Available online at www.sciencedirect.com
ScienceDirect
Mathematics and Computers in Simulation 139 (2017) 23–36
MATHEMATICS AND COMPUTERS IN SIMULATION
www.elsevier.com/locate/matcom

Original articles

Solving Poisson equation with Robin boundary condition on a curvilinear mesh using high order mimetic discretization methods

Mohammad Abouali*, Jose E. Castillo

Computational Science Research Center at San Diego State University, 5500 Campanile Drive, San Diego, CA 92182-1245, USA

Received 6 December 2013; received in revised form 1 June 2014; accepted 10 October 2014
Available online 3 December 2014

Abstract

This paper introduces a new software package, written in MATLAB®, that generates an extended discrete Laplacian ($L = DG = \nabla \cdot \nabla$) based on the Castillo–Grone Mimetic difference operators over a general curvilinear grid. The boundary conditions are included in the extended discrete Laplacian Operator, i.e. Dirichlet, Neumann, as well as Robin boundary conditions. The user only needs to provide a curvilinear grid, a desired operator order, and the degree of the interpolating polynomial. The operator is tested in different condition and its performance is reported. Finally, based on accuracy of the results, amount of required memory, and the computation that is needed, it is concluded that the 4th order operator is the best one.

Keywords: Poisson's equation; Castillo–Grone's mimetic operators; Higher-order method; General curvilinear grids

1. Introduction

Poisson's equation, named after French mathematician Simeon Denis Poisson, is an elliptic partial differential equation with wide applications in many fields, such as electrostatics, magnetism, mechanical engineering, fluid dynamics [3,11,12], ground water, and dispersion of pollutants. Poisson's equation is written as follows:

$$\nabla \cdot \nabla f = F, \tag{1}$$

where f, in this paper, is an unknown 2D function, i.e. $f(x, y)$ that would be determined after solving the above equation and $F(x, y)$ is a 2D function, called Right Hand Side (RHS) function, which is either known analytically or at least a discretized estimation if it is known. $L = \nabla \cdot \nabla$ is called Laplace operator, not to be mistaken by Laplace equation, which is pretty much Poisson's equation once $F(x, y)$ is zero.

Due to its importance, there are many attempts and studies devoted to Poisson's equation. There are many solvers available online. However, most of these studies or software focus on Poisson's equation in Cartesian rectilinear grids, i.e. regularly spaced rectangular domain. The majority of the software, make use of a certain discretization scheme and convert Poisson's equation in matrix form as follows:

$$\nabla \cdot \nabla f = F \rightarrow Af = F, \tag{2}$$

* Corresponding author.
E-mail address: maboualiedu@gmail.com (M. Abouali).

http://dx.doi.org/10.1016/j.matcom.2014.10.004

where f and F are $n \times 1$ vectors and:

$$\nabla = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right]^T, \tag{3}$$

n is the number of discretized points, and A is a $n \times n$ matrix that is obtained based on the numerical discretization method in use along with the specified boundary condition. Castillo–Grone's Mimetic (CGM) difference operators are used here to construct a discrete analog of Poisson's equation. CGM has shown to perform very well in various past studies [10]. Abouali and Castillo showed that using CGM operators to solve advective equation results into a stable scheme for Courant number larger than one [2]. Abouali and Castillo also showed that the discrete Poisson's equation using second order CGM operators along with linear interpolation, performs up to %25 better relative to conservative discretization method in curvilinear grids in terms of the accuracy of the results [1]. This paper focuses on the higher order CGM operators in conjunction with higher degree interpolating polynomials. Since, it was already shown that the second order CGM operators perform better for Poisson's equations in curvilinear grids, i.e. a baseline of performance is established, the other variants of the CGM operators are compared to this established baseline.

Next section shortly describes how the matrix A is constructed. Later, CGM operators are tested in various conditions and the results are presented.

2. Methods

2.1. Poisson equation in curvilinear grids

Poisson's equation can be written in vector form using the gradient and divergence operator, as follows:

$$\nabla \cdot (\kappa \nabla f) = F, \tag{4}$$

where $\nabla = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right]^T$ and κ is the identity matrix, here. Poisson's equation can be written also in matrix form as follows:

$$\nabla^T \kappa \nabla f = F. \tag{5}$$

In order to solve the above equation, one needs to transform the equation from the curvilinear physical domain to the regularly spaced computational space. Full description of this procedure can be found on many manuscripts [5,13,14]. A summary of this procedure is provided here for the sake of completeness. Assuming that (x, y, z) domain is transformed to (ξ, η, ζ) space, the operators in these two domains are related to each other by $\tilde{\nabla} = T\nabla$, where $\tilde{\nabla} = \left[\frac{\partial}{\partial \xi}, \frac{\partial}{\partial \eta}, \frac{\partial}{\partial \zeta}\right]^T$, and

$$T = \begin{bmatrix} x_\xi & y_\xi & z_\xi \\ x_\eta & y_\eta & z_\eta \\ x_\zeta & y_\zeta & z_\zeta \end{bmatrix} \tag{6}$$

where x_ξ is the shorthand for $\partial x/\partial \xi$. Inverting the above equation, one gets $\nabla = T^{-1}\tilde{\nabla} = \frac{1}{\mathbf{J}} C \tilde{\nabla}$, where $\mathbf{J}$ is the Jacobian of the transformation matrix and C is the cofactor of the transformation matrix. By substituting this in Eq. (4), one can rewrite Poisson's equation as:

$$\nabla^T \kappa \nabla f = \underbrace{\frac{1}{\mathbf{J}} \tilde{\nabla}^T \tilde{\kappa} \tilde{\nabla}}_{A} f = F, \tag{7}$$

where:

$$\tilde{\kappa} = \frac{1}{\mathbf{J}} C^T k C. \tag{8}$$

It has to be noted that $\tilde{\kappa}$ is no longer the identity matrix and all the off diagonal entries are not necessarily zero anymore. This means that Poisson's equation will be transformed into a fully elliptic equation in computational domain

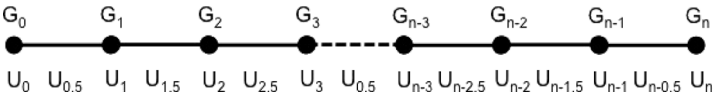


Fig. 1. Staggered computation of the CGM Gradient operator in 1D. U_i represents the data and G_i represents the calculated gradient.

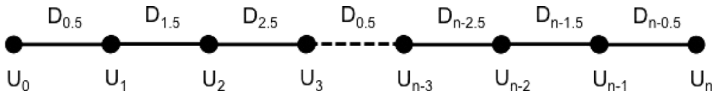


Fig. 2. Staggered computation of the CGM Divergence operator in 1D. U_i represents the data and D_i represents the calculated divergence.

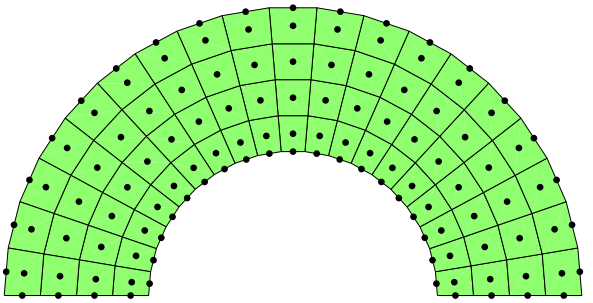


Fig. 3. Staggered computation of the CGM Gradient operator in 2D. Black dots represent the location where the data should be stored for the gradient operator.

with all cross-derivatives being present. This adds additional complexity to the solution of Poisson's equation. As an example, in a 2D curvilinear grid $\tilde{\kappa}$ can be written as follows:

$$\mathbf{J}\tilde{\kappa} = \begin{bmatrix} (y_\eta^2 + x_\eta^2) & -(y_\xi y_\eta + x_\xi x_\eta) \\ -(y_\xi y_\eta + x_\xi x_\eta) & (y_\xi^2 + x_\xi^2) \end{bmatrix}. \tag{9}$$

Therefore, Poisson's equation will be transformed into a diffusion problem, where the diffusion coefficient is neither homogeneous nor isotropic. It can be easily shown that when the grid is orthogonal (or near orthogonal) the off diagonal elements are zero.

2.2. Castillo–Grone's mimetic (CGM) difference operators

The Castillo–Grone mimetic approach provides the exact equivalent of the gradient and the divergence operator in a discrete domain [10,6]. It has been shown that these discrete operators satisfy all the properties of their continuous version in the discrete sense, are conservative, and provide the same accuracy in the entire domain, including the boundaries without the need for dummy or ghost nodes [7,9,4,3]. The details of these operators and how they are generated are not discussed here and the reader is encouraged to refer to one of the references. Castillo–Grone mimetic approach uses a staggered grid to calculate the gradient and the divergence. For better understanding refer to Figs. 1 and 2 for 1D representation and Figs. 3 and 4 for 2D representation.

2.3. MATLAB® command

The MATLAB® command that generates the discrete extended Laplace operator based on CGM operators is:

```
constructCGMLap(xn,yn,BC, OpOrder,InterpOrder,nInterp,verbose)
```

where:

- xn and yn are the coordinates of the nodal curvilinear grid,
- OpOrder is the CGM operator order, (default value is 2),

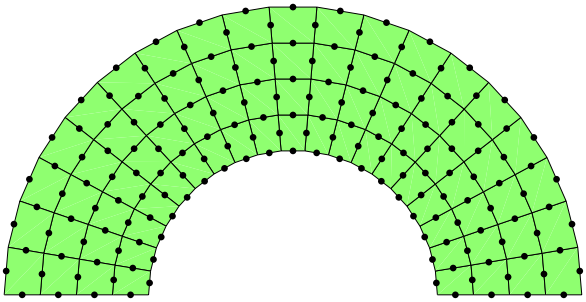


Fig. 4. Staggered computation of the CGM Divergence operator in 2D. Black dots represent the location where the data should be stored for the divergence operator.

- InterpOrder is the degree of the interpolating polynomial, (default is 1),
- nInterp is the number of nodes to be used to construct the interpolator,
- verbose, is a logical variable that defines the level of the output to be printed on the screen.

BC is $n_b \times 2$ matrix, where $n_b = 2(M + N)$ is the number of nodes at the boundary, and M and N are the number of cells along ξ and η direction. The first $2M$ entries are related to the bottom and top boundaries, i.e. $\eta = 1$ and $\eta = N + 1$, and the last $2N$ entries are related to left and right boundaries, i.e. $\xi = 1$ and $\xi = M + 1$. The first column of BC defines the α and the second column defines the β coefficient in Robin boundary condition, i.e.:

$$\alpha f + \beta \nabla f \cdot \vec{n} = \gamma, \tag{10}$$

where $\vec{n}$ is the normal vector pointing outward at the boundary. If β is set to zero, Dirichlet boundary condition is obtained and if α is set to zero Neumann boundary condition is obtained. Once the command is executed the resulting Laplacian operator which includes the mimetic boundary conditions is calculated along with the coordinate of the CGM grid, cell centers, and the index of the points that fall on the border.

3. Results and discussion

In order to find out how sparse the operators are and how long does it take to build these operators a series of tests were performed on a uniformly spaced rectangular domain with 80×80 nodes. It was seen that relatively all operators require the same amount of time to be built regardless of the order of the operators. On a machine with Intel® i3 CPU and 16 GB of memory the second order operator took 14.7 [s], fourth order operator took 15.1 [s], and sixth order operator took 15.59 [s]. Although this difference gets larger by increasing the problem size, i.e. number of nodes in the grid, however, it would be still negligible. The number of non-zero elements increases with the order of the operators though. In this example the second order operator had 31,521 non-zero elements, fourth order operator had 81,449 non-zero elements, and sixth order operator had 13,0429 non-zero elements. This means that there was a 258% increase in non-zero elements by choosing 4th order operators relative to the 2nd order operator and 414% increase by choosing 6th order operator. However, the density of the matrix, i.e. the number of non-zero elements to the total number of elements, did not reached even 0.5% for the most dense operator, i.e. 6th order. The density of these matrices decreases by increasing the number of nodes in the grid. This is summarized in Table 1.

A measure of error is required to understand how accurate these operators are. Three different measures were chosen as follows:

- **Root Mean Square Error** or RMSE which is calculated as follows:

$$\text{RMSE} = \sqrt{\sum_{i}^{n} \frac{(f_i - f_i^*)^2}{n}}. \tag{11}$$

- L_2 **norm**, which is computed as follows:

$$\|\text{Error}\|_2 = \sqrt{\sum_{i}^{n} (f_i - f_i^*)^2}. \tag{12}$$

Table 1
Required time in seconds and number of non-zero operator.

Operator order	Time (s)	# non-zero	Density	% increase
2nd	14.7	31 521	0.07	–
4th	15.1	81 449	0.19	258
6th	15.59	130 429	0.30	414

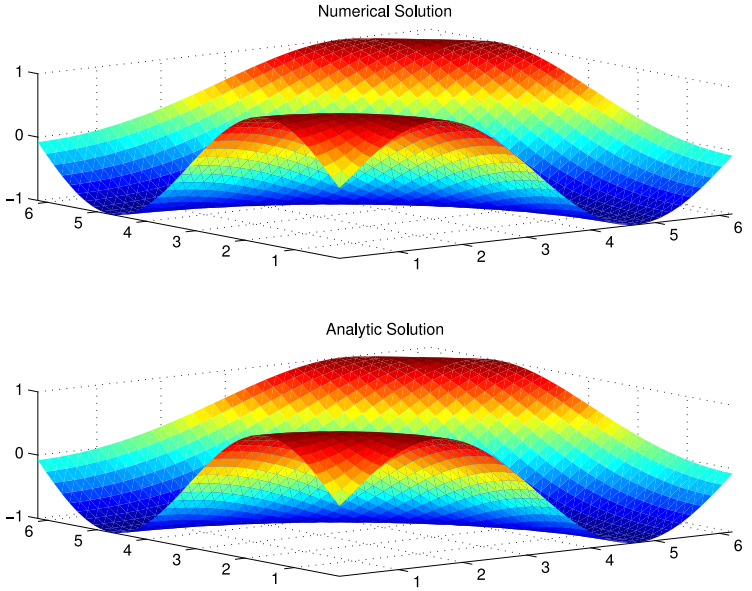


Fig. 5. Sample numerical and analytic solution using 4th order operators.

- **Max error** or L_∞ norm, which is calculated as follows:

$$\|\text{Error}\|_\infty = \max_i |f_i - f_i^*|. \tag{13}$$

In above equations, n is the number of nodes, f_i is the numerical solution, and f_i^* is the analytic or exact solution. In order to test the accuracy, the right hand side of Poisson's equation, i.e. $\nabla \cdot \nabla f = F$, was set to:

$$F(x, y) = \frac{\cos(\sqrt{x^2 + y^2})}{\sqrt{x^2 + y^2}} - \sin(\sqrt{x^2 + y^2}), \tag{14}$$

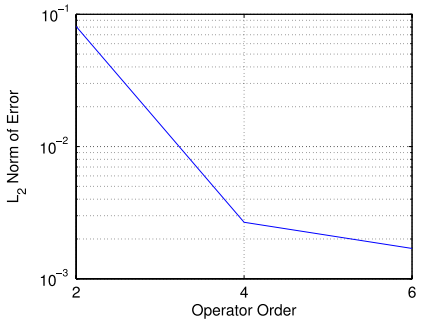
where $x, y \in [0, 2\pi] \times [0, 2\pi]$ and Dirichlet boundary condition was used. The exact solution, i.e. f, would be:

$$f(x, y) = \sin(\sqrt{x^2 + y^2}). \tag{15}$$

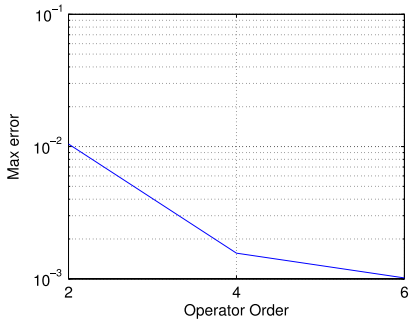
Fig. 5 shows a sample solution to the above equation using the 4th order Castillo–Grone's operators. The grid had 40×40 nodes and Algebraic Multi-Grid solver, implemented by Yvan Notay [18,16,15,17], was used to solve the resulting system of equations. Fig. 6 shows how the error is changing by increasing the operator order. These graphs were generated over a uniform rectangular grid with 20×20 nodes; therefore, $n = 190, 969$.

One property of each numerical method is the global operator order. This measure shows how the error decreases by refining the grid spacing. Therefore, four different grid spacing were used as follows:

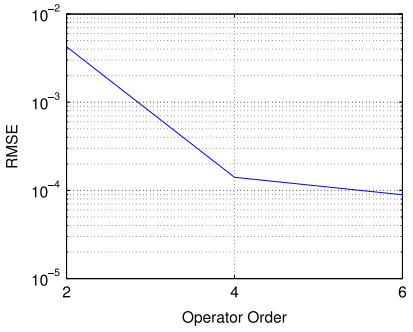
- Grid 1: $10 \times 10 \rightarrow dx = dy = 0.6981$,
- Grid 2: $20 \times 20 \rightarrow dx = dy = 0.3307$,
- Grid 3: $40 \times 40 \rightarrow dx = dy = 0.1611$,
- Grid 4: $80 \times 80 \rightarrow dx = dy = 0.0795$.



(a) L_2 norm of the error.



(b) Max error or L_∞.



(c) RMSE.

Fig. 6. Changes of error by increasing operator order.

Fig. 7 shows how the error decreases by refining the grid spacing or increasing the number of nodes. Although different error measures show different rate of which the error is decreased by refining the grid; however, almost all of them show that this rate is greater in coarser grids and decreases in finer grids. There is only one exception for the 4th order operator while using max error. It appears that error drops faster for finer grids than for coarser grids, Fig. 7(b).

So far only the Dirichlet boundary condition was used. Further tests were performed in order to test for Neumann boundary condition. Note that the Neumann boundary condition is defined as:

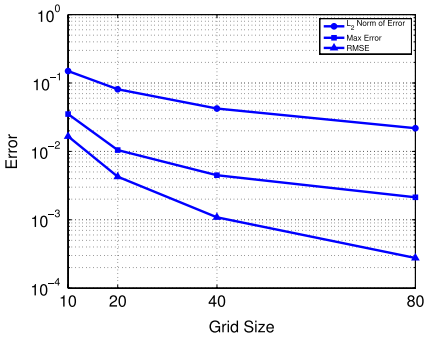
$$\frac{\partial f}{\partial \vec{n}} = \nabla f \cdot \vec{n} = \gamma(x, y), \tag{16}$$

where $\vec{n}$ is the normal vector to the boundary facing outward. To test the Neumann boundary condition the same right hand side function was used; however, at $y = 0$ and $y = 2\pi$ Dirichlet boundary condition was used and at $x = 0$ and $x = 2\pi$ Neumann boundary condition was used. The sparse matrix that is generated, i.e. the matrix A in $\nabla \cdot \nabla f = F \rightarrow Af = F$, includes the mimetic boundary conditions. Again as the operator orders increase the RMSE is lowered. However, in all cases using the Neumann boundary condition resulted in more error. It appears that higher order operators are more susceptible to Neumann boundary condition, Fig. 8. Similar results were obtained once Dirichlet boundary condition was used at $x = 0$ and $x = 2\pi$ and Neumann boundary condition was used at $y = 0$ and $y = 2\pi$.

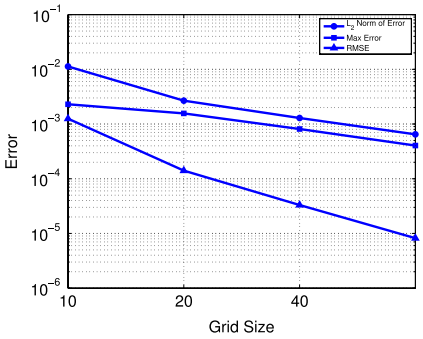
The Robin boundary condition was also tested. Note that the Robin boundary condition is defined as:

$$\alpha f + \beta \nabla f \cdot \vec{n} = \gamma. \tag{17}$$

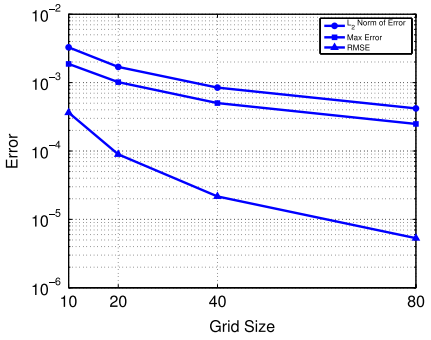
Note that by setting β to zero the Robin boundary condition is reduced to Dirichlet boundary condition and by setting α to zero Neumann boundary condition is obtained. Hence, the Robin boundary condition is considered to be



(a) 2nd order operators.



(b) 4th order operators.



(c) 6th order operators.

Fig. 7. Global operator order.

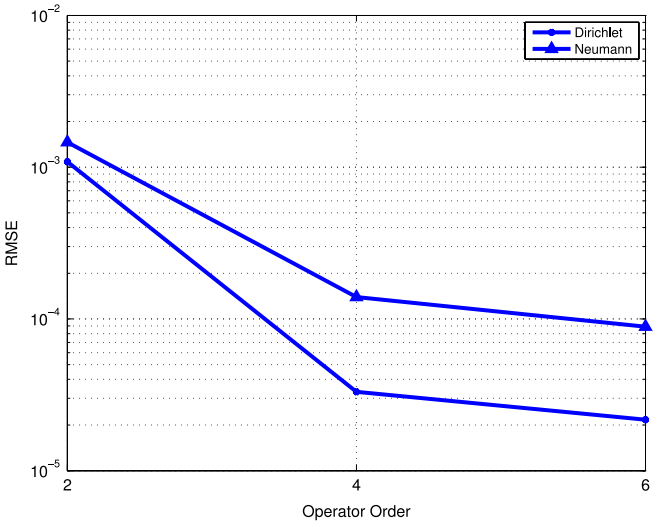


Fig. 8. Error comparison between Neumann and Dirichlet boundary condition.

the most complete boundary condition. To test Robin boundary condition the right hand side function was changed to:

$$F(x, y) = \frac{\lambda^2 e^{\lambda x}}{e^{\lambda} - 1}, \tag{18}$$

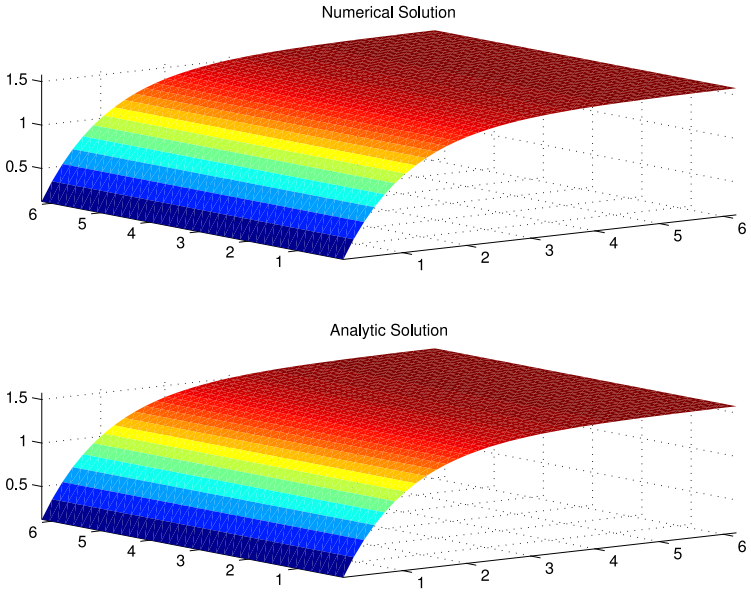


Fig. 9. Sample numerical and analytic solution using 4th order operators for Eq. (19).

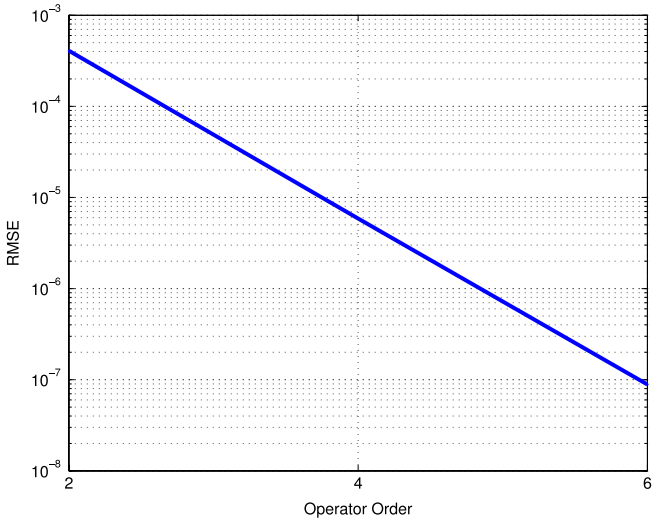


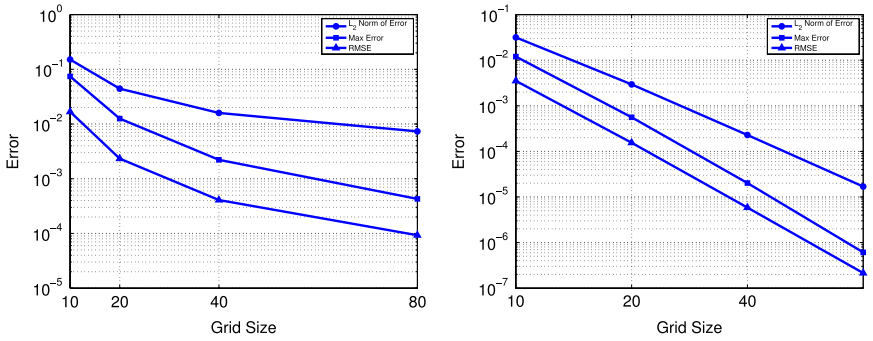
Fig. 10. The accuracy improves by increasing the operator order once Robin boundary condition is used.

where $\lambda = -1$ and $x, y \in [0, 2\pi] \times [0, 2\pi]$. Dirichlet boundary condition was applied at $y = 0$ and $y = 2\pi$, and Robin boundary condition was applied at $x = 0$ and $x = 2\pi$. The exact or analytic solution to this problem is

$$f(x, y) = \frac{e^{\lambda x} - 1}{e^{\lambda} - 1}. \tag{19}$$

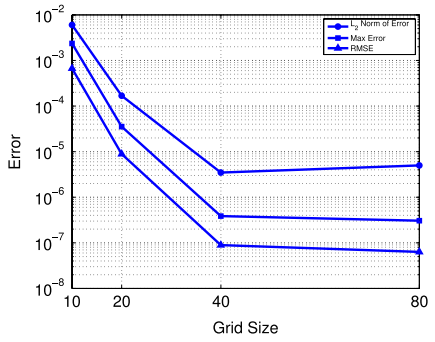
Fig. 9 shows a sample solution to the above equation using 4th order operator and 40×40 grid. As in two previous cases, the error decreases by increasing the operator orders. However, in previous cases, i.e. when Dirichlet and Neumann boundary conditions were used, the 6th order operator did not show much of improvement in accuracy and its accuracy was comparable to that of the 4th order operator. However, when Robin boundary condition is used the accuracy improvement of the 6th order operator is considerable relative to that of the 4th order operator, Fig. 10.

Once Robin boundary condition is used, the relation between the grid spacing and the operator order shows no considerable change for the second order operator, i.e. the rate of improvement in the accuracy tends to slow down as



(a) 2nd order operators.

(b) 4th order operators.



(c) 6th order operators.

Fig. 11. Improvement in the accuracy as the grid is refined once Robin boundary condition is used.

the grid is refined. However, 4th order operator shows a completely different pattern. As it can be seen in Fig. 11, the rate at which the accuracy increases by refining the grid tends to be slightly increasing as the grids are refined. Sixth order operators also behave completely differently. Although the rate of improvement in accuracy slows down in the beginning once Robin boundary condition is used; this slowing down is not as harsh as in the case when Dirichlet boundary condition is used. However, once a very fine grid is used, i.e. 80×80 grid, suddenly the error increases if L_2 norm is used and the improvement in the accuracy is not noticeable for L_∞ and RMSE, Fig. 11.

In order to test the operators better, a more challenging problem is chosen. The right hand side function is set to:

$$F(x, y) = (f_1 + f_2 + f_3 + f_4 - f_5 - f_6)/R^3, \tag{20}$$

where:

$$\begin{aligned}
f_1 &= 12R^2 - 40R^3,\\
f_2 &= 90\pi x^4 \cos(\varphi(R)),\\
f_3 &= 90\pi y^4 \cos(\varphi(R)),\\
f_4 &= 180\pi x^2 y^2 \cos(\varphi(R)),\\
f_5 &= 1800\pi^2 x^2 \sin(\varphi(R)) R^5,\\
f_6 &= 1800\pi^2 y^2 \sin(\varphi(R)) R^5,\\
\varphi(R) &= 20\pi R^3,\\
R &= \sqrt{x^2 + y^2}.
\end{aligned} \tag{21}$$

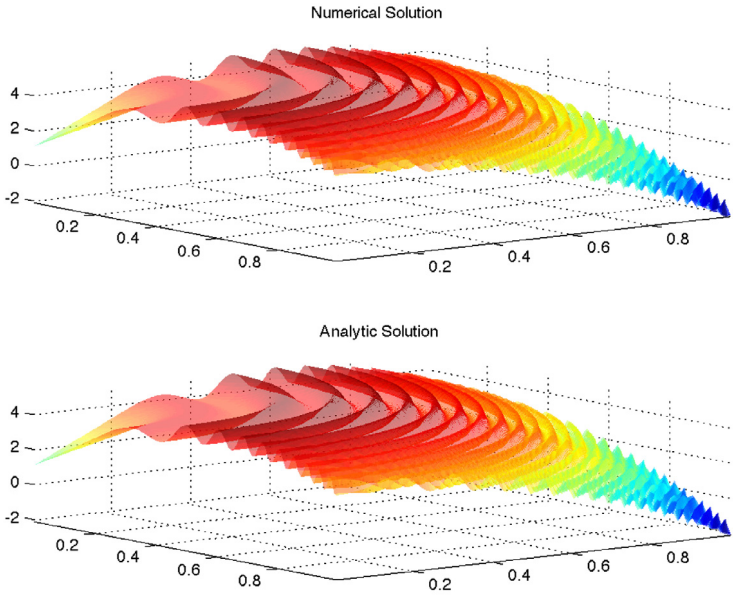


Fig. 12. A hard sample problem with lots of oscillations.

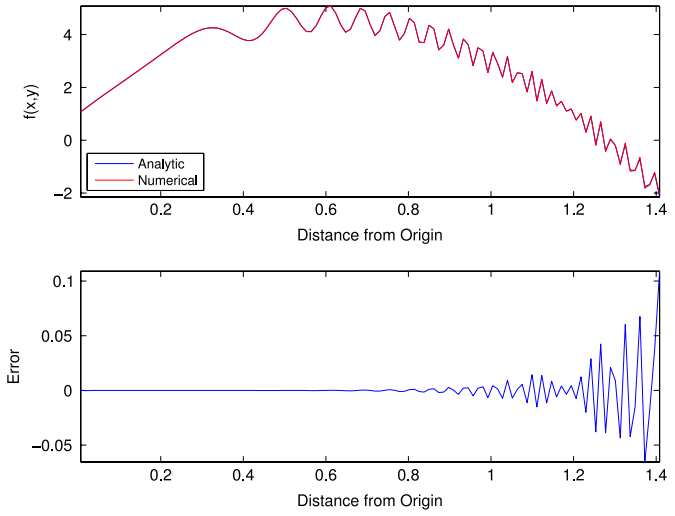


Fig. 13. Diagonal profile of the hard sample case and the error.

Dirichlet boundary condition was used at all boundaries and $x, y \in [0, 1] \times [0, 1]$. The analytic or exact solution to the above equation is:

$$f(x, y) = 1 + 12R(x, y) - 10R(x, y)^2 + \frac{1}{2}\sin(\varphi(R(x, y))). \tag{22}$$

A grid consisting of 120×120 nodes was used to discretize the domain. A sample solution using the 4th order operators along with the analytic solution is shown in Fig. 12. As it can be seen this problem contains a lot of oscillations and various frequencies, which makes it challenging. Existence of these high frequency oscillations in the solution requires a very fine grid. The profile of both numerical and analytic solution along the diagonal of the domain is shown in Fig. 13. As it can be seen the two solution almost fall on each other. The error is also shown in Fig. 13. It can be seen that the error is zero on the left, where there are not much oscillations, and the error starts to grow as the solution starts to oscillate at a higher frequency. It has to be noted that most of the error is due to lack of enough

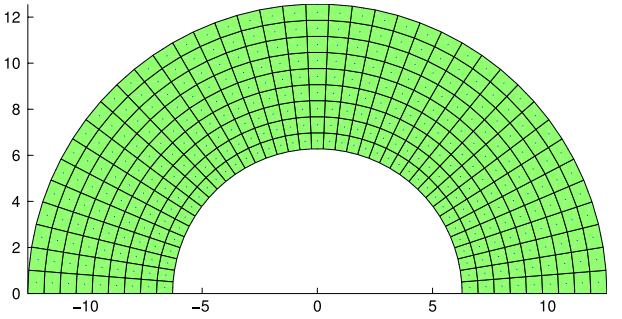


Fig. 14. A coarse curvilinear grid.

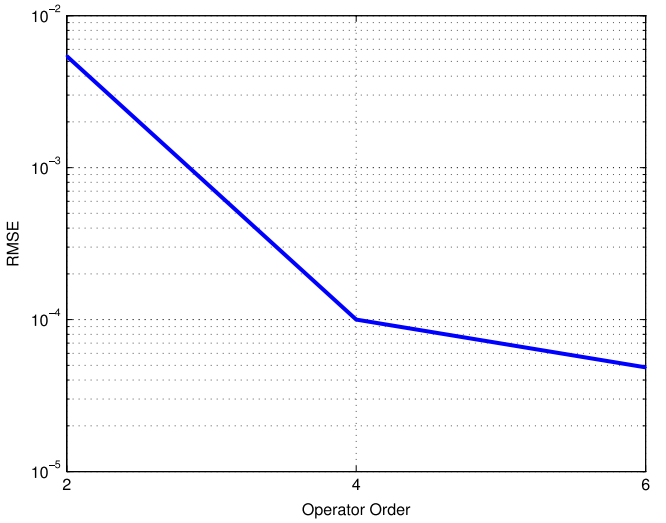


Fig. 15. Error variation by operator orders in curvilinear grid.

sampling in the domain, i.e. the lack of enough grid resolutions. The error in that region can be reduced by refining the grid in that part. This makes a perfect example for adaptive grid problem.

The operators were also tested on a curvilinear grid with curved boundaries. A sample coarse resolution curvilinear grid is shown in Fig. 14. The operators behaved exactly the same as in non-curvilinear grids. Fig. 15 shows how RMSE changes with the operator order.

It was also interesting to test the operators on non-smooth grids. To obtain a non-smooth grid, first a regularly spaced orthogonal non-curvilinear grid was generated on $[0, 2\pi] \times [0, 2\pi]$. Once the coordinates of the nodes were obtained a Gaussian noise was added to the coordinates. The lowest possible Signal to Noise Ratio (SNR), without causing the grid to fold on itself, was 25 once the domain was discretized using 20×20 nodes. This lowest possible SNR increased to 30 once the domain was discretized with 40×40 nodes. A sample noisy grid is shown in Fig. 16. A sample numerical solution along with the analytic solution is shown in Fig. 17. It has to be noted that Castillo et al. did study the non-smooth grids [8]. However, all the cases studied by them had perfectly straight boundaries and only the internal nodes were allowed to move.

Both the order of the operators and the degree of interpolating polynomial affects the accuracy in this case. Table 2 shows the RMSE and how it changes with the operator order and the degree of the interpolating polynomial. In the case of the non-smooth grids, as the operator order increases the RMSE also increases, i.e. higher order operators are more susceptible to non-smoothness in the grid. However, increasing the degree of the interpolating polynomial causes decrease in the RMSE, with one exception in the 2nd order operator and 4th degree interpolating polynomial. It has to be noted that as SNR increases, i.e. having a smoother grid, the results become very similar as in previous cases.

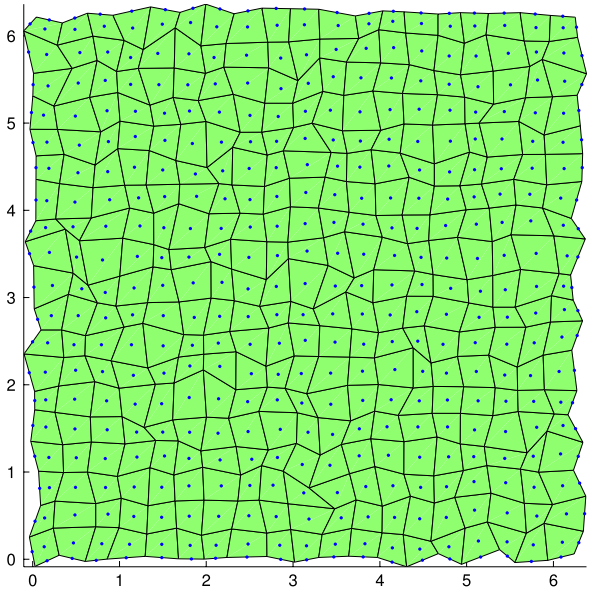


Fig. 16. Sample noisy grid ($SNR = 25$).

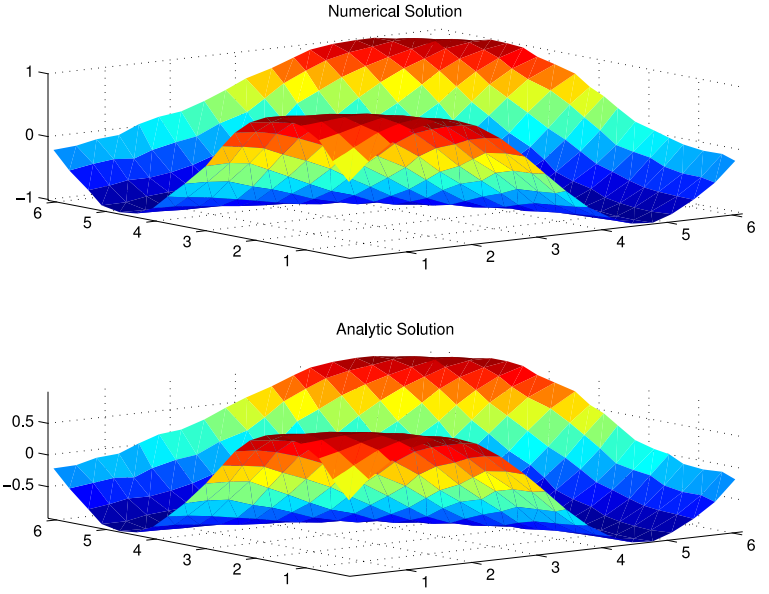


Fig. 17. Numerical and analytic solution on the sample noisy grid.

Table 2
RMSE on noisy grid using the minimum required number of nodes for interpolation.

Operator order	Degree of interpolating polynomial			
	1	2	3	4
2	0.0479	0.0161	0.0074	0.0229
4	0.0184	0.1042	0.0156	0.0128
6	0.5129	0.0855	0.0351	0.0154

The minimum number of nodes were used to construct the interpolating polynomial in Table 2. An interpolating polynomial of degree p_d has $(p_d + 1)^2$ coefficients that need to be determined; hence, the minimum number of nodes

Table 3
RMSE on noisy grid using more nodes than the minimum required for interpolation.

Operator order	Degree of interpolating polynomial			
	1	2	3	4
2	0.0042	0.0041	0.0045	0.0045
4	0.0057	0.0054	0.0061	0.0066
6	0.0086	0.0085	0.0091	0.0104

to form this polynomial would be equal to the number of coefficients. It is possible to use more node than the minimum required, to construct the interpolating polynomials. This is known as curve fitting and the resulting polynomial does not necessarily match the function value at each node. In order to understand the effect of curve fitting instead of interpolation, it was decided to use $(p_d + 2)^2 - 1$. Table 3 shows the RMSE in this case. It can be seen that the RMSE has been reduced considerably. By increasing the operator order the RMSE is still going to be increased; however, there would be no considerable changes in the accuracy once the degree of the interpolating polynomial is increased, i.e. the RMSE becomes independent of the degree of the interpolating polynomial. It has to be noted again that as SNR increases the results become very similar as in previous cases.

4. Conclusion

In previous studies, the accuracy of the extended Laplacian operator resulting from the second order CGM operators along with linear interpolation was investigated and a baseline was established showing that these operators perform better. In this paper, that study is extended to higher order CGM operators along with higher degree interpolating polynomial. It was seen that in general as the operator order increases the accuracy of the numerical solution also increases. However, the amount of increase in the accuracy by switching from 4th order operators to 6th order operators are not that significant. Although relatively all operators tend to take the same amount of time to be constructed (the difference was negligible); however, the amount of memory that is required to store the operators increases almost by a factor of 1.6. Moreover 4th order operators showed the highest rate of convergence for both fine grids and coarse grids. Based on these factors, The 4th order operator is recommended.

It was also shown that by increasing the degree of the interpolating polynomial the accuracy of the numerical solution increases. However, it was shown that if instead of interpolation, a curve fitting approach is used the accuracy increases considerably (in some cases by a factor of 10) and, more important, the accuracy becomes relatively independent of the degree of the interpolating polynomial in use. This finding is very important, particularly once non-smooth grids are in use.

The authors used aggregated multi-grid solver to solve the resulting system of linear equations and they would like to study the effect of various solvers along with different preconditioners in the near future.

Software availability

The software and the MATLAB codes are made available free of charge for academic and non-commercial use online at MATLAB File Exchange Central. The link to download the code is: http://www.mathworks.com/matlabcentral/fileexchange/44354-curvilinear-2d-grid-Poisson.

Please start with one of the test scripts to get acquainted with how to use the code.

References

[1] M. Abouali, J. Castillo, The castillo-grone's mimetic difference operators in 2d and 3d fully curvilinear grids: Case study of Poisson's equation, in: B. Soni, R.M. Spitaleri, J.P. Suarez (Eds.), MASCOT12 & ISGG12 Proceedings, in: IMACS Series in Computational and Applied Mathematics, vol. 18, IMACS, Rome, 2014, p. 300.

[2] M. Abouali, J. Castillo, Stability and performance analysis of the Castillo–Grone mimetic operators in conjunction with rk3 time discretization in solving advective equations, in: International Conference on Computational Science (ICCS), 2013, pp. 465–472.

[3] M. Abouali, J. Castillo, Unified curvilinear ocean atmosphere model (ucoam): A vertical velocity case study, J. Math. Comput. Model. 57 (2013) 2158–2168.

[4] E. Batista, J. Castillo, Mimetic schemes on non-uniform structured meshes, Electron. Trans. Numer. Anal. 34 (2009) 152–162.

[5] J. Castillo, Mathematical Aspects of Numerical Grid Generation, in: Frontiers in Applied Mathematics, Society for Industrial Mathematics, 1987.
[6] J. Castillo, R. Grone, Matrix analysis to high-order approximations for divergence and gradients satisfying a global conservation law, SIAM Matrix Anal. Appl. 25 (2003) 128–142.
[7] J. Castillo, J. Hyman, M. Shashkov, S. Steinberg, The sensitivity and accuracy of fourth order finite-difference schemes on nonuniform grids in one dimension, Compters Math. Appl. 30 (8) (1995) 41–55.
[8] J. Castillo, J. Hyman, M. Shashkov, S. Steinberg, High-order mimetic finite difference methods on nonuniform grids, in: A.V. Ilin, L. R Scott (Eds.), ICOSAHOM.95 : Proceedings of the Third International Conference on Spectral and High Order Methods, in: Houston Journal of Mathematics, University of Houston, 1996, pp. 347–362.
[9] J. Castillo, J. Hyman, M. Shashkov, S. Steinberg, Fourth-and sixth-order conservative finite difference approximations of the divergence and gradient, Appl. Numer. Math. 37 (2001) 171–187.
[10] J. Castillo, G. Miranda, Mimetic Discretization Methods, CRC Press, 2013.
[11] J. Ferziger, P. Millovan, Computational Methods for Fluid Dynamics, Vol. 1, Springer, 2001.
[12] B. Geurts, Elements of Direct and Large-Eddy Simulation, Edwards, 2004.
[13] K. Hoffmann, S. Chiang, Computational Fluid Dynamics, Vol. 2, Engineering Education System Book, 2000.
[14] P. Knupp, S. Steinberg, Fundamentals of Grid Generation, CRC Press, 1993.
[15] A. Napov, Y. Notay, An algebraic multigrid method with guaranteed convergence rate, SIAM J. Sci. Comput. 34 (2012) A1079–A1109.
[16] Y. Notay, An aggregation-based algebraic multigrid method, Electron. Trans. Numer. Anal. 37 (2010) 123–146.
[17] Y. Notay, Aggregation-based algebraic multigrid for convection–diffusion equations, SIAM J. Sci. Comput. 34 (2012) 2288–2316.
[18] Y. Notay, Agmg software and documentation (2013). http://homepages.ulb.ac.be/~ynotay/agmg.