionalidad de la graficación de acuerdo con el tipo de malla del modelo.
- Agregamos a esta visualización lo siguiente:
  - `plot_array()` : genera un mapa de color con los datos de la carga cidráulica.
  - `contour_array()` : genera contornos con los datos de la carga cidráulica.
  - `plot_grid` : dibuja la malla del modelo.
- Usamos también la función `xmf6.cax()` para obtener un espacio adecuado para la barra de color.

```
fig = plt.figure()
ax = fig.gca() # Ejes donde se hará la graficación

# Mapa visual del modelo. En este caso el modelo es 'o_gwf'.
mapview = flopy.plot.PlotMapView(model=o_gwf, ax=ax)

# Visualización de la carga hidráulica usando un mapa de color.
cb = mapview.plot_array(head, cmap="viridis", vmin=np.nanmin(head), vmax=np.nanmax(head))

# Visualización de la carga hidráulica usando contornos.
ct = mapview.contour_array(head, levels=20, linewidths=1.0, colors='silver')

# Visualización de la malla, con una transparencia.
mapview.plot_grid(linewidths=0.5, alpha =0.5)

# Barra de color.
plt.colorbar(cb, ax=ax, label="h (m)", cax = xmf6.vis.cax(ax, cb))

# Personalización de la gráfica.
ax.set_title("Carga hidráulica (capa 1)")
ax.set_xlabel("X (m)")
ax.set_ylabel("Y (m)")
ax.set_aspect('equal')
plt.show()
```



11.9.3 Recuperación de la descarga específica.

También es posible calcular y graficar la descarga específica, la cual se define como

$$q = -K\nabla h$$

Donde: * q : vector de descarga específica (flujo Darciano) [L/T]. * K : conductividad hidráulica del medio [L/T]. * ∇h: gradiente hidráulico (variación espacial de la carga hidráulica).

Objetos, funciones y archivos a usar.

- En este caso usaremos un objeto de la clase `flopy.utils.CellBudgetFile`.
- Luego para obtener los datos usamos la función `get_data()`. En este caso se requiere del parámetro `text = "DATA-SPDIS"` que indica que se obtendrá la descarga específica.
- Para que esto funcione, es necesario agregar el parámetro `save_specific_discharge = True` en el paquete NPF (*Node Property Flow*).
- La función `get_data` regresará una lista que contiene un solo elemento, el cual es un arreglo de tipo `numpy.rec.recarray`.

```
# construimos el nombre con la ruta y extensión correcta
budfile = os.path.join(f"{sim_ws}", f"{sim_name}.bud")

# Objeto para obtener y manipular arreglos o series de tiempo de arreglos con datos
# de una o más celdas con información del archivo binario del budget.
bud  = flopy.utils.CellBudgetFile(budfile)

# Cargar resultados de la descarga específica
spdis = bud.get_data(text="DATA-SPDIS")[0]

# Imprimimos la información
print("spdis array:", spdis.shape)
xmf6.info_array(spdis)
```

```
spdis array: (150,)
 tipo  : <class 'numpy.rec.recarray'>
 dtype : (numpy.record, [('node', '<i4'), ('node2', '<i4'), ('q', '<f8'), ('qx', '<f8'), ('q
 dim   : 1
 shape : (150,)
 size(bytes) : 40
 size(elements) : 150
```

Observaciones.

- El objeto `spdis` es de tipo `numpy.rec.recarray` con 6 columnas de información: `node`, `node2`, `q`, `qx`, `qy`, `qz`.
- Cada renglón de este arreglo corresponde a la información de una celda (en este caso $15 \times 10 = 150$ celdas).

Columna	Descripción	Unidades
`node`	Índice del nodo origen: celda desde la que sale el flujo	—
`node2`	Índice del nodo destino: celda hacia la que va el flujo	—
`q`	Caudal total entre `node` y `node2`	m^3/d
`qx`	Componente del flujo en dirección x	m^3/d
`qy`	Componente del flujo en dirección y	m^3/d
`qz`	Componente del flujo en dirección z (positivo hacia abajo)	m^3/d

- Podemos ver el contenido de este arreglo como sigue:

```
print("\nSPDIS:\n", spdis)
```

```
SPDIS:
 [(  1,   1, 0., 0.35714286,  0.00000000e+00, 0.)
 (  2,   2, 0., 0.35714286,  5.47633050e-10, 0.)
 (  3,   3, 0., 0.35714286,  7.50866036e-12, 0.)
 (  4,   4, 0., 0.35714286, -1.00981623e-09, 0.)
 (  5,   5, 0., 0.35714286, -1.53560009e-09, 0.)
 (  6,   6, 0., 0.35714285, -8.93553675e-10, 0.)
 (  7,   7, 0., 0.35714285,  2.87018409e-09, 0.)
 (  8,   8, 0., 0.35714286,  1.05430837e-08, 0.)
 (  9,   9, 0., 0.35714286,  1.47557371e-08, 0.)
 ( 10,  10, 0., 0.35714286,  8.12601186e-09, 0.)
 ( 11,  11, 0., 0.35714286,  4.20858903e-10, 0.)
 ( 12,  12, 0., 0.35714286,  1.66894321e-09, 0.)
 ( 13,  13, 0., 0.35714285,  4.89590590e-09, 0.)
 ( 14,  14, 0., 0.35714284,  5.71419445e-09, 0.)
 ( 15,  15, 0., 0.35714284,  0.00000000e+00, 0.)
 ( 16,  16, 0., 0.35714286,  0.00000000e+00, 0.)
 ( 17,  17, 0., 0.35714286,  5.43539436e-10, 0.)
 ( 18,  18, 0., 0.35714286,  8.54491589e-10, 0.)
 ( 19,  19, 0., 0.35714286,  2.06802397e-09, 0.)
 ( 20,  20, 0., 0.35714286,  3.72881459e-09, 0.)
 ( 21,  21, 0., 0.35714285,  3.38642092e-09, 0.)
 ( 22,  22, 0., 0.35714285,  1.03656950e-09, 0.)
 ( 23,  23, 0., 0.35714285,  1.75400272e-09, 0.)
 ( 24,  24, 0., 0.35714286,  6.03943651e-09, 0.)
 ( 25,  25, 0., 0.35714287,  6.14190743e-09, 0.)
 ( 26,  26, 0., 0.35714286,  3.57975916e-09, 0.)
 ( 27,  27, 0., 0.35714286,  6.56130039e-09, 0.)
 ( 28,  28, 0., 0.35714285,  9.53989865e-09, 0.)
 ( 29,  29, 0., 0.35714285,  7.52138662e-09, 0.)
 ( 30,  30, 0., 0.35714285,  0.00000000e+00, 0.)
 ( 31,  31, 0., 0.35714286,  0.00000000e+00, 0.)
 ( 32,  32, 0., 0.35714286,  6.91943391e-10, 0.)
 ( 33,  33, 0., 0.35714286,  1.86103044e-09, 0.)
 ( 34,  34, 0., 0.35714286,  4.83698237e-09, 0.)
 ( 35,  35, 0., 0.35714286,  8.76451445e-09, 0.)
 ( 36,  36, 0., 0.35714286,  9.74681047e-09, 0.)
 ( 37,  37, 0., 0.35714286,  4.02223854e-09, 0.)
 ( 38,  38, 0., 0.35714285, -3.16989590e-09, 0.)
```

```
( 39,  39, 0., 0.35714286, -1.20540644e-09, 0.)
( 40,  40, 0., 0.35714286,  6.99057878e-09, 0.)
( 41,  41, 0., 0.35714286,  1.08003615e-08, 0.)
( 42,  42, 0., 0.35714286,  1.34577243e-08, 0.)
( 43,  43, 0., 0.35714285,  1.34282678e-08, 0.)
( 44,  44, 0., 0.35714285,  8.25992919e-09, 0.)
( 45,  45, 0., 0.35714286,  0.00000000e+00, 0.)
( 46,  46, 0., 0.35714286,  0.00000000e+00, 0.)
( 47,  47, 0., 0.35714286,  1.39541267e-09, 0.)
( 48,  48, 0., 0.35714286,  2.75352452e-09, 0.)
( 49,  49, 0., 0.35714285,  3.49437812e-09, 0.)
( 50,  50, 0., 0.35714285,  3.80157950e-09, 0.)
( 51,  51, 0., 0.35714286,  5.19847188e-09, 0.)
( 52,  52, 0., 0.35714286,  5.56624391e-09, 0.)
( 53,  53, 0., 0.35714286,  1.45992685e-09, 0.)
( 54,  54, 0., 0.35714285,  5.92713878e-10, 0.)
( 55,  55, 0., 0.35714286,  8.73039330e-09, 0.)
( 56,  56, 0., 0.35714286,  1.47507939e-08, 0.)
( 57,  57, 0., 0.35714286,  1.49993009e-08, 0.)
( 58,  58, 0., 0.35714286,  1.11664762e-08, 0.)
( 59,  59, 0., 0.35714286,  5.95146332e-09, 0.)
( 60,  60, 0., 0.35714286,  0.00000000e+00, 0.)
( 61,  61, 0., 0.35714285,  0.00000000e+00, 0.)
( 62,  62, 0., 0.35714285,  1.48180401e-09, 0.)
( 63,  63, 0., 0.35714286,  2.84630985e-09, 0.)
( 64,  64, 0., 0.35714286,  2.11817763e-09, 0.)
( 65,  65, 0., 0.35714286, -1.27107391e-09, 0.)
( 66,  66, 0., 0.35714286, -2.88448909e-09, 0.)
( 67,  67, 0., 0.35714286,  1.54385305e-09, 0.)
( 68,  68, 0., 0.35714286,  4.45305082e-09, 0.)
( 69,  69, 0., 0.35714285,  2.14465024e-09, 0.)
( 70,  70, 0., 0.35714285,  3.12278559e-09, 0.)
( 71,  71, 0., 0.35714285,  7.03745329e-09, 0.)
( 72,  72, 0., 0.35714286,  7.94326427e-09, 0.)
( 73,  73, 0., 0.35714286,  4.83928586e-09, 0.)
( 74,  74, 0., 0.35714287,  2.62405475e-09, 0.)
( 75,  75, 0., 0.35714287,  0.00000000e+00, 0.)
( 76,  76, 0., 0.35714285,  0.00000000e+00, 0.)
( 77,  77, 0., 0.35714285, -1.52249768e-10, 0.)
( 78,  78, 0., 0.35714286, -4.79487561e-10, 0.)
( 79,  79, 0., 0.35714286, -1.24683552e-09, 0.)
( 80,  80, 0., 0.35714286, -2.99411163e-09, 0.)
( 81,  81, 0., 0.35714285, -5.73222803e-09, 0.)
```

```
( 82,  82, 0., 0.35714285, -3.64090491e-09, 0.)
( 83,  83, 0., 0.35714286,  1.97418881e-09, 0.)
( 84,  84, 0., 0.35714286,  2.17832863e-10, 0.)
( 85,  85, 0., 0.35714285, -4.28550528e-09, 0.)
( 86,  86, 0., 0.35714285, -4.40030812e-09, 0.)
( 87,  87, 0., 0.35714286, -1.86558724e-09, 0.)
( 88,  88, 0., 0.35714286, -4.82911977e-09, 0.)
( 89,  89, 0., 0.35714287, -4.75692685e-09, 0.)
( 90,  90, 0., 0.35714287,  0.00000000e+00, 0.)
( 91,  91, 0., 0.35714285,  0.00000000e+00, 0.)
( 92,  92, 0., 0.35714286, -1.60843427e-09, 0.)
( 93,  93, 0., 0.35714286, -2.96047187e-09, 0.)
( 94,  94, 0., 0.35714286, -2.56772736e-09, 0.)
( 95,  95, 0., 0.35714286, -5.72208059e-10, 0.)
( 96,  96, 0., 0.35714286, -3.13721937e-09, 0.)
( 97,  97, 0., 0.35714285, -9.26611987e-09, 0.)
( 98,  98, 0., 0.35714285, -7.99172506e-09, 0.)
( 99,  99, 0., 0.35714286, -6.16383966e-09, 0.)
(100, 100, 0., 0.35714286, -8.32303249e-09, 0.)
(101, 101, 0., 0.35714285, -1.31341396e-08, 0.)
(102, 102, 0., 0.35714286, -9.14399534e-09, 0.)
(103, 103, 0., 0.35714287, -5.49348167e-09, 0.)
(104, 104, 0., 0.35714286, -7.28141591e-09, 0.)
(105, 105, 0., 0.35714286,  0.00000000e+00, 0.)
(106, 106, 0., 0.35714286,  0.00000000e+00, 0.)
(107, 107, 0., 0.35714286, -2.46356535e-09, 0.)
(108, 108, 0., 0.35714286, -3.95433553e-09, 0.)
(109, 109, 0., 0.35714286, -3.08912806e-09, 0.)
(110, 110, 0., 0.35714286,  1.89291249e-10, 0.)
(111, 111, 0., 0.35714286,  7.32352845e-10, 0.)
(112, 112, 0., 0.35714286, -7.39661310e-09, 0.)
(113, 113, 0., 0.35714285, -1.37140770e-08, 0.)
(114, 114, 0., 0.35714286, -1.39238296e-08, 0.)
(115, 115, 0., 0.35714286, -9.43181933e-09, 0.)
(116, 116, 0., 0.35714285, -1.25264670e-08, 0.)
(117, 117, 0., 0.35714285, -1.39834171e-08, 0.)
(118, 118, 0., 0.35714286, -3.06157011e-09, 0.)
(119, 119, 0., 0.35714286, -4.16186641e-10, 0.)
(120, 120, 0., 0.35714285,  0.00000000e+00, 0.)
(121, 121, 0., 0.35714286,  0.00000000e+00, 0.)
(122, 122, 0., 0.35714286, -9.88087834e-10, 0.)
(123, 123, 0., 0.35714286, -1.73246573e-09, 0.)
(124, 124, 0., 0.35714285, -3.12913340e-09, 0.)
```

```
 (125, 125, 0., 0.35714286, -4.17634727e-09, 0.)
 (126, 126, 0., 0.35714286, -2.74673351e-09, 0.)
 (127, 127, 0., 0.35714287, -3.69824837e-09, 0.)
 (128, 128, 0., 0.35714286, -7.74172815e-09, 0.)
 (129, 129, 0., 0.35714285, -1.20199997e-08, 0.)
 (130, 130, 0., 0.35714285, -1.00885398e-08, 0.)
 (131, 131, 0., 0.35714286, -6.53736354e-09, 0.)
 (132, 132, 0., 0.35714285, -1.19671197e-08, 0.)
 (133, 133, 0., 0.35714285, -8.38436565e-09, 0.)
 (134, 134, 0., 0.35714286, -6.92109925e-10, 0.)
 (135, 135, 0., 0.35714286,  0.00000000e+00, 0.)
 (136, 136, 0., 0.35714286,  0.00000000e+00, 0.)
 (137, 137, 0., 0.35714286,  1.05580966e-09, 0.)
 (138, 138, 0., 0.35714286,  1.68792802e-09, 0.)
 (139, 139, 0., 0.35714286, -9.46094758e-10, 0.)
 (140, 140, 0., 0.35714286, -5.49697710e-09, 0.)
 (141, 141, 0., 0.35714286, -6.10685191e-09, 0.)
 (142, 142, 0., 0.35714286, -3.05696002e-09, 0.)
 (143, 143, 0., 0.35714286, -1.30559563e-09, 0.)
 (144, 144, 0., 0.35714286, -5.58576119e-09, 0.)
 (145, 145, 0., 0.35714285, -1.07803562e-08, 0.)
 (146, 146, 0., 0.35714286, -6.05407369e-09, 0.)
 (147, 147, 0., 0.35714285, -9.17450382e-09, 0.)
 (148, 148, 0., 0.35714285, -1.19560513e-08, 0.)
 (149, 149, 0., 0.35714285, -5.06436226e-09, 0.)
 (150, 150, 0., 0.35714285,  0.00000000e+00, 0.)]
```

Podemos checar cada una de las columnas por separado:

```
print("\nnode:", spdis.node.shape)
xmf6.info_array(spdis.node)
print(spdis.node)

print("\nnode2:", spdis.node2.shape)
xmf6.info_array(spdis.node2)
print(spdis.node2)

print("\nq:", spdis.q.shape)
xmf6.info_array(spdis.q)
print(spdis.q)

print("\nqx:", spdis.qx.shape)
```

```
xmf6.info_array(spdis.qx)
print(spdis.qx)

print("\nqy:", spdis.qy.shape)
xmf6.info_array(spdis.qy)
print(spdis.qy)

print("\nqz:", spdis.qz.shape)
xmf6.info_array(spdis.qz)
print(spdis.qz)
```

```
node: (150,)
 tipo  : <class 'numpy.ndarray'>
 dtype : int32
 dim   : 1
 shape : (150,)
 size(bytes) : 4
 size(elements) : 150
[  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
 145 146 147 148 149 150]

node2: (150,)
 tipo  : <class 'numpy.ndarray'>
 dtype : int32
 dim   : 1
 shape : (150,)
 size(bytes) : 4
 size(elements) : 150
[  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
```

```
 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
 145 146 147 148 149 150]

q: (150,)
 tipo  : <class 'numpy.ndarray'>
 dtype : float64
 dim   : 1
 shape : (150,)
 size(bytes) : 8
 size(elements) : 150
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0.]

qx: (150,)
 tipo  : <class 'numpy.ndarray'>
 dtype : float64
 dim   : 1
 shape : (150,)
 size(bytes) : 8
 size(elements) : 150
[0.35714286 0.35714286 0.35714286 0.35714286 0.35714286 0.35714285
 0.35714285 0.35714286 0.35714286 0.35714286 0.35714286 0.35714286
 0.35714285 0.35714284 0.35714284 0.35714286 0.35714286 0.35714286
 0.35714286 0.35714286 0.35714285 0.35714285 0.35714285 0.35714286
 0.35714287 0.35714286 0.35714286 0.35714285 0.35714285 0.35714285
 0.35714286 0.35714286 0.35714286 0.35714286 0.35714286 0.35714286
 0.35714286 0.35714285 0.35714286 0.35714286 0.35714286 0.35714286
 0.35714285 0.35714285 0.35714286 0.35714286 0.35714286 0.35714286
 0.35714285 0.35714285 0.35714286 0.35714286 0.35714286 0.35714285
 0.35714286 0.35714286 0.35714286 0.35714286 0.35714286 0.35714286
 0.35714285 0.35714285 0.35714286 0.35714286 0.35714286 0.35714286
 0.35714286 0.35714286 0.35714285 0.35714285 0.35714285 0.35714286
 0.35714286 0.35714287 0.35714287 0.35714285 0.35714285 0.35714286
 0.35714286 0.35714286 0.35714285 0.35714285 0.35714286 0.35714286
 0.35714285 0.35714285 0.35714286 0.35714286 0.35714287 0.35714287
 0.35714285 0.35714286 0.35714286 0.35714286 0.35714286 0.35714286
 0.35714285 0.35714285 0.35714286 0.35714286 0.35714285 0.35714286
```

```
 0.35714287 0.35714286 0.35714286 0.35714286 0.35714286 0.35714286
 0.35714286 0.35714286 0.35714286 0.35714286 0.35714285 0.35714286
 0.35714286 0.35714285 0.35714285 0.35714286 0.35714286 0.35714285
 0.35714286 0.35714286 0.35714286 0.35714285 0.35714286 0.35714286
 0.35714287 0.35714286 0.35714285 0.35714285 0.35714286 0.35714285
 0.35714285 0.35714286 0.35714286 0.35714286 0.35714286 0.35714286
 0.35714286 0.35714286 0.35714286 0.35714286 0.35714286 0.35714286
 0.35714285 0.35714286 0.35714285 0.35714285 0.35714285 0.35714285]

qy: (150,)
 tipo  : <class 'numpy.ndarray'>
 dtype : float64
 dim   : 1
 shape : (150,)
 size(bytes) : 8
 size(elements) : 150
[ 0.00000000e+00  5.47633050e-10  7.50866036e-12 -1.00981623e-09
 -1.53560009e-09 -8.93553675e-10  2.87018409e-09  1.05430837e-08
  1.47557371e-08  8.12601186e-09  4.20858903e-10  1.66894321e-09
  4.89590590e-09  5.71419445e-09  0.00000000e+00  0.00000000e+00
  5.43539436e-10  8.54491589e-10  2.06802397e-09  3.72881459e-09
  3.38642092e-09  1.03656950e-09  1.75400272e-09  6.03943651e-09
  6.14190743e-09  3.57975916e-09  6.56130039e-09  9.53989865e-09
  7.52138662e-09  0.00000000e+00  0.00000000e+00  6.91943391e-10
  1.86103044e-09  4.83698237e-09  8.76451445e-09  9.74681047e-09
  4.02223854e-09 -3.16989590e-09 -1.20540644e-09  6.99057878e-09
  1.08003615e-08  1.34577243e-08  1.34282678e-08  8.25992919e-09
  0.00000000e+00  0.00000000e+00  1.39541267e-09  2.75352452e-09
  3.49437812e-09  3.80157950e-09  5.19847188e-09  5.56624391e-09
  1.45992685e-09  5.92713878e-10  8.73039330e-09  1.47507939e-08
  1.49993009e-08  1.11664762e-08  5.95146332e-09  0.00000000e+00
  0.00000000e+00  1.48180401e-09  2.84630985e-09  2.11817763e-09
 -1.27107391e-09 -2.88448909e-09  1.54385305e-09  4.45305082e-09
  2.14465024e-09  3.12278559e-09  7.03745329e-09  7.94326427e-09
  4.83928586e-09  2.62405475e-09  0.00000000e+00  0.00000000e+00
 -1.52249768e-10 -4.79487561e-10 -1.24683552e-09 -2.99411163e-09
 -5.73222803e-09 -3.64090491e-09  1.97418881e-09  2.17832863e-10
 -4.28550528e-09 -4.40030812e-09 -1.86558724e-09 -4.82911977e-09
 -4.75692685e-09  0.00000000e+00  0.00000000e+00 -1.60843427e-09
 -2.96047187e-09 -2.56772736e-09 -5.72208059e-10 -3.13721937e-09
 -9.26611987e-09 -7.99172506e-09 -6.16383966e-09 -8.32303249e-09
 -1.31341396e-08 -9.14399534e-09 -5.49348167e-09 -7.28141591e-09
  0.00000000e+00  0.00000000e+00 -2.46356535e-09 -3.95433553e-09
```

```
 -3.08912806e-09  1.89291249e-10  7.32352845e-10 -7.39661310e-09
 -1.37140770e-08 -1.39238296e-08 -9.43181933e-09 -1.25264670e-08
 -1.39834171e-08 -3.06157011e-09 -4.16186641e-10  0.00000000e+00
  0.00000000e+00 -9.88087834e-10 -1.73246573e-09 -3.12913340e-09
 -4.17634727e-09 -2.74673351e-09 -3.69824837e-09 -7.74172815e-09
 -1.20199997e-08 -1.00885398e-08 -6.53736354e-09 -1.19671197e-08
 -8.38436565e-09 -6.92109925e-10  0.00000000e+00  0.00000000e+00
  1.05580966e-09  1.68792802e-09 -9.46094758e-10 -5.49697710e-09
 -6.10685191e-09 -3.05696002e-09 -1.30559563e-09 -5.58576119e-09
 -1.07803562e-08 -6.05407369e-09 -9.17450382e-09 -1.19560513e-08
 -5.06436226e-09  0.00000000e+00]

qz: (150,)
 tipo   : <class 'numpy.ndarray'>
 dtype : float64
 dim   : 1
 shape : (150,)
 size(bytes) : 8
 size(elements) : 150
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0.]
```

```python
# Extraemos las componentes del flujo en un arreglo 3D para graficación
qx, qy, qz = flopy.utils.postprocessing.get_specific_discharge(spdis, o_gwf)

print("\nqx:", qx.shape)
xmf6.info_array(qx)
print(qx)

print("\nqy:", qy.shape)
xmf6.info_array(qy)
print(qy)
```

```
qx: (1, 10, 15)
 tipo   : <class 'numpy.ndarray'>
 dtype : float64
```

```
 dim   : 3
 shape : (1, 10, 15)
 size(bytes) : 8
 size(elements) : 150
[[[0.35714286 0.35714286 0.35714286 0.35714286 0.35714286 0.35714285
   0.35714285 0.35714286 0.35714286 0.35714286 0.35714286 0.35714286
   0.35714285 0.35714284 0.35714284]
  [0.35714286 0.35714286 0.35714286 0.35714286 0.35714286 0.35714285
   0.35714285 0.35714285 0.35714286 0.35714287 0.35714286 0.35714286
   0.35714285 0.35714285 0.35714285]
  [0.35714286 0.35714286 0.35714286 0.35714286 0.35714286 0.35714286
   0.35714286 0.35714285 0.35714286 0.35714286 0.35714286 0.35714286
   0.35714285 0.35714285 0.35714286]
  [0.35714286 0.35714286 0.35714286 0.35714285 0.35714285 0.35714286
   0.35714286 0.35714286 0.35714285 0.35714286 0.35714286 0.35714286
   0.35714286 0.35714286 0.35714286]
  [0.35714285 0.35714285 0.35714286 0.35714286 0.35714286 0.35714286
   0.35714286 0.35714286 0.35714285 0.35714285 0.35714285 0.35714286
   0.35714286 0.35714287 0.35714287]
  [0.35714285 0.35714285 0.35714286 0.35714286 0.35714286 0.35714285
   0.35714285 0.35714286 0.35714286 0.35714285 0.35714285 0.35714286
   0.35714286 0.35714287 0.35714287]
  [0.35714285 0.35714286 0.35714286 0.35714286 0.35714286 0.35714286
   0.35714285 0.35714285 0.35714286 0.35714286 0.35714285 0.35714286
   0.35714287 0.35714286 0.35714286]
  [0.35714286 0.35714286 0.35714286 0.35714286 0.35714286 0.35714286
   0.35714286 0.35714285 0.35714286 0.35714286 0.35714285 0.35714285
   0.35714286 0.35714286 0.35714285]
  [0.35714286 0.35714286 0.35714286 0.35714285 0.35714286 0.35714286
   0.35714287 0.35714286 0.35714285 0.35714285 0.35714286 0.35714285
   0.35714285 0.35714286 0.35714286]
  [0.35714286 0.35714286 0.35714286 0.35714286 0.35714286 0.35714286
   0.35714286 0.35714286 0.35714286 0.35714285 0.35714286 0.35714285
   0.35714285 0.35714285 0.35714285]]]

qy: (1, 10, 15)
 tipo  : <class 'numpy.ndarray'>
 dtype : float64
 dim   : 3
 shape : (1, 10, 15)
 size(bytes) : 8
 size(elements) : 150
[[[ 0.00000000e+00  5.47633050e-10  7.50866036e-12 -1.00981623e-09
```

```
  -1.53560009e-09 -8.93553675e-10  2.87018409e-09  1.05430837e-08
   1.47557371e-08  8.12601186e-09  4.20858903e-10  1.66894321e-09
   4.89590590e-09  5.71419445e-09  0.00000000e+00]
 [ 0.00000000e+00  5.43539436e-10  8.54491589e-10  2.06802397e-09
   3.72881459e-09  3.38642092e-09  1.03656950e-09  1.75400272e-09
   6.03943651e-09  6.14190743e-09  3.57975916e-09  6.56130039e-09
   9.53989865e-09  7.52138662e-09  0.00000000e+00]
 [ 0.00000000e+00  6.91943391e-10  1.86103044e-09  4.83698237e-09
   8.76451445e-09  9.74681047e-09  4.02223854e-09 -3.16989590e-09
  -1.20540644e-09  6.99057878e-09  1.08003615e-08  1.34577243e-08
   1.34282678e-08  8.25992919e-09  0.00000000e+00]
 [ 0.00000000e+00  1.39541267e-09  2.75352452e-09  3.49437812e-09
   3.80157950e-09  5.19847188e-09  5.56624391e-09  1.45992685e-09
   5.92713878e-10  8.73039330e-09  1.47507939e-08  1.49993009e-08
   1.11664762e-08  5.95146332e-09  0.00000000e+00]
 [ 0.00000000e+00  1.48180401e-09  2.84630985e-09  2.11817763e-09
  -1.27107391e-09 -2.88448909e-09  1.54385305e-09  4.45305082e-09
   2.14465024e-09  3.12278559e-09  7.03745329e-09  7.94326427e-09
   4.83928586e-09  2.62405475e-09  0.00000000e+00]
 [ 0.00000000e+00 -1.52249768e-10 -4.79487561e-10 -1.24683552e-09
  -2.99411163e-09 -5.73222803e-09 -3.64090491e-09  1.97418881e-09
   2.17832863e-10 -4.28550528e-09 -4.40030812e-09 -1.86558724e-09
  -4.82911977e-09 -4.75692685e-09  0.00000000e+00]
 [ 0.00000000e+00 -1.60843427e-09 -2.96047187e-09 -2.56772736e-09
  -5.72208059e-10 -3.13721937e-09 -9.26611987e-09 -7.99172506e-09
  -6.16383966e-09 -8.32303249e-09 -1.31341396e-08 -9.14399534e-09
  -5.49348167e-09 -7.28141591e-09  0.00000000e+00]
 [ 0.00000000e+00 -2.46356535e-09 -3.95433553e-09 -3.08912806e-09
   1.89291249e-10  7.32352845e-10 -7.39661310e-09 -1.37140770e-08
  -1.39238296e-08 -9.43181933e-09 -1.25264670e-08 -1.39834171e-08
  -3.06157011e-09 -4.16186641e-10  0.00000000e+00]
 [ 0.00000000e+00 -9.88087834e-10 -1.73246573e-09 -3.12913340e-09
  -4.17634727e-09 -2.74673351e-09 -3.69824837e-09 -7.74172815e-09
  -1.20199997e-08 -1.00885398e-08 -6.53736354e-09 -1.19671197e-08
  -8.38436565e-09 -6.92109925e-10  0.00000000e+00]
 [ 0.00000000e+00  1.05580966e-09  1.68792802e-09 -9.46094758e-10
  -5.49697710e-09 -6.10685191e-09 -3.05696002e-09 -1.30559563e-09
  -5.58576119e-09 -1.07803562e-08 -6.05407369e-09 -9.17450382e-09
  -1.19560513e-08 -5.06436226e-09  0.00000000e+00]]]
```

```
# Coordenadas del centro de celdas para graficación
x, y, z = o_gwf.modelgrid.xyzcellcenters
```

```python
print("\nx:", x.shape)
xmf6.info_array(x)
print(x)

print("\ny:", y.shape)
xmf6.info_array(y)
print(y)
```

```
x: (10, 15)
 tipo  : <class 'numpy.ndarray'>
 dtype : float64
 dim   : 2
 shape : (10, 15)
 size(bytes) : 8
 size(elements) : 150
[[ 0.5  1.5  2.5  3.5  4.5  5.5  6.5  7.5  8.5  9.5 10.5 11.5 12.5 13.5
  14.5]
 [ 0.5  1.5  2.5  3.5  4.5  5.5  6.5  7.5  8.5  9.5 10.5 11.5 12.5 13.5
  14.5]
 [ 0.5  1.5  2.5  3.5  4.5  5.5  6.5  7.5  8.5  9.5 10.5 11.5 12.5 13.5
  14.5]
 [ 0.5  1.5  2.5  3.5  4.5  5.5  6.5  7.5  8.5  9.5 10.5 11.5 12.5 13.5
  14.5]
 [ 0.5  1.5  2.5  3.5  4.5  5.5  6.5  7.5  8.5  9.5 10.5 11.5 12.5 13.5
  14.5]
 [ 0.5  1.5  2.5  3.5  4.5  5.5  6.5  7.5  8.5  9.5 10.5 11.5 12.5 13.5
  14.5]
 [ 0.5  1.5  2.5  3.5  4.5  5.5  6.5  7.5  8.5  9.5 10.5 11.5 12.5 13.5
  14.5]
 [ 0.5  1.5  2.5  3.5  4.5  5.5  6.5  7.5  8.5  9.5 10.5 11.5 12.5 13.5
  14.5]
 [ 0.5  1.5  2.5  3.5  4.5  5.5  6.5  7.5  8.5  9.5 10.5 11.5 12.5 13.5
  14.5]
 [ 0.5  1.5  2.5  3.5  4.5  5.5  6.5  7.5  8.5  9.5 10.5 11.5 12.5 13.5
  14.5]]

y: (10, 15)
 tipo  : <class 'numpy.ndarray'>
 dtype : float64
 dim   : 2
```

```
 shape : (10, 15)
 size(bytes) : 8
 size(elements) : 150
[[9.5 9.5 9.5 9.5 9.5 9.5 9.5 9.5 9.5 9.5 9.5 9.5 9.5 9.5 9.5]
 [8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5 8.5]
 [7.5 7.5 7.5 7.5 7.5 7.5 7.5 7.5 7.5 7.5 7.5 7.5 7.5 7.5 7.5]
 [6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5 6.5]
 [5.5 5.5 5.5 5.5 5.5 5.5 5.5 5.5 5.5 5.5 5.5 5.5 5.5 5.5 5.5]
 [4.5 4.5 4.5 4.5 4.5 4.5 4.5 4.5 4.5 4.5 4.5 4.5 4.5 4.5 4.5]
 [3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5]
 [2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5]
 [1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5]
 [0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5]]
```

11.9.4 Visualización de la descarga específica.

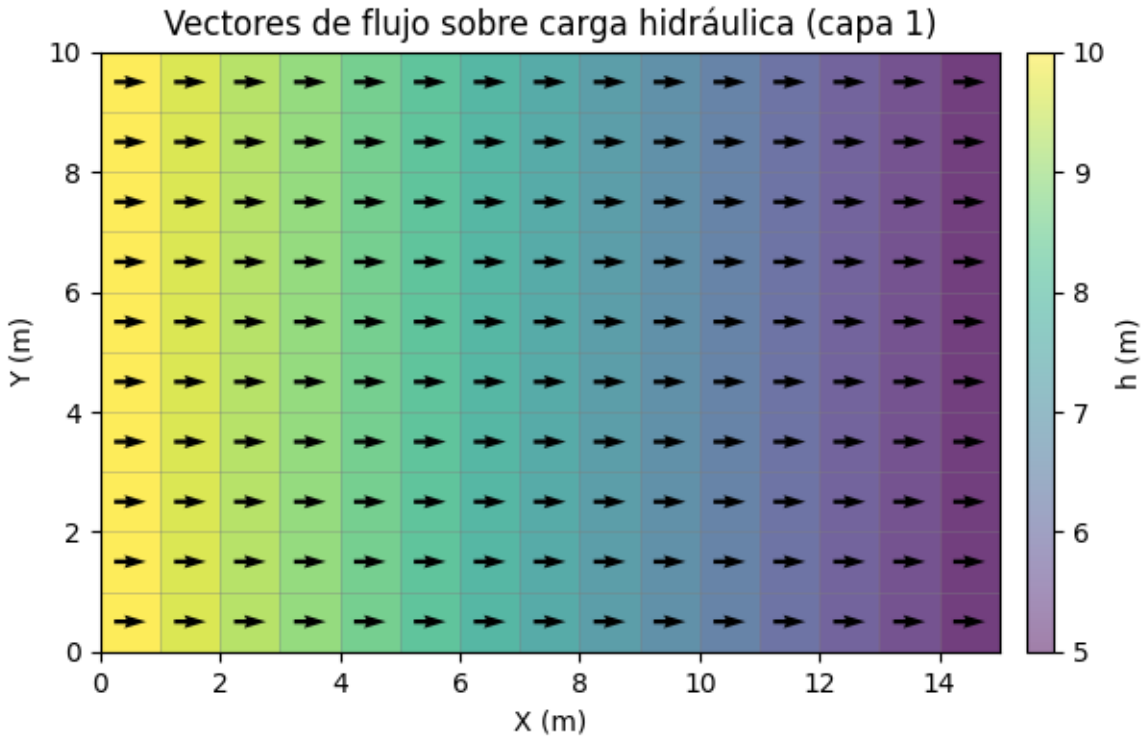
En este caso, se usan los mismos objetos que en la graficación de la carga hidráulica y solo se agregan los vectors de la descarga hidráulica.

Observación.

Para dibujar los vectores usamos la función `quiver()` de matplotlib, pues ya tenemos los arreglos `x`, `y`, `qx` y `qy` en el formato adecuado para graficarlos en 2D.

```
fig = plt.figure()
ax =fig.gca()
mapview = flopy.plot.PlotMapView(model=o_gwf, ax=ax)

# Visualización de la carga hidráulica usando un mapa de color.
cb = mapview.plot_array(head, cmap="viridis", alpha=0.5)

# Visualización de los vectores de la descarga específica.
ax.quiver(x, y, qx[0], qy[0], scale=10, color='k', pivot='middle')

# Visualización de la malla, con una transparencia.
mapview.plot_grid(linewidths=0.5, alpha =0.5)

# Barra de color.
plt.colorbar(cb, ax=ax, label="h (m)", cax = xmf6.vis.cax(ax, cb))

# Personalización de la gráfica.
ax.set_title("Vectores de flujo sobre carga hidráulica (capa 1)")
ax.set_xlabel("X (m)")
```

```
ax.set_ylabel("Y (m)")
ax.set_aspect('equal')
plt.show()
```



12 Referencias

[1] Langevin, C.D., Hughes, J.D., Provost, A.M., Banta, E.R., Niswonger, R.G., and Panday, Sorab, 2017, Documentation for the MODFLOW 6 Groundwater Flow (GWF) Model: U.S. Geological Survey Techniques and Methods, book 6, chap. A55, 197 p., accessed August 4, 2017. https://doi.org/10.3133/tm6A55.

[2] MODFLOW 6 – Description of Input and Output. Version mf6.4.4—February 13, 2024. U.S. Department of the Interior. U.S. Geological Survey.. Archivo: mf6io.pdf de la documentación de MODFLOW 6 que se puede obtener de https://github.com/MODFLOW-ORG/modflow6/releases.

13 MODFLOW 6: simulación de flujo v 1.0

13.1 Resumen.

Se realiza una simulación de flujo en dos dimensiones explicando paso a paso el proceso, similar a lo realizado en la notebook 02_MF6_GWF_paq.ipynb pero con una conductividad hidráulica variable. Adicionalmente, se encapsulan en una función de python varias acciones para hacer más modular la simulación.

14 Introducción.

Usando las herramientas de Flopy es posible generar los archivos de entrada para MODFLOW de una forma más simple y automatizada. Además, dentro del ambiente de Python se pueden construir estructuras de datos que permiten definir los parámetros de la simulación y modificarlos fácilmente. Dado que usando Flopy se tiene una secuencia bien definida para crear la simulación y construir los archivos de entrada, en este ejemplo se construye una función que recibe los parámetros adecuados para definir dicha secuencia y regresar los objetos de la simulación y del modelof GWF, con los cuales será posible ejecutar la simulación y posteriormente realizar el post-procesamiento.

15 Ejemplo: Simulación de flujo en 2D con k variable.

En este ejemplo se simula una distribución de carga hidráulica en un dominio rectangular de 15 m $\times$ 10 m con una sola capa con espesor de 1 m. La carga va de 10 m en el lado izquierdo a 5 m en lado derecho, formando un gradiente horizontal de flujo. Los lados superior e inferior del dominio son impermeables. No hay fuentes ni sumideros. El valor de la conductividad hidráulica se genera de manera aleatoria dentro de todo el dominio y se define una sección con una permeabilidad constante.

15.1 Paso 0. Importación de bibliotecas y módulos.

Debemos incluir todas las bibliotecas que se van a usar. En este caso vamos a incluir el módulo `gwf_build_1` que contiene las funciones que se describen en este documento.

```
import flopy
import numpy as np
import matplotlib.pyplot as plt
import os
import xmf6
from gwf_model import build
```

15.2 Paso 1. Parámetros de la simulación.

Con base en lo realizado en la notebook 02_MF6_GWF_paq.ipynb, se define una función con la secuencia de pasos necesaria para construir el modelo GWF. Esta función tiene la siguiente firma:

```
def build(init, time, mesh, ic_data, chd_data, k_data):
```

La función `build()`recibe los siguientes parámetros:

- `init`. Parámetros de configuración de la simulación (nombre del ejecutable, nombre del modelo, espacio de trabajo, etc).
- `time`. Parámetros para la discretización temporal.
- `mesh`. Parámetros para la discretización espacial.
- `ic_data`. Parámetros para las condiciones iniciales.
- `chd_data`. Parámetros para las condiciones de frontera.
- `k_data`. Parámetros para las propiedades del flujo.

Estos parámetros se definen en diccionarios y en arreglos como se muesta a continuación.

```
# --- Datos para la configuración de la simulación ---
# Opciones de SO's: linux, macos, macosarm, windows
init = {
    'exe_name' : "C:\\Users\\luiggi\\Documents\\GitSites\\xmf6\\mf6\\windows\\mf6",
#    'exe_name' : "../../../mf6/macosarm/mf6",
    'sim_name' : "flow",
    'sim_ws' : "sandbox3"
}

# --- Datos para la discretización temporal ---
time = {
    'units': "DAYS",
    'nper' : 1,
    'perioddata': [(1.0, 1, 1.0)]
}

# --- Datos para la discretización espacial ---
dis = {
    'nlay': 1,
    'nrow': 20,
    'ncol': 30,
    'delr': 0.5,
    'delc': 0.5,
    'top' : 0.0,
    'botm': -1.0
}

# --- Datos para las condiciones iniciales ---
ic_data = {
    'strt': 10
}
```

Se puede ver el contenido de los diccionarios como sigue:

```
print(init)
print(time)
print(dis)
print(ic_data)
```

```
{'exe_name': 'C:\\Users\\luiggi\\Documents\\GitSites\\xmf6\\mf6\\windows\\mf6', 'sim_name':
{'units': 'DAYS', 'nper': 1, 'perioddata': [(1.0, 1, 1.0)]}
{'nlay': 1, 'nrow': 20, 'ncol': 30, 'delr': 0.5, 'delc': 0.5, 'top': 0.0, 'botm': -1.0}
{'strt': 10}
```

Para los datos de las condiciones de frontera creamos listas y arreglos de numpy como se muestra a continuación:

```
# --- Datos para las condiciones de frontera ---
chd_data = []
for row in range(dis['nrow']):
    chd_data.append([(0, row, 0), 10.0])       # Condición en la pared izquierda
    chd_data.append([(0, row, dis['ncol'] - 1), 5.0]) # Condición en la pared derecha

#print(chd_data)
```

```
# --- Datos para las propiedades de flujo ---
# Creamos un arreglo de conductividad hidráulica para toda la malla
k_data = np.random.rand(dis['nlay'], dis['nrow'], dis['ncol'])*1.0

# Asignamos una conductividad constante en una sección del dominio
k_data[:,5:15,10:20] = 0.1

#print(k_data)
```

15.3 Paso 2. Función `build()`.

La función `build()` define la siguiente secuencia de pasos (véase la sección **Paquetes** en la notebook 00_MF6_Intro.ipynb):

1. Inicialización de la simulación.
2. Componentes:
    - Tiempos de simulación.
    - Solucionador numérico.

- Modelo GWF.

3. Paquetes:

    - Discretización espacial (DIS).
    - Condiciones iniciales (IC).
    - Condiciones de frontera (CHD).
    - Propiedades de flujo (NPF).
    - Configuración de la salida (OC).

La función regresa un objeto para la simulación (`o_sim`) y un objeto del modelo GWF (`o_gwf`).

```
def build(init, time, mesh, ic_data, chd_data, k_data):

    ...

    return o_sim, o_gwf
```

La implementación de la función se puede ver el archivo gwf_model_1.py.

Ejecutamos la función como sigue:

```
# --- Construcción de la simulación y del modelo GWF ---
o_sim, o_gwf = build(init, time, dis, ic_data, chd_data, k_data)
```

```
# --- Impresión del estado de la simulación ---
#print(o_sim)
```

```
# --- Impresión del estado del modelo GWF ---
#print(o_gwf)
```

15.4 Paso 3. Escritura de archivos.

A través del objeto de la simulación, `o_sim`, escribimos los archivos de entrada para MODFLOW 6:

```
# --- Escribimos los archivos ---
o_sim.write_simulation()
```

```
writing simulation...
  writing simulation name file...
  writing simulation tdis package...
  writing solution package ims_-1...
  writing model flow...
    writing model name file...
    writing package dis...
    writing package ic...
    writing package chd_0...
INFORMATION: maxbound in ('gwf6', 'chd', 'dimensions') changed to 40 based on size of stress_
    writing package npf...
    writing package oc...
```

15.5 Paso 4. Ejecución de la simulación.

A través del objeto de la simulación, `o_sim`, ejecutamos la simulación:

```
# --- ejecutamos la simulación ---
o_sim.run_simulation(silent=False)
```

```
FloPy is using the following executable to run the model: ..\..\..\..\mf6\windows\mf6.exe
                                    MODFLOW 6
                U.S. GEOLOGICAL SURVEY MODULAR HYDROLOGIC MODEL
                            VERSION 6.6.1 02/10/2025

   MODFLOW 6 compiled Feb 10 2025 17:37:25 with Intel(R) Fortran Intel(R) 64
   Compiler Classic for applications running on Intel(R) 64, Version 2021.7.0
                               Build 20220726_000000

This software has been approved for release by the U.S. Geological
Survey (USGS). Although the software has been subjected to rigorous
review, the USGS reserves the right to update the software as needed
pursuant to further analysis and review. No warranty, expressed or
implied, is made by the USGS or the U.S. Government as to the
functionality of the software and related material nor shall the
fact of release constitute any such warranty. Furthermore, the
software is released on condition that neither the USGS nor the U.S.
Government shall be held liable for any damages resulting from its
authorized or unauthorized use. Also refer to the USGS Water
Resources Software User Rights Notice for complete use, copyright,
and distribution information.
```

```
 MODFLOW runs in SEQUENTIAL mode

 Run start date and time (yyyy/mm/dd hh:mm:ss): 2025/05/18 16:33:31

 Writing simulation list file: mfsim.lst
 Using Simulation name file: mfsim.nam

    Solving:  Stress period:     1    Time step:     1

 Run end date and time (yyyy/mm/dd hh:mm:ss): 2025/05/18 16:33:31
 Elapsed run time:  0.122 Seconds

 Normal termination of simulation.

(True, [])
```

15.6 Paso 5. Recuperación de los resultados.

Los datos los vamos a recuperar de los archivos generados por la simualción, en este caso de `flow.hds` y `flow.bud`.

- La carga hidráulica se almacenará en el arreglo `head`.
- La descarga específica se almacenará en los arreglos `qx`, `qy` y `qz`.

```
### --- Recuperación de la carga hidráulica ---

# Construimos el nombre con la ruta y extensión correcta
headfile = os.path.join(f"{init["sim_ws"]}", f"{init["sim_name"]}.hds")

# Objeto para obtener y manipular arreglos o series de tiempo de arreglos con datos
# de una o más celdas con información del archivo binario de la carga hidráulica.
hds = flopy.utils.HeadFile(headfile)

# Solo se obtiene la capa 0, cuando no se pone ningún valor en el parámetro mlay se obtienen
head = hds.get_data(mflay=0)

### --- Recuperación de la descarga específica ---

# Construimos el nombre con la ruta y extensión correcta
```

```
budfile = os.path.join(f"{init["sim_ws"]}", f"{init["sim_name"]}.bud")

# Objeto para obtener y manipular arreglos o series de tiempo de arreglos con datos
# de una o más celdas con información del archivo binario del budget.
bud  = flopy.utils.CellBudgetFile(budfile)

# Cargamos resultados de la descarga específica
spdis = bud.get_data(text="DATA-SPDIS")[0]

# Extraemos las componentes del flujo en un arreglo 3D para graficación
qx, qy, qz = flopy.utils.postprocessing.get_specific_discharge(spdis, o_gwf)

# Calculamos la norma de la descarga específica
n_q = np.sqrt(np.square(qx[0]) + np.square(qy[0]))
```

En este momento tenemos los siguientes arreglos: * `head`, carga hidráulica. * `spdis`, descarga específica en un arreglo de tipo `recarray` con 6 columnas. * `qx`, `qy`, `qz`, componentes de la descarga específica. * `n_q`, norma del vector de descarga específica para graficación.

15.7 Paso 6. Visualización de los resultados.

Se visualiza los siguiente: * Permeabilidad (k, `k_data`). * Carga hidráulica (h, `head`). * Descarga específica (q, `(qx, qy)`)

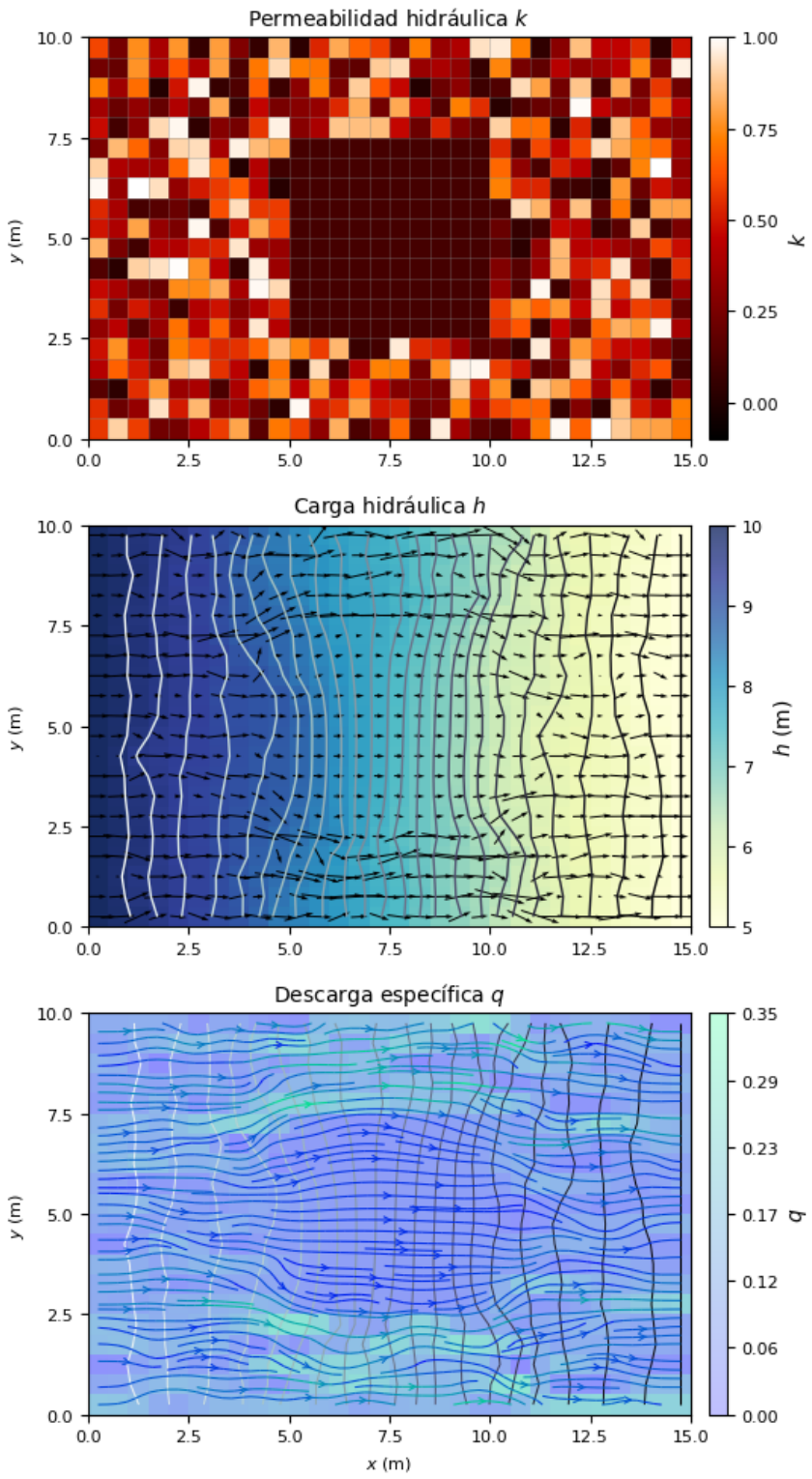
```
# --- Parámetros para las gráficas
grid = o_gwf.modelgrid
x, y, z = grid.xyzcellcenters
xticks = np.linspace(grid.extent[0], grid.extent[1], 7)
yticks = np.linspace(grid.extent[2], grid.extent[3], 5)
xlabels = [f'{x:1.1f}' for x in xticks]
ylabels = [f'{y:1.1f}' for y in yticks]
kvmin = 1.0 #np.nanmin(k_data)
kvmax = 0.0 #np.nanmax(k_data)
hvmin = np.nanmin(head)
hvmax = np.nanmax(head)
qvmin = 0.00 #np.nanmin(n_q)
qvmax = 0.35 #np.nanmax(n_q)
```

```
# --- Definición de la figura ---
```

```
# Se harán tres gráficas en una sola figura.
fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize =(10,10))

#--- Gráfica 1. ---
# Mapa visual del modelo. En este caso el modelo es 'o_gwf'.
kview = flopy.plot.PlotMapView(model = o_gwf, ax = ax1)

# Visualización de la malla, con una transparencia.
kview.plot_grid(linewidths = 0.5, alpha = 0.5)

# Visualización de la permeabilidad hidráulica usando un mapa de color.
k_ac = kview.plot_array(k_data, cmap = "gist_heat", vmin = kvmin, vmax = kvmax)

# Barra de color para la permeabilidad hidráulica.
k_cb = plt.colorbar(k_ac, ax = ax1, label = "$k$",
                    ticks = [0.0, 0.25, 0.50, 0.75, 1.0],
                    cax = xmf6.vis.cax(ax1, k_ac))
k_cb.ax.tick_params(labelsize=8)

# Personalización de la gráfica.
ax1.set_title("Permeabilidad hidráulica $k$", fontsize=10)
ax1.set_ylabel("$y$ (m)", fontsize = 8)
ax1.set_xticks(ticks = xticks, labels = xlabels, fontsize = 8)
ax1.set_yticks(ticks = yticks, labels = ylabels, fontsize = 8)
ax1.set_aspect('equal')

### Gráfica 2.

# Mapa visual del modelo. En este caso el modelo es 'o_gwf'.
hview = flopy.plot.PlotMapView(model = o_gwf, ax = ax2)

# Visualización de la carga hidráulica usando un mapa de color.
h_ac = hview.plot_array(head, cmap = "YlGnBu", vmin = hvmin, vmax = hvmax, alpha = 0.75)

# Visualización de la carga hidráulica usando contornos.
hview.contour_array(head, levels = 30, cmap = "bone", linewidths = 1.0)

# Visualización de los vectores de la descarga específica.
ax2.quiver(x, y, qx[0], qy[0], scale = 3,
           color = 'k', linewidth = 0.95, pivot = 'middle')

# Barra de color para la carga hidráulica.
```

```
h_cb = plt.colorbar(h_ac, ax = ax2, label = "$h$ (m)",
                    cax = xmf6.vis.cax(ax2, h_ac))
h_cb.ax.tick_params(labelsize=8)

# Personalización de la gráfica.
ax2.set_title("Carga hidráulica $h$", fontsize=10)
ax2.set_ylabel("$y$ (m)", fontsize = 8)
ax2.set_xticks(ticks = xticks, labels = xlabels, fontsize = 8)
ax2.set_yticks(ticks = yticks, labels = ylabels, fontsize = 8)
ax2.set_aspect('equal')

### Gráfica 3.

# Mapa visual del modelo. En este caso el modelo es 'o_gwf'.
fview = flopy.plot.PlotMapView(model = o_gwf, ax = ax3)

# Visualización de la descarga hidráulica usando un mapa de color.
q_ac = fview.plot_array(n_q, cmap = "winter", vmin = qvmin, vmax = qvmax, alpha = 0.25)

# Visualización de la carga hidráulica usando contornos.
fview.contour_array(head, levels = 20, cmap = 'bone', linewidths = 0.75, )

# Visualización de líneas de corriente de la descarga específica.
ax3.streamplot(x, y[::-1][:], qx[0], qy[0][::-1],
               density = [2, 1.5], linewidth = 0.75, broken_streamlines = True,
               color = n_q, cmap = "winter",
               arrowstyle = "->", arrowsize = 0.75,  )

# Barra de color para la norma de la descarga hidráulica.
q_cb = plt.colorbar(q_ac, ax=ax3, label="$q$",
                    ticks = np.linspace(0.0, 0.35, 7),
                    format = "{x:3.2f}",
                    cax = xmf6.vis.cax(ax3, q_ac))
q_cb.ax.tick_params(labelsize=8)

# Personalización de la gráfica.
ax3.set_title("Descarga específica $q$", fontsize=10)
ax3.set_xlabel("$x$ (m)", fontsize = 8)
ax3.set_ylabel("$y$ (m)", fontsize = 8)
ax3.set_xticks(ticks = xticks, labels = xlabels, fontsize = 8)
ax3.set_yticks(ticks = yticks, labels = ylabels, fontsize = 8)
ax3.set_aspect('equal')
```

```
plt.tight_layout()
plt.savefig("03_MF6.pdf")
plt.show()
```


Permeabilidad hidráulica k
y (m)
k
Carga hidráulica h
y (m)
h (m)
Descarga específica q
y (m)
q
x (m)

16 sim_ws = sandbox4

```
import numpy as np
import matplotlib.pyplot as plt
import flopy
import os
import xmf6
```

```
init = {
    'sim_name' : "flow",
    'exe_name' : "C:\\Users\\luiggi\\Documents\\GitSites\\xmf6\\mf6\\windows\\mf6",
    'sim_ws' : "sandbox4"
}

time = {
    'units': "DAYS",
    'nper' : 1,
    'perioddata': [(1.0, 1, 1.0)]
}

ims = {}

gwf = {
    'modelname': init["sim_name"],
    'model_nam_file': f"{init["sim_name"]}.nam",
    'save_flows': True
}

dis = {
    'nlay': 1,
    'nrow': 20,
    'ncol': 30,
    'delr': 0.5,
    'delc': 0.5,
    'top' : 0.0,
    'botm': -1.0
```

```
}

ic = {
    'strt': 10
}

chd_data = []
for row in range(dis['nrow']):
    chd_data.append([(0, row, 0), 10.0])       # Condición en la pared izquierda
    chd_data.append([(0, row, dis['ncol'] - 1), 5.0]) # Condición en la pared derecha

chd = {
    'stress_period_data': chd_data,
}

k_data = np.random.rand(dis['nlay'], dis['nrow'], dis['ncol'])*1.0
k_data[:,5:15,10:20] = 0.1

npf = {
    'save_specific_discharge': True,
    'k': k_data,
}

oc = {
    'budget_filerecord': f"{init['sim_name']}.bud",
    'head_filerecord': f"{init['sim_name']}.hds",
    'saverecord': [("HEAD", "ALL"), ("BUDGET", "ALL")],
    'printrecord': [("HEAD", "ALL")]
}
```

- Función `xmf6.gwf.initialize()`.
  - SIM
  - TDIS
  - IMS
- Función `xmf6.gwf.build()`.
  - GWF
    - DIS
    - IC
    - CHD
    - NPF
    - WEL

* OC

```
o_sim = xmf6.gwf.initialize(init = init, time = time, ims = ims)
```

```
sim configuration::
  sim_name = flow
  exe_name = C:\Users\luiggi\Documents\GitSites\xmf6\mf6\windows\mf6
```

```
time configuration::
  units = DAYS
  nper = 1
  perioddata = [(1.0, 1, 1.0)]
---
```

```
numerical solution configuration::
---
```

```
o_gwf = xmf6.gwf.build(o_sim,
                       gwf = gwf, dis = dis, ic = ic, chd = chd, npf = npf, oc = oc)
```

```
numerical model configuration::
  modelname = flow
  model_nam_file = flow.nam
  save_flows = True
---
```

```
spatial discretization configuration::
  nlay = 1
  nrow = 20
  ncol = 30
  delr = 0.5
```

```
  delc = 0.5
  top = 0.0
  botm = -1.0
---



initial conditions configuration::
  strt = 10
---



boundary conditions configuration::
  stress_period_data = [[(0, 0, 0), 10.0], [(0, 0, 29), 5.0], [(0, 1, 0), 10.0], [(0, 1, 29)
---



flow properties configuration::
  save_specific_discharge = True
  k = [[[0.60954214 0.4165278  0.59880469 0.77742402 0.83425926 0.29178523
   0.07997176 0.63055316 0.10080799 0.84034342 0.58251966 0.14685616
   0.71997601 0.19212944 0.16469966 0.24887592 0.73077801 0.90353812
   0.02421148 0.3209033  0.42660233 0.33009569 0.71149392 0.06640044
   0.53761123 0.15403321 0.73331497 0.6325598  0.4233338  0.10060297]
  [0.58172865 0.59489213 0.47720888 0.7186139  0.17088769 0.27663864
   0.4431125  0.10531354 0.56255331 0.18103481 0.87588637 0.60414895
   0.07070339 0.61333832 0.58070383 0.24453826 0.84468363 0.31074807
   0.3366561  0.01063396 0.86642706 0.33885709 0.52127048 0.43854741
   0.63139824 0.88296731 0.61651247 0.62249954 0.53600603 0.79623583]
  [0.42651659 0.85189251 0.76862176 0.0192641  0.92486633 0.45859273
   0.95222947 0.74718305 0.7303584  0.96669004 0.03797059 0.41581452
   0.55995294 0.29258672 0.50915158 0.67583881 0.36547336 0.06630294
   0.97127454 0.60900597 0.15396387 0.60133809 0.8868297  0.36459134
   0.47613916 0.1834512  0.15922238 0.74681327 0.80542287 0.63698214]
  [0.08493063 0.21007637 0.31864139 0.10171217 0.10084855 0.65461473
   0.27637435 0.76873855 0.80473152 0.23698279 0.21767036 0.7080493
   0.49738631 0.09256222 0.74180426 0.85496122 0.29785593 0.30898477
   0.68138992 0.21398019 0.78980619 0.91025677 0.84595157 0.33737856
   0.62253958 0.71005846 0.38156734 0.73047296 0.08941825 0.5618577 ]
  [0.71061168 0.29468957 0.16549578 0.66360553 0.72325996 0.71499216
   0.08569351 0.37260317 0.131688   0.5795041  0.95926536 0.11343725
```

```
 0.27230613 0.83758215 0.30964065 0.37883134 0.29639367 0.63075808
 0.81085514 0.49520185 0.97593826 0.14202662 0.67044399 0.92963538
 0.80701181 0.56924532 0.75538746 0.96009188 0.15138022 0.21820589]
[0.92162446 0.11777162 0.66816536 0.91985051 0.93065289 0.24674941
 0.51027642 0.35290523 0.15075646 0.21425436 0.1        0.1
 0.1        0.1        0.1        0.1        0.1        0.1
 0.1        0.1        0.03633807 0.32857986 0.29322688 0.00196099
 0.52364293 0.26822774 0.37733374 0.61720164 0.79750417 0.63872012]
[0.40899421 0.89045582 0.63089737 0.40582278 0.1574124  0.97798622
 0.25798593 0.92841246 0.77109303 0.66305755 0.1        0.1
 0.1        0.1        0.1        0.1        0.1        0.1
 0.1        0.1        0.4260733  0.5711068  0.06047622 0.58723017
 0.37066854 0.03590139 0.83174182 0.68410482 0.19430228 0.74566653]
[0.48909608 0.40143397 0.71766863 0.88823988 0.19841955 0.46523447
 0.23884811 0.57743066 0.11683424 0.44784673 0.1        0.1
 0.1        0.1        0.1        0.1        0.1        0.1
 0.1        0.1        0.11639044 0.74145373 0.14097811 0.88323276
 0.32756397 0.6785058  0.59312826 0.72339001 0.35930303 0.07056934]
[0.47188999 0.67402566 0.0598525  0.72428385 0.88550466 0.3263359
 0.56709129 0.11379937 0.8986357  0.17379803 0.1        0.1
 0.1        0.1        0.1        0.1        0.1        0.1
 0.1        0.1        0.72221205 0.89874218 0.69565134 0.36238807
 0.11179486 0.78635862 0.57046664 0.58615268 0.59553973 0.94976285]
[0.50503726 0.6355614  0.44647873 0.03569859 0.44456797 0.65613192
 0.01214675 0.41611311 0.76761505 0.66311039 0.1        0.1
 0.1        0.1        0.1        0.1        0.1        0.1
 0.1        0.1        0.56194922 0.03204619 0.95461408 0.75246119
 0.73386853 0.21528988 0.87108699 0.31797345 0.84903037 0.19728488]
[0.16191081 0.20698614 0.36144665 0.18948286 0.81532856 0.96762703
 0.26893315 0.75286238 0.27806896 0.98215767 0.1        0.1
 0.1        0.1        0.1        0.1        0.1        0.1
 0.1        0.1        0.09628858 0.46389259 0.57139433 0.05555563
 0.06641261 0.10354844 0.63821466 0.64589331 0.5987501  0.33380669]
[0.21081234 0.46304405 0.94228682 0.57480994 0.99385803 0.87440511
 0.24412777 0.3378692  0.80118833 0.94706171 0.1        0.1
 0.1        0.1        0.1        0.1        0.1        0.1
 0.1        0.1        0.54011187 0.70446813 0.85334526 0.53270045
 0.91712264 0.97645601 0.9776483  0.94335103 0.55395428 0.49716771]
[0.7351477  0.1976568  0.04880156 0.77312046 0.30244749 0.7289515
 0.96047552 0.17623309 0.86618092 0.51502403 0.1        0.1
 0.1        0.1        0.1        0.1        0.1        0.1
 0.1        0.1        0.55417786 0.46583961 0.65164091 0.46724248
 0.67813855 0.97581013 0.35308116 0.23586903 0.60014789 0.15751739]
```

```
  [0.71182831 0.81341408 0.24928742 0.81940459 0.09119011 0.14351805
   0.87831421 0.78086573 0.24781307 0.72275756 0.1        0.1
   0.1        0.1        0.1        0.1        0.1        0.1
   0.1        0.1        0.14295445 0.17638646 0.27632532 0.40086105
   0.75169229 0.64590893 0.44500937 0.97433426 0.85282528 0.47198021]
  [0.97070155 0.79323625 0.0926608  0.33863488 0.65114285 0.05857631
   0.86041702 0.58136518 0.02877632 0.62039913 0.1        0.1
   0.1        0.1        0.1        0.1        0.1        0.1
   0.1        0.1        0.8772467  0.71246371 0.64517283 0.6291897
   0.5008465  0.10873085 0.64639637 0.23427497 0.80667509 0.60830658]
  [0.67902921 0.41613792 0.15886561 0.01517551 0.32729675 0.31170932
   0.45055506 0.7638732  0.90504783 0.27400773 0.04283054 0.42026607
   0.20658033 0.97060952 0.73686589 0.07027035 0.27332108 0.75062355
   0.23843715 0.81012994 0.07389765 0.64320421 0.95219903 0.69276063
   0.72732776 0.86836107 0.96046007 0.90047045 0.5935871  0.18096066]
  [0.74038223 0.88442429 0.94867736 0.23565713 0.27983582 0.92766227
   0.23025879 0.29135481 0.00782693 0.89991441 0.52855816 0.3398044
   0.12428893 0.09020395 0.7062567  0.6790088  0.76948659 0.81606104
   0.82018279 0.77556398 0.0196206  0.44562157 0.85888991 0.75357674
   0.73229664 0.04684407 0.99578841 0.62992741 0.44665919 0.64277267]
  [0.89412357 0.64700497 0.98000352 0.1885739  0.99814334 0.89585734
   0.85589407 0.15125979 0.84876382 0.12537594 0.2655866  0.62314632
   0.88980534 0.4433292  0.41013799 0.85601778 0.97011184 0.76082126
   0.35077519 0.86173279 0.53425341 0.09267336 0.24040696 0.32353725
   0.3740524  0.94398973 0.63865311 0.53522329 0.98313504 0.33025271]
  [0.3454997  0.19925824 0.82498561 0.61767584 0.3233694  0.30111378
   0.88094893 0.37886199 0.25491263 0.12571527 0.89747098 0.60414257
   0.21775654 0.32055469 0.25358899 0.75421632 0.56402424 0.70097943
   0.26981733 0.70447768 0.12850656 0.08597061 0.32757756 0.61899271
   0.85444533 0.00866161 0.62179413 0.54558051 0.64146646 0.72862054]
  [0.18398859 0.25501667 0.19902577 0.69882713 0.00929394 0.51422457
   0.59863361 0.27425144 0.62329835 0.57342179 0.7367898  0.9850804
   0.01306854 0.76514079 0.41054708 0.87316507 0.170627   0.07714944
   0.3820619  0.96639174 0.33743642 0.29670213 0.40844918 0.86048357
   0.46868387 0.90048778 0.24200684 0.40585159 0.01574999 0.98852646]]]
---



output configuration::
  budget_filerecord = flow.bud
  head_filerecord = flow.hds
  saverecord = [('HEAD', 'ALL'), ('BUDGET', 'ALL')]
```

```
  printrecord = [('HEAD', 'ALL')]
---
```

```
o_sim.write_simulation(silent = True)
```

```
o_sim.run_simulation(silent=False)
```

```
FloPy is using the following executable to run the model: ..\..\..\..\mf6\windows\mf6.exe
                                    MODFLOW 6
                U.S. GEOLOGICAL SURVEY MODULAR HYDROLOGIC MODEL
                                VERSION 6.6.1 02/10/2025

   MODFLOW 6 compiled Feb 10 2025 17:37:25 with Intel(R) Fortran Intel(R) 64
   Compiler Classic for applications running on Intel(R) 64, Version 2021.7.0
                               Build 20220726_000000

This software has been approved for release by the U.S. Geological
Survey (USGS). Although the software has been subjected to rigorous
review, the USGS reserves the right to update the software as needed
pursuant to further analysis and review. No warranty, expressed or
implied, is made by the USGS or the U.S. Government as to the
functionality of the software and related material nor shall the
fact of release constitute any such warranty. Furthermore, the
software is released on condition that neither the USGS nor the U.S.
Government shall be held liable for any damages resulting from its
authorized or unauthorized use. Also refer to the USGS Water
Resources Software User Rights Notice for complete use, copyright,
and distribution information.


 MODFLOW runs in SEQUENTIAL mode

 Run start date and time (yyyy/mm/dd hh:mm:ss): 2025/05/18 16:35:52

 Writing simulation list file: mfsim.lst
 Using Simulation name file: mfsim.nam

    Solving:  Stress period:     1    Time step:     1

 Run end date and time (yyyy/mm/dd hh:mm:ss): 2025/05/18 16:35:52
 Elapsed run time:  0.118 Seconds
```

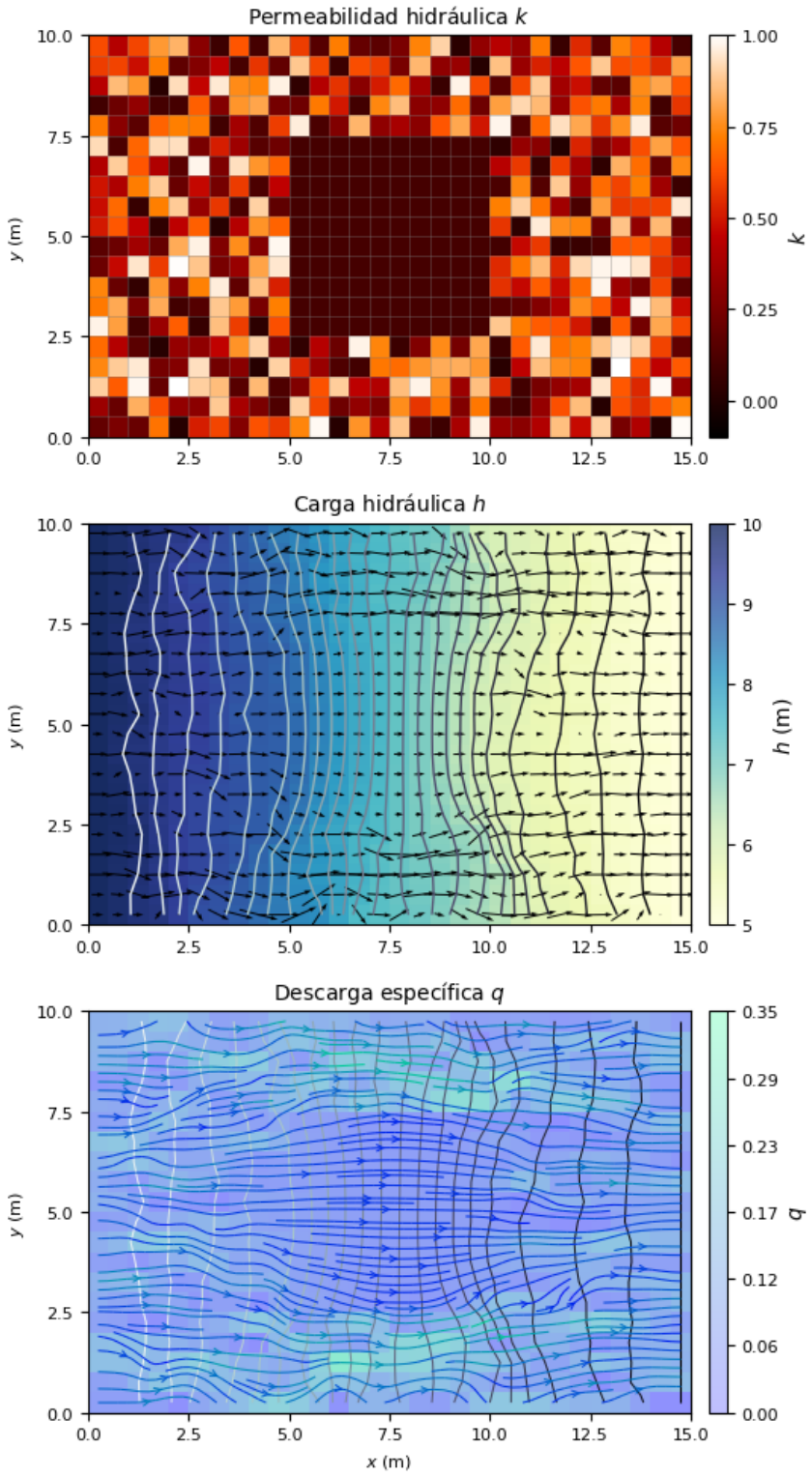
```
 Normal termination of simulation.
```

```
(True, [])
```

```
# --- Recuperamos los resultados de la simulación ---
head = xmf6.gwf.get_head(o_sim, o_gwf)
qx, qy, qz, n_q = xmf6.gwf.get_specific_discharge(o_sim, o_gwf)
```

```
# --- Parámetros para las gráficas ---
grid = o_gwf.modelgrid
x, y, z = grid.xyzcellcenters
xticks = np.linspace(grid.extent[0], grid.extent[1], 7)
yticks = np.linspace(grid.extent[2], grid.extent[3], 5)
xlabels = [f'{x:1.1f}' for x in xticks]
ylabels = [f'{y:1.1f}' for y in yticks]
kvmin = 1.0 #np.nanmin(k_data)
kvmax = 0.0 #np.nanmax(k_data)
hvmin = np.nanmin(head)
hvmax = np.nanmax(head)
qvmin = 0.00 #np.nanmin(n_q)
qvmax = 0.35 #np.nanmax(n_q)

# --- Definición de la figura ---
fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize =(10,10))

# --- Gráfica 1. ---
kview = flopy.plot.PlotMapView(model = o_gwf, ax = ax1)
kview.plot_grid(linewidths = 0.5, alpha = 0.5)
k_ac = kview.plot_array(k_data, cmap = "gist_heat", vmin = kvmin, vmax = kvmax)
k_cb = plt.colorbar(k_ac, ax = ax1, label = "$k$",
                    ticks = [0.0, 0.25, 0.50, 0.75, 1.0],
                    cax = xmf6.vis.cax(ax1, k_ac))
k_cb.ax.tick_params(labelsize=8)
ax1.set_title("Permeabilidad hidráulica $k$", fontsize=10)
ax1.set_ylabel("$y$ (m)", fontsize = 8)
ax1.set_xticks(ticks = xticks, labels = xlabels, fontsize = 8)
ax1.set_yticks(ticks = yticks, labels = ylabels, fontsize = 8)
ax1.set_aspect('equal')

# --- Gráfica 2. ---
hview = flopy.plot.PlotMapView(model = o_gwf, ax = ax2)
h_ac = hview.plot_array(head, cmap = "YlGnBu", vmin = hvmin, vmax = hvmax, alpha = 0.75)
```

```
hview.contour_array(head, levels = 30, cmap = "bone", linewidths = 1.0)
ax2.quiver(x, y, qx[0], qy[0], scale = 3,
           color = 'k', linewidth = 0.95, pivot = 'middle')
h_cb = plt.colorbar(h_ac, ax = ax2, label = "$h$ (m)",
                    cax = xmf6.vis.cax(ax2, h_ac))
h_cb.ax.tick_params(labelsize=8)
ax2.set_title("Carga hidráulica $h$", fontsize=10)
ax2.set_ylabel("$y$ (m)", fontsize = 8)
ax2.set_xticks(ticks = xticks, labels = xlabels, fontsize = 8)
ax2.set_yticks(ticks = yticks, labels = ylabels, fontsize = 8)
ax2.set_aspect('equal')

# --- Gráfica 3. ---
fview = flopy.plot.PlotMapView(model = o_gwf, ax = ax3)
q_ac = fview.plot_array(n_q, cmap = "winter", vmin = qvmin, vmax = qvmax, alpha = 0.25)
fview.contour_array(head, levels = 20, cmap = 'bone', linewidths = 0.75, )
ax3.streamplot(x, y[::-1][:], qx[0], qy[0][::-1],
               density = [2, 1.5], linewidth = 0.75, broken_streamlines = True,
               color = n_q, cmap = "winter",
               arrowstyle = "->", arrowsize = 0.75,  )
q_cb = plt.colorbar(q_ac, ax=ax3, label="$q$",
                    ticks = np.linspace(0.0, 0.35, 7),
                    format = "{x:3.2f}",
                    cax = xmf6.vis.cax(ax3, q_ac))
q_cb.ax.tick_params(labelsize=8)
ax3.set_title("Descarga específica $q$", fontsize=10)
ax3.set_xlabel("$x$ (m)", fontsize = 8)
ax3.set_ylabel("$y$ (m)", fontsize = 8)
ax3.set_xticks(ticks = xticks, labels = xlabels, fontsize = 8)
ax3.set_yticks(ticks = yticks, labels = ylabels, fontsize = 8)
ax3.set_aspect('equal')

plt.tight_layout()
plt.savefig("04_MF6.pdf")
plt.show()
```


Permeabilidad hidráulica k
y (m)
k
Carga hidráulica h
y (m)
h (m)
Descarga específica q
y (m)
q
x (m)

17 — Recuperamos los resultados de la simulación —

```
import numpy as np
import matplotlib.pyplot as plt
import flopy
import os
import xmf6
```

```
init = {
    'sim_name' : "flow",
    'exe_name' : "C:\\Users\\luiggi\\Documents\\GitSites\\xmf6\\mf6\\windows\\mf6",
    'sim_ws' : "sandbox5"
}

time = {
    'units': "DAYS",
    'nper' : 1,
    'perioddata': [(1.0, 1, 1.0)]
}

ims = {}

gwf = {
    'modelname': init["sim_name"],
    'model_nam_file': f"{init["sim_name"]}.nam",
    'save_flows': True
}

dis = {
    'nlay': 1,
    'nrow': 20,
    'ncol': 30,
    'delr': 0.5,
    'delc': 0.5,
```

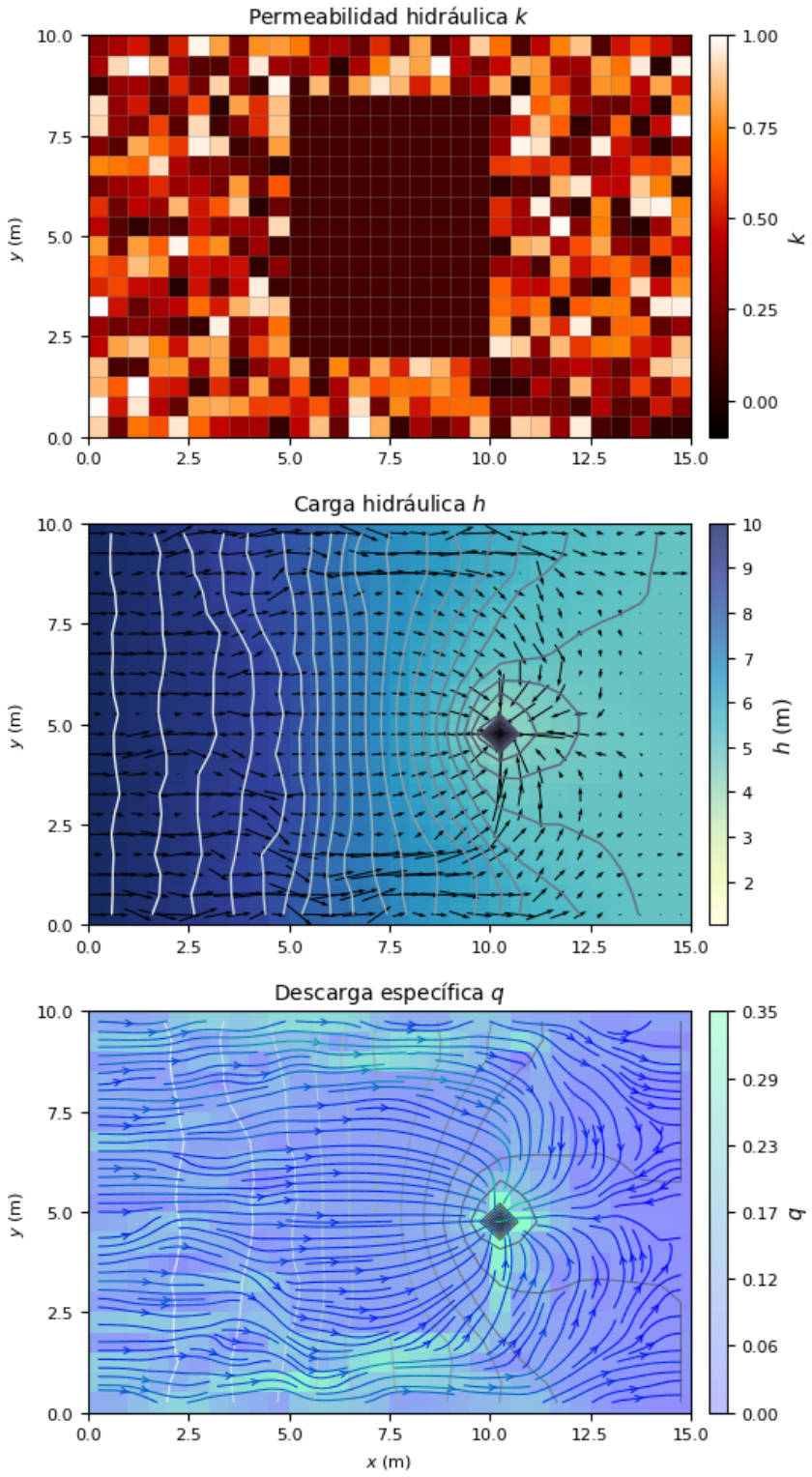
```
    'top' : 0.0,
    'botm': -1.0
}

ic = {
    'strt': 10
}

chd_data = []
for row in range(dis['nrow']):
    chd_data.append([(0, row, 0), 10.0])       # Condición en la pared izquierda
    chd_data.append([(0, row, dis['ncol'] - 1), 5.0]) # Condición en la pared derecha

chd = {
    'stress_period_data': chd_data,
}

k_data = np.random.rand(dis['nlay'], dis['nrow'], dis['ncol'])*1.0
k_data[:, dis['nrow']//6 : dis['nrow']*5//6, dis['ncol']//3 : dis['ncol']*2//3] = 0.1

npf = {
    'save_specific_discharge': True,
    'k': k_data,
}

well_data = [((0, dis['nrow']//2, dis['ncol']*2//3), -1.0)]

well = {
    'stress_period_data': well_data,
    'pname': "WEL-1",
    'save_flows': True
}

oc = {
    'budget_filerecord': f"{init['sim_name']}.bud",
    'head_filerecord': f"{init['sim_name']}.hds",
    'saverecord': [("HEAD", "ALL"), ("BUDGET", "ALL")],
    'printrecord': [("HEAD", "ALL")]
}

o_sim = xmf6.gwf.initialize(silent = True, init = init, time = time, ims = ims)
o_gwf = xmf6.gwf.build(o_sim, silent = True,
```

```
                        gwf = gwf, dis = dis, ic = ic, chd = chd, npf = npf, oc = oc, well = w

o_sim.write_simulation(silent = True)
o_sim.run_simulation(silent = True)
```

```
(True, [])
```

```
head = xmf6.gwf.get_head(o_sim, o_gwf)
qx, qy, qz, n_q = xmf6.gwf.get_specific_discharge(o_sim, o_gwf)

# --- Parámetros para las gráficas ---
grid = o_gwf.modelgrid
x, y, z = grid.xyzcellcenters
xticks = np.linspace(grid.extent[0], grid.extent[1], 7)
yticks = np.linspace(grid.extent[2], grid.extent[3], 5)
xlabels = [f'{x:1.1f}' for x in xticks]
ylabels = [f'{y:1.1f}' for y in yticks]
kvmin = 1.0 #np.nanmin(k_data)
kvmax = 0.0 #np.nanmax(k_data)
hvmin = np.nanmin(head)
hvmax = np.nanmax(head)
qvmin = 0.00 #np.nanmin(n_q)
qvmax = 0.35 #np.nanmax(n_q)

# --- Definición de la figura ---
fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize =(10,10))

# --- Gráfica 1. ---
kview = flopy.plot.PlotMapView(model = o_gwf, ax = ax1)
kview.plot_grid(linewidths = 0.5, alpha = 0.5)
k_ac = kview.plot_array(k_data, cmap = "gist_heat", vmin = kvmin, vmax = kvmax)
k_cb = plt.colorbar(k_ac, ax = ax1, label = "$k$",
                    ticks = [0.0, 0.25, 0.50, 0.75, 1.0],
                    cax = xmf6.vis.cax(ax1, k_ac))
k_cb.ax.tick_params(labelsize=8)
ax1.set_title("Permeabilidad hidráulica $k$", fontsize=10)
ax1.set_ylabel("$y$ (m)", fontsize = 8)
ax1.set_xticks(ticks = xticks, labels = xlabels, fontsize = 8)
ax1.set_yticks(ticks = yticks, labels = ylabels, fontsize = 8)
ax1.set_aspect('equal')
```

```
# --- Gráfica 2. ---
hview = flopy.plot.PlotMapView(model = o_gwf, ax = ax2)
h_ac = hview.plot_array(head, cmap = "YlGnBu", vmin = hvmin, vmax = hvmax, alpha = 0.75)
hview.contour_array(head, levels = 30, cmap = "bone", linewidths = 1.0)
ax2.quiver(x, y, qx[0], qy[0], scale = 3,
           color = 'k', linewidth = 0.95, pivot = 'middle')
h_cb = plt.colorbar(h_ac, ax = ax2, label = "$h$ (m)",
                    cax = xmf6.vis.cax(ax2, h_ac))
h_cb.ax.tick_params(labelsize=8)
ax2.set_title("Carga hidráulica $h$", fontsize=10)
ax2.set_ylabel("$y$ (m)", fontsize = 8)
ax2.set_xticks(ticks = xticks, labels = xlabels, fontsize = 8)
ax2.set_yticks(ticks = yticks, labels = ylabels, fontsize = 8)
ax2.set_aspect('equal')

# --- Gráfica 3. ---
fview = flopy.plot.PlotMapView(model = o_gwf, ax = ax3)
q_ac = fview.plot_array(n_q, cmap = "winter", vmin = qvmin, vmax = qvmax, alpha = 0.25)
fview.contour_array(head, levels = 20, cmap = 'bone', linewidths = 0.75, )
ax3.streamplot(x, y[::-1][:], qx[0], qy[0][::-1],
               density = [2, 1.5], linewidth = 0.75, broken_streamlines = True,
               color = n_q, cmap = "winter",
               arrowstyle = "->", arrowsize = 0.75,  )
q_cb = plt.colorbar(q_ac, ax=ax3, label="$q$",
                    ticks = np.linspace(0.0, 0.35, 7),
                    format = "{x:3.2f}",
                    cax = xmf6.vis.cax(ax3, q_ac))
q_cb.ax.tick_params(labelsize=8)
ax3.set_title("Descarga específica $q$", fontsize=10)
ax3.set_xlabel("$x$ (m)", fontsize = 8)
ax3.set_ylabel("$y$ (m)", fontsize = 8)
ax3.set_xticks(ticks = xticks, labels = xlabels, fontsize = 8)
ax3.set_yticks(ticks = yticks, labels = ylabels, fontsize = 8)
ax3.set_aspect('equal')

plt.tight_layout()
plt.savefig("05_MF6.pdf")
plt.show()
```


Permeabilidad hidráulica k
y (m)
k
Carga hidráulica h
h (m)
Descarga específica q
q
x (m)