BioDeg

Biodegradation and Corrosion Simulation using Finite Element

User Manual

Version 0.8

(generated September 9, 2021)

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 manage 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.

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

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 on 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 installing/building FreeFEM and the required libraries, and then, cloning the BioDeg core repository and running the code according 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. 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.

2. 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.

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

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-cc=/usr/ --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 nay 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 abale 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`.

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

4 Running BioDeg

4.1 Configuring the simulation

4.1.1 Example 1

4.1.2 Example 2

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 element 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 a number of resources:

- 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 A.4.

A.1 Geometry and mesh parameters

- *Parameter name:* `mesh_file`

  Default: Should be provided

  Description: [Standard]

  Possible values: Any string value

- *Parameter name:* `label_scaffold`

  Default: 1

  Description: [Standard]

  Possible values: Any positive integer value

- *Parameter name:* `label_medium`

  Default: 2

  Description: [Standard]

  Possible values: Any positive integer value

- *Parameter name:* `label_wall`

  Default: 3

  Description: [Advanced]

  Possible values: Any positive integer value

- *Parameter name:* `label_inlet`

  Default: 4

  Description: [Advanced]

  Possible values: Any positive integer value

- *Parameter name:* `label_outlet`

  Default: 5

  Description: [Advanced]

  Possible values: Any positive integer value

- *Parameter name:* `box_length`

  Default: 20.0

  Description: [Standard]

  Possible values: Any positive floating point number

- *Parameter name:* `cube_size_x`

  Default: 13.0

  Description: [Standard]

  Possible values: Any positive floating point number

- *Parameter name:* `cube_size_y`

  Default: 13.0

  Description: [Standard]

  Possible values: Any positive floating point number

- *Parameter name:* `cube_size_z`

  Default: 4.0

  Description: [Standard]

  Possible values: Any positive floating point number

- *Parameter name:* `mesh_size`

  Default: 32

  Description: [Standard]

  Possible values: Any positive integer number

- *Parameter name:* `refine_initial_mesh`

  Default: false (0)

  Description: [Advanced]

  Possible values: A boolean value (1 or 0)

- *Parameter name:* `mshmet_error`

  Default: 0.01

  Description: [Advanced]

  Possible values: Any floating point number

- *Parameter name:* `mesh_size_min`

  Default: 0.04

  Description: [Advanced]

  Possible values: Any floating point number

- *Parameter name:* `mesh_size_max`

  Default: 0.8

  Description: [Advanced]

  Possible values: Any floating point number

A.2 Materials and boundary conditions parameters

- *Parameter name:* `material_density`

  Default: 1.735e-3

  Description: [Standard]

  Possible values: Any floating point number

- *Parameter name:* `film_density`

  Default: 2.3446e-3

  Description: [Standard]

  Possible values: Any floating point number

- *Parameter name:* `material_satur`

  Default: 0.134e-3

  Description: [Advanced]

  Possible values: Any floating point number

- *Parameter name:* `material_eps`

  Default: 0.55

  Description: [Advanced]

  Possible values: Any floating point number between 0 and 1

- *Parameter name:* `material_tau`

  Default: 1.0

  Description: [Advanced]

  Possible values: Any floating point number

- *Parameter name:* `d_mg`

  Default: 0.05

  Description: [Standard]

  Possible values: Any floating point number

- *Parameter name:* `d_cl`

  Default: 0.05

  Description: [Standard]

  Possible values: Any floating point number

- *Parameter name:* `d_oh`

  Default: 25.2

  Description: [Standard]

  Possible values: Any floating point number

- *Parameter name:* `k1`

  Default: 7.0

  Description: [Standard]

  Possible values: Any floating point number

- *Parameter name:* `k2`

  Default: 1e15

  Description: [Standard]

  Possible values: Any floating point number

- *Parameter name:* `fluid_nu`

  Default: 0.85

  Description: [Advanced]

  Possible values: Any floating point number

- *Parameter name:* `fluid_in_x`

  Default: 0.1

  Description: [Advanced]

  Possible values: Any floating point number

- *Parameter name:* `fluid_in_y`

  Default: 0

  Description: [Advanced]

  Possible values: Any floating point number

- *Parameter name:* `fluid_in_z`

  Default: 0

  Description: [Advanced]

  Possible values: Any floating point number

- *Parameter name:* `initial_cl`

  Default: 5.175e-6

  Description: [Standard]

  Possible values: Any floating point number

- *Parameter name:* `initial_oh`

  Default: 1e-7

  Description: [Standard]

  Possible values: Any floating point number

A.3 Solver parameters

- *Parameter name:* `solve_mg`

  Default: true (1)

  Description: [Standard]

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `solve_film`

  Default: true (1)

  Description: [Standard]

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `solve_cl`

  Default: true (1)

  Description: [Standard]

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `solve_oh`

  Default: true (1)

  Description: [Standard]

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `solve_ls`

  Default: true (1)

  Description: [Standard]

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `solve_fluid`

  Default: false (0)

  Description: [Standard]

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `solve_full_ns`

  Default: true (1)

  Description: [Advanced]

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `write_fluid_output`

  Default: true (1)

  Description: [Advanced]

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `solve_fluid_each`

  Default: 10

  Description: [Advanced]

  Possible values: Any positive integer number

- *Parameter name:* `time_step`

  Default: 0.025

  Description: [Advanced]

  Possible values: Any floating point number

- *Parameter name:* `final_time`

  Default: 21.0

  Description: [Standard]

  Possible values: Any floating point number

- *Parameter name:* `do_redistance`

  Default: true (1)

  Description: [Advanced]

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `redistance_time`

  Default: 1.0

  Description: [Advanced]

  Possible values: Any floating point number

A.4 Output parameters

- *Parameter name:* `text_output_file`

  Default: "output/result.txt"

  Description: [Standard]

  Possible values: Any string value referring to a valid path

- *Parameter name:* `write_vtk`

  Default: true (1)

  Description: [Standard]

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `vtk_output_name`

  Default: "output/output"

  Description: [Standard]

  Possible values: Any string value referring to a valid path

- *Parameter name:* `save_each`

  Default: 0.25

  Description: [Standard]

  Possible values: Any floating point number

- *Parameter name:* `save_last_state`

  Default: true (1)

  Description: [Standard]

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `output_per_area`

  Default: false (0)

  Description: [Advanced]

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `save_multiplier`

  Default: 1.0

  Description: [Advanced]

  Possible values: Any floating point number

- *Parameter name:* `export_scaffold`

  Default: false (0)

  Description: [Advanced]

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `export_scaffold_each`

  Default: 1.0

  Description: [Advanced]

  Possible values: Any floating point number

- *Parameter name:* `export_scaffold_volume`

  Default: true (1)

  Description: [Advanced]

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `export_scaffold_surface`

  Default: true (1)

  Description: [Advanced]

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `save_initial_mesh`

  Default: false (0)

  Description: [Advanced]

  Possible values: Any boolean value (1 or 0)

- *Parameter name:* `save_initial_partitioned_mesh`

  Default: false (0)

  Description: [Advanced]

  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.