MOLE: Mimetic Operators Library Enhanced

The Open-Source Library for Solving Partial Differential Equations using Mimetic Methods

Johnny Corbino and Jose Castillo

October 10, 2023

Publication Number: CSRCR2023-07

Computational Science & Engineering Faculty and Students Research Articles

Database Powered by the Computational Science Research Center Computing Group & Visualization Lab

COMPUTATIONAL SCIENCE & ENGINEERING

SAN DIEGO STATE UNIVERSITY

Computational Science Research Center
College of Sciences
5500 Campanile Drive
San Diego, CA 92182-1245
(619) 594-3430

MOLE: Mimetic Operators Library Enhanced

The Open-Source Library for Solving Partial Differential Equations Using Mimetic Methods

Johnny Corbino* and Jose Castillo*

October 10, 2023

Abstract

MOLE is a high-quality (C++ & MATLAB/Octave) library that implements high-order mimetic methods to solve 1D, 2D, and 3D partial differential equations. It provides discrete analogs of the most common vector calculus operators: Gradient, Divergence, Curl, and Laplacian. These operators (matrices) act on staggered grids (uniform, non-uniform, and curvilinear) and satisfy local and global conservation laws. Mathematics are based on the work of [Corbino and Castillo 2020]. However, the user may find useful previous publications, such as [Castillo and Grone 2003], in which similar operators were derived using a matrix analysis approach.

*Computational Science Research Center, 5500 Campanile Drive, San Diego, CA 92182

1 Introduction

Physical phenomena are typically modelled as a set of differential equations subject to conservation laws. Numerical methods used to solve these equations are of vital importance in the paradigm of computational science. In this work, we talk about MOLE, an open-source library that implements mimetic discretization methods (MDM) to intuitively solve partial differential equations (PDE).

Mimetic operators are derived by constructing discrete analogs of the continuum differential operators $\nabla, \nabla\cdot, \nabla\times$, and ∇^2. Since most continuum models are described in terms of these operators, the MDM approach has recently gained a lot of space in the context of numerical PDEs.

Qualities of a mimetic operator:

- It is a discrete analog of the continuum operator
- It satisfies essential identities from vector calculus
- It satisfies global and local conservation laws
- It provides uniform order of accuracy
- It is easy to use (and reusable)

In 2003, Castillo and Grone came up with a matrix analysis approach to construct high-order approximations of divergence and gradient operators [1]. However, in their approach, the 4^{th}-order operators have three free-parameters. The mimetic operators implemented in MOLE are based on the work of [Corbino and Castillo 2020] which are a substantial improvement over the operators introduced in [1]. These new operators have no free-parameters, have optimal bandwidth, are more accurate, and in the worst case they deliver the same accuracy as the ones from 2003.

There are many applications of MDM in solving continuum problems, including in the geosciences (porous media) [2], [3]; fluid dynamics (Navier-Stokes) [4] [5]; image processing [6]; general relativity [7]; and electromagnetism [8].

2 On the Mathematics

MDM not only provide uniform order of accuracy (all the way to the boundary), but they also satisfy fundamental identities from vector calculus,

✓ Gradient of a constant	$Gf = 0$
✓ Free stream preservation	$Dv = 0$
✓ Curl of the gradient	$CGf = 0$
✓ Divergence of the curl	$DCv = 0$
✓ Divergence of the gradient	$DGf = Lf$

In addition, the discrete version of Gauss' extended divergence theorem is also satisfied:

$$\langle Dv, f \rangle_Q + \langle v, Gf \rangle_P = \langle Bv, f \rangle \qquad (1)$$

the deduction of (1) can be found on https://www.crcpress.com/Mimetic-Discretization-Methods/Castillo-Miranda/p/book/9781466513433

When using MDM we are not discretizing the equations (as it is done with standard finite-difference methods (FDM)), but instead we construct a discrete analog to the differential operator,

$$\frac{\partial^2 f}{\partial x^2} \quad \text{vs} \quad \frac{\partial^2 f}{\partial x^2} = \frac{\partial^2}{\partial x^2} f$$

$$\downarrow \qquad\qquad \downarrow$$

$$\frac{f_{j+1} - 2f_j + f_{j-1}}{\Delta x^2} \qquad \mathbf{DG}f$$

3 Installing the Library

Once you download the library from:

https://github.com/csrc-sdsu/mole

you need to add the path to it on your MATLAB/Octave script,

```
addpath('path_to_mole')
```

after this, you will be able to call any of the 29 functions provided by MOLE to construct mimetic operators.

4 Getting the Operators

Functions syntax is consistent throughout the library. In addition, you can see a brief explanation of each function by typing the following command:

```
help 'function_name'
```

for instance,

```
>> help div
  Returns a m+2 by m+1 one-dimensional mimetic divergence operator

  Parameters:
                 k : Order of accuracy
                 m : Number of cells
                dx : Step size
```

or,

```
>> help grad2D
  Returns a two-dimensional mimetic gradient operator

  Parameters:
                  k : Order of accuracy
                  m : Number of cells along x-axis
                 dx : Step size along x-axis
                  n : Number of cells along y-axis
                 dy : Step size along y-axis
```

All functions in MOLE return a sparse matrix representation of the requested operator.

To obtain a 2nd-order 1D mimetic Laplacian we can type:

```
div(k, m, dx)*grad(k, m, dx);
```

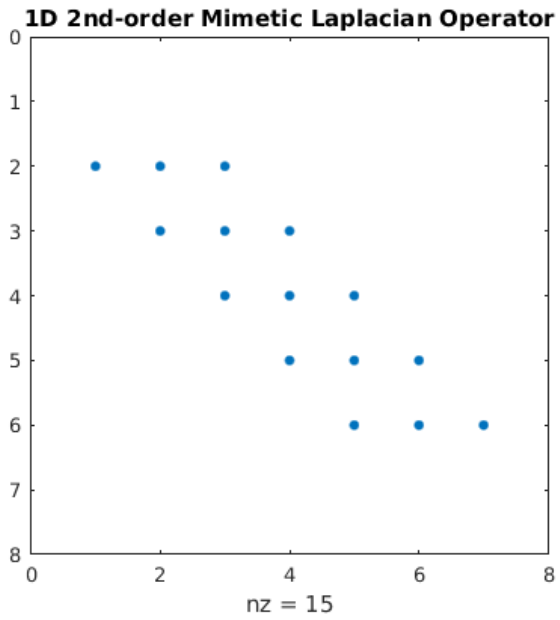
or

```
lap(k, m, dx);
```

where '`k`' is the desired order of accuracy (in this case 2), '`m`' is the number of cells, and '`dx`' is the step size.

```
spy(lap(2, 5, 1));
```

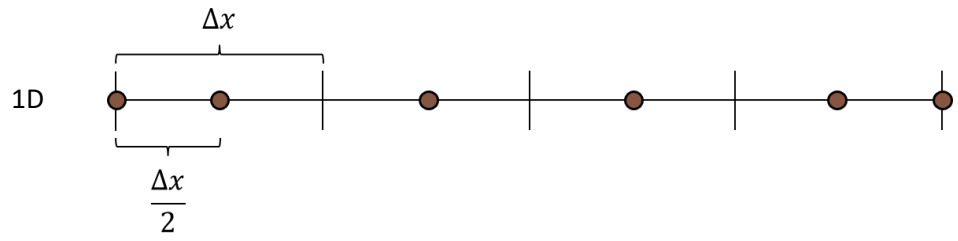
yields:



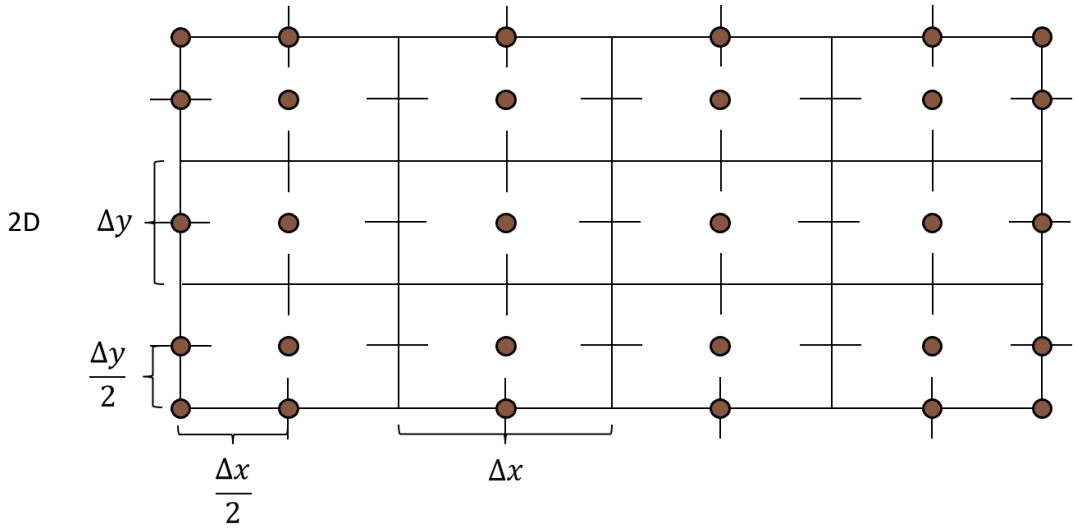
a 7x7 sparse matrix with 15 nonzero elements.

5 Staggered Grids

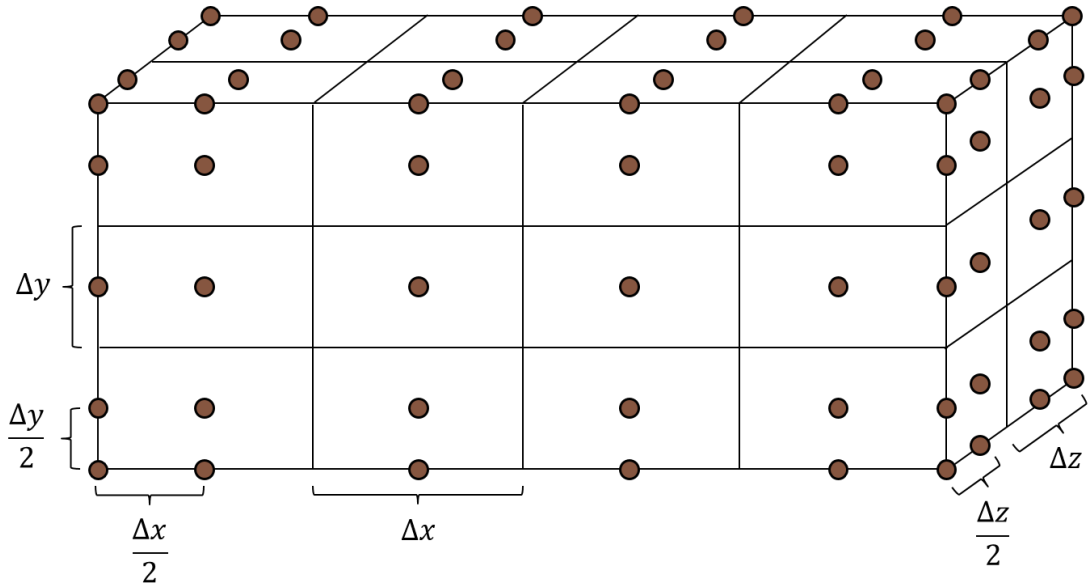
Mimetic operators are defined over `staggered grids`.



● Scalar quantities. | Vector quantities.



and finally 3D,



There are several ways to generate uniform structured staggered grids (in MATLAB). What is important is to remember that we need to store the coordinates for two different quantities.

suppose,

```
west = 0; east = 1; m = 10; dx = (east-west)/m;
xgridSca = [west west+dx/2 : dx : east-dx/2 east];
```

This grid holds the coordinates of all scalar quantities (cell centers + boundaries). Vector quantities coordinates are just:

```
xgridVec = west : dx : east;
```

The previous commands create a couple of one-dimensional arrays with the coordinates of each field (scalar and vectorial). Now we need a couple of arrays to hold the actual values of these fields.

NOTE: Depending on the problem, you may need or not to explicitly create such arrays. It is often a good practice to preallocate the memory for better performance:

```
scalarField = zeros(numel(xgridSca), 1);
```

and,

```
vectorField = zeros(numel(xgridVec), 1);
```

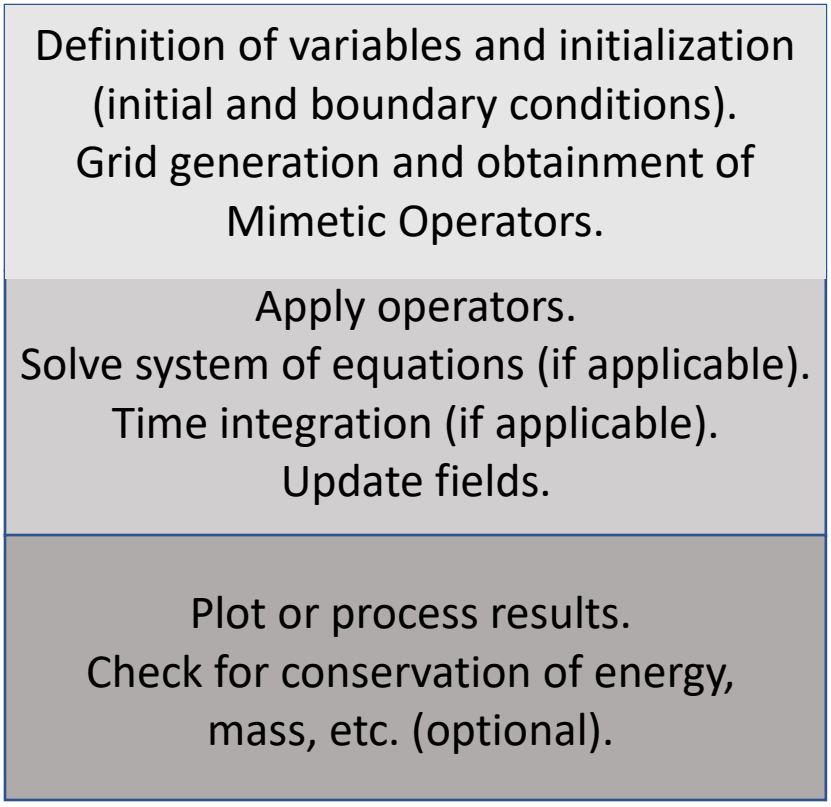
6 Using the Operators

Inside the "`examples_MATLAB`" folder you will find several MATLAB scripts that use MOLE to solve well known partial differential equations.

Our selection includes steady-state and time-dependent problems:

- Burger's eq.
- Richards' eq. (highly nonlinear, mixed form)
- Wave eq. (with symplectic schemes)
- Heat eq. (explicitly and implicitly)
- Etc.

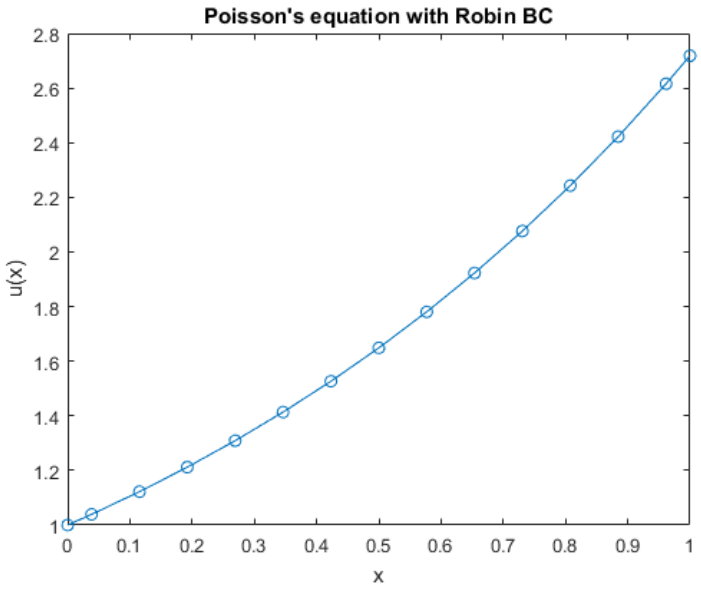
Each script in the "`examples_MATLAB`" folder is adequately commented. You may notice that all programs have the same taxonomy:



7 Code Snippet

```
% Solves the 1D Poisson's equation with Robin boundary
conditions

clc; close all

addpath('../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 size


% 1D Staggered grid

xgrid = [west west+dx/2 : dx : east-dx/2 east];


% RHS

RHS = exp(xgrid)';

RHS(1) = 0;            % West BC

RHS(end) = 2*exp(1);  % East BC
```

```
% Get 1D 6th-order mimetic laplacian operator from MOLE
L = lap(k, m, dx);

% Impose Robin BC on laplacian operator
a = 1;  % The ‘a’ coefficient in Robin BC
b = 1;  % The ‘b’ coefficient in Robin BC
L = L + robinBC(k, m, dx, a, b);   % robinBC also provided by MOLE

U = L\RHS;  % Solve a linear system of equations
plot(xgrid, U, 'o-')
title('Poisson''s equation with Robin BC')
xlabel('x')
ylabel('u(x)')
% End of elliptic1D.m
```



Result

As you can see, the previous script followed the structure outlined on page 16. It is easy to solve 1D, 2D and 3D problems using MOLE!!!

Remember: MOLE is also available in C++.

8 Non-uniform Operators

MOLE is also equipped with mimetic operators that act on structured non-uniform staggered grids. Depending on the problem, it is sometimes useful to have more spatial resolution on specific locations of the physical domain. To obtain non-uniform mimetic operators from MOLE, the user must type:

```
divNonUniform(k, grid);
```

or

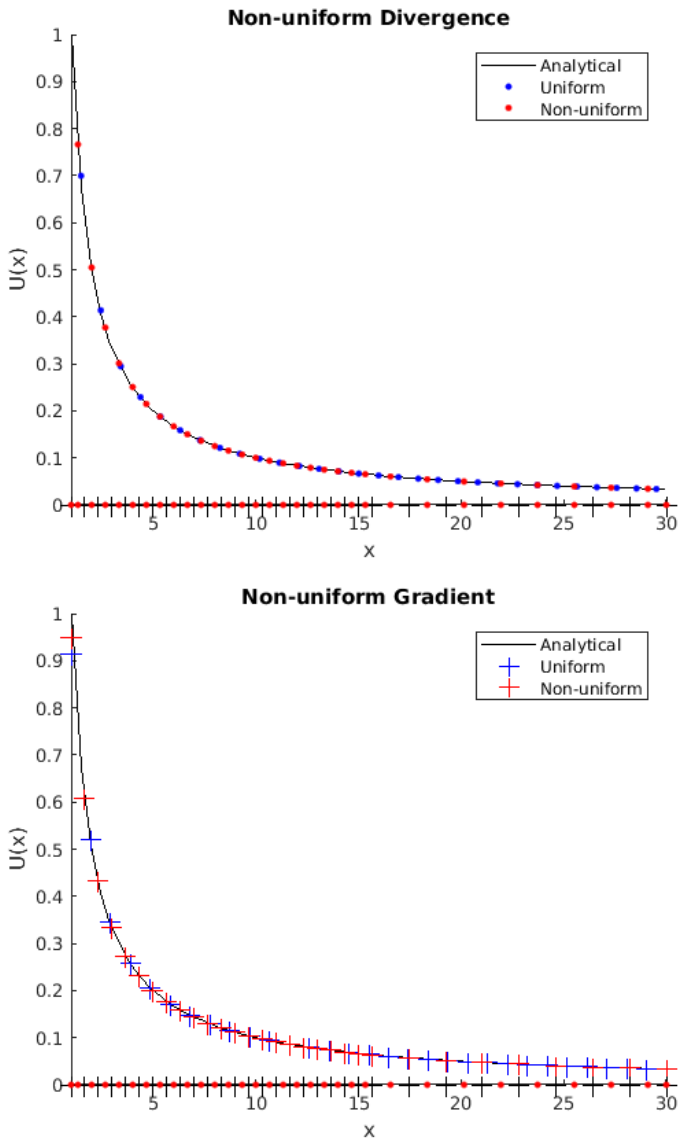
```
gradNonUniform(k, grid);
```

for higher dimensions,

```
div2DNonUniform(k, grid), div3DNonUniform(k, grid),
grad2DNonUniform(k, grid), and
grad3DNonUniform(k, grid);
```

where '`grid`' is the array that contains the coordinates of each cell center or each edge (depending on the operator being computed).

MOLE computes these non-uniform operators by multiplying the corresponding inverted Jacobian matrix to the corresponding uniform operator. For more information on non-uniform operators, please see [9].



9 Compact Operators

Compact representation of higher-order mimetic operators (order 4th and up) is also implemented in MOLE. It consists on the factorization of higher-order divergence and gradient operators in the following way,

let D_2 and G_2 denote the 2nd-order divergence and gradient operators. Let R_k and L_k be the right and left factor matrices of the kth-order divergence and gradient operators, respectively.

Thus, a k[th]-order mimetic Laplacian operator can be represented as:

$$\check{L}_k = D_2 R_k L_k G_2$$

where $R_k L_k$ is known as the "star*" operator. This operator is the tensor that contains all the material properties.

For more information on compact operators, look at the folder named "`compact_operators`" and [10].

References

[1] A Matrix Analysis Approach to Higher-Order Approximations for Divergence and Gradients Satisfying a Global Conservation Law. SIAM journal on matrix analysis and applications. 2003.

[2] J. Aarnes, S. Krogstad, and K-A Lie. Multiscale mixed/mimetic methods on corner-point grids. Computational Geoscience. 2008.

[3] J.M. Hyman, J. Morel, M. Shashkov, and S. Steinberg. Mimetic finite-difference methods for diffusion equations. Computational Geoscience. 2002.

[4] A. Abba and L. Bonaventura. A mimetic finite-difference method for large eddy simulation of incompressible flow. Technical Report. Politecnico di Milano, Milano, Italy, August 2010.

[5] E. Barbosa and O. Daube. A finite-difference method for 3D incompressible flows in cylindrical coordinates. Computational Fluids. 2005.

[6] E. Haber and J. Modersitzki. A multilevel method for image registration. SIAM journal on scientific computing. 2006.

[7] C. Di Bartolo, R. Gambini, and J. Pullin. Consistent and mimetic discretizations in general relativity. Journal of mathematical physics. 2005.

[8] J.M. Hyman and M. Shashkov. Mimetic discretizations for Maxwell's equations and equations of magnetic diffusion. Fourth International

Conference on Mathematical and Numerical Aspects of Wave Propagation, Golden, Colorado. SIAM. 1998.

[9] Mimetic schemes on non-uniform structured meshes. E. D. Batista and Jose E. Castillo. Electronic transactions on numerical analysis. Vol. 34. 2009.

[10] High order compact mimetic differences and discrete energy decay in 2D wave motions. Jose E. Castillo and Guillermo F. Miranda. Spectral and high order methods for partial differential equations. ICOSAHOM. 2017.