Carlos Aznarán
Lima, 19 de enero de 2025

Problema 1 (Primer ejemplo elíptico)

Antes de modificar el programa 3, responda las siguientes preguntas.

1) ¿Cuál ecuación diferencial se está resolviendo?

Solución Se está resolviendo un problema de valor de frontera Robin de 2 puntos [**CORBINO2020112326**].

$$
\begin{cases}
\dfrac{\mathrm{d}^2 u}{\mathrm{d}x^2} = e^x, \text{ para } x \in [0,1]. \\
0 = u(0) - \left(\dfrac{\mathrm{d}u}{\mathrm{d}x}\right)_{x=0}. \\
2e = u(1) + \left(\dfrac{\mathrm{d}u}{\mathrm{d}x}\right)_{x=1}.
\end{cases} \tag{1}
$$

2) ¿Cuál es la función de fuerza o lado derecho?

Solución La función de fuerza es e^x.

3) ¿Cuál es el tipo de condiciones de contorno y cuáles son los valores de su lado derecho?

Solución La condición de contorno es Robin con los coeficientes Dirichlet y Neumann iguales a 1. Los valores son $g(x=0) = 0$ y $g(x=1) = 2e$.

4) ¿Cuál es la solución exacta?

Solución Integramos dos veces y obtenemos la solución general.

$$
\begin{aligned}
\iint \frac{\mathrm{d}^2 u}{\mathrm{d}x^2}\,\mathrm{d}x\,\mathrm{d}x &= \iint e^x\,\mathrm{d}x\,\mathrm{d}x. \\
\int \frac{\mathrm{d}u}{\mathrm{d}x}\,\mathrm{d}x &= \int \left(e^x + C_1\right)\,\mathrm{d}x. \\
u(x) &= e^x + C_1 x + C_2.
\end{aligned}
$$

Ahora, apliquemos las condiciones de frontera Robin.

$$
\begin{cases}
0 = u(0) - \left(\dfrac{\mathrm{d}u}{\mathrm{d}x}\right)_{x=0} = e^0 + C_1(0) + C_2 - \left(e^0 + C_1\right) = C_2 - C_1. \\
2e = u(1) + \left(\dfrac{\mathrm{d}u}{\mathrm{d}x}\right)_{x=1} = e^1 + C_1(1) + C_2 + \left(e^1 + C_1\right) = 2e + 2C_1 + C_2.
\end{cases} \tag{2}
$$

El sistema (2) tiene como solución $C_1 = C_2 = 0$. $\therefore$ la solución de (1) es $u(x) = e^x$.

Explicación del Programa 3

- En la línea 1 encontramos el *shebang*[1], esto permite ejecutar un script de Octave `./elliptic1D.m` con la opción de modo de procesamiento por lotes (batch), para esto se necesita tener permisos de ejecución (por ejemplo, `chmod +x elliptic1D.m`).

[1] https://en.wikipedia.org/wiki/Shebang_(Unix)

- En las líneas 2 al 10 tenemos un comentario sobre el programa de modo que ayude al codificador a obtener un contexto del problema a resolver.

- En la línea 12, la función `addpath` agrega el directorio `"/usr/share/mole/matlab/"` a la ruta de búsqueda de la función. Allí se encuentran el conjunto de scripts Octave / MATLAB de la biblioteca MOLE. Vea el Programa 8.

- En las líneas 14 y 15, se inicializan los identicadores `west` (oeste, izquierda), `east` (este, derecha) con los valores de 0 y 1, respectivamente, estos representan los valores de frontera del dominio espacial en (1).

- En la línea 21, llamamos a la función `lap`, este genera un operador Laplaciano discreto extendido que requiere como argumentos obligatorios el orden de precisión `k`, el número de celdas `m` y el tamaño de paso `dx`.

$$L = L^{(k)} = D^{(k)} G^{(k)} = DG. \qquad \left(\triangle = \nabla \cdot \nabla\right),$$

  donde D y G son los operadores miméticos de divergencia y gradiente, respectivamente. Dado que $D \in \mathbb{R}^{(m+2)\times(m+1)}$ y $G \in \mathbb{R}^{(m+1)\times(m+2)}$, entonces $L \in \mathbb{R}^{(m+2)\times(m+2)}$.

- En la línea 22, con la función `figure` desactivamos que se muestre la figura en la pantalla, preferimos solamente guardar la gráfica.

- En la línea 23, con la función `spy` graficamos (no se mostrará) el patrón de dispersidad de L.

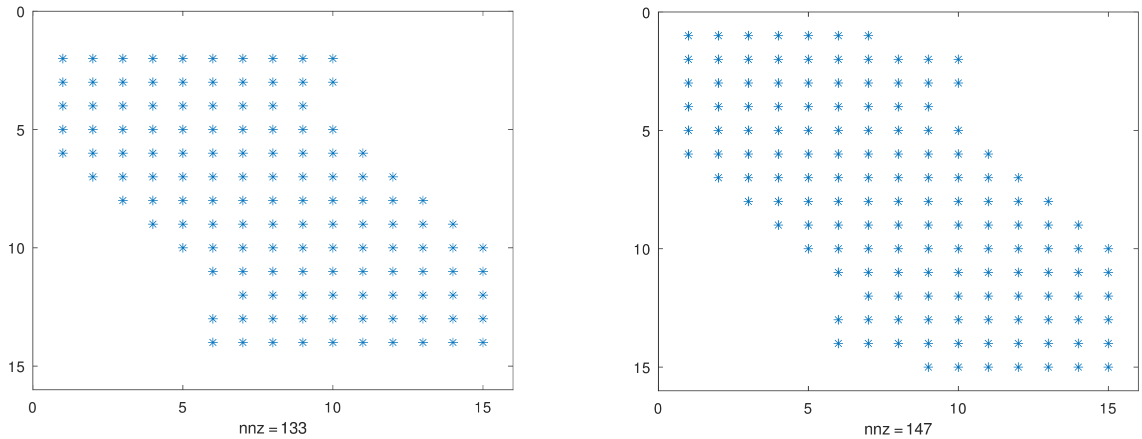


Figura 1: Izquierda: Representación dispersa de L hasta la línea 21. La primera y la última fila son vectores de ceros. Derecha: Representación dispersa de L hasta la línea 29. La matriz $L \in \mathbb{R}^{15\times 15}$.

- En la línea 24, con la función `saveas` guardamos esta gráfica en formato PDF y recortado.

- En la línea 29, llamamos a la función `robinBC`, este requiere como argumentos obligatorios el orden de precisión `k`, el número de celdas `m`, el tamaño de paso `dx`, el coeficiente Dirichlet `a` y el coeficiente Neumann `b`. Esta función devuelve una matriz en $\mathbb{R}^{(m+2)\times(m+2)}$. Actualizamos la matriz `L` según el Algoritmo 1.

Algoritmo 1: Actualizaciones del operador Laplaciano discreto extendido.

$A \leftarrow L$;
$F \leftarrow f$;
$A \leftarrow A + R_G$;
$U \leftarrow \text{solve}\,(A, F)$;

- En la línea 34, creamos la malla escalonada unidimensional, note que los puntos internos son los centros de las celdas equiespaciados por `dx`. La distancia entre el extremo izquierdo y el posterior punto malla, así como del extremo derecho y el anterior punto malla es `dx/2`.

- En las líneas 35 y 43, guardamos `save` la malla computacional y la solución en el formato HDF5 para posterior post procesamiento.

- En la líneas 39 y 40, aplicamos las condiciones de frontera Robin, empleamos la función `exp`. El signo menos que antecede al coeficiente Neumann `b` se debe a que en el borde izquierdo de la malla el vector normal hacia afuera apunta hacia la izquierda, mientras que en el borde derecho el vector normal hacia afuera apunta hacia la derecha.

- En la línea 42, resolvemos el sistema de ecuaciones lineales disperso con la función `mldivide`.

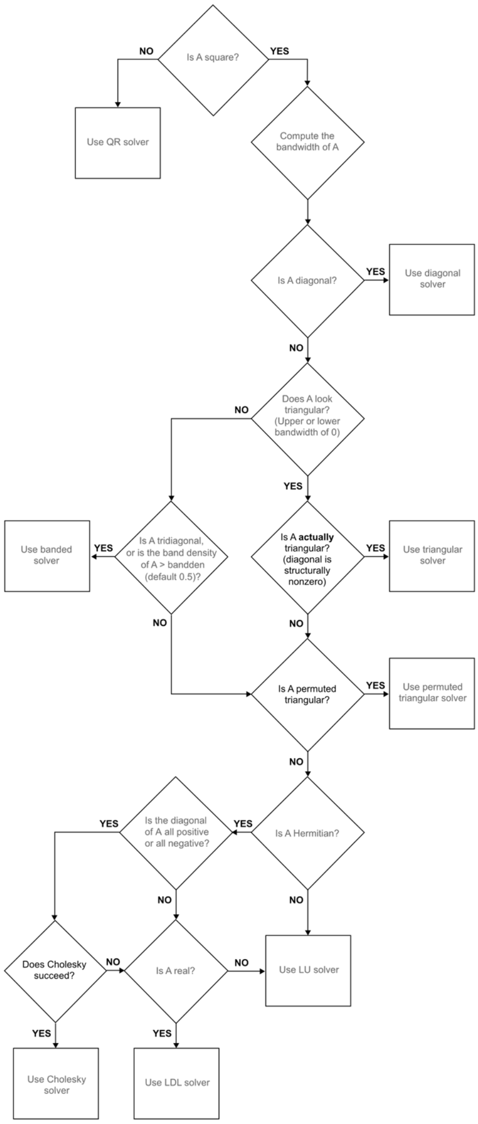


Figura 2: Diagrama de flujo del solucionador `mldivide` para matrices disperas que emplea MATLAB. Recuperado de https://www.mathworks.com/help/matlab/ref/double.mldivide.html.

Resultados del Programa 3

En primer lugar, mostramos la gráfica a escala 1:1 de la solución exacta y de la solución mimética obtenida en el Programa 3.

En segundo lugar, mostramos una gráfica del error en cada punto de la malla computacional dada por

$$\text{Error de } u \text{ en } x_{j-\frac{1}{2}} = \left| u\left(x_{j-\frac{1}{2}}\right) - u_{j-\frac{1}{2}} \right|.$$

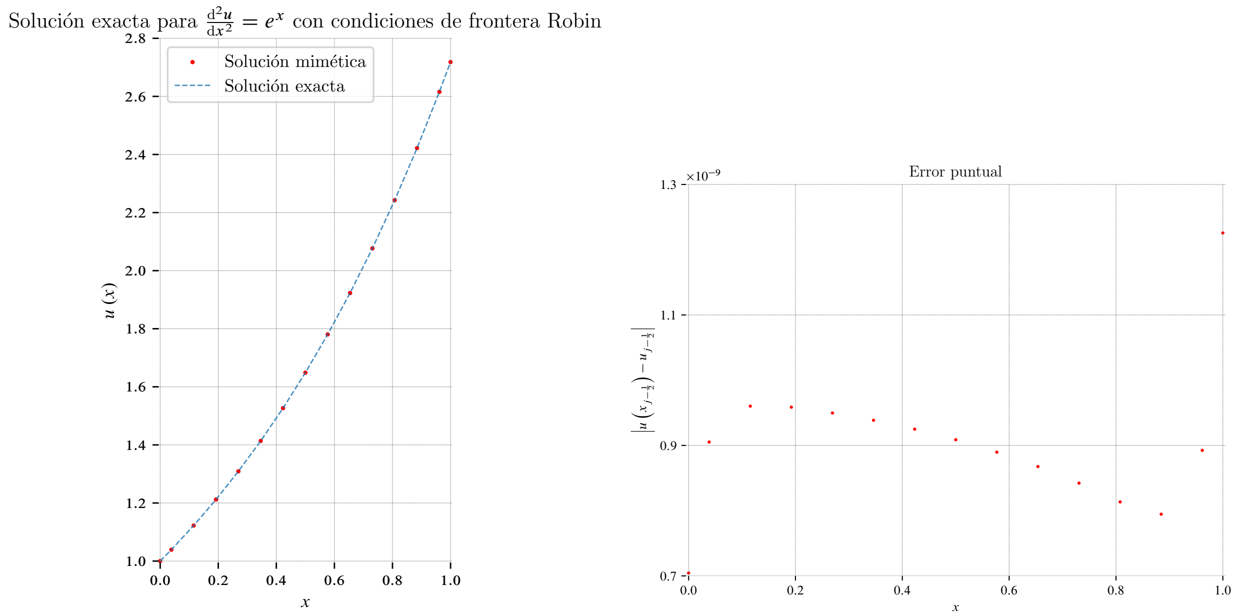


Figura 3: Izquierda: Solución de (1) usando `k=6` y `m=2k+1=13`. Derecha: Error en la malla escalonada $\left\{0, \ldots, x_{j-\frac{1}{2}} \ldots, 1\right\}$

Por último, mostramos la tabla de los errores y el orden de convergencia numérico.

Δx	Error ℓ_1	Orden
1.562×10^{-2}	4.410×10^{-2}	-
1.535×10^{-2}	4.464×10^{-2}	-0.678

Cuadro 1: Tabla de errores de aproximación de U en $x_{j-\frac{1}{2}}$ y el orden convergencia numérico obtenido.

Explicación del Programa 4

- En las tres primeras líneas tenemos un comentario sobre el programa de modo que ayude al codificador a obtener un contexto del problema a resolver.
- En la línea 5 `ARMA_USE_HDF5` para habilitar la lectura y/o escritura de archivos HDF5.
- En la línea 16, se inicializa el orden de precisión del operador.
- En las líneas 17 y 18, se inicializan los valores que representan los bordes del dominio espacial.
- `using Real = double` es un alias que se ha definido en `#include <mole/robinbc.h>`.
- En la línea 23 se instancia la clase `Laplacian`
- En la línea 26 se instancia la clase `RobinBC`
- `arma::vec`.
- .

Problema 2

Copie el programa 3 y modifique lo siguiente:

- Cambie el orden de precisión a 4.

- Asigne el número de celdas a 10.
- Mantenga las condiciones de frontera de tipo Robin.
- Cambie la función de fuerza a $f(x) = x + 1$.
- Ejecute el programa con todos los cambios que ha realizado.
- Verifique que la gráfica es apropiada (títulos, valores de los ejes, etc.).

<u>Solución</u> Se está resolviendo la ecuación de Poisson con el problema de valor de frontera Robin

$$\begin{cases} \dfrac{\mathrm{d}^2u}{\mathrm{d}x^2} = x+1, \text{ para } x \in [0,1]. \\ 0 = u(0) - \left(\dfrac{\mathrm{d}u}{\mathrm{d}x}\right)_{x=0}. \\ 2e = u(1) + \left(\dfrac{\mathrm{d}u}{\mathrm{d}x}\right)_{x=1}. \end{cases} \tag{3}$$

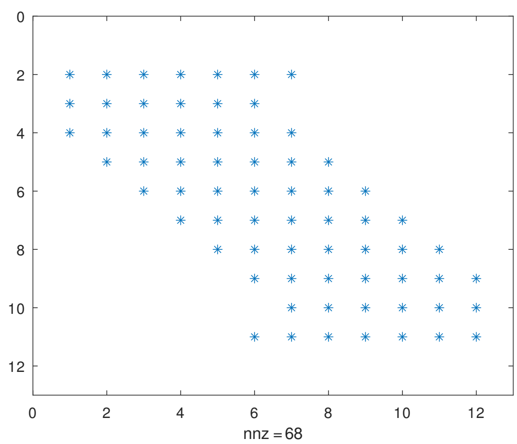
Integramos dos veces y obtenemos la solución general.

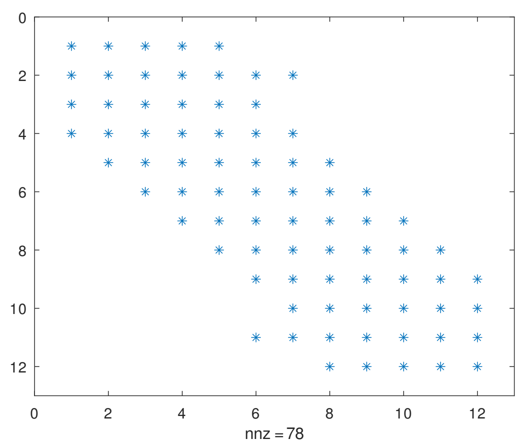
$$\begin{aligned} \iint \frac{\mathrm{d}^2u}{\mathrm{d}x^2}\,\mathrm{d}x\,\mathrm{d}x &= \iint (x+1)\,\mathrm{d}x\,\mathrm{d}x. \\ \int \frac{\mathrm{d}u}{\mathrm{d}x}\,\mathrm{d}x &= \int \left(\frac{x^2}{2} + x + C_1\right)\mathrm{d}x. \\ u(x) &= \frac{x^3}{6} + \frac{x^2}{2} + C_1 x + C_2. \end{aligned}$$

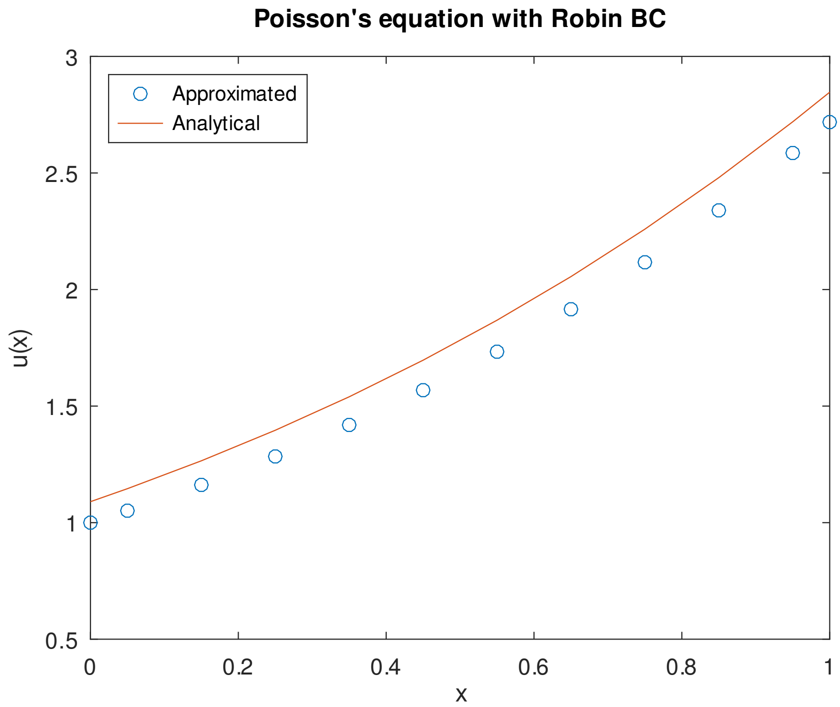
Ahora, apliquemos las condiciones de frontera Robin.

$$\begin{cases} 0 = u(0) - \left(\dfrac{\mathrm{d}u}{\mathrm{d}x}\right)_{x=0} = \dfrac{0^3}{6} + \dfrac{0^2}{2} + C_1(0) + C_2 - \dfrac{0^2}{2} - 0 - C_1 = C_2 - C_1. \\ 2e = u(1) + \left(\dfrac{\mathrm{d}u}{\mathrm{d}x}\right)_{x=1} = \dfrac{1^3}{6} + \dfrac{1^2}{2} + C_1(1) + C_2 + \dfrac{1^2}{2} + 1 + C_1 = \dfrac{13}{6} + 2C_1 + C_2. \end{cases} \tag{4}$$

El sistema (4) tiene como solución $C_1 = C_2 = \frac{2e}{3} - \frac{13}{18}$. $\therefore$ la solución de (3) es $u(x) = \frac{x^2}{2} + \frac{x^3}{6} + (x+1)\left(\frac{2e}{3} - \frac{13}{18}\right)$.







Problema 3

Copie el programa 1 y modifique lo siguiente:

1) ¿Cuál es el dominio de la EDP (la región donde la PDE se cumple)?

   Solución El dominio de la EDP es $\Omega = [0,7] \times [0,8] \times [0,9] \subset \mathbb{R}^3$.

2) ¿Cuál es la ecuación que resuelve este ejemplo?

   Solución La ecuación que resuelve es la ecuación de Laplace.

3) ¿Cuál es la función de fuerza o el lado derecho?

   Solución Es cero.

4) ¿Qué tipo de condiciones de contorno?

   Solución Se tiene condiciones Dirichlet.

5) Cambie la condición de contorno a 100 en la cara frontal y posterior del cubo.

   Solución

6) Ejecute el ejemplo con todos los cambios que ha realizado.

   Solución

Problema 4

En este ejemplo no modificarás el código fuente

1) ¿Cuál es el dominio de la EDP (la región donde se cumple la EDP)?

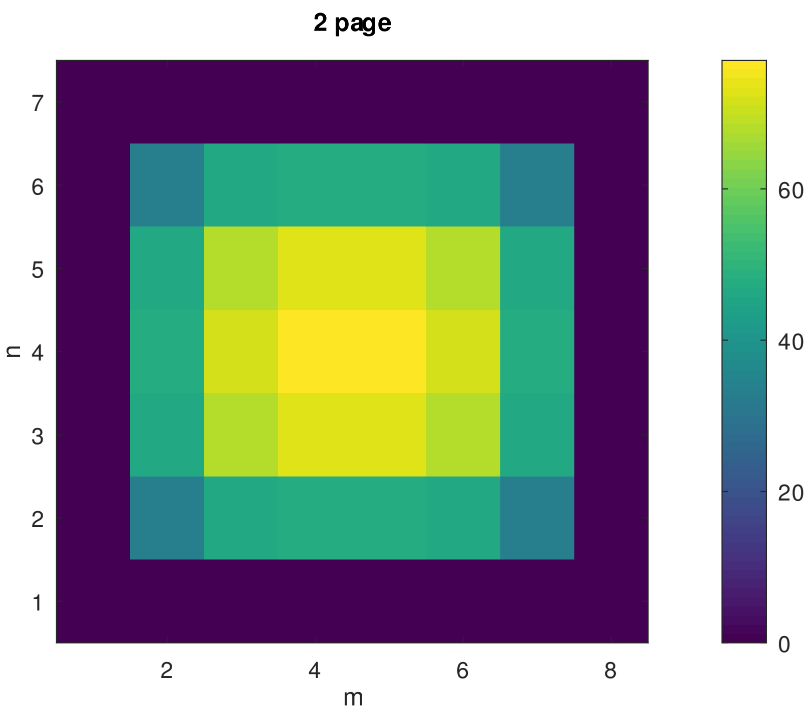
2) ¿Cuál es la ecuación que resuelve este ejemplo?

3) ¿Cuál es la condición inicial?

4) ¿Cuáles son las condiciones de contorno?

```
#!/usr/bin/env -S octave -qf
% 3D Staggering example using a 3D Mimetic laplacian

clc
close all

addpath('/usr/share/mole/matlab/')

k = 2; % Order of accuracy
m = 5; % -> 7
n = 6; % -> 8
o = 7; % -> 9

L = lap3D(k, m, 1, n, 1, o, 1); % 3D Mimetic laplacian operator
L = L + robinBC3D(k, m, 1, n, 1, o, 1, 1, 0); % Dirichlet BC

RHS = zeros(m+2, n+2, o+2);

RHS(:, :, 1) = 100; % Known value at the cube's front face
RHS(:, :, end) = 100; % Known value at the cube's back face

RHS = reshape(RHS, (m+2)*(n+2)*(o+2), 1);

SOL = L\RHS;

SOL = reshape(SOL, m+2, n+2, o+2);

p = 2; % Page to be displayed 1 ... o+2

page = SOL(:, :, p);
figure('visible', 'off');

imagesc(page)
title([num2str(p) ' page'])
xlabel('m')
ylabel('n')
set(gca, 'YDir', 'Normal')
colorbar
```

Programa 1: Programa `elliptic3D.m`



5) ¿Cuál es la función de fuerza o el lado derecho?

6) Explora y explica qué es el criterio de estabilidad de Neumann.

7) ¿Cómo se discretiza la derivada parcial con respecto al tiempo en el esquema explícito de este código?

8) ¿Cómo se discretiza la derivada parcial con respecto al tiempo en el esquema implícito de este código?

9) Ejecuta el ejemplo con el esquema explícito y almacena su solución numérica.

10) Ejecuta el ejemplo con el esquema implícito y almacena su solución numérica.

11) Haz un gráfico de ambas soluciones, explícita en azul e implícita en rojo.

□

Referencias

[1] Daniele Andreucci et al. «Some Numerical Results on Chemotactic Phenomena in Stem Cell Therapy for Cardiac Regeneration». En: *Mathematics* 12.13 (2024). ISSN: 2227-7390. DOI: 10.3390/math12131937. URL: https://www.mdpi.com/2227-7390/12/13/1937.

[2] Daniel Arrigo. *An Introduction to Partial Differential Equations*. Cham: Springer International Publishing, 2023. ISBN: 978-3-031-22087-6.

[3] Jared Brzenski. «Building an Ocean Model Using Mimetic Operators». Doctoral Dissertation. California: UC Irvine, 2024. URL: https://escholarship.org/uc/item/5bt2c69f.

[4] Jared Brzenski y Jose E. Castillo. «Solving Navier-Stokes with mimetic operators». En: *Computers & Fluids* 254 (2023), pág. 105817. ISSN: 0045-7930. DOI: 10.1016/j.compfluid.2023.105817.

[5] Rustum Choksi. *Partial Differential Equations: A First Course*. American Mathematical Society, abr. de 2022. ISBN: 978-1-4704-6491-2.

[6] Johnny Corbino y Jose E. Castillo. «High-order mimetic finite-difference operators satisfying the extended Gauss divergence theorem». En: *Journal of Computational and Applied Mathematics* 364 (2020), pág. 112326. ISSN: 0377-0427. DOI: 10.1016/j.cam.2019.06.042.

[7] Johnny Corbino, Miguel A. Dumett y Jose E. Castillo. «MOLE: Mimetic Operators Library Enhanced». En: *Journal of Open Source Software* 9.99 (2024), pág. 6288. DOI: 10.21105/joss.06288. URL: https://doi.org/10.21105/joss.06288.

[8] Yessica Judith Gonzales Aredo. «Convergencia del método mimético para la ecuación de difusión no estática». Tesis de mtría. Trujillo: Universidad Nacional de Trujillo, 2023. URL: https://dspace.unitru.edu.pe/items/1742d2ae-7727-4c4b-9bf9-0cadbbb75d3c.

[9] Konstantin Lipnikov, Gianmarco Manzini y Mikhail Shashkov. «Mimetic finite difference method». En: *Journal of Computational Physics* 257 (2014). Physics-compatible numerical methods, págs. 1163-1227. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2013.07.031.

[10] Bertha K. Rodriguez-Chavez y Yessica E. Zarate-Pedrera. «Spectral Differentiation and Mimetic Methods for Solving the Scalar Burger's Equation». En: *Selecciones Matemáticas* 11.02 (dic. de 2024), págs. 259-270. DOI: 10.17268/sel.mat.2024.02.05. URL: https://revistas.unitru.edu.pe/index.php/SSMM/article/view/6157.

[11] Franco Rubio López y Mardo Gonzales Herrera. «Algorítmo para la Ecuación de Difusión en Estado Estacionario 2D usando el Método Mimético en Diferencias Finitas.» En: *Selecciones Matemáticas* 1.01 (abr. de 2015). DOI: 10.17268/sel.mat.2014.01.04. URL: https://revistas.unitru.edu.pe/index.php/SSMM/article/view/826.

[12] Conrad Sanderson y Ryan Curtin. *Armadillo: An Efficient Framework for Numerical Linear Algebra*. 2025. DOI: 10.48550/arXiv.2502.03000. arXiv: 2502.03000 [cs.MS].

[13] G. Sosa Jones, J. Arteaga y O. Jiménez. «A study of mimetic and finite difference methods for the static diffusion equation». En: *Computers & Mathematics with Applications* 76.3 (2018), págs. 633-648. ISSN: 0898-1221. DOI: 10.1016/j.camwa.2018.05.004.

[14] Angel Boada Velazco. «High order mimetic finite differences on non-trivial problems». Doctoral Dissertation. San Diego: San Diego State University, 2021. URL: https://digitalcollections.sdsu.edu/do/e67f17cb-b906-4847-a574-a8781e581024.

A Instalación

La única dependencia dura es `armadillo`. Los pasos están descritos en el Programa 2.

```bash
#!/bin/bash

# [1 / 3] Update the system
sudo pacman --noconfirm -Syu
# [1.5 / 3] Optionally install Intel MKL as a replacement for LAPACK linear algebra library
sudo pacman --noconfirm -S intel-oneapi-mkl # or one of them depending on your needs: intel-oneapi-basekit, intel-oneapi-hpckit

# [2 / 3] Install armadillo
git clone https://aur.archlinux.org/armadillo.git
pushd armadillo
makepkg -s --noconfirm
popd

# [3 / 3] Install MOLE C++/Octave
git clone https://aur.archlinux.org/libmole.git
pushd libmole
makepkg -s --noconfirm
popd
```

Programa 2: Instalación de MOLE vía `installer.sh` en Arch Linux.

B Programas utilizados

B.1 Programa 3

```matlab
#!/usr/bin/env -S octave -qf
% Solves the 1D Poisson Equation with Robin Boundary Conditions and a
% non-constant forcing right hand side using the Mimetic Method with
% MOLE in Octave / MATLAB.
%
%     Δu = f
%
% u : Vertical Displacement of a membrane
% f : Forcing right hand side
% Δ : Laplace Operator

addpath('/usr/share/mole/matlab/')

west = 0; % Domain's limits
east = 1;

k = 6; % Operator's order of accuracy
m = 2 * k + 1; % Minimum number of cells to attain the desired accuracy
dx = (east - west) / m; % Step length

L = lap(k, m, dx); % 1D Mimetic laplacian operator
L_before_name = sprintf("L_before.h5");
save("-hdf5", L_before_name, "L")
figure('visible', 'off');
spy(L);
saveas(gcf, "elliptic1Dsparsebefore.pdf", 'pdfcrop')

% Impose Robin BC on laplacian operator
a = 1;
b = 1;
L = L + robinBC(k, m, dx, a, b);
L_after_name = sprintf("L_after.h5");
save("-hdf5", L_after_name, "L")
spy(L);
saveas(gcf, "elliptic1Dsparseafter.pdf", 'pdfcrop')

% 1D Staggered grid
grid = [west west + dx / 2:dx:east - dx / 2 east];
save("-hdf5", "grid")

% RHS
U = exp(grid)';
U(1) = 0; % West BC
U(end) = 2 * exp(1); % East BC
tic
U = L \ U; % Solve a linear system of equations
toc
save("-hdf5", "U")
% Plot result
plot(grid, U, 'o')
hold on
plot(grid, exp(grid))
legend('Approximated', 'Analytical', 'Location', 'NorthWest')
```

Programa 3: Programa `elliptic1D.m`

```cpp
/**
 * This example uses MOLE to solve a 1D BVP
 */

#define ARMA_USE_HDF5
#include <mole/laplacian.h>
#include <mole/operators.h>
#include <mole/robinbc.h>

int main()
{
  int k = 6;             // Operators' order of accuracy
  Real a = 0;            // Left boundary
  Real b = 1;            // Right boundary
  int m = 2 * k + 1;     // Number of cells
  Real dx = (b - a) / m; // Step size

  // Get mimetic operators
  Laplacian L(k, m, dx);
  Real d = 1; // Dirichlet coefficient
  Real n = 1; // Neumann coefficient
  RobinBC BC(k, m, dx, d, n);
  L = L + BC;

  // 1D Staggered grid
  arma::vec grid(m + 2);
  grid(0) = a;
  grid(1) = a + dx / 2.0;
  int i;
  for (i = 2; i <= m; i++) {
    grid(i) = grid(i - 1) + dx;
  }
  grid(i) = b;

  // Build RHS for system of equations
  arma::vec rhs(m + 2);
  rhs = arma::exp(grid); // rhs(0) = 1
  rhs(0) = 0;
  rhs(m + 1) = 2 * std::exp(1); // rhs(1) = 2e

  arma::wall_clock timer;
  // Solve the system of linear equations
#ifdef EIGEN
  // Use Eigen only if SuperLU (faster) is not available
  std::cout << "With Eigen" << std::endl;
  timer.tic();
  vec sol = Utils::spsolve_eigen(L, rhs);
  Real elapsed_time = timer.toc();
  std::cout << "Elapased time with Eigen: " << elapsed_time << endl;
#elif LAPACK
  std::cout << "With LAPACK" << std::endl;
  timer.tic();
  arma::vec sol = arma::spsolve(L, rhs, "lapack"); // Will use LAPACK
  Real elapsed_time = timer.toc();
  std::cout << "Elapased time with LAPACK: " << elapsed_time << endl;
#else
  std::cout << "With SuperLU" << std::endl;
#define ARMA_USE_SUPERLU 1
  arma::superlu_opts opts;
  opts.allow_ugly = true;
  opts.equilibrate = true;
  timer.tic();
  arma::vec sol = arma::spsolve(L, rhs, "superlu", opts); // Will use SuperLU
  Real elapsed_time = timer.toc();
  std::cout << "Elapased time with SuperLU: " << elapsed_time << endl;
#endif

  // Save grid
  grid.save(arma::hdf5_name("elliptic1d_grid.h5", "elliptic1d_grid"));
  // Save the solution
  sol.save(arma::hdf5_name("elliptic1d_solution.h5", "elliptic1d_solution"));
  // Print out the solution
  std::cout << sol;

  return 0;
}
```

Programa 4: Programa `elliptic1D.cpp`

```cmake
cmake_minimum_required(VERSION 3.28)
project(Tutorial VERSION 1.0)

set(CMAKE_CXX_STANDARD 20)
set(CMAKE_CXX_STANDARD_REQUIRED True)

find_package(Armadillo REQUIRED)
include_directories(${ARMADILLO_INCLUDE_DIRS})
add_executable(elliptic1D elliptic1D.cpp)
# target_link_libraries(elliptic1D PUBLIC ${ARMADILLO_LIBRARIES} mole_C++ Eigen3::Eigen hdf5)
  POST_BUILD
  TARGET elliptic2D
)

add_custom_command(
```

Programa 5: Programa `CMakeLists.txt`

```python
#!/usr/bin/env python

import sys

import matplotlib.pyplot as plt
import numpy as np
from h5py import File

try:
    with File(name="../examples/cpp/build/elliptic1d_grid.h5", mode="r") as f:
        grid = np.array(f["elliptic1d_grid"][:][0])
    with File(name="../examples/cpp/build/elliptic1d_solution.h5", mode="r") as f:
        solution = np.array(f["elliptic1d_solution"][:][0])
    with File(name="../examples/octave/L_after.h5", mode="r") as f:
        L_after = np.array(f["L"]["value"]["data"][:])
    with File(name="../examples/octave/L_before.h5", mode="r") as f:
        L_before = np.array(f["L"])
except OSError as err:
    print(f"OS error: {err}")
    sys.exit(0)

print(L_after)
print(L_before)

# exact = np.exp(grid)
# fig, ax = plt.subplots(layout="constrained")
# ax.set_xlabel(xlabel=r"$x$")
# ax.set_ylabel(ylabel=r"$u\left(x\right)$")
# ax.set_xlim(left=grid[0] - 5e-3, right=grid[-1] + 5e-3)
# ax.set_ylim(bottom=exact.min() - 5e-3, top=exact.max() + 5e-3)
# ax.set_xticks(ticks=np.linspace(start=grid[0], stop=grid[-1], num=6))
# ax.set_yticks(
#     ticks=np.linspace(
#         start=solution.min(), stop=np.ceil(solution[-1] * 10) / 10, num=10
#     )
# )
# ax.set_xticklabels(labels=ax.get_xticklabels(which="major"), fontdict=dict(size=8))
# ax.set_yticklabels(labels=ax.get_yticklabels(which="major"), fontdict=dict(size=8))
# ax.scatter(x=grid, y=solution, c="red", s=2, label="Solución mimética")
# ax.plot(
#     grid,
#     exact,
#     alpha=0.8,
#     label="Solución exacta",
#     linestyle="dashed",
#     linewidth=0.8,
# )
# ax.legend(loc="best")
# ax.set_title(
#     label=r"Solución exacta para $\diff[2]{{u}}{{x}}=e^{{x}}$ con condiciones de frontera Robin",
#     loc="center",
#     fontsize=12,
# )
# ax.grid(c="gray", linewidth=0.1, linestyle="dashed")
# ax.set_aspect("equal", adjustable="box")
# ax.spines["bottom"].set_color("none")
# ax.spines["top"].set_color("none")
# ax.spines["left"].set_color("none")
# ax.spines["right"].set_color("none")
# plt.savefig("elliptic1D.pdf", transparent=True, bbox_inches="tight")
# plt.clf()

# error = np.abs(exact - solution)
# fig, ax = plt.subplots(layout="constrained")
# ax.scatter(x=grid, y=error, c="red", s=2)
# ax.set_xticks(np.linspace(start=grid[0], stop=grid[-1], num=5))
# ax.set_xlabel(xlabel=r"$x$")
# ax.set_ylabel(ylabel=r"$\left|u\left(x_{j-\frac12}\right)-u_{j-\frac12}\right|$")
# ax.set_xlim(left=grid[0] - 5e-3, right=grid[-1] + 5e-3)
# ax.set_ylim(
#     bottom=np.floor(error.min() * 1e10) * 1e-10, top=np.ceil(error.max() * 1e10) * 1e-10
# )
# ax.set_xticks(ticks=np.linspace(start=grid[0], stop=grid[-1], num=6))
# ax.set_yticks(
#     ticks=np.linspace(
#         start=np.floor(error.min() * 1e10) * 1e-10,
#         stop=np.ceil(error.max() * 1e10) * 1e-10,
#         num=4,
#     )
# )
# ax.set_title(
#     label="Error puntual",
#     loc="center",
#     fontsize=12,
# )
# ax.grid(c="gray", linewidth=0.1, linestyle="dashed")
# ax.spines["bottom"].set_color("none")
# ax.spines["top"].set_color("none")
# ax.spines["left"].set_color("none")
# ax.spines["right"].set_color("none")
# plt.savefig("elliptic1Derror.pdf", transparent=True, bbox_inches="tight")
# plt.clf()

# for m in range(10, 1000, 100):
#     try:
#         with File(name=f"sandbox/grid{m}.h5", mode="r") as f:
#             grid = np.array(f["elliptic1d_grid"][:][0])
#         with File(name="../examples/cpp/build/elliptic1d_solution.h5", mode="r") as f:
#             solution = np.array(f["elliptic1d_solution"][:][0])
#     except OSError as err:
#         print(f"OS error: {err}")
#         sys.exit(0)
#     error = np.abs(exact - solution)

# Δx = np.logspace(start=-15, stop=0, num=16, base=2)
# fig, ax = plt.subplots(layout="constrained")
# ax.loglog(
#     Δx,
#     Δx,
#     "--",
#     mfc="none",
#     label=r"$\mathcal{O}\left(\Delta x\right)$",
#     linewidth=0.8,
# )
# ax.loglog(
#     Δx,
#     np.power(Δx, 2),
#     "--",
#     mfc="none",
#     label=r"$\mathcal{O}\left(\Delta x^{2}\right)$",
#     linewidth=0.8,
# )
# ax.set_xlabel(r"Tamaño malla equivalente $\Delta x$")
# ax.set_ylabel("Error")
# ax.set_xlim(left=Δx[0], right=Δx[-1])
# ax.set_ylim(bottom=1e-5, top=1e-1)
# ax.grid(c="gray", linewidth=0.1, linestyle="dashed")
```

```
$ octave --help
GNU Octave, version 9.3.0
Copyright (C) 1993-2024 The Octave Project Developers.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.

Octave was configured for "x86_64-pc-linux-gnu".

Usage: octave [options] [FILE]

Options:

  --built-in-docstrings-file FILE Use docs for built-ins from FILE.
  --debug, -d             Enter parser debugging mode.
  --doc-cache-file FILE   Use doc cache file FILE.
  --echo-commands, -x     Echo commands as they are executed.
  --eval CODE             Evaluate CODE.  Exit when done unless --persist.
  --exec-path PATH        Set path for executing subprograms.
  --experimental-terminal-widget
                          Use new experimental terminal widget in the GUI.
  --gui                   Start the graphical user interface.
  --help, -h,             Print short help message and exit.
  --image-path PATH       Add PATH to head of image search path.
  --info-file FILE        Use top-level info file FILE.
  --info-program PROGRAM  Use PROGRAM for reading info files.
  --interactive, -i       Force interactive behavior.
  --line-editing          Force readline use for command-line editing.
  --no-gui                Disable the graphical user interface.
  --no-history, -H        Don't save commands to the history list
  --no-init-file          Don't read the ~/.octaverc or .octaverc files.
  --no-init-path          Don't initialize function search path.
  --no-line-editing       Don't use readline for command-line editing.
  --no-site-file          Don't read the site-wide octaverc file.
  --no-window-system, -W  Disable window system, including graphics.
  --norc, -f              Don't read any initialization files.
  --path PATH, -p PATH    Add PATH to head of function search path.
  --persist               Go interactive after --eval or reading from FILE.
  --server                Enter server mode at startup.
  --silent, --quiet, -q   Don't print message at startup.
  --texi-macros-file FILE Use Texinfo macros in FILE for makeinfo command.
  --traditional           Set variables for closer MATLAB compatibility.
  --verbose, -V           Enable verbose output in some cases.
  --version, -v           Print version number and exit.

  FILE                    Execute commands from FILE.  Exit when done
                          unless --persist is also specified.

Home page:            https://octave.org

Improve Octave:       https://octave.org/get-involved

Support resources:    https://octave.org/support
```

Programa 7: `octave-help.txt` muestra la lista de opciones de Octave por la línea de comandos.

```
$ tree /usr/share/mole/matlab/
/usr/share/mole/matlab/
├── amean.m
├── boundaryIdx2D.m
├── curl2D.m
├── DI2.m
├── DI3.m
├── div2DCurv.m
├── div2D.m
├── div2DNonUniform.m
├── div3DCurv.m
├── div3D.m
├── div3DNonUniform.m
├── div.m
├── divNonUniform.m
├── GI13.m
├── GI1.m
├── GI2.m
├── grad2DCurv.m
├── grad2D.m
├── grad2DNonUniform.m
├── grad3DCurv.m
├── grad3D.m
├── grad3DNonUniform.m
├── grad.m
├── gradNonUniform.m
├── gridGen.m
├── grids
│   ├── chevron
│   │   ├── bottom.m
│   │   ├── left.m
│   │   ├── right.m
│   │   └── top.m
│   ├── horseshoe
│   │   ├── bottom.m
│   │   ├── left.m
│   │   ├── right.m
│   │   └── top.m
│   └── swan
│       ├── bottom.m
│       ├── left.m
│       ├── right.m
│       └── top.m
├── hmean.m
├── interpol2D.m
├── interpol3D.m
├── interpolCentersToFacesD1D.m
├── interpolCentersToFacesD2D.m
├── interpolCentersToFacesD3D.m
├── interpolCentersToNodes1D.m
├── interpolCentersToNodes2D.m
├── interpolCentersToNodes3D.m
├── interpolD2D.m
├── interpolD3D.m
├── interpolD.m
├── interpolFacesToCentersG1D.m
├── interpolFacesToCentersG2D.m
├── interpolFacesToCentersG3D.m
├── interpol.m
├── interpolNodesToCenters1D.m
├── interpolNodesToCenters2D.m
├── interpolNodesToCenters3D.m
├── jacobian2D.m
├── jacobian3D.m
├── lap2D.m
├── lap3D.m
├── lap.m
├── mimeticB.m
├── mixedBC2D.m
├── mixedBC3D.m
├── mixedBC.m
├── neumann2DCurv.m
├── neumann3DCurv.m
├── nodal2DCurv.m
├── nodal2D.m
├── nodal3DCurv.m
├── nodal3D.m
├── nodal.m
├── rk4.m
├── robinBC2D.m
├── robinBC3D.m
├── robinBC.m
├── sidedNodal.m
├── tensorGrad2D.m
├── tfi.m
├── ttm.m
├── weightsP2D.m
├── weightsP.m
├── weightsQ2D.m
└── weightsQ.m

5 directories, 84 files
```

Programa 8: `moledirectoriesoctave.txt` muestran la estructura de árbol de directorios de las funciones Octave / MATLAB de la biblioteca MOLE.

```
$ tree /usr/include/mole
/usr/include/mole
├── divergence.h
├── gradient.h
├── interpol.h
├── laplacian.h
├── mixedbc.h
├── mole.h
├── operators.h
├── robinbc.h
└── utils.h

1 directory, 9 files

$ tree /usr/lib/libmole.so

/usr/lib/libmole.so

0 directories, 1 file
```

Programa 9: `moledirectoriescpp.txt` muestran la estructura de árbol de directorios de las cabeceras de C++ de la biblioteca MOLE.

```
#!/usr/bin/env -S octave -qf
% Solves the 1D Poisson Equation with Robin Boundary Conditions and a
% non-constant forcing right hand side using the Mimetic Method with
% MOLE in Octave / MATLAB.
%
%     Δu = f
%
% u : Vertical Displacement of a membrane
% f : Forcing right hand side
% Δ : Laplace Operator

addpath('/usr/share/mole/matlab/')

west = 0; % Domain's limits
east = 1;

k = 4; % Operator's order of accuracy
m = 10; % Minimum number of cells to attain the desired accuracy
dx = (east - west) / m; % Step length

L = lap(k, m, dx); % 1D Mimetic laplacian operator
figure('visible', 'off');
spy(L);
saveas(gcf, "elliptic1DAznaransparsebefore.pdf", 'pdfcrop')

% Impose Robin BC on laplacian operator
a = 1;
b = 1;
L = L + robinBC(k, m, dx, a, b);
spy(L);
saveas(gcf, "elliptic1DAznaransparseafter.pdf", 'pdfcrop')

% 1D Staggered grid
grid = [west west + dx / 2:dx:east - dx / 2 east];
save("-hdf5", "grid")

% RHS
U = exp(grid)';
U(1) = 0; % West BC
U(end) = 2 * exp(1); % East BC
tic
U = L \ U; % Solve a linear system of equations
toc
save("-hdf5", "U")
% Plot result
plot(grid, U, 'o')
hold on
plot(grid, grid.^2/2 + grid.^3/6 + (grid + 1) * (2/3 * exp(1) - 13/18))
legend('Approximated', 'Analytical', 'Location', 'NorthWest')
title('Poisson''s equation with Robin BC')
xlabel('x')
ylabel('u(x)')
saveas(gcf, "elliptic1DAznaran.pdf", 'pdfcrop')
```

Programa 10: Programa `elliptic1DAznaran.m`