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.

DOI: 10.21105/joss.03380

Software

- Review
- Repository
- Archive

Editor: Jed Brown

Reviewers:

- @ketch
- @jshipton

Submitted: 30 December 2020
Published: 18 June 2021

Summary

MOLE is a high quality (C++ & MATLAB) library that implements high-order mimetic operators to solve partial differential equations. 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.

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

Mimetic operators

All linear transformations can be represented by a matrix multiplication, integration and differentiation are linear transformations. Mimetic operators are essentially matrices that when applied to discrete scalar or vector fields produce high-order approximations that are faithful to the physics.

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 Gradient (**G**) and Divergence (**D**) operators. These operators can be reused throughout the model and they provide a higher level of abstraction at the time of solving differential equations.

The library

MOLE was designed to be an intuitive software package to numerically solve partial differential equations using mimetic methods. 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.

For example, if the user wants to get a one-dimensional *k*-order mimetic Laplacian, just need to invoke:

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

*Corresponding author.

The Journal of Open Source Software

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).

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)')
```

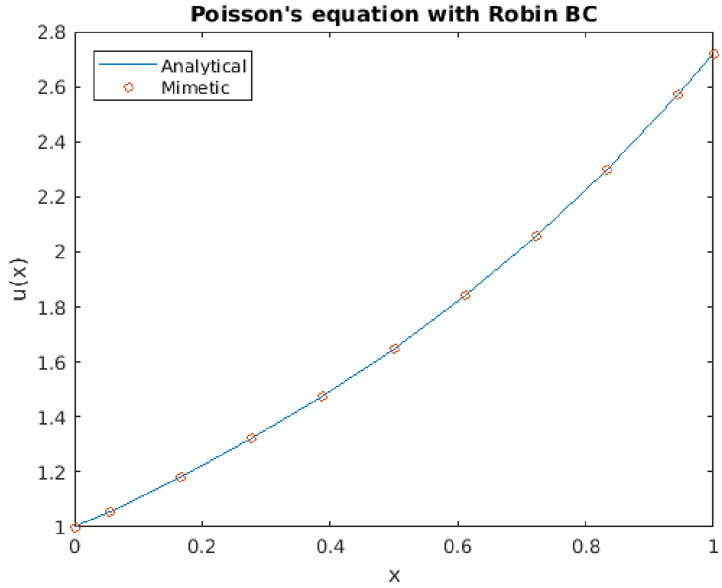

Figure 1: Solution to BVP using *k=4* and *m=9*.

References

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

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

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