# The *spin-transport* documentation

Rico A.R. Picone*

Aug 30, 2017

**Abstract**

The *spin-transport* software (GitHub) is for the dynamic simulation of bulk spin transport—diffusion and separation—in solid media. The project is open-source and still in development.

## Contents



# 1 Installation

# 2 spin-transport: introduction

This repository contains the (developing) open-source code for simulating bulk spin transport—diffusion and separation—in solid media. Multi-spin-species and magnetic resonance simulations are in development.

This is a Python and FEniCS project. FEniCS is used to numerically solve the spin transport governing partial differential equations.

End users of this project write Python code to interface with FEniCS.

*Email: rpicone@stmartin.edu. Department of Mechanical Engineering, Saint Martin's University.

## 2.1 Installation

One must first have a working installation of FEniCS. This README assumes the use of Docker for installation, which is documented here.

Then clone this repository to the host machine.

## 2.2 Workflow

The FEniCS docs have a section on workflow. There are many ways to instantiate these good practices, but if you're using a nix system, the following may be the easiest.

With the cloned `spin-transport` repository as your working directory, create a link in your path to **spin-transport**'s `fenics` executable bash script.

```
ln fenics /usr/local/bin
```

Now a FEniCS Python script `foo.py` can be started with the command `fenics foo.py` **from the host** instead of manually starting it from a Docker container. This has several advantages, including that there is no need to move scripts into the container and that the complicated syntax need not be remembered.

## 2.3 Testing the installation

To verify that everything is installed correctly, run the Poisson equation demo `ft01_poisson.py` (source) in your container.

If you installed the `fenics` bash script per the instructions above, you can use the following command (working directory: `spin-transport`).

```
$ fenics ft01_poisson.py
```

If everything is working fine, the output should look something like the following.

```
$ fenics ft01_poisson.py
Calling DOLFIN just-in-time (JIT) compiler, this may take some time.
--- Instant: compiling ---
Calling FFC just-in-time (JIT) compiler, this may take some time.
Calling FFC just-in-time (JIT) compiler, this may take some time.
Calling FFC just-in-time (JIT) compiler, this may take some time.
Solving linear variational problem.
*** Warning: Degree of exact solution may be inadequate for accurate result in errornorm.
Calling FFC just-in-time (JIT) compiler, this may take some time.
Calling FFC just-in-time (JIT) compiler, this may take some time.
  Ignoring precision in integral metadata compiled using quadrature representation. Not implemented.
Calling FFC just-in-time (JIT) compiler, this may take some time.
Calling FFC just-in-time (JIT) compiler, this may take some time.
Calling FFC just-in-time (JIT) compiler, this may take some time.
error_L2  = 0.00823509807335
error_max = 1.33226762955e-15
```

The directory `spin-transport/poisson` should have been created and should contain two files: `solution.pvd` and `solution000000.vtu`. These files contain the solution data.

### 2.4 Acknowledgement

This work is supported by a grant from the Army Research Office, Materials Science Division under grant proposal **Nanoscale Spin Hyperpolarization and Imaging** with PI John Marohn, PhD.

### 2.5 Contributors

This project stems from a collaboration among three institutions:

**[Cornell University** (http://www.cornell.edu/),

**[Saint Martin's University** (https://www.stmartin.edu/), and the

**[University of Washington** (http://www.washington.edu/).

The lead contributor to this project is Rico Picone, PhD of Saint Martin's University, co-PI on the ARO grant. Other contributors include John Marohn, PhD (Cornell, PI), John A. Sidles, PhD (Washington), Joseph L. Garbini, PhD (Washington), and Corinne Isaac (Cornell).

## 3 Short theoretical introduction

Here's some code.

```
x = 5
if x > 3:
  print('big!')
```