A general purpose library for solving multiphysics problems

Gustavo P. Exel, Hermínio T. Honório, Clovis, R. Maliska

Dept. of Mechanical Engineering, Federal University of Santa Catarina
R. Eng. Agronômico Andrei Cristian Ferreira, s/n - Trindade, Florianópolis, 88040-900, Santa Catarina, Brazil
gustavoexelgpe@gmail.com, herminio.eng@gmail.com, clovis.maliska@gmail.com

Abstract. This article presents a numerical library specifically developed for solving partial differential equations (PDEs) using the Element-based Finite Volume Method (EbFVM). The library supports two and three-dimensional unstructured grids composed of different types of elements. The user is responsible for providing the mesh file, boundary and initial conditions, the physical properties and typing the model equations in a math-like text format. The library is responsible for integrating the equations in time and space, obtaining the set of algebraic equations, assembling the linear system and running the simulation. In this manner, the user is not required to possess any previous knowledge of the underlying numerical scheme, i.e. the EbFVM. Another appealing feature is that coupled multiphysics problems can be easily configured through a user-friendly interface.

For demonstration purposes, we present three well known test cases. In the first one we solve a simple heat conduction problem with the main goal of introducing the basic guidelines of the library. Next, we present the procedure for solving a linear elasticity problem, in which the three displacement components are the unknowns. Finally, a coupled consolidation problem modelled by Biot's [1] equations is presented. In this case, the unknowns are pressure and displacements, which is easily handled by the library. We remark that, to the best of our knowledge, there is no general purpose and open source code available on the internet that uses the EbFVM as a numerical scheme.

Keywords: Element-based Finite Volume Method, Numerical Methods, Multiphysics, Physics Coupling

1 Introduction

Mathematical formulations for physical problems are often expressed in the form of partial differential equations (PDEs). This happens because the formulations relate physical properties at neighboring points in time and space. Examples of equations expressed by PDEs are thermal (diffusion, heat conduction, fluid mechanics), structural, electromagnetic and wave phenomena.

In addition to the examples mentioned above for PDE applications, in different situations, it is important to couple different physics, for example: thermal expansion (couples thermal conduction equations with mechanical equilibrium equations), geomechanics (couples fluid mechanics equations with mechanical equilibrium equations) and the simulation of an electric motor (which can couple equations of electromagnetism, solid mechanics and heat conduction).

However, the analytical solutions to these problems are limited to simple problems, and are not found for more complex configurations, and therefore there is a need for numerical methods to solve these PDEs. Among the most common formulations are the finite volume method, the finite difference method, and the finite element method, and all their respective variants.

There are already consolidated software for numerical simulation, among which are some commercial software such as Ansys and FlexSim, as well as open source simulation software such as OpenFOAM, PorePy, FEniCS, DUNE. In the programs listed previously (with the exception of FEniCS and OpenFOAM) the user must choose a previously implemented model that best suits the desired situation, and enter the settings of their problem to solve it. The proposal in this paper is similar to the FEniCS and OpenFOAM approach, which consists of receiving the user's differential equations instead of choosing a previously implemented model.

The proposed library, called bellbird, is written in Python (known for its ease of use), and uses the Element-

based Finite Volume Method, which unlike the finite element and finite difference method guarantees conservation laws even for less refined meshes (by integrating the equations in the conservative form). The user must inform the system of differential equations to be solved, the initial and boundary conditions, the properties of the medium and the mesh, and the library generates a solver for the informed problem, which can be then modified by the user as needed.

In section 2 of this article, the characteristics of the numerical method will be presented, in section 3 it will be explained how the library is used, in section 4 some numerical examples will be solved, and in section 5 the conclusions will be presented.

2 Element-based Finite Volume Method

The Element-based Finite Volume method (see Maliska [2] and Honório [3]) uses a cell vertex approach (see Fig. 1), which means that the vertices (which store the variables) are at the center of the control volumes. Furthermore, the EbFVM also uses shape functions to approximate the field values and derivatives (just like the finite element method, hence the "element" similarity in the naming). Another signature of the method is integrating the differential equations in the conservative form, and using Gauss's divergence theorem on the appropriate terms.

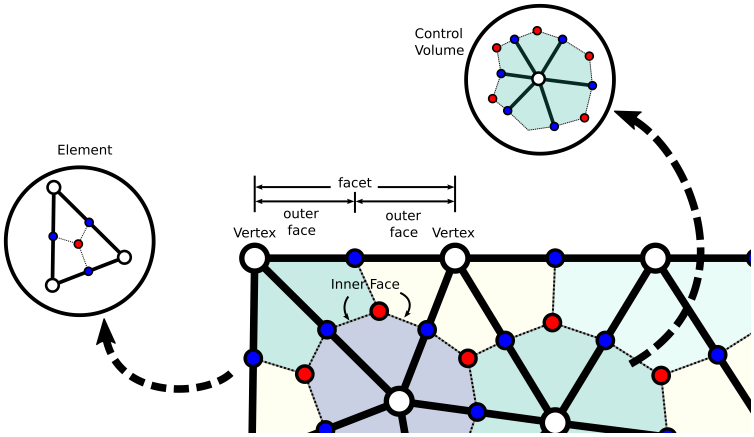

Figure 1. Geometrical Entities used in the EbFVM

3 Bellbird

The method used to discretize and solve the differential equations, while simple, can become quite extensive as we increase the complexity of the PDE system. The chance of implementation errors can increase, and implementation becomes rather unfriendly. As it is a very methodical process, it is possible to automate the entire step of integration and discretization of the differential equations, as well as their numerical implementation in code. For that, from the differential equations, the library creates a solver for the problem, informing the initial and boundary conditions, properties of the medium and mesh, solves the problem and exports the results. An example of usage will be shown below.

Consider the problem of heat conduction through a solid body modelled by the heat equation:

$$\nabla \cdot (k \nabla T) + q''' - \rho c_p \frac{\partial T}{\partial t} = 0. \tag{1}$$

Using the initial condition $T(x, y, 0) = 300K$, the boundary conditions $T(0, y, t) = 350K$, $T(1, y, t) = 400K$ and $\frac{\partial T(x,0,t)}{\partial y} = \frac{\partial T(x,1,t)}{\partial y} = 0$ in a 2D $1m \times 1m$ plate, and $\rho = 8970.0kg/m^3$, $c_p = 377.0J/kg \cdot K$, $k = 22.0W/m \cdot K$, $q = 0.0W/m^3$ the problem would be solved as such:

```
import bellbird

model = bellbird.Model(
  name = "Heat Transfer",
  equationsStr = ["k * div( grad(T) ) + q = rho * cp * d/dt(T)"],
  variables   = ["T"], # temperature [K]
  properties = {
    "Body":{
      "k"   : 22.0,   # [W/m.K]  - conductivity
      "rho" : 8960.0, # [kg/m3]  - density
      "cp"  : 377.0,  # [J/kg.K] - specific heat
      "q"   : 0.0,    # [W/m3]   - heat generation
    },
  },
  boundaryConditions = [
    bellbird.InitialCondition("T", 300.0),
    bellbird.BoundaryCondition("T", bellbird.Dirichlet, "West", 350.0),
    bellbird.BoundaryCondition("T", bellbird.Dirichlet, "East", 400.0),
    bellbird.BoundaryCondition("T", bellbird.Neumann, "South", 0.0),
    bellbird.BoundaryCondition("T", bellbird.Neumann, "North", 0.0),
  ],
  meshPath = "mesh.msh",
  timeStep = 1000.0,
  tolerance = 1e-4,
  sparse = False,
)

model.run()
```

The mesh is generated by Gmsh, and the results can be read using Paraview

4 Numerical Examples

Now we're going to solve some problems to show the ease of use of the library.

4.1 Beam Load

Consider a solid beam fixed in the ground, and with a load applied to its end in the downwards direction. We can model the phenomenon from the elastic equation:

$$\nabla \cdot \left[\mathbb{C} \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T \right) \right] + \rho \mathbf{g} = \mathbf{0}, \tag{2}$$

with $\mathbb{C}$ being the constitutive tensor, $\mathbf{u}$ the displacement vector, ρ the beam density and $\mathbf{g}$ the gravity vector.

Where $\sigma_L = 10kN$ is the load applied, and the dimensions of the beam are $L = 1m$, $h = 0.1m$ and $a = 0.1m$. Bellow the implementation of this problem using bellbird is shown.

```
import bellbird

model = bellbird.Model(
    name = "Beam Load",
    equationsStr = [
        "div(Ce * grad_s(u)) + rho * g = vec(0)",
    ],
    variables   = ["u_x", "u_y"],
    definitions = [
        "g = np.array([0.0, -9.81])",
        "lame_p = 2*G*nu/(1-2*nu)",
        "Ce = np.array([[2*G+lame_p, lame_p, 0], [lame_p, 2*G+lame_p, 0], [0, 0, G]])"
    ],
    properties = {
        "Body":{
            "rho": 1800.0,
            "nu": 0.4,
            "G": 6.0e+06,
```

```
        },
    },
    boundaryConditions = [
        bellbird.InitialCondition("u_x", 0.0),
        bellbird.BoundaryCondition("u_x", bellbird.Dirichlet, "West", 0.0),
        bellbird.BoundaryCondition("u_x", bellbird.Dirichlet, "East", 0.0),
        bellbird.BoundaryCondition("u_x", bellbird.Neumann, "South", 0.0),
        bellbird.BoundaryCondition("u_x", bellbird.Neumann, "North", 0.0),
        bellbird.InitialCondition("u_y", 0.0),
        bellbird.BoundaryCondition("u_y", bellbird.Neumann, "West", 0.0),
        bellbird.BoundaryCondition("u_y", bellbird.Neumann, "East", 0.0),
        bellbird.BoundaryCondition("u_y", bellbird.Dirichlet, "South", 0.0),
        bellbird.BoundaryCondition("u_y", bellbird.Neumann, "North", 1e4),
    ],
    meshPath = "mesh.msh",
)

model.run()
```

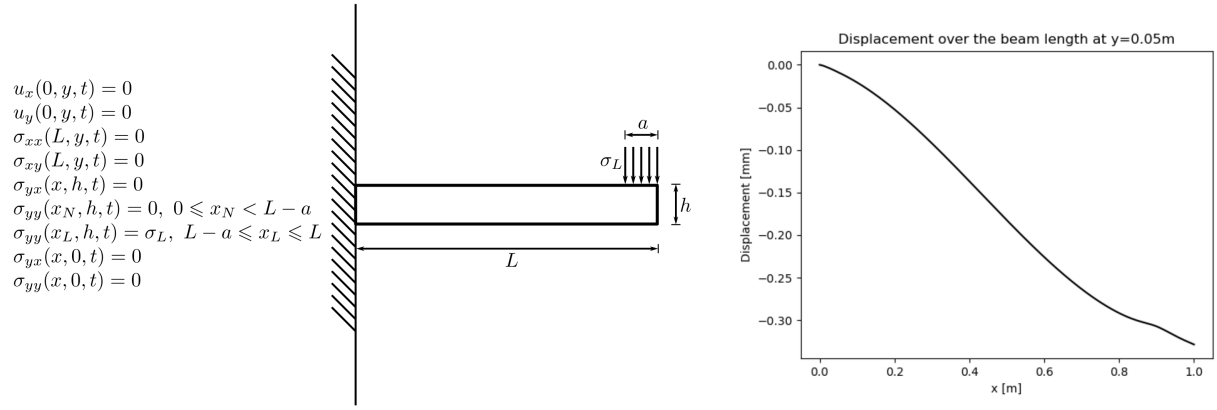

Figure 2. Beam Load Example

4.2 Cryer Sphere

The next problem consists of a consolidation problem for a spherical soil sample, proposed by Cryer in 1963, and serves as a good benchmark in the geomechanical simulation scenario. The mass conservation equation (for solid and fluid) and Newton's second law are coupled through Darcy's law and Terzaghi's effective stress principle. The equations are shown below:

$$\frac{1}{M}\frac{\partial p}{\partial t} + \alpha\frac{\partial}{\partial t}(\nabla\cdot\mathbf{u}) + \nabla\cdot\left[\frac{\mathbf{K}}{\mu}(\rho\mathbf{g} - \nabla p)\right] = 0. \tag{3}$$

Where M is the Biot modulus, p is the pore pressure, t is time, α is the Biot's coefficient, $\mathbf{u}$ is the displacement vector, $\mathbf{K}$ is the permeability tensor, μ is the fluid's viscosity, ρ is the density of the medium and $\mathbf{g}$ is the gravity vector.

$$\nabla\cdot\left[\mathbb{C}\left(\nabla\mathbf{u} + \nabla\mathbf{u}^T\right) + \alpha p\mathbb{I}\right] + \rho\mathbf{g} = 0, \tag{4}$$

with $\mathbb{C}$ being the constitutive tensor, and $\mathbb{I}$ the identity tensor.

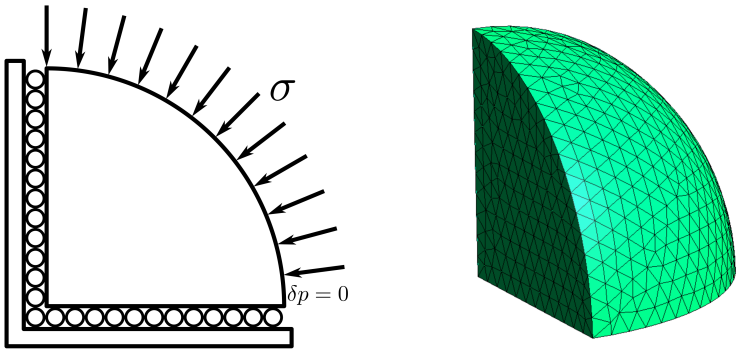

Figure 3. Cryer Sphere Problem

The way this problem is implemented with bellbird is shown below.

```
import bellbird

model = bellbird.Model(
    name = "Cryer Sphere",
    equationsStr = [
        "(1/M) * d/dt(p) + alpha*d/dt(div(u)) + div( (k/mu) * ( rho*g - grad(p) ) ) = 0",
        "div( Ce*grad_s(u) ) - grad(alpha*p) + rho * g = vec(0)",
    ],
    variables   = ["u_x", "u_y", "u_z", "p"],
    definitions = [
        "g = np.array([0.0, 0.0, 0.0])",
        "lame = 2*G*nu/(1-2*nu)",
        "Ce = np.array([[2*G+lame, lame, lame, 0.0, 0.0, 0.0], [lame, 2*G+lame, lame, 0.0,
    0.0, 0.0], [lame, lame, 2*G+lame, 0.0, 0.0, 0.0], [0.0, 0.0, 0.0, G, 0.0, 0.0], [0.0, 0.0,
     0.0, 0.0, G, 0.0], [0.0, 0.0, 0.0, 0.0, 0.0, G]])",

        "rho = phi * rhof + (1-phi) * rhos",     # medium density
        "K = 2*G*(1 + nu) / 3*(1-2*nu)",         # Bulk modulus
        "cb = 1 / K",                            # Bulk compressibility
        "alpha = 1 - cs / cb",                   # Biot coefficient
        "M = 1/(phi * cf + (alpha-phi) * cs)",   # Biot modulus
    ],
    properties = {
        "Body":{
            "nu": 2.0e-1,      # Poisson's ratio  (Soil)
            "G": 6.0e+9,       # Shear modulus    (Soil)
            "cs": 0.0,         # Compressibility  (Soil)
            "phi": 1.9e-1,     # Porosity         (Soil)
            "k": 1.9e-15,      # Permeability     (Soil)
            "rhos": 2.7e+3,    # Density          (Soil)
            "cf": 3.0303e-10,  # Compressibility  (Fluid)
            "mu": 1/1.0e-03,   # Viscosity        (Fluid)
            "rhof": 1.0e+3,    # Density          (Fluid)
        },
    },
    # ...
    # Define boundary conditions, the mesh path, the time step, and the tolerance
    # Zero displacement prescribed in the normal direction to the flat surfaces of
    # the eight of the sphere. Zero tension in the other faces.
    # Zero pressure gradient in all of the faces.
```

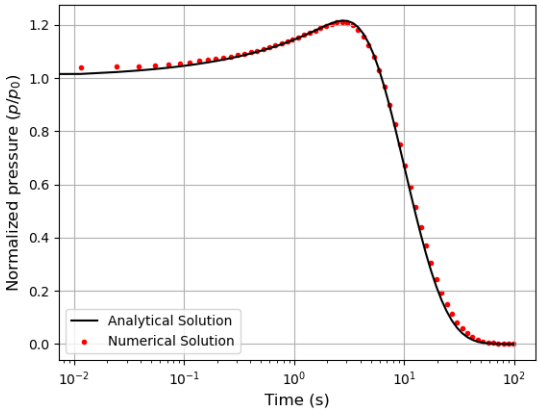

Figure 4. Internal pressure over time

5 Conclusions

The presented approach allows an easier implementation of the element-based Finite Volume Method, which had not been found in the literature. The computational tools developed help not only in the implementation of terms from different differential equations, but also in the effortless coupling of different physics. The library still has some disadvantages, such as the need to prepare it for the implementation of new terms. The Navier-Stokes equations, for example, cannot yet be implemented using the library, as there are terms in the equations that need to be studied and implemented, but the implementation can be done, thus allowing the formulation of equations involving advective terms.

Another improvement that can be made in the library is to allow the implementation of non-linear equations through iterative methods. Currently, only linear equations are easily implemented, but otherwise the library generates a user-accessible script and easily implements an iterative loop that can solve non-linear equations using iterative methods.

In general, Bellbird allows the introduction of engineers to the finite volume method, which can serve as a basis for other studies, as it was already being used by other fellows in the SINMEC laboratory.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] M. A. Biot. General theory of three-dimensional consolidation. *Journal of applied physics*, vol. 12, n. 2, pp. 155–164, 1941.

[2] C. Maliska. Transferência de calor e mecânica dos fluidos computacional. 2ª. *Edição. Rio de Janeiro: Livros Técnicos e Científicos Editora*, vol. 200, 2004.

[3] H. T. Honório and others. A stabilization technique for treating numerical instabilities in three-dimensional poroelasticity, 2018.