SciCell++

Release 0.6.0

Julio César Pérez Sansalvador

Nov 01, 2022

CONTENTS

SciCell++ is an object-oriented framework for the simulation of biological and physical phenomena modelled as continuous or discrete processes.

CHAPTER

ONE

A FRAMEWORK, NOT A LIBRARY

SciCell++ is a **software framework**, that means, we provide you with the main code structure and you only have to fill-in the specific details for your project. In a library, you are in charge of the main flow of the program and the tools that you include.

The code that you write becomes part of the framework, this will help others to build on your code just as you built on the code wrote by them. We encourage you to personalise SciCell++ based on your needs **by extending** its current behavior to provide it with extra functionalities.

1.1 Initial steps

This document shows you how to *install*, *start* and *configure* SciCell++ on Linux and Windows systems.

1. The default installation strategy is based on containers so that all software dependencies are preinstalled and ready to use by SciCell++. The *installation* section will show you how to install the container software for your operating system.
2. Then continue with the *starting SciCell++* section that shows you how to start SciCell++ using a container.
3. Once running the container with SciCell++ the *configuration* section will show you how to compile and enable advanced SciCell++'s features.

1.1.1 Installation

The default installations use a container strategy such that all software dependencies are preinstalled and ready to use for SciCell++.

initial_steps/installation/linux_installation For Linux systems
: users. Use this if you are not an experience Linux user. If you feel like challenging yourself then try our non-container based installation instructions and set aside enough sparse time.

  Advanced installation: Use this installation only if you are familiar with Unix based systems. This installation provides you with full control over the versions of the third-part packages used by SciCell++.

initial_steps/installation/windows_installation For Windows
: systems users. Use this instructions only if you are not able to install a Linux distribution in your local machine.

1.1.2 Starting SciCell++

Start the container that provides you with all packages required by SciCell++ and let it ready for the configuration step.

initial_steps/start_scicellxx/start_scicellxx_linux Use this
option to run SciCell++ by either using the container based installation or the advanced installation.

initial_steps/start_scicellxx/start_scicellxx_windows Use this
option to run SciCell++ using the container based installation.

1.1.3 Configuration

This section guides you through the configuration process of SciCell++.

Important: At this point you should have successfully installed and executued SciCell++ on either Linux or Windows. Also you have started the docker container.

The configuration is performed with help of the `autogen.sh` script which lives in the main SciCell++ folder.

1. In the running terminal make sure you are in the `scicellxx` folder.
2. Execute the automatic generator script by typing:

```
./autogen.sh
```

This command executes a full compilation of SciCell++ and runs all the demos and tests to make sure you are working with an stable copy. If you want a full list of available parameters for this script then add the `-h` parameter and review the *additional options for autogen.sh* section.

A summary of the compilation and testing processes is shown once they have finished. If no errors were reported then SciCell++ is ready to go.

Note: If you require to enable advanced features then check the configuration files at the `./configs` folder and have a look at the *options for these files* to enabled/disable advanced features.

Note: If you are NOT running SciCell++ within a container then use the appropiate config files in the `./configs/advanced/` folder.

If this is the first time you compiled SciCell++ then you need to create your own private folder, check the *Workflows* section on how to do this and many other common uses for SciCell++. Also check the *Tutorials* and *Demos* documents.

Additional options for `autogen.sh`

Activate the interactive mode for full configuration by passing the `-i` parameter.

```
./autogen.sh -i
```

We encourage you to check the full list of options by passing the `-h` parameter, the following may not reflect the full list of options for the script.

This script builds [and runs the test suite of] SciCell++

OPTIONS:

-h	Show this message
-t	Indicates to generate a 'STATIC' or 'SHARED' version of library files
-b	Build version 'DEBUG' or 'RELEASE'
-c	Configuration file for additional building tools
-n	Number of processors to build the framework
-d	Number of processors to run demos (set to '0' to skip demos testing)
-i	Interative mode, launches the interactive mode to prompt for FULL configuration options (any other parameters are ignored)
-r	Generate code coverage report
-v	Verbose

Options for the configuration file

- SCICELLXX_LIB_TYPE

  Specify the building type of the library `STATIC` or `SHARED`.

  Example:

  ```
  SCICELLXX_LIB_TYPE=STATIC
  ```

- SCICELLXX_RANGE_CHECK

  Specify whether to check for out-of-range in vectors or not. This significantly increase the running time of your application. We recommend you to enable this option only for developing purposes to ease you finding errors in the code. When using this option consider to compile with debugging options when prompted by the `autogen.sh` file. Once you move into the release mode disable this option and compile with full optimisation when prompted by the `autogen.sh` file.

  Example:

  ```
  SCICELLXX_RANGE_CHECK=TRUE
  ```

- SCICELLXX_USES_DOUBLE_PRECISION

  Specify whether to use single (`float`) or double (`double`) precision at running time.

  Example:

```
SCICELLXX_USES_DOUBLE_PRECISION=TRUE
```

- SCICELLXX_USES_ARMADILLO

  Specify whether to use the external library Armadillo for linear algebra. You would need to install Armadillo to enable this option.

  Example:

```
SCICELLXX_USES_ARMADILLO=FALSE
```

- SCICELLXX_AUTO_FIND_ARMADILLO_PATHS

  Use this option to automatically find the corresponding installation folder of Armadillo. Enable this option only if the option SCICELLXX_USES_ARMADILLO was set to TRUE.

  Example:

```
SCICELLXX_AUTO_FIND_ARMADILLO_PATHS=TRUE
```

- ARMADILLO_AUTO_FIND_FOLDER

  If you want ot use Armadillo and you set the variable SCICELLXX_AUTO_FIND_ARMADILLO_PATHS to TRUE you may indicate an starting folder for the automatic finding of the Armadillo installation.

  Example:

```
ARMADILLO_AUTO_FIND_FOLDER=/home/tachidok/local/working/research/armadillo-8.300.3
```

- ARMADILLO_INCLUDE_DIRS

  If you want to use Armadillo but set the variable SCICELLXX_AUTO_FIND_ARMADILLO_PATHS to FALSE you must indicate the include directory in this option.

  Example:

```
ARMADILLO_INCLUDE_DIRS=/home/tachidok/local/working/research/armadillo-8.300.3/
↪installation/include
```

- ARMADILLO_LIBRARIES

  If you want to use Armadillo but set the variable SCICELLXX_AUTO_FIND_ARMADILLO_PATHS to FALSE you must indicate the lib directory in this option.

  Example:

```
ARMADILLO_LIBRARIES=/home/tachidok/local/working/research/armadillo-8.300.3/
↪installation/lib/libarmadillo.so.8.300.3
```

- SCICELLXX_USES_VTK

  Specify whether to use the external library VTK for results visualization. Some demos require VTK to generate output. You would need to install VTK to enable this option.

  Example:

```
SCICELLXX_USES_VTK=FALSE
```

- SCICELLXX_AUTO_FIND_VTK_PATHS

Use this option to automatically find the corresponding installation folder of VTK. Enable this option only if the option SCICELLXX_USES_VTK was set to TRUE.

Example:

```
SCICELLXX_AUTO_FIND_VTK_PATHS=TRUE
```

- VTK_AUTO_FIND_FOLDER

  If you want ot use VTK and you set the variable SCICELLXX_AUTO_FIND_VTK_PATHS to TRUE you may indicate an starting folder for the automatic finding of the VTK installation.

  Example:

```
VTK_AUTO_FIND_FOLDER=/home/tachidok/local/working/research/VTK-8.1.1/VTK-bin
```

- VTK_INCLUDE_DIRS

  If you want to use VTK but set the variable SCICELLXX_AUTO_FIND_VTK_PATHS to FALSE you must indicate the `include` directory in this option.

  Example:

```
VTK_INCLUDE_DIRS=/home/tachidok/local/working/research/VTK-8.1.1/VTK-bin/
↪installation/include/vtk-8.1
```

- VTK_LIBRARIES

  If you want to use VTK but set the variable SCICELLXX_AUTO_FIND_VTK_PATHS to FALSE you must indicate the `lib` directory in this option.

  Example:

```
VTK_LIBRARIES=/home/tachidok/local/working/research/VTK-8.1.1/VTK-bin/installation/
↪lib/libvtkalglib-8.1.so.1
```

- SCICELLXX_PANIC_MODE

  This option enables a large number of validations at running time, it also enables error messages that may help you to identify problems in your code. However, this considerably increase the running time of your application. Use this option only at developing time. Deactivate this function when runnig on release mode, also make sure to activate full optimisation at compilation time when prompted by the `autogen.sh` script.

  Example:

```
SCICELLXX_PANIC_MODE=TRUE
```

1.2 Workflows

The most common uses for SciCell++ are described in this section, these workflows show you how to create your private folder, run demos, create your own project and include it as part of the demos of SciCell++.

The main differences on the workflow for Windows and Linux users are on the graphic interfaces. We provide you with the details for the graphic interfaces when required.

Important: All these workflows suppose you have executed SciCell++ on a container and the terminal prompt is at the main folder of SciCell++.

1.2.1 Beginner workflows

workflow/beginner/create_your_private_folder_workflow
Every new user requires to create it own private folder to store all of his/her work (new features, demos, tests).

workflow/beginner/compiling_and_running_demos_workflow
The easiest way to start working with SciCell++ is to run any of its demos and change the values of the parameters to review its effects on the outputs.

workflow/beginner/creating_your_own_project_workflow
For Windows systems users.

workflow/beginner/daily_workflow
For Linux systems users.

1.2.2 Intermediate workflows

workflow/intermediate/add_your_project_to_the_demos_folder_workflow
For Windows systems users.

workflow/intermediate/generate_tar_file_workflow
Generate a `tar` file to distribute or move SciCell++ into a supercomputing cluster.

workflow/advanced/generate_doxygen_documentation_workflow
Generate full documentation from source code with class diagrams.

1.2.3 Advanced workflows

1.2.4 Expert workflows

1.3 Demos

1.3.1 Examples and test cases

Demos, examples or test cases are in the *demos* folder. These demos are also used to self test the implementation. You should run all of them to make sure nothing is broken. You can select to perform a full test at installation time, otherwise type

`` `shell ./ctest ` `` into the specified build folder (the default one is *build*).

A large number of examples is expected to live in the *demos* folder so check there for any feature available in the project.

1.3.2 Current demos for specific functionalities

- Interpolation
- Linear solvers
- Matrices operations
- Newton's method
- Solution of ODE's * Lotka-Volterra solved with different time steppers * N-body problem (only 3-body and 4-body cases) * Explicit time steppers * Implicit time steppers (full implicit and *E(PC)^k E* implementations) * Adaptive time steppers

Here is some example text:

```
make 'this'
>> output
```

1.4 Tutorials

In this section you will find a large set of tutorials (some of them with exercises) to help your way through SciCell++.

1.4.1 General

Create a tutorial

How to create documentation

Python

Sphinx

Web tutorials

A quick starting-up guide

In this section we provide you with the basic tool to start working with SciCell++, we present the main folder structure and how to create your very first project. There is also a section for including your project as a demo for further reference of new collaborators.

Running demos

SciCell++ is released with a set of demos that show you some of its main features. We recommend you to explore the demos section of the documentation and the demos folder. Whenever you want to run a demo just go to the demo folder which you are interested, create a folder called `RESLT` if it is not already there and type `./bin/` followed by the name of the demo.

- **Example:** Lets say you want to run the Lotka-Volterra demo in the folder `/demos/lotka_volterra/`, once you are in that folder create the `RESLT` folder where the output is stored (all the demos are configured to store its output in a folder with that name, if the folder does not exist then the output is not generated) and run the demo:

  ```
  mkdir RESLT
  ./bin/demo_lotka_volterra
  ```

  Once the demo has started you should see output messages on the terminal with general information about the results of the computations. You can check the produced results in the `RESLT` folder.

Note: Observe that some demos are equipped with Python or GNUPlot script to visualise the results. Try to run them as `python <name-of-the-python-script.py>` or `gnuplot <name-of-the-gnu-script.gp>`.

Input arguments

Some demos require input arguments to run, if you try to run one of those and pass nothing you will get a message indicating what you need to pass. You can also check what input arguments a demo needs by passing the `--help` or `-h` options at running time.

Create your `private` folder

Every user has its own private folder, use this folder to store all of your work, in-development demos and any of your new developed features for SciCell++. One of the first things that you should do in order to start developing new features for SciCell++ is to create your private folder, to do so follow theses instructions:

1. Open a terminal and go to the `private` folder of SciCell++ and typet the following (make sure to substitute `john_cool` by your name):

```
cd private
mkdir john_cool
cd john_cool
```

2. Update the `CMakeLists.txt` file in the private folder by adding your folder name at the end of the file as follow (make sure to substitute `john_cool` by your name):

```
ADD_SUBDIRECTORY(john_cool)
```

3. Run the `autogen.sh` script at the root folder of SciCell++ and make sure no problems are found. If there are any problem double-check that you added your folder inside the `private` folder of SciCell++ and that you are modifying the correct `CMakeLists.txt` file.

Creating your own project

The easiest way to start a new project is to use a demo as a template. For this example we are going to copy the demo driver `demo_basic_interpolation.cpp` from the folder `demos/interpolation/basic_interpolation`.

1. Open a terminal and go to your private folder.
2. Type the following to copy the demo driver into your private folder:

```
cp ../../demos/interpolation/basic_interpolation/demo_basic_interpolation.cpp demo_
↪john.cpp
```

3. Copy the `CMakeLists.txt.private_template` file from the `tools` folder into your private directory and change its name to `CMakeLists.txt`

```
cp ../../tools/CMakeLists.txt.private_template CMakeLists.txt
```

4. Change the content of the `CMakeLists.txt` file as follow:

- Change all the instances of the tag `SRC_demo_john` for your own tag to identify your source code. For example: `SRC_project_sophy`.
- Change all the instances of `demo_john.cpp` for the name of your source code file. For example: `project_sophy.cpp`.
- Change all the instances of `demo_john`, this will be the name of your executable and the name you need to type at the terminal to compile your project. For example:`project_sophy`.

- Change all the instances of the tag `LIB_demo_john` for your own tag to identify libraries required for your code. For example: `LIB_project_sophy`.
- Include the modules you need. In the template we only include the `general_lib` and the `problem_lib` modules. Check the modules document for the full list of module and their details.

5. Go to the root folder of SciCell++ and execute the `./autogen.sh` script. If you find errors please make sure you correctly changed all the tags indicated in the previous step. Once building has finished without errors you can build your own project.

Building and executing your project

Open a terminal and follow these instructions:

1. Go to the `build` folder in the root SciCell++ folder and type

```
make demo_sophy
```

   The building output should be displayed at your screen. Once no errors have been reported you may run your code.

2. Go to your `private` folder, create a `RESLT` folder if you have no one, and type:

```
./bin/demo_sophy
```

3. You should see the output of your project at the terminal.

Important: As you noticed, the generation and execution of your project is performed in two different folders:

- the `build` folder (building)
- your `private` folder (execution)

We use this two-folders strategy to avoid cluttering the folder structure of SciCell++ with files automatically generated by CMake. By following this strategy we keep a clean folder structure for SciCell++ and group all files generated by CMake in the `build` folder. This help us to keep track for changes easily since we can exclude the whole `build` folder from the git repository.

Just keep in mind the following:

- Whenever you want to build your project you need to do so in the `build` folder, inthere just type `make` followed by the name of your project.
- Whenever you want to execute your project go to your `private` folder and type `./bin/the-name-of-your-project`.

Add your project to the `demos` folder

If you add a new feature to SciCell++ we encourage you to *Create a tutorial* and a demo showing these new features. Here we detail the process to include your project as part of the demos of SciCell++. We divide this process in two parts, the first one guides you to create your folder and your validation files, the second part shows you how to configure the SciCell++ to build and execute your demo. In both sections we suppose that your demo is called `demo_sophy`.

Create your demo and validation folder for your demo

The initial steps to include your demo as part of SciCell++ involve create a folder in the SciCell++ demos folder structure and to generate the validation files.

1. Execute your project and save its output into a file. We encorage you to execute it using single and double precision so that we have two different outputs. The files that you generate should be named:
   - `validate_demo_sophy.dat` for the single precision generated output.
   - `validate_double_demo_sophy.dat` for the double precision generated output.
2. Create a new folder into the `demos` folder structure. Use a name that captures the intent of your project.

```
mkdir <your-folder-name>
```

3. Add the following line at the end of the `CMakeLists.txt` file that lives at the same level of the folder that you created:

```
ADD_SUBDIRECTORY(your-folder-name)
```

4. Step into your demo folder and create a folder called `validate`.
5. Copy the two output files (or copy all of them if you have more than two) generated at step 1 into the `validate` folder.

Configure SciCell++ to build and execute your demo

Once you have created your folder and copied the validation files there you are ready to configure SciCell++ to build and execute your demo.

1. Copy the source code for your project into your demo folder, in this case we suppose that the source code for your project is the file `demo_sophy.cpp`.
2. Copy the `CMakeLists.txt.demo_template` from the `/tools/` folder into your demo folder. Rename this file as `CMakeLists.txt`.
3. Change the content of the `CMakeLists.txt` file as follow:
   - Change all the instances of the tag `SRC_demo_john` for your own tag to identify your source code. For example: `SRC_demo_sophy`.
   - Change all the instances of `demo_john.cpp` for the name of your source code file. For example: `demo_sophy.cpp`.
   - Change all the instances of `demo_john`, this will be the name of your executable and the name you need to type at the terminal to compile your project. For example:`demo_sophy`.
   - Change all the instances of the tag `LIB_demo_john` for your own tag to identify libraries required for your code. For example: `LIB_demo_sophy`.
   - Include the modules you need. In the template we only include the `general_lib` and the `problem_lib` modules. Check the modules document for the full list of module and their details.
4. In the same file perform the following changes in the `Test section`.
   - Change all the instances of `TEST_demo_john_run` by the name of your demo. For example: `TEST_demo_sophy_run`.

Important: Make sure to keep the `TEST` and `_run` prefix and postfix, respectively.

- Change all the instances of `demo_john` with the name of your demo. For example: `demo_sophy`.
- Change all the instances of `VALIDATE_FILENAME_demo_john` with the name of your tag for the validation file. For example: `VALIDATE_FILENAME_demo_sophy`.
- Change the name of the validation file `validate_double_demo_john.dat` by yours. Recall that this file should store the output of your project executed using double precision. For example: `validate_double_demo_sophy.dat`.
- Change the name of the validation file `validate_demo_john.dat` by yours. Recall that this file should store the output of your project executed using single double precision. For example: `validate_demo_sophy.dat`.
- Change all instances of `TEST_demo_john_check_output` with the name of your demo. For example: `TEST_demo_sophy_check_output`.

Important: Make sure to keep the `TEST` and `_output` prefix and postfix, respectively.

5. Make sure that the computations of your demo are stored in an output file. If the file that you generate is called differently than `output_test.dat` then modify any instance of that name in the `CMakeLists.txt` file.
6. Go to the root folder of SciCell++ and execute the `./autogen.sh` script and enable the execution of the demos. If you find errors please make sure you correctly changed all the tags indicated in the previous steps. Your project should be automatically built, executed and validated.

1.4.2 Maths

Differential Equations

What are differential equations?

- Move the solution of a differential equation ot other section * You may start by stating that a solution is a function and not a

  value

- In the section of why DE are important check the YouTube video https://youtu.be/p_di4Zn4wz4

A differential equation is any expression where a function y is related to its derivative [Strang2014], for example:

$$\frac{d}{dt}y(t) = y(t)(1.1)(1.1)$$

This equation states that the change in the function y with respect to a change in the variable t is equal to the function itself evaluated at time t. A solution for such equation should be a function such that its derivative is the same function y.

A different way to think about a solution for that equation may be

"what function has the property that its derivative is the very same function?", or "what function satisfy the relation given by (1.1)?"

A function with such properties is the exponential function e^t, therefore the solution for (1.1) is:

$$y(t) = e^t \tag{1.2}$$

If you do the maths, you will notice that the derivative of e^t, that is $\frac{d}{dt}e^t = e^t$; and you are done.

Some more examples of differential equations are the following:

$$\begin{aligned}
\frac{d}{dt}y(t) &= 2ty(t) \\
\frac{d}{dt}y(t) &= y(t)^3 \\
\frac{d}{dt}y(t) &= 3y(t)^2 \sin(t + y(t)) \\
\frac{d^2}{dt^2}y(t) &= \sin t + 3y(t) + \left(\frac{d}{dt}y(t)\right)^2 \\
\frac{d^3}{dt^3}y(t) &= e^{-y(t)} + t + \frac{d^2}{dt^2}y(t)
\end{aligned} \tag{1.3}$$

Observe that differential equations may represent complex relations between a function and its derivatives. Have a second look at the previous equations, you will notice that there are relations regarding the second and the third derivate of a function, expressed as $\frac{d^2}{dt^2}$ and $\frac{d^3}{dt^3}$, respectively. You may find differential equations relating a function with its n derivate; which is expressed as $\frac{d^n}{dt^n}$.

Why are differential equations important?

Differential equations are very useful to study a wide variety of phenomena found in nature. Differential equations connect maths with physics, biology and chemestry.

Differential equations describe changes

Differential equations are commonly used when it is easier to describe changes on a phenomena rather than state why a phenome is at a particular state.

Important: It is a common practice to write the dependent function without its parameters, for example, y is commonly used instead of $y(t)$. Therefore the notation is simplified as $\frac{dy}{dt}$ instead of $\frac{d}{dt}y(t)$. You should be careful that a similar notation may be used for dependant function with multiple parameters, for example, $u(t, x)$ may be wrote as u. By now we will only deal with functions with one parameter.

Types of differential equations

Differential equations can be classified according to its degree, its number of variables, ...

Ordinary Differential Equations vs Partial Differential Equations

How to interpret a differential equation

References

1.4.3 Cellular Automata

Cellular Motors

TASEP

1.5 Contributions

Information that may help you to start your contributions to SciCell++, please feel free to contact the developer's team at any time.

1.5.1 How to contribute?

Create new issues in the GitHub repository or send an email directly to the developer's team (jcp.sansalvador@inaoep.mx)

1.5.2 Facts and curiosities

How many developers are currently working on the library?

At Monday, September/20, 2021 there is one and only one developer.

When did this project start?

The first commit to GitHub of this project was on Friday, 11 March 2016.

A list (in wikipedia) of some software packages implement the FEM

https://en.wikipedia.org/wiki/List_of_finite_element_software_packages

List of contributors

- tachidok
  - GitHub repository (https://github.com/tachidok)
  - Official webpage (https://ccc.inaoep.mx/~jcp.sansalvador/)

1.6 License

GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <https://fsf.org/> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work,

and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions

infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the

parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found. <one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>

> This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program. If not, see <https://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

> <program> Copyright (C) <year> <name of author> This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

> You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <https://www.gnu.org/licenses/>.
>
> The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <https://www.gnu.org/licenses/why-not-lgpl.html>.

1.7 Help

For help please contact the developers team at jcp.sansalvador@inaoep.mx

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[Strang2014] Strang, G. (2014). Differential Equations and Linear Algebra. Wellesley, MA: Wellesley-Cambridge Press.