Solving Poisson's Equation with Different Boundary Conditions with MFEM.

Juan Sebastián Castaño Franco[1,a]

[a]jcastanof@unal.edu.co

Abstract

In this article we are going to talk about the Finite element method to solve partial differential equations, we show the general idea of this discretization method, and how can we approach the problem when it has different boundary conditions (Dirichlet, Neumann and Robin). Finally we show the power and how useful the MFEM library is when we want to solve a partial differential equation. We give some important tips to keep in mind when you are using this library.

Keywords: Finite element method, FEM, MFEM, Partial differential equations, PDEs, Poisson's equation.

1. Introduction

A lot of problems in physics, math, engineering are described by partial differential equations and almost all of them cannot be solved by analytical methods, for this reason numerical methods had to be implemented in order to get an approximation of the solution. To achieve this goal typically it can be use a discretization methods which can be solved using numerical methods. The return of all this process is an approximation to the solution of the PDE. The finite element method (FEM) is precissely the way to calculate such approximation. The goal of this document is give a summary of how FEM works, study a MFEM library through an example of Poisson's equation in certain domain with different boundary conditions.

Remark 1. MFEM is an open-source C++ library oriented to solve partial differential equations using the finite element method. All the documentation of MFEM can be find at [2].

2. Finite element method (FEM)

Let us introduce the finite element method in two dimensional case with a Dirichlet boundary condition. Let $\vec{F}(x)$ a vector-valued function defined in Ω to $\mathbb{R}^2$, we want to find a solution for the following equation:

$$\begin{aligned} -\operatorname{div}(\vec{F}(u)) &= f \text{ in } \Omega, \\ u &= g \text{ on } \partial\Omega. \end{aligned} \tag{1}$$

To achieve this, we are going to take a test function v in V the space of all test functions, which V depends on the problem what you are working for (generally it takes a space of polynomials). If we multiply v by both sides of the equations and integrate over the domain we get that,

$$\int_\Omega -\operatorname{div}(\vec{F}(u))v\,dx = \int_\Omega fv\,dx.$$

Now, if we use Integration by Parts and the Dirvergence Theorem we get the following,

$$\int_\Omega \operatorname{div}(v\vec{F})\,dx = \int_{\partial\Omega} v\vec{F}\cdot\hat{\eta}\,dS_x = \int_\Omega \nabla v\cdot\vec{F}\,dx + \int_\Omega v\operatorname{div}\vec{F}\,dx,$$

thus,

$$-\int_\Omega v \operatorname{div} \vec{F}\, dx = \int_\Omega \nabla v \cdot \vec{F}\, dx - \int_{\partial\Omega} v\vec{F} \cdot \hat{\eta}\, dS_x,$$

finally we have that,

$$\int_\Omega \nabla v \cdot \vec{F}\, dx - \int_{\partial\Omega} v\vec{F} \cdot \hat{\eta}\, dS_x = \int_\Omega f v\, dx. \tag{2}$$

If we suppose that $v(x) = 0$ on $\partial\Omega$, the problem reduce to

$$\int_\Omega \nabla v \cdot \vec{F}\, dx = \int_\Omega f v\, dx.$$

Using the Petrov-Galerking method, we choose a subspace $V_N \subset V$ and the idea is to see u as a linear combination of the elements of the basis of V_N (solution space) i.e,

$$u = \sum_{j=1}^N \alpha_j \phi_j,$$

then, we want to find this scalars α_j with $j = 1, 2, \ldots, N$. Therefore if we take M test functions we have that,

$$\int_\Omega \nabla\varphi_i \cdot \vec{F}\left(\sum_{j=1}^N \alpha_j\phi_j\right) dx = \int_\Omega f\varphi_i\, dx \quad i = 1, \ldots, M.$$

As $\vec{F}$ is a linear transformation we can do the following,

$$\sum_{j=1}^N \alpha_j \left(\int_\Omega \nabla\varphi_i \cdot \vec{F}(\phi_j)\, dx\right) = \int_\Omega f\varphi_i\, dx, \quad i = 1, \ldots, M. \tag{3}$$

We have built the discretization of the problem, where our linear system is given by

$$\sum_{j=1}^N a_{ij}\alpha_j = b_i, \text{ where } a_{ij} = \int_\Omega \nabla\varphi_i \cdot \vec{F}(\phi_j)\, dx \text{ and } b_i = \int_\Omega f\varphi_i\, dx.$$

In the case of the Poisson's equation it is enough to consider the function $\vec{F}(u) = \nabla u$ and our new problem is

$$\begin{aligned} -\operatorname{div}(\nabla u) &= f \text{ in } \Omega, \\ u &= g \text{ on } \partial\Omega. \end{aligned}$$

Precisely this equation will be the object of study in this article adding a coefficient $\rho(x)$ which can gives the characteristics of the material, conductivity or permeability, among others.

2.1. Boundary conditions.

To understand what we are going to do, we need to first study the concept of different boundary conditions; Dirichlet boundary condition, Neumann boundary condition, Robin boundary condition.

2.1.1. Dirichlet boundary condition

Given a problem of the partial differential equation, we say that the problem has a Dirichlet boundary condition if it knows the function on the boundary e.g the equation (1).

2.1.2. Neumann boundary condition

Given a problem of the partial differential equation, we say that the problem has a Neumann boundary condition if it knows the derivative of the function (spacial derivative) on the boundary. In several dimensions it knows the normal derivative on the boundary e.g

$$\begin{aligned}\Delta u &= 0 \text{ in } \Omega,\\ \partial_{\hat{n}} u &= f \text{ on } \partial\Omega.\end{aligned}$$

2.1.3. Robin boundary condition

Given a problem of the partial differential equation, we say that the problem has a Robin boundary condition if it knows a linear combination of a Dirichlet and Neumann boundary conditions e.g

$$\begin{aligned}\Delta u &= 0 \text{ in } \Omega,\\ a\partial_{\hat{n}} u + bu &= f \text{ on } \partial\Omega.\end{aligned}$$

2.2. Finite element method with different boundary conditions.

We are going to study the behavior of the finite element method when the problem has a inhomogeneous Neumann, Dirichlet and Robin boundary conditions in the Poisson's equation. The problem what we want to study is the following

$$\begin{aligned}-\operatorname{div}(\nabla u) &= f \text{ in } \Omega,\\ u &= g_1 \text{ on } \Gamma_1 \text{ (Dirichlet)},\\ \partial_{\hat{n}} u &= g_2 \text{ on } \Gamma_2 \text{ (Inhomogeneous Neumann)},\\ a\partial_{\hat{n}} u + bu &= g_3 \text{ on } \Gamma_3 \text{ (Robin) }.\end{aligned} \tag{4}$$

where, $\partial\Omega = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$. Doing the same what we did at the beginning of this section, we have te equation (2);

$$\int_\Omega \nabla v \cdot \nabla u \, dx - \int_{\partial\Omega} v \nabla u \cdot \hat{\eta} \, dS_x = \int_\Omega f v \, dx,$$

we can now divide the boundary integral as follow,

$$\int_\Omega \nabla v \cdot \nabla u \, dx - \left(\int_{\Gamma_1} v \nabla u \cdot \hat{\eta} \, dS_x + \int_{\Gamma_2} v \nabla u \cdot \hat{\eta} \, dS_x + \int_{\Gamma_3} v \nabla u \cdot \hat{\eta} \, dS_x \right) = \int_\Omega f v \, dx. \tag{5}$$

Let $u = u_1 + u_0$ where $u_0(x) = 0$ for all $x \in \partial\Omega - \Gamma_1$, $u_0(x) = g_1$ for $x \in \Gamma_1$ and $u_1(x) = 0$ on Γ_1. Replacing u in (5) we get that,

$$\begin{aligned}\int_\Omega \nabla v \cdot \nabla (u_1 + u_0) \, dx - \int_{\Gamma_1} v \nabla (u_1 + u_0) \cdot \hat{\eta} \, dS_x - \int_{\Gamma_2} v \nabla (u_1 + u_0) \cdot \hat{\eta} \, dS_x &\\ - \int_{\Gamma_3} v \nabla (u_1 + u_0) \cdot \hat{\eta} \, dS_x &= \int_\Omega f v \, dx.\end{aligned}$$

If we supposed that $v = 0$ on Γ_1 and applying the conditions over $u_1 + u_0$ we have that,

$$\begin{aligned}\int_\Omega \nabla v \cdot \nabla u_1 \, dx &= \int_\Omega f v \, dx - \int_\Omega \nabla v \cdot \nabla u_0 \, dx + \int_{\Gamma_2} v \nabla u_1 \cdot \hat{\eta} \, dS_x + \int_{\Gamma_3} v \nabla u_1 \cdot \hat{\eta} \, dS_x,\\ &= \int_\Omega f v \, dx - \int_\Omega \nabla v \cdot \nabla u_0 \, dx + \int_{\Gamma_2} v g_2 \, dS_x + \int_{\Gamma_3} v \frac{1}{a}(g_3 - b u_1) \, dS_x,\\ &= \int_\Omega f v \, dx - \int_\Omega \nabla v \cdot \nabla u_0 \, dx + \int_{\Gamma_2} v g_2 \, dS_x + \int_{\Gamma_3} \frac{1}{a} v g_3 \, dS_x - \int_{\Gamma_3} \frac{1}{a} b u_1 v \, dS_x.\end{aligned}$$

This can be rewritten as,

$$\int_\Omega \nabla v \cdot \nabla u_1 \, dx + \frac{b}{a}\int_{\Gamma_3} u_1 v \, dS_x = \int_\Omega f v \, dx - \int_\Omega \nabla v \cdot \nabla u_0 \, dx + \int_{\Gamma_2} v g_2 \, dS_x + \int_{\Gamma_3} \frac{1}{a} v g_3 \, dS_x,$$

finally u_0 can be written as a linear combination of the solution space on Γ_1, and u_1 as a linear combination of solution space i.e

$$u_0 = \sum_{\phi_l \in \Gamma_1} \alpha_l \phi_l, \quad u_1 = \sum_{j=1}^{N} \beta_j \phi_j.$$

Therefore, we obtain the discretization of the problem with different boundary conditions,

$$\sum_{j=1}^{N} \beta_j \int_\Omega \nabla\phi_j \nabla\varphi_i \, dx + \frac{b}{a}\sum_{j=1}^{N} \beta_j \int_{\Gamma_3} \phi_j \varphi_i \, dS_x = \int_\Omega f\varphi_i \, dx - \sum_{\phi_l \in \Gamma_1} \alpha_l \int_\Omega \nabla\phi_l \nabla\varphi_i \, dx + \int_{\Gamma_2} g_2 \varphi_i \, dS_x + \frac{1}{a}\int_{\Gamma_3} g_3 \varphi_i \, dS_x, \quad (6)$$

where $i = 1, \ldots, M$.

Now, as we know how FEM works, let us continue with the second part; solve a partial differential equation using MFEM library.

3. Poisson's equation with Dirichlet and Neumann Boundary conditions in MFEM

The main idea is to study the follow PDE,

$$\begin{aligned} -\operatorname{div}(\rho(x)\nabla u) &= f \text{ in } \Omega, \\ u &= g_1 \text{ on } \Gamma_1. \\ \partial_{\hat{n}} u &= g_2 \text{ on } \Gamma_2. \end{aligned} \quad (7)$$

We are going to work the Poisson's equation on a rectangle domain which has two holes removed as you can see in the Figure 1. The narrow ends of the mesh are connected to form a Periodic boundary condition. The upper edge Γ_2, lower edge Γ_1 and the right hole Γ_4 receives an inhomogeneous Neumann boundary condition. The left hole Γ_3 receives a Dirichlet boundary condition. So, what we want is to see how to put this on MFEM and verify that method converge to solution.

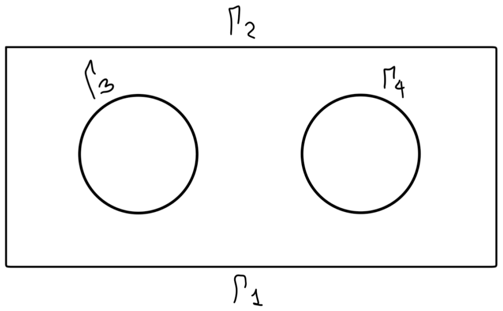


Figure 1: Domain of the function to study.

For this, take u as a periodic function on the x_1 coordinate this could be $\cos 2\pi x_1$ and for the x_2 coordinate we can take e^{x_2}, then $u = e^{x_2}\cos 2\pi x_1$. Let's take $\rho(x) = \begin{bmatrix} e^{-(x_1+x_2)} & 0 \\ 0 & e^{-(x_2+x_2)} \end{bmatrix}$, with this information we can find the function f in (7) we just have to calculate ∇u, multiply $\rho(x)\nabla u$ and do $-\operatorname{div}(\rho(x)\nabla u)$. The final result for f is,

$$f = 2\pi e^{-x_1}(2\pi \cos(2\pi x_1) - \sin(2\pi x_1)).$$

Now, we have to determine the boundary conditions as we want, for Γ_1 we consider de vector $\hat{n}_1^t = (0, -1)$, then,

$$\partial_{\hat{n}_1} u = \begin{bmatrix} \partial_{x_1} u \\ \partial_{x_2} u \end{bmatrix} \begin{bmatrix} 0 \\ -1 \end{bmatrix} = -e^{x_2}\cos(2\pi x_1) = g_1(x_1, x_2).$$

For Γ_2 we take $\hat{n}_1^t = (0, 1)$, and the boundary condition is,

$$\partial_{\hat{n}_2} u = \begin{bmatrix} \partial_{x_1} u \\ \partial_{x_2} u \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = e^{x_2}\cos(2\pi x_1) = g_2(x_1, x_2).$$

For Γ_3 we can take the same function u, then,

$$u = e^{x_2}\cos(2\pi x_1) = g_3(x_1, x_2).$$

To find the condition on Γ_4 we consider the function $t(x_1, x_2) = (x_1 - 1/2)^2 + x_2^2 - r^2$ and calculate $(\nabla t)^t = (2x_1 - 1, 2x_2)$ and we normalize dividing by the norm of the vector, doing this, we have that $\hat{n}_3^t = -(\sqrt{(2x_1 - 1)^2 + 4x_2^2})^{-1}(2x_1 - 1, 2x_2)$, thus,

$$\partial_{\hat{n}_3} u = -\begin{bmatrix} \partial_{x_1} u \\ \partial_{x_2} u \end{bmatrix} \begin{bmatrix} \frac{2x_1 - 1}{\sqrt{(2x_1-1)^2 + 4x_2^2}} \\ \frac{2x_2}{\sqrt{(2x_1-1)^2 + 4x_2^2}} \end{bmatrix} = \frac{e^{x_2}[2\pi(2x_1 - 1)\sin(2\pi x_1) - 2x_2\cos(2\pi x_1)]}{\sqrt{(2x_1 - 1)^2 + 4x_2^2}} = g_4(x_1, x_2).$$

Finally, what we want is to solve the following PDE,

$$\begin{aligned} -\operatorname{div}(\rho(x)\nabla u) &= 2\pi e^{-x_1}(2\pi\cos(2\pi x_1) - \sin(2\pi x_1)) \text{ in } \Omega, \\ \partial_{\hat{n}_1} u &= -e^{x_2}\cos(2\pi x_1) = g_1(x_1, x_2) \text{ on } \Gamma_1, \\ \partial_{\hat{n}_2} u &= e^{x_2}\cos(2\pi x_1) = g_2(x_1, x_2) \text{ on } \Gamma_2, \\ u &= e^{x_2}\cos(2\pi x_1) \text{ on } \Gamma_3, \\ \partial_{\hat{n}_3} u &= \frac{e^{x_2}[2\pi(2x_1 - 1)\sin(2\pi x_1) - 2x_2\cos(2\pi x_1)]}{\sqrt{(2x_1 - 1)^2 + 4x_2^2}} = g_4(x_1, x_2) \text{ on } \Gamma_4. \end{aligned} \tag{8}$$

Now, the idea is use MFEM to verify that effectively the solution is the u function. We are going to talk about the essential things which you need to know to do the FEM method with different boundary conditions and adding a coefficient in MFEM. Let's skip how initialize, how to do the parse command, how construct the mesh and how to define the parallel finite element space on the parallel mesh, because all of this you can find it at the MFEM examples [5]. Nevertheless, you can find the entire code in [4].

3.1. Define the boundary conditions.

To define the boundary conditions we need to use the"marker arrays" where we can identify each of the boundary conditions. These arrays have an entry corresponding to each boundary attribute. If you put 1 in entry i marks attribute $i + 1$ is being active and if you put 0, is inactive. A simple examples is to thing in our problem, we want an inhomogeneuous boundary condition in Γ_1 thus, the correct form to activate this is writting the next code.

```
Array<int> nbc_bdr(pmesh.bdr_attributes.Max());
nbc_bdr = 0; nbc_bdr[0] = 1;
```

With this, we activate the condition over Γ_1.

3.2. Set up the various coefficients and add ρ coefficient.

The next step is set up the various coefficients needed for the Poisson's operator and the all boundary conditions. To add the ρ coefficient, first, we need to programm the function what we want to put, then use `FunctionCoefficient` and finally turn it in to a grid function as follows.

```
FunctionCoefficient p(funCoef);
ParGridFunction rho1(&fespace);
rho1.ProjectCoefficient(p);
GridFunctionCoefficient rho(&rho1);
```

Here, `funCoef` is the function that determines ρ in our case this function is $e^{-(x_1+x_2)}$.. As we mentioned, we need to convert the boundary conditions, and f in the Poisson's equation in to `FunctionCoefficient`, this can be done like this,

```
FunctionCoefficient g(gbc);
FunctionCoefficient f_an(fanalytic);
```

where `gbc` is the function imposed at the boundary condition (it can be Dirichlet, Neumann, Robin), and `fanalytic` is the result f of Poisson's problem.

3.3. Incorporate a coefficient on the Neumann boundary condition.

Now, we need to incorporate the $\rho(x)$ coefficient on the Neumann boundary conditions, since as wee see, $\partial_{\hat{n}} u$ terms arise by integrating $-\operatorname{div}(\rho(x)\nabla u)$ by parts. Therefore, what we have to do is enforce $\rho(x)$ i.e, we have that the new Neumann boundary conditions are,

$$\rho(x)\partial_{\hat{n}} u = \rho(x) g_i \text{ with } i = 1, 2, 4.$$

This can be done in MFEM using `ProductCoefficient`, e.g,

```
 ProductCoefficient m_nbcCoef(g, rho);
```

At the problem what we are studying there are three parts of the boundary which have Neumann boundary conditions, then in our case we have to do three products. Now we need to define the solution vector u as a parallel finite element grid function corresponding to the finite element space (fespace), we just need to use `ParGridFunction` and for simplicity we can initialize u as zero.

```
ParGridFunction u(&fespace);
u = 0.0;
```

3.4. Set up the FEM method.

Now set up the left side of the equation (6) (it can be see as a bilinear form) on the finite element space, by adding the diffusion domain itegrator, this is done with `ParBilinearForm, BilinearFormIntegrator, DiffusionIntegrator` as follows:

```
ParBilinearForm a(&fespace);
BilinearFormIntegrator *integ = new DiffusionIntegrator(rho);
a.AddDomainIntegrator(integ);
a.Assemble();
```

To finish the part of the FEM method we must to calculate the right side of the equation (6) considering all the boundary conditions, so we create the vector b that allow us to save the result of compute all the integrals described above, and then we calculate all of them with MFEM help. That can be done with `DomainLFIntegrator, BoundaryLFIntegrator` as follows,

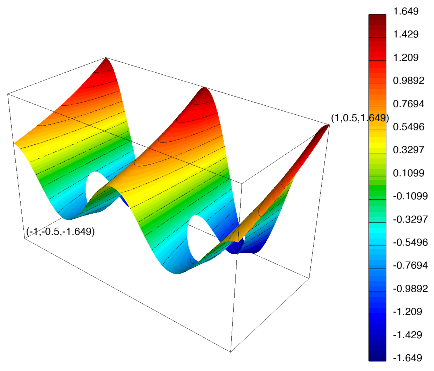
```
ParLinearForm b(&fespace);
u.ProjectBdrCoefficient(g3,dbc_bdr3);
b.AddDomainIntegrator(new DomainLFIntegrator(f_an));
b.AddBoundaryIntegrator(new BoundaryLFIntegrator(m_nbcCoef),nbc_bdr);
b.Assemble();
```

3.5. Construct the linear system and solve the system.

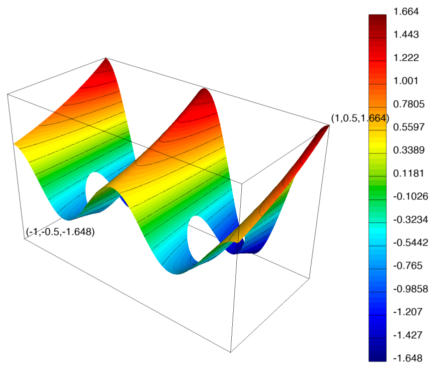
The next step is to construct the linear system and work it out. You can see how to do this in [4] or [5].

3.6. Plots of the exact solution and solution calculate with MFEM with GLVIS.

Now we can see the exact solution and the solution given by MFEM of the problem with GLVIS.



(a) Exact Solution.



(b) MFEM Solution.

Figure 2: (a) is the exact solution of the function u at the domain and (b) is the solution calculated by MFEM library with the FEM method.

We can compute the norms error in H^1, to do this, we need to write the exact solution as `FunctionCoefficient` and compute the gradient of u as `VectorFunctionCoefficient`, after that we write the next code,

```
double h1_err_prev = 0.0;
double h_prev = 0.0;
double h1_err = u.ComputeH1Error(&u1,&u_grad,&p,1.0,1.0);
mfem::out <<"Calculate of the error: "
          << h1_err << endl;
```

When we run the code, we get a 0.0683466 difference between the norms and if we refine the mesh, this value reduces to the half, to be exact the result is 0.0341603, this tell us that the code is going well.

4. Conclusions

1. In order to use this tool you need to know and understand very well the concepts of the Finite element method and other things.
2. With this project I could see the powerful that MFEM has when you want to get a solution of a PDE numerically.
3. Learn how to use MFEM is a little complicate, I think that the library needs more documentation on the web page, nevertheless if you search deep, maybe you can find what you want to do. One think useful is to watch a github repositories, there many people ask how to do many things so I think it could be helpful.

References

[1] Claes Johnson. *Numerical solution of partial differential equations by the finite ele- ment method.* Courier Corporation, 2012.

[2] https://mfem.org/

[3] Robert Anderson[1], Andrew Barker[1], Jamie Bramwell[1] , Jakub Cerveny[2] , Johann Dahm[3] , Veselin Dobrev[1] , Yohann Dudouit[1] , Aaron Fisher[1] , Tzanio Kolev[1] , Mark Stowell[1] , and Vladimir Tomov[1]. *MFEM: A Modular Finite Element Method Library.* [1] Lawrence Livermore National Laboratory, [2] University of West Bohemia. [3]IBM Research, July 2, 2018.

[4] `https://github.com/Juan051099/Beyond-Research-project.git`

[5] `https://mfem.org/examples/`