The Journal of Open Source Software

MOLE: Mimetic Operators Library Enhanced

Johnny Corbino[1¶] **and Jose E. Castillo**[1]

1 Computational Science Research Center, San Diego State University, 5500 Campanile Dr, San Diego, California, 92182. ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Jed Brown

Reviewers:

- @ketch
- @jshipton

Submitted: 30 December 2020
Published: unpublished

Summary

MOLE is a high quality (C++ & MATLAB) library that implements high-order mimetic operators. It provides discrete analogs of the most common vector calculus operators: Gradient, Divergence, Laplacian and Curl. These operators (matrices) act on staggered grids (uniform and nonuniform) and they satisfy local and global conservation laws. These operators can be used to build codes to solve partial differential equations (PDEs).

The mathematics is based on the work of (Corbino & Castillo, 2020). In addition, the user may find useful previous publications such as (J. E. Castillo & Grone, 2006), in which similar operators are derived using a matrix analysis approach.

Mimetic operators

Mimetic finite-difference operators, divergence (**D**), gradient (**G**), curl (**C**), and laplacian (**L**) are discrete analogs of their corresponding continuum operators. These mimetic finite-difference operators satisfy in the discrete sense the vector identities that the continuum ones do, making them more faithful to the physics (Corbino & Castillo, 2020).

The basis of higher-dimensional operators, as well of more sophisticated operators such as the laplacian or the biharmonic operator are the one-dimensional mimetic **G** and **D** operators. These finite-dimensional operators can be reused throughout the model and they provide a higher level of abstraction at the time of solving differential equations.

These operators, have been used to write codes to solve PDEs of different types (Bazan et al., 2011), (Boada et al., 2020), (Velazco et al., 2020), (Rojas et al., 2008), (Villamizar et al., 2021), (Puente et al., 2014), (Abouali & Castillo, 2013). For an overview of mimetic methods of different types see the book by Castillo and Miranda and the references there in (José E. Castillo & Miranda, 2013).

Statement of need

Implementing mimetic operators is not a trivial matter, particularly in three dimensions, this is substantially facilitated by MOLE relieving the user to devote their time to focus on the problem of interest. The user interested in solving, for example, a Poisson equation $-\nabla^2 u = f$, will be solving a discrete analog of this equation, $-DG\bar{u} = \bar{f}$, by using MOLE with a few lines of code.

State of the field

A previous library (Sanchez et al., 2014) was developed to implement the mimetic operators presented in (J. E. Castillo & Grone, 2006). This library was only capable of handling dense matrices so it was limited to solve small problems hence its development was stopped. MOLE implements the operators presented in the Corbino and Castillo paper (Corbino & Castillo, 2020). These operators are optimal from the number of points in each stencil and produce more accurate results. MOLE deals with sparse matrices efficiently and is capable of solving problems with millions of cells. To the best of the authors' knowledge, there are no other libraries that implement mimetic methods as the ones presented in this paper.

The library

MOLE was designed to be an intuitive software package to construct mimetic operators based on (Corbino & Castillo, 2020) method. MOLE is implemented in C++ and in MATLAB scripting language (these are two independent flavors) and every single function in MOLE returns a sparse matrix of the requested mimetic operator. For information on the installation or usage of the library, please read the User's Manual which is included in the repository.

Mimetic operators can be easily used to build codes to solve PDEs with a few lines of code. For example, if the user wants to get a one-dimensional k-order mimetic Laplacian, just need to invoke:

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

where **k** is the desired order of accuracy, **m** is the number of cell centers (spatial resolution), and **dx** is the step length. All functions in MOLE are quite consistent with this syntax, and more information regarding the signature of the function can be accessed via the `help` command. The C++ version of the library only depends on *Armadillo*, which is an open-source package for dense and sparse linear algebra (Sanderson & Curtin, 2016).

It is important to mention that MOLE's main role is the construction of matrices that represent spatial derivative operators and boundary conditions; other components such as solvers and time steppers are only provided via self-contained examples.

The following code snippet shows how easy is to solve a boundary value problem (with Robin's boundary conditions) through MOLE:

```
addpath('../mole_MATLAB')  % Add path to library

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

k = 4;  % 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

% Impose Robin BC on laplacian operator
a = 1;  % Dirichlet coefficient
b = 1;  % Neumann coefficient
L = L + robinBC(k, m, dx, a, b);  % Add BCs to laplacian operator

% 1D Staggered grid
grid = [west west+dx/2 : dx : east-dx/2 east];

% RHS
U = exp(grid)';
U(1) = 0;  % West BC
U(end) = 2*exp(1);  % East BC
```

```
U = L\U;  % Solve a system of linear equations

% Plot result
plot(grid, U, 'o-')
title('Poisson''s equation with Robin BC')
xlabel('x')
ylabel('u(x)')
```

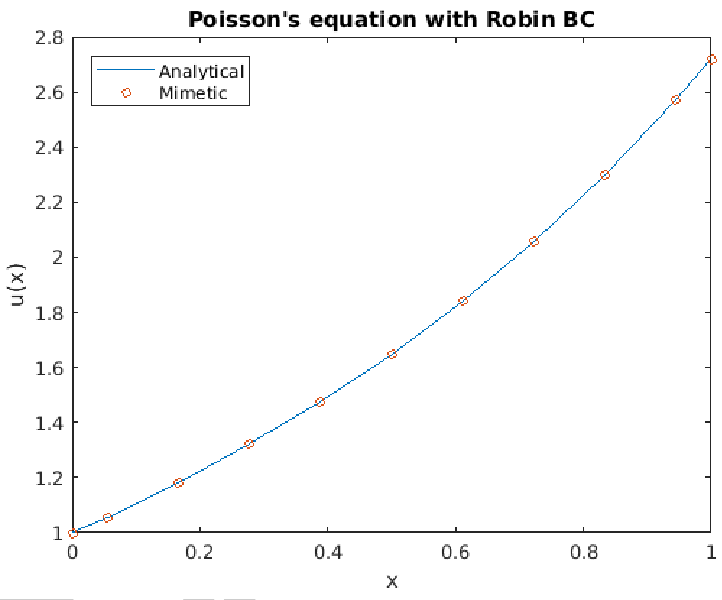

Figure 1: Solution to BVP using *k=4* and *m=9*.

Concluding remarks

In this short article we introduced MOLE, an open-source library that implements the mimetic operators from (Corbino & Castillo, 2020). For conciseness purposes, we showed a one-dimensional Poisson problem as example, however, MOLE comes with over 30 examples that range from the one-way wave equation to highly nonlinear and computationally demanding problems such as Richard's equation for unsaturated flow in porous media. The user can find such examples in the Examples folder.

References

Abouali, M., & Castillo, J. E. (2013). Stability and performance analysis of the castillo-grone mimetic operators in conjunction with RK3 time discretization in solving advective equations. *Procedia Computer Science*, *18*, 465–472.

Bazan, C., Abouali, M., Castillo, J., & Blomgren, P. (2011). Mimetic finite difference methods in image processing. *Computational & Applied Mathematics*, *30*(3), 701–720.

Boada, A., Paolini, C., & Castillo, J. E. (2020). High-order mimetic finite differences for anisotropic elliptic equations. *Computers & Fluids*, *213*, 104746.

Castillo, J. E., & Grone, R. D. (2006). A Matrix Analysis Approach to Higher-Order Approximations for Divergence and Gradients Satisfying a Global Conservation Law. *Matrix Analysis and Applications*, *25*. https://doi.org/10.1137/S0895479801398025

Castillo, José E., & Miranda, G. F. (2013). *Mimetic discretization methods*. CRC Press.

Corbino, J., & Castillo, J. E. (2020). High-order mimetic finite-difference operators satisfying the extended Gauss divergence theorem. *Computational and Applied Mathematics*, *364*. https://doi.org/10.1016/j.cam.2019.06.042

Puente, J. de la, Ferrer, M., Hanzich, M., Castillo, J. E., & Cela, J. M. (2014). Mimetic seismic wave modeling including topography on deformed staggered grids. *Geophysics*, *79*(3), T125–T141.

Rojas, O., Day, S., Castillo, J., & Dalguer, L. A. (2008). Modelling of rupture propagation using high-order mimetic finite differences. *Geophysical Journal International*, *172*(2), 631–650.

Sanchez, E. J., Paolini, C. P., & Castillo, J. E. (2014). The mimetic methods toolkit: An object-oriented api for mimetic finite differences. *Journal of Computational and Applied Mathematics*, *270*, 308–322.

Sanderson, C., & Curtin, R. (2016). Armadillo: a template-based C++ library for linear algebra. *Open Source Software*, *1*. https://doi.org/10.21105/joss.00026

Velazco, A. B., Corbino, J., & Castillo, J. (2020). High order mimetic difference simulation of unsaturated flow using richards equation. *Mathematics in Applied Sciences and Engineering*, *1*(4), 401–409.

Villamizar, J., Calderón, G., Carrillo, J., Bautista Rozo, L., Carrillo, J., Rueda, J., & Castillo, J. (2021). Mimetic finite difference methods for restoration of fundus images for automatic detection of glaucoma suspects. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, 1–8.