BioDeg

Biodegradation and Corrosion Simulation using Finite Element

User Manual

Version 0.8

(generated March 22, 2022)

Mojtaba Barzegari
Liesbet Geris

BioDeg website

KU LEUVEN

Contents

1 Introduction

BioDeg is an open-source software written in FreeFEM (a domain-specific language for finite element programming), C++, and Python for modeling the degradation of metallic biomaterials and simulating the biodegradation behavior of medical devices, implants, and scaffolds in corrosion experiments. It can handle any geometry of desire and supports parallel computing to simulate large-scale models.

1.1 Authors

BioDeg is developed by the Biomechanics Research group at KU Leuven and University of Liege. The code is currently maintained by its principal developer, who manages the development of the mathematical models and the core functionalities.

Principal developer

- Mojtaba Barzegari (University of Leuven, Belgium)

Previous Contributors

- Yann Guyot (University of Liege, Belgium)
- Piotr Bajger (University of Oxford, UK)

Mentor

- Liesbet Geris (University of Leuven, Belgium)

Chemist contributors

(who has helped to validate the models)

- Sviatlana V. Lamaka (Helmholtz-Zentrum Hereon, Gremany)
- Di Mei (Zhengzhou University, China)
- Cheng Wang (Helmholtz-Zentrum Hereon, Gremany)

1.2 Acknowledgments

The development of BioDeg open-source code is financially supported by the Prosperos project, funded by the Interreg VA Flanders – The Netherlands program, CCI grant no. 2014TC16RFCB046 and by the Fund for Scientific Research Flanders (FWO), grant G085018N. The developers also acknowledge support from the European Research Council under the European Union's Horizon 2020 research and innovation programmen, ERC CoG 772418.

1.3 Referencing BioDeg

Please refer to BioDeg repository, section "Publications and referencing" to properly cite the use of BioDeg in your scientific work.

2 Useful background information

You may refer to the following articles for a background of the methods and algorithms implemented in BioDeg:

1. M. Barzegari, D. Mei, S.V. Lamaka, L. Geris, Computational modeling of degradation process of biodegradable magnesium biomaterials, *Corrosion Science*, 190, 109674, 2021.
2. M. Barzegari, L. Geris, Highly scalable numerical simulation of coupled reaction–diffusion systems with moving interfaces, *The International Journal of High Performance Computing Applications*, 2021.

In addition, below are some useful references on the finite element method and some online resources that provide a background of finite elements and their application to the solution of partial differential equations.

1. H.P. Langtangen, K.A. Mardal, Introduction to Numerical Methods for Variational Problems, Springer, 2019: a nice book to understand variational formulation required for finite element computations.
2. An introduction to applied numerical computing: A set of Jupyter notebooks on various aspects of numerical methods, including the notebooks covering the topics of the book mentioned above.
3. YouTube video series discussing different numerical methods concepts including the finite difference and finite element methods.

3 Installation

Installing BioDeg is a straightforward procedure. You need to install a couple of prerequisites and download and run BioDeg; that's all you need to do. But for advanced users, it might be necessary or more interesting to build everything from scratch to have more control over customizing features and improving performance. As a result, we have provided 2 sets of instructions, one for easy installation using the compiled binaries and one for building things from the source codes. The installation instructions are provided for Linux and Windows operating systems, but the procedure should be very similar for macOS.

It is possible to use BioDeg without the user interface (UI) if this scenario is required by the user (like for running it on a super-computer). The core of BioDeg is written in FreeFEM, so in this case, all you need to do is to install/build FreeFEM and the required libraries, and then, clone the BioDeg core repository and run the code according to the provided instruction in the README file of the repository.

The BioDeg UI contains all the bundles for pre-processing and post-processing simulation input/results. These features are being hosted on their own repositories, but with obtaining the user interface, you can have them all together. If you choose to use BioDeg without the user interface and still want to use the provided script for pre/post-processing, you may need to obtain them separately.

For building the source codes, we assume standard tools and libraries like CMake, compilers (for C, C++ and Fortran), and MPI libraries are pre-installed on your machine. If you are going to build BioDeg and required dependencies on a super-computer or cluster, you should notice that most high-performance computers would have the latest version of these compilers and libraries in the default environment.

3.1 Easy installation

The simplest way to install and run BioDeg is via the pre-built binaries you can download from GitHub. The same principle applies to prerequisites, which is in this case FreeFEM only. So, following these steps will install BioDeg on your machine:

1. Install an MPI implementation on your system, which has different installation instructions depending on the operating system and the implementation you choose to use. In Windows, the simplest option is the open-source Microsoft MPI, which can be downloaded and installed from its official website. In Linux, the best options are OpenMPI and MPICH, which can be installed by executing `sudo apt-get install openmpi-bin` or `sudo apt-get install mpich` in terminal (in Ubuntu).

2. Download FreeFEM installer for the platform you use and install it. You can find the .exe installer for Windows and the .deb installer for Linux (Ubuntu) in the Release page of FreeFEM repository. You will find these files under the Assets section of the latest (or any other) version. Execute the download file and follow the installation procedure appearing on your screen. More information can be found here for various platforms.

3. Download BioDeg tarballs for your preferred platform (Windows or Linux) from the Release page of BioDeg repository. This is indeed the BioDeg UI bundle that contains the BioDeg core, the user interface, the pre-processor, and the post-processor.

4. Extract the downloaded tarball (zip) file and execute `runBioDeg.cmd` in Windows or `runBioDeg.sh` in Linux. By doing this, you see the BioDeg interface showing up on the screen.

5. For visualizing the results, ParaView should be installed on your system, which can be downloaded and installed from the official website for different platforms. Moreover, Python is required for using the basic postprocessing templates provided by the BioDeg-UI, so it should be separately installed if one is interested to use that feature.

3.2 Advanced installation for improved performance/flexibility

Building BioDeg and required libraries from source code will increase the performance since the program and the libraries will get optimized for the platform in which they are going to run. Moreover, this enables users to customize the software in the way they want. Additionally, this is an inevitable aspect if you are going to use BioDeg for development purposes or you want to contribute to it.

3.2.1 Compiling and installing external libraries

3.2.1.1 PETSc and Qt

BioDeg uses PETSc for parallel computing. You may choose to build a customized version of PETSc or use the version that comes with FreeFEM. The version that is bundled with FreeFEM has the following build configuration:

```
--with-debugging=0 COPTFLAGS="-O3 -mtune=native" CXXOPTFLAGS="-O3 -mtune=native"
```

```
FOPTFLAGS="-O3 -mtune=native" --with-cxx-dialect=C++11 --with-ssl=0
--with-x=0 --with-fortran-bindings=0 --with-scalar-type=complex
--with-blaslapack-include= --with-blaslapack-lib="-llapack -lblas"
--with-scalapack --with-metis --with-ptscotch --with-suitesparse --with-suitesparse-lib=
"-Wl, -lumfpack -lklu -lcholmod -lbtf -lccolamd -lcolamd -lcamd -lamd -lsuitesparseconfig"
--with-mumps --with-parmetis --with-tetgen --download-slepc --download-hpddm PETSC_ARCH=fc
```

You may need to build your own version if this configuration is not suitable for you. You can find the instruction for building custom version of PETSc here.

BioDeg UI is developed using Qt, so it should be installed on your system if you want to compile BioDeg UI. You can find the installation instruction for various platforms here.

3.2.1.2 FreeFEM

The full build documentation of FreeFEM is available here, but the following steps is what you need to do to build it on any platform. By default, FreeFEM downloads and builds PETSc during the build process.

1. Install required prerequisites

   ```
   $ sudo apt-get install cpp freeglut3-dev g++ gcc gfortran m4 make patch pkg-config
     wget python unzip liblapack-dev libhdf5-dev libgsl-dev autoconf automake
     autotools-dev bison flex gdb git cmake
   $ sudo apt-get install mpich
   ```

2. Make a new directory for FreeFEM and navigate to it

   ```
   $ cd
   $ mkdir FreeFEM
   $ cd FreeFEM/
   ```

3. Clone the source code repository and navigate to the downloaded directory

   ```
   $ git clone https://github.com/FreeFem/FreeFem-sources.git
   $ cd FreeFem-sources/
   ```

4. Generate the configure scripts

   ```
   $ autoreconf -i
   ```

5. Run the configure script to specify build options, including the location to install the program

   ```
   $ ./configure --enable-download --enable-optim
     --prefix=/home/<your_profile>/FreeFEM/freefem-install
   ```

6. Download the source code of 3rd-party libraries

```
$ ./3rdparty/getall -a
```

7. Build PETSc and all the 3rd-party libraries

```
$ cd 3rdparty/ff-petsc/
$ make petsc-slepc
```

8. Navigate back and reconfigure the build

```
$ cd -
$ ./reconfigure
```

9. Build the source code of FreeFEM using 4 parallel processes (or any other number you like

```
$ make -j4
```

10. Check the build by running some examples

```
$ make -j2 check
```

11. Install the built binaries to the specified directory

```
$ make install
```

12. Navigate to the installation location and run FreeFEM

```
$ cd ../freefem-install/
$ cd bin/
$ ./FreeFem++
```

13. Navigate to the home directory and add FreeFEM to the PATH variable in the `.bashrc‘` file

```
$ cd
$ nano .bashrc
```

and add `export PATH=$PATH:/home/<your_profile>/FreeFEM/freefem-install/bin` to the end of the file and save (press *Ctrl+X* and then *Y*).

After doing this, you should be able to run FreeFEM. Start a new terminal and run `FreeFem++` and `FreeFem++-mpi`. See ing no error in the output means that you have successfully installed it.

3.2.2 Building and installing BioDeg

Since BioDeg UI is developed using Qt, compiling the source files is quite straightforward. Upon installing Qt on your machine, clone the BioDeg UI repository and follow one of the following scenarios to build it.

3.2.2.1 Build BioDeg UI using Qt Creator IDE

This is the simplest technique to build the program, and it has a similar procedure for all the supported platforms. Qt Creator is the default IDE for Qt development, so it is automatically installed along with Qt. Simply open the Qt project file (`CMakeLists.txt`) in Qt Creator (by executing `qtcreator CMakeLists.txt` or selecting *File->Open Project* from the IDE) and build the project (by pressing *Shift+B*).

3.2.2.2 Build BioDeg UI using Qt tools

Building the source files using CMake is also quite simple. Navigate to the source files directory (the cloned repository) and run the following commands (this assumes that you have already added Qt `bin` directory to the `PATH` variable so that the CMake script can find Qt libraries and binaries):

```
$ mkdir build
$ cd build
$ cmake ..
$ make
```

In Windows, you may need to call the correct build system installed along with Qt. For example, by assuming that you have installed the MinGW integration for Qt, the `make` command should be written as `mingw32-make`. Moreover, in this case, you need to call CMake with a suitable generator, so the `cmake` command should be replaced by something like `cmake -G "MinGW Makefiles" ..` (don't forget to insert the double dots).

After doing this, you can find the BioDeg UI executable in the build (or bin) directory and run it by executing `./BioDeg-UI` in Linux or `.\BioDeg-UI.exe` in Windows.

3.2.3 Testing BioDeg

The testing module of BioDeg is developed using Google Testing framework. The tests cover some basic and advanced evaluation of the functionality of FreeFEM installation and BioDeg-core, so running them may take several minutes to complete.

After building BioDeg using the process mentioned in Section 3.2.2, you can run the tests simply by executing the following command (in the build directory):

```
$ make test
```

which can be `mingw32-make test` in Windows. You can review the results upon completion of all the tests. In a healthy installation of FreeFEM and BioDeg, all tests should pass successfully.

4 Running BioDeg

After installing/compiling BioDeg as described in Section 3, we are ready to run it. There are 2 ways to run BioDeg simulations: 1) using the UI to configure and execute BioDeg, or 2) running BioDeg directly from the command line and providing simulation parameters via command-line arguments. Moreover, in a hybrid approach, the UI can be used to configure and generate the command for method #2, which can

be useful when you want to configure the simulation only and run it later in another environment like on a super-computer or cluster.

The UI can be run simply by double-clicking on the `BioDeg-UI.exe` in Windows or by executing `./BioDeg-UI` in Linux. For running BioDeg directly, one need to execute the following command:

```
$ mpirun -n N FreeFem++-mpi core/src/main.edp <command line args>
```

in which N defines the number of MPI processes to be used. The full list of command-line arguments can be found in Section Index.

In order to clarify and demonstrate the procedure of performing simulations using BioDeg, 2 step-by-step examples are provided in Section 4.1, showing how to configure and run simulations with and without the UI. Moreover, a third example shows how to combine these approaches and use the UI to generate the execution command. The mesh files needed to run these examples can be found in the `demo` directory. Additionally, Section 4.2 provides some guidelines on the postprocessing of the BioDeg simulation results.

4.1 Configuring the simulation

4.1.1 Example 1 - degradation of a simple screw

Let us consider the first example given in the `demo` directory, where we simulate the biodegradation of a small screw. The size of the screw was chosen to be very small intentionally to make the simulation shorter such that the user can see the effect of the degradation much faster. The input mesh file is named `screw.mesh`. For this example, we use the BioDeg-UI interface to perform the simulation.

Let's conduct the first simulation with most of the parameters left with their default values. Run the user interface, and mark Geometry & Mesh » Import external mesh. Then click the browse button in front of the "File" box (Geometry & Mesh » Import mesh » File), navigate to the `demo` directory, and select the `screw.mesh` file. The path should be inserted in the "File" box. Selecting appropriate label numbers of the external mesh is very crucial in BioDeg, so you should always check the labels before importing the mesh. One of the best tools to do this is GMSH, in which you can view the labels of mesh files by opening the mesh and navigating to Tools » Visibility. Doing this on the `screw.mesh` shows that the label number of the scaffold is 2 while the medium has a label number of 1 (Fig. 1). So, we need to switch the default labels by selecting 2 for the scaffold (Geometry & Mesh » Import mesh » Scaffold label) and 1 for the medium (Geometry & Mesh » Import mesh » Medium label).

We need to enable the VTK output if we want to see the graphical output of the simulations. Navigate to Output » Output options » Write VTK output and mark it. You should also specify the output directory by clicking on the browse button in front of the "Output directory" (Output » Output options » Output directory) and select a directory of desire.

That was all we needed to do to setup a simulation in BioDeg, and you can start the simulation by pressing the Run simulation button located in the middle of the main window. There are lot more parameters we can deal with (see Section Index), but the input and output are the most essential ones.

Although you can start the simulation of this example right now, there is a couple of more things you may want to change:

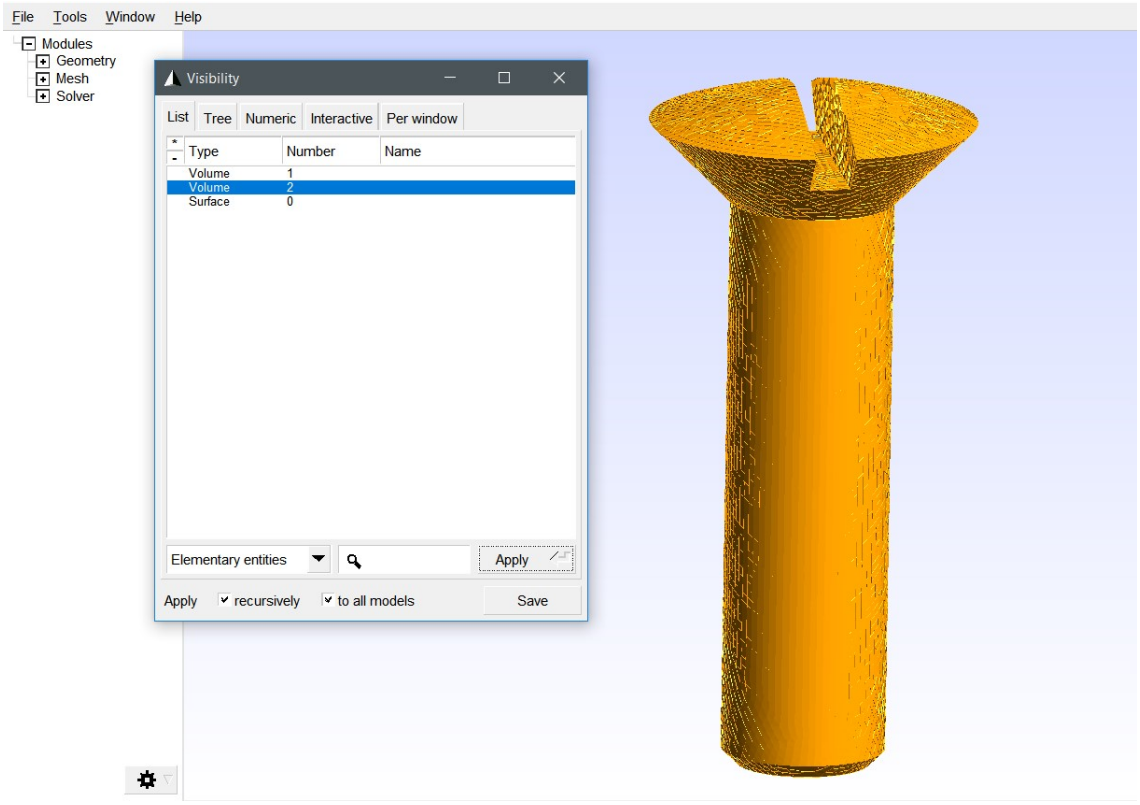

Figure 1: Using GMSH to read the labels of the mesh

- In BioDeg, simulations are by default carried out in parallel using domain decomposition, meaning that the simulation is distributed among available computing nodes. If you are running BioDeg on your local machine, you may need to adjust the parallel computing settings. Navigate to Solver › Parallel computing › Enable parallel computing and disable it if you don't want to parallelize the simulation. If you want to continue with parallelization enabled (default behavior), you may need to adjust the number of parallel processes to match the number of free CPU cores you have on your machine. You can change it in Solver › Parallel computing › CPU/MPI cores.

- The default degradation rate is quite fast, so you may want to decrease it by reducing the diffusion coefficient of the metallic ions (please refer to "Theory Guide" if you want to know more about diffusion controls the rate of degradation). The default diffusion rate is the value we have estimated for saline solutions, which leads to a high rate of corrosion. You can apply this by changing the value in Material & BCs › Reaction-diffusion properties › Metal ion diffusion coefficient and reduce it to something like 0.0005, which is its order when it comes to buffered solutions and simulated body fluids.

- The results write interval, implying how frequently you want to store the results, affects the resource consumption (which is the storage in this case) and the quality of the graphical postprocessing. So, you should configure this carefully to keep the balance of quality and resource consumption. The default save interval is 0.25 hours of simulation time, but you can change it in Output › Output options › Save results every ... hours. For this simulation, since the screw geometry is small and degrades very fast,

you can reduce this to 0.1 to be able to see the degradation steps better.

- Final simulation time does not affect the simulation progress, but it is always a good practice to adjust it, enabling us to track the progress of the simulation better and avoid wasting resources (both computing power and storage). The default simulation time is 21 hours, but you can reduce it in Solver » Time control » Final simulation time (hour). For this simulation, you can reduce it to 2.

After running the simulation (by pressing the Run simulation button), you can view the progress of the simulation on the UI, showing you how many steps have been taken and how much material degradation has happened (Fig 2). The UI also shows you the details info of the size of the problem, including the degrees of freedom (DOF) of each equation and the number of elements, as well as the number of DOFs for each sub-domain after mesh partitioning (for parallel computing).

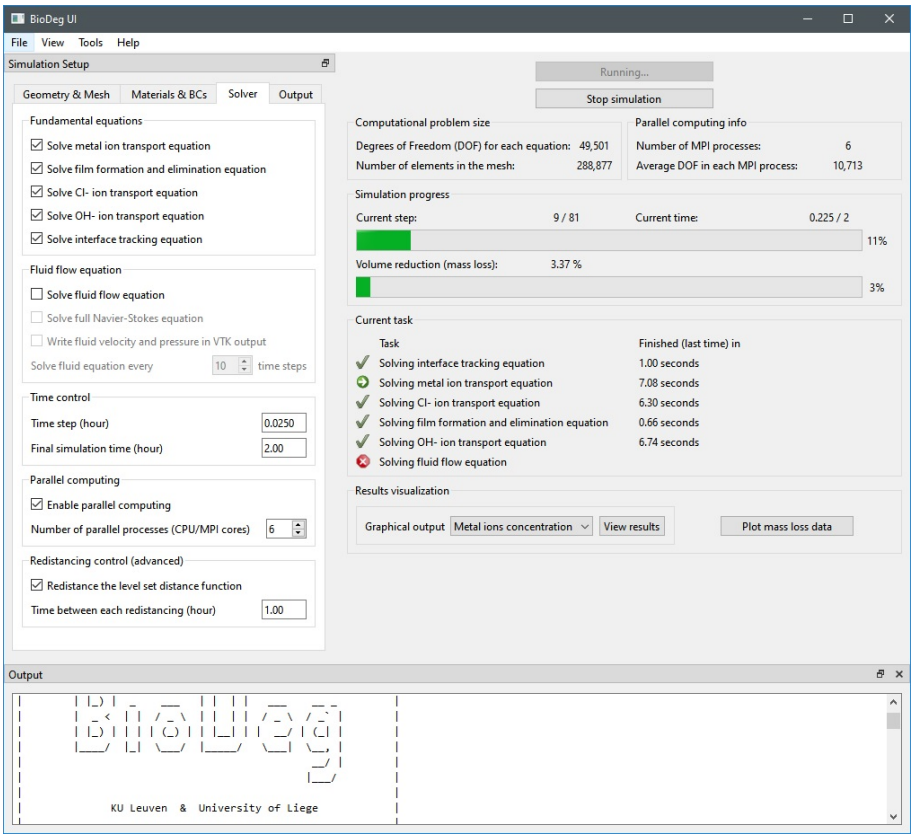

Figure 2: BioDeg-UI running the screw degradation example.

Running this simulation leads to the results demonstrated in Fig. 3, showing how the screw degrades. For more information on how to postprocess the results, please refer to the postprocessing section.

4.1.2 Example 2 - degradation of a helical shape (Biomech logo)

In the second example, we want to run a heavier biodegradation simulation (with 1,484,412 elements and a DOF of 254,157 for each equation), so let's do it by calling BioDeg directly on the command line. This

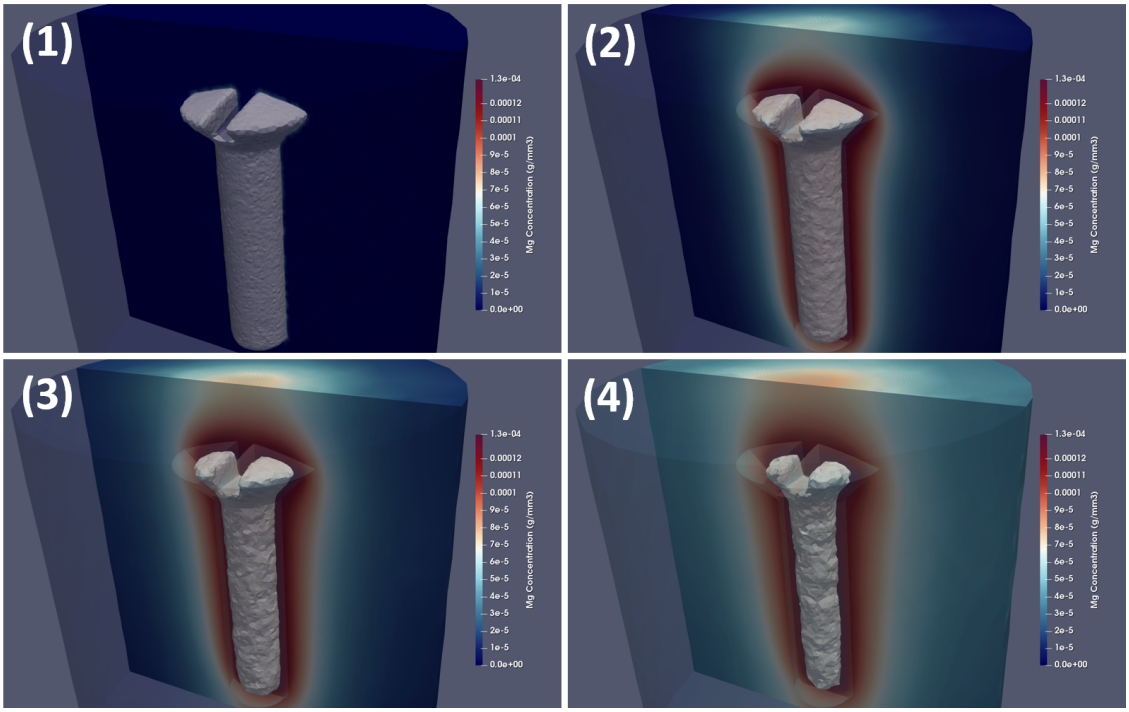

Figure 3: Simulation of the biodegradation of a simple screw, showing how the material is released and how the implant degrades.

approach can be taken on remote clusters and HPC environments to run BioDeg on hundreds or thousands of computing nodes.

The model in this example has a helical shape, which is actually the logo of the lab in which BioDeg has been developed. The input mesh file is called `biomech_logo.mesh` and is located in the `demo` directory. Similar to the previous example, we try to use the default value of parameters and only change the crucial ones. You should notice that the default value of parameters might be different between the core BioDeg and the UI, so it's always better to define them explicitly in the execution command. The default values of parameters can be checked in Section Index.

The main file of the BioDeg code is called `main.edp`, and since it's a parallel code, we should run it with the `mpiexec` command. So, calling the code with all the default parameters on 6 MPI cores can be done like this:

```
$ mpiexec -n 6 FreeFem++-mpi core/src/main.edp -v 0
```

The `-v 0` is added to suppress the messages that FreeFEM writes to the terminal, and it's recommended to include it on any call you make to BioDeg. Calling BioDeg like this does nothing for us, so let's complete the command by adding more configuration to it (remember that boolean values are passed by their integer equivalents, 0 for false and 1 for true):

- Turn off the fluid flow simulation and ignore the convection effect: `-solve_fluid 0`
- Indicate that we want to import an external mesh: `-import_mesh 1`

- Specify the name of the mesh, located in the `demo` directory: `-mesh_file "demo/biomech_logo.mesh"`
- Specify the labels of different regions, which are crucial parameters as discussed in previous example (Section 4.1.1): `-label_scaffold 3 -label_medium 4`
- You can lower the degradation rate a little bit if you like. You may run the simulation several times and see the effect of this parameter in action: `-d_mg 0.005`

Adding all the above arguments forms the final execution command to run:

```
$ mpiexec -n 6 FreeFem++-mpi core/src/main.edp -v 0 -solve_fluid 0 -import_mesh 1
-mesh_file "demo/biomech_logo.mesh" -label_scaffold 3 -label_medium 4 -d_mg 0.005
```

The above command executes BioDeg, which writes its output to the terminal (Fig 4) and stores the simulation results to the default directory called `output` (make sure it exists before calling BioDeg).

Figure 4: Output of BioDeg code for the logo example.

The output of this simulation is shown in Fig 5, which is postprocessed using the NVIDIA IndeX. Please refer to the postprocessing section to see how to do it.

4.1.3 Example 3 - biodegradation of a cuboid

The aim of this example is to show how one can combine the approaches taken in the last two examples and use the UI to generate the execution command so that it can be used to run the model in another

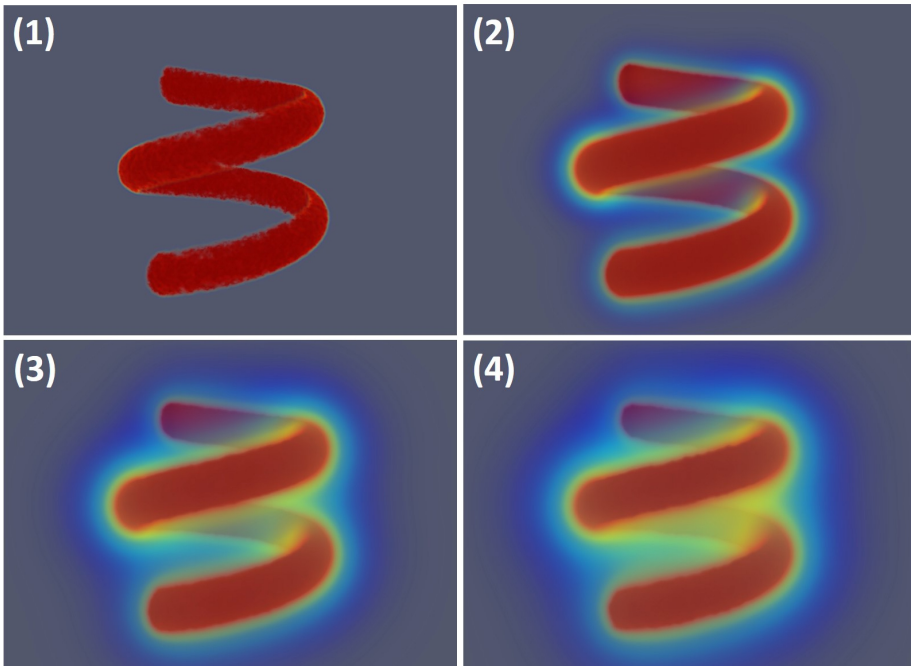

Figure 5: Simulation result of the logo example.

environment from the command line. The idea is simple: BioDeg-UI writes the run command and the output of the core model in the "Output" panel, so we can have the execution command if we setup the simulation, run it, stop it immediately, and have a look at the "Output" panel.

For demonstrating this, we simulate the degradation of a cuboid inside a cubic container. The mesh file is called `cuboid.mesh`, which has 830,808 elements. The mesh labels for scaffold and medium are the same as the default values on the BioDeg-UI (1 for scaffold and 2 for medium), so we don't need to change them. The setup will be straightforward since we just need to adjust the input and output, but the UI will generate a command with a full list of arguments, making it easy to modify it later if required.

Start the UI, make sure Geometry & Mesh › Import external mesh is checked, click the browse button in front of the "File" box (Geometry & Mesh › Import mesh › File), navigate to the `demo` directory, and select the `cuboid.mesh` file. Since we don't want to change anything else, go directory to the output settings, mark Output › Output options › Write VTK output (since we want to have graphical output for postprocessing), and specify the output directory by clicking on the browse button in front of the "Output directory" (Output › Output options › Output directory) and select a directory of desire.

As the setup is over, we are ready to run the simulation. Click the Run simulation button, wait a moment for the command to appear in the "Output" panel (written like `Executing: <command>`), and then click Stop simulation. You can now copy the command from the "Output" panel by selecting it, right-clicking on it, and choosing `Copy`. For this sample case, the command looks like this:

```
$ mpiexec -n 3 FreeFem++-mpi core/src/main.edp -v 0 -import_mesh 1
-mesh_file "/home/mojtaba/BioDeg/demo/cuboid.mesh" -label_scaffold 1 -label_medium 2
-refine_initial_mesh 0  -material_density 0.001735 -film_density 0.002344
```

```
-material_satur 0.000134 -material_eps 0.55 -material_tau 1 -k1 7 -k2 1e+10 -d_mg 0.05
-d_cl 0.05 -d_oh 25 -initial_cl 5.17e-06 -initial_oh 1e-07 -solve_mg 1 -solve_film 1
-solve_cl 1 -solve_oh 1 -solve_ls 1 -time_step 0.025 -final_time 21 -do_redistance 1
-redistance_time 1 -solve_fluid 0 -write_fluid_output 0 -text_output_file
"/home/mojtaba/BioDeg/cuboid_output/output.txt" -write_vtk 1 -vtk_output_name
"/home/mojtaba/BioDeg/cuboid_output/output" -save_last_state 0 -save_each 0.25
-output_per_area 0 -save_multiplier 1 -export_scaffold 0  -save_initial_mesh 0
-save_initial_partitioned_mesh 0
```

More details of these parameters can be found in Section Index. You can always omit the parameters with the default value, resulting in a command like the one we made in the second example (Section 4.1.2).

4.2 Postprocessing of the results using ParaView

Although the BioDeg-UI provides some basic postprocessing features such as the total amount of mass loss (via clicking on Plot mass loss data which reads the output text file) and a couple of ParaView templates (via selecting one of the variables on Graphical output and clicking View results), it can always be beneficial to visualize the results the way you want. Doing this enables you to reproduce the demonstration you see in Figs. 3 and 5 as well as any plot output you want, such as plotting variables over a line.

The details of the postprocessing steps of BioDeg results are discussed in the following YouTube videos, describing how to use ParaView to visualize the degrading scaffold beside plotting other output variables such as film formation and pH changes:

- https://www.youtube.com/watch?v=yeBPGwP3L80&ab_channel=TuxRiders
- https://www.youtube.com/watch?v=Sz-eBML2pxs&ab_channel=TuxRiders

Please note that there are 2 minor differences between the result files shown on the above videos and the current results that BioDeg generates:

1. The VTK files shown on the videos are actually replaced by VTU files. When you select the option to save the VTK output, BioDeg stores the results in a set of VTU files, written in requested time steps for each partition of the mesh separately.
2. Since BioDeg uses parallel computing to perform the simulation, a PVD file is written as a base container to organize all the generated output of different employed CPU cores (or computational nodes). So, instead of opening the grouped VTK files shown on the video, the user should open the PVD file in ParaView, which loads the VTU files automatically.

The following video gives you an idea of how to plot the values of different variables over a line and how to do some quantitative analysis on the results:

- https://www.youtube.com/watch?v=tGi-jk2UE2U&ab_channel=TuxRiders

For visualizing the fluid flow, more advanced techniques should be used. The following videos demonstrate how to visualize the fluid field around a degrading object simulated by BioDeg:

- https://www.youtube.com/watch?v=CByh84hOslU&ab_channel=TuxRiders
- https://www.youtube.com/watch?v=Xzwe94bvGJI&ab_channel=TuxRiders

5 Future extensions to BioDeg

The future versions of BioDeg will focus on implementing the following methodologies/features.

- Extending the core models to capture more complex chemistry of biodegradation by considering more reactions occurring in buffered solutions.
- Adding support for more base materials like Fe and Zn.
- Considering the effect of alloying elements and complex compositions.
- Adding more post-processing features to BioDeg UI.
- Adding basic visualization to BioDeg UI using ParaView Glance.
- Improving the performance of fluid flow solver by employing a gradient-based solver.
- Considering GPU support by enabling GPU computing in recent versions of PETSc.

6 Finding answers to more questions

If you have questions that go beyond this manual, there are several resources you may refer to:

- For questions/suggestions about BioDeg installation, bugs, or similar stuff please use the BioDeg issue tracker.
- BioDeg is primarily based on the FreeFEM. If you have particular questions about FreeFEM, contact the community at https://community.freefem.org/.
- If you have specific questions about BioDeg that are not suitable for public and archived mailing lists, you can contact the primary developer and mentor:
  - Mojtaba Barzegari: mojtaba.barzegari@kuleuven.be.
  - Liesbet Geris: liesbet.geris@kuleuven.be (Mentor).

A Run-time input parameters

The underlying description of the input parameters also includes a "Standard/Advanced" label, which signifies whether an input parameter is a standard one or an advanced level parameter. The default values of the "Advanced" parameters are good enough for almost all cases. However, in some cases, user may need to use "Advanced" labeled parameters. For user convenience, all input parameters are also indexed at the end of this manual in Section Index.

A.1 Geometry and mesh parameters

- *Parameter name:* `import_mesh`

  Default: true (1)

  Description: [Standard] Boolean parameter specifying whether an external mesh file is imported or a container box, as well as a cubic scaffold, would be created on the fly for simulation.

  Possible values: A boolean value (1 or 0)

- *Parameter name:* `mesh_file`

  Default: Should be provided

  Description: [Standard] Path to the input mesh file (in MEDIT `.mesh` format), which can be either absolute or relative. Is relevant only if parameter `import_mesh` is set to TRUE.

  Possible values: Any string value

- *Parameter name:* `label_scaffold`

  Default: 1

  Description: [Standard] The label of the (volume) region which is supposed to be scaffold in the input mesh (can be viewed in programs like GMSH before importing into BioDeg).

  Possible values: Any positive integer value

- *Parameter name:* `label_medium`

  Default: 2

  Description: [Standard] The label of the (volume) region which is supposed to be the medium (electrolyte) in the input mesh.

  Possible values: Any positive integer value

- *Parameter name:* `label_wall`

  Default: 3

  Description: [Advanced] The label of the surface in the input mesh to be assigned as wall (no-slip boundary condition) in the fluid flow simulations.

  Possible values: Any positive integer value

- *Parameter name:* `label_inlet`

  Default: 4

  Description: [Advanced] The label of the surface in the input mesh to be assigned as flow inlet (constant velocity boundary condition) in the fluid flow simulations.

  Possible values: Any positive integer value

- *Parameter name:* `label_outlet`

  Default: 5

  Description: [Advanced] The label of the surface in the input mesh to be assigned as flow outlet (zero pressure boundary condition) in the fluid flow simulations.

  Possible values: Any positive integer value

- *Parameter name:* `box_length`

  Default: 20.0

  Description: [Standard] In case of `import_mesh` being FALSE, specifies the length of the container box (for the electrolyte) in mm.

  Possible values: Any positive floating-point number

- *Parameter name:* `cube_size_x`

  Default: 13.0

  Description: [Standard] In case of `import_mesh` being FALSE, specifies the length of the scaffold cuboid along the x axis in mm.

  Possible values: Any positive floating-point number

- *Parameter name:* `cube_size_y`

  Default: 13.0

  Description: [Standard] In case of `import_mesh` being FALSE, specifies the length of the scaffold cuboid along the y axis in mm.

  Possible values: Any positive floating-point number

- *Parameter name:* `cube_size_z`

  Default: 4.0

  Description: [Standard] In case of `import_mesh` being FALSE, specifies the length of the scaffold cuboid along the z axis in mm.

  Possible values: Any positive floating-point number

- *Parameter name:* `mesh_size`

  Default: 32

  Description: [Standard] Number of elements on each edge of the container box, so a higher number means a finer mesh. The mesh size of the cuboid will be adjusted accordingly or can be adaptively refined by setting parameter `refine_initial_mesh` to TRUE.

  Possible values: Any positive integer number

- *Parameter name:* `refine_initial_mesh`

  Default: false (0)

Description: [Advanced] A boolean parameter specifying if the mesh (no matter if imported or generated) should be adaptively refined on the metal-medium interface (corrosion surface). This affects the beginning of the simulation only (on the initial mesh).

Possible values: A boolean value (1 or 0)

- *Parameter name:* `mshmet_error`

  Default: 0.01

  Description: [Advanced] Since the open-source tool `mshmet` is used for creating a metric for refining the mesh on the level set signed distance function, a tolerance should be specified for it. A lower value results to a finer mesh.

  Possible values: Any floating-point number

- *Parameter name:* `mesh_size_min`

  Default: 0.04

  Description: [Advanced] Specifies the smallest element size to be passed to the `tetgen` mesh generator for refining the initial mesh.

  Possible values: Any floating-point number

- *Parameter name:* `mesh_size_max`

  Default: 0.8

  Description: [Advanced] Specifies the largest element size to be passed to the `tetgen` mesh generator for refining the initial mesh.

  Possible values: Any floating-point number

A.2 Materials and boundary conditions parameters

- *Parameter name:* `material_density`

  Default: 1.735e-3

  Description: [Standard] The density of the metallic material. The default value is the density of magnesium.

  Possible values: Any floating-point number

- *Parameter name:* `film_density`

  Default: 2.3446e-3

  Description: [Standard] The density of the protective film that forms on the corrosion surface. The default value is the density of magnesium hydroxide.

  Possible values: Any floating-point number

- *Parameter name:* `material_satur`

  Default: 0.134e-3

  Description: [Advanced] The saturation concentration at which the metallic material (here, the ions) saturates through the medium. The default value is defined for magnesium ions.

  Possible values: Any floating-point number

- *Parameter name:* `material_eps`

  Default: 0.55

  Description: [Advanced] The porosity of the formed protective layer in the range [0, 1].

  Possible values: Any floating-point number between 0 and 1

- *Parameter name:* `material_tau`

  Default: 1.0

  Description: [Advanced] The tortuosity of the formed protective layer.

  Possible values: Any floating-point number

- *Parameter name:* `d_mg`

  Default: 0.05

  Description: [Standard]

  Possible values: The diffusion coefficient of the metallic ions transport. This parameter is one of the most effective ones on the rate of degradation.

- *Parameter name:* `d_cl`

  Default: 0.05

  Description: [Standard] The diffusion coefficient of the chloride ions transport

  Possible values: Any floating-point number

- *Parameter name:* `d_oh`

  Default: 25.2

  Description: [Standard] The diffusion coefficient of the hydroxide ions transport

  Possible values: Any floating-point number

- *Parameter name:* `k1`

  Default: 7.0

  Description: [Standard] The reaction rate at which the chemical reaction of the protective film formation occurs.

  Possible values: Any floating-point number

- *Parameter name:* `k2`

  Default: 1e15

  Description: [Standard] The reaction rate at which the chemical reaction of the protective film dissolution occurs.

  Possible values: Any floating-point number

- *Parameter name:* `fluid_nu`

  Default: 0.85

  Description: [Advanced] The effective viscosity of the fluid used to simulate hydrodynamics conditions.

  Possible values: Any floating-point number

- *Parameter name:* `fluid_in_x`

  Default: 0.1

  Description: [Advanced] The X component of the fluid velocity defined on the inlet (see `label_inlet`) as the boundary condition for the fluid flow.

  Possible values: Any floating-point number

- *Parameter name:* `fluid_in_y`

  Default: 0

  Description: [Advanced] The Y component of the fluid velocity defined on the inlet (see `label_inlet`) as the boundary condition for the fluid flow.

  Possible values: Any floating-point number

- *Parameter name:* `fluid_in_z`

  Default: 0

  Description: [Advanced] The Z component of the fluid velocity defined on the inlet (see `label_inlet`) as the boundary condition for the fluid flow.

  Possible values: Any floating-point number

- *Parameter name:* `initial_cl`

  Default: 5.175e-6

  Description: [Standard] Initial concentration of chloride ions in the medium.

  Possible values: Any floating-point number

- *Parameter name:* `initial_oh`

  Default: 1e-7

  Description: [Standard] Initial concentration of hydroxide ions in the medium, used for computing pH. The default value indicates a pH of 7.

  Possible values: Any floating-point number

A.3 Solver parameters

- *Parameter name:* `solve_mg`

  Default: true (1)

  Description: [Standard] Boolean parameter indicating whether the equation for material dissolution and ions release should be solved or not. This equation is the most essential equation and in most use-cases should be solved.

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `solve_film`

  Default: true (1)

  Description: [Standard] Boolean parameter indicating whether the equation for the protective film formation should be solved or not.

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `solve_cl`

  Default: true (1)

  Description: [Standard] Boolean parameter indicating whether the equation for the transport of chloride ions should be solved or not.

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `solve_oh`

  Default: true (1)

  Description: [Standard] Boolean parameter indicating whether the equation for the transport of hydroxide ions should be solved or not. This equation is essential for calculating the pH changes, if desired.

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `solve_ls`

  Default: true (1)

  Description: [Standard] Boolean parameter indicating whether the level set surface tracking equation should be solved or not. Surface tracking is essential for computing mass loss.

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `solve_fluid`

  Default: false (0)

  Description: [Standard] Boolean parameter indicating whether the fluid flow equation should be solved or not.

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `solve_full_ns`

  Default: true (1)

  Description: [Advanced] Boolean parameter specifying which fluid flow equation to solve: the full transient Navier-Stokes equations or a steady-state Stokes equation. A true value (1) results in BioDeg solving the former equation.

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `write_fluid_output`

  Default: true (1)

  Description: [Advanced] Boolean parameter specifying whether the fluid flow quantities (velocity field components and pressure) should be saved in the output VTK file or not. Requires `write_vtk` to be TRUE.

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `solve_fluid_each`

  Default: 10

  Description: [Advanced] Determines the number of time steps to skip before solving the specified fluid flow equation. For example, if the value is set to 10 (default value), the fluid equation gets solved in time steps 1, 11, 21, ...

  Possible values: Any positive integer number

- *Parameter name:* `time_step`

  Default: 0.025

  Description: [Advanced] The time step value of the simulations.

  Possible values: Any floating-point number

- *Parameter name:* `final_time`

  Default: 21.0

  Description: [Standard] The final simulation time, meaning the duration of interest for the biodegradation model.

  Possible values: Any floating-point number

- *Parameter name:* `do_redistance`

  Default: true (1)

  Description: [Advanced] Indicates whether the redistancing of the level-est distance function should be done or not. Please refer to the theory guides to see how this affects the simulation.

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `redistance_time`

  Default: 1.0

  Description: [Advanced] In case `do_redistance` is true, this parameter indicates the interval between each level set function re-initialization.

  Possible values: Any floating-point number

A.4 Output parameters

- *Parameter name:* `text_output_file`

  Default: "output/result.txt"

  Description: [Standard] Path to the text file in which text output, like the mass loss and evolved hydrogen production, is written over time. The path can be relative or absolute.

  Possible values: Any string value referring to a valid path

- *Parameter name:* `write_vtk`

  Default: true (1)

  Description: [Standard] Indicates whether VTK output (in the VTU format) should be written or not. This is required for further post-processing of the results using ParaView.

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `vtk_output_name`

  Default: "output/output"

  Description: [Standard] The naming scheme for the VTK output. This is mainly for the PVD file, and the final name of the rest of the files will be determined by the number of employed MPI computing nodes and the time step. The number of VTU files saved per time step equals the number of employed MPI nodes.

  Possible values: Any string value referring to a valid path

- *Parameter name:* `save_each`

  Default: 0.25

  Description: [Standard] The interval of saving results, text and VTK (if selected), to disk.

  Possible values: Any floating-point number

- *Parameter name:* `save_last_state`

  Default: true (1)

  Description: [Standard] Indicates whether the last state of the system should be saved or not. The last state will be always saved on a global (non-partitioned) mesh, meaning that it will be a single VTU file, in contrast to a normal save in which the number of written files equals to the number of computing nodes.

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `output_per_area`

  Default: false (0)

  Description: [Advanced] Indicated whether the side hydrogen evolution results should be computed per unit area of the exposed surface.

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `save_multiplier`

  Default: 1.0

  Description: [Advanced] In the case of symmetrical conditions, this parameter can be used to multiply the obtained quantitative results to have an easier comparison with experimental results. A value of 1.0 indicates no multiplication.

  Possible values: Any floating-point number

- *Parameter name:* `export_scaffold`

  Default: false (0)

  Description: [Advanced] Indicates if the degrading material should be exported as a single entity for further analysis like in a structural mechanics simulation. It works based on Mmg level set meshing.

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `export_scaffold_each`

  Default: 1.0

  Description: [Advanced] In case `export_scaffold` is true, this parameter determines the interval of saving the material mesh to the disk.

  Possible values: Any floating-point number

- *Parameter name:* `export_scaffold_volume`

  Default: true (1)

  Description: [Advanced] In case `export_scaffold` is true, this parameter indicates whether a volume mesh should be saved as output or not.

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `export_scaffold_surface`

  Default: true (1)

  Description: [Advanced] In case `export_scaffold` is true, this parameter indicates whether a surface mesh should be saved as output or not.

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `save_initial_mesh`

  Default: false (0)

Description: [Advanced] Determines whether the initial computational mesh should be saved for further debugging or not. Can be used to investigate the mesh refinement if it is asked by setting `refine_initial_mesh` to true.

Possible values: Any boolean value (1 or 0)

- *Parameter name:* `save_initial_partitioned_mesh`

  Default: false (0)

  Description: [Advanced] Determines whether the partitioned computational mesh should be saved for further debugging or not. Can be used to debug and view the output of the mesh partitioning process before going for the actual simulation.

  Possible values: Any boolean value (1 or 0)

Index of run-time parameters with section names

The following is a listing of all run-time parameters, sorted by the section in which they appear.