`nard` Documentation

Jacob Cooper Vandenberg

7 August 2021

Contents

1 Introduction

1.1 What it does

`nard` is a numerical partial differential equations solver for the Reaction-Diffusion equation. It solves the Reaction-Diffusion equation, written in the following form.

$$\frac{\partial \mathbf{u}}{\partial \mathbf{t}} = \mathsf{D}(\mathbf{u}, \mathbf{x}, t)\nabla^2 \mathbf{u} + \mathbf{F}(\mathbf{u}, \mathbf{x}, t) \tag{1}$$

In this equation, D is a user supplied function referring to the diffusion constant at a given point in space and time, and can depend on the chemical concentration. $\mathbf{F}$ is a user supplied function which determines the reaction term. $\mathbf{u}$ is a vector of concentrations, and this is the function being solved for. $\mathbf{x}$ and t are the space and time coordinates respectively. ∇^2 is the Laplacian operator.

Currently it only works in two spatial dimensions, but three spatial dimensions will be implemented. Currently advection is not supported, but this is to be implemented in future.

1.2 Time discretisation

`nard` currently has two options for time stepping methods. Both are IMEX (Implicit-Explicit) methods, which means that part of the equation is handled by an implicit Runge-Kutta method, and part of the equation is handled by an explicit Runge-Kutta method. The diffusion term is handled by the implicit method to improve stability, and the reaction term is handled by the explicit method to improve versatility and computational cost. The following table shows the time stepping methods.

Reference #	Implicit Method	Explicit method	Order
1	Backwards Euler	Euler	1
2	Heun	Crank-Nicolson	2

To choose which timestepping method you want, you will set `iparms(6)` to the Reference number of the desired method.

1.3 Space discretisation

`nard` uses the standard Finite-Difference approximation for the Laplacian operator. This is order 2, and allows for great flexibility with the timestep size.

2 Installation

2.1 Prerequisites

`nard` has been tested and developed on 64 bit Linux (Ubuntu 18.04 and 20.04). In order to install `nard`, you need to have the following packages installed and added to PATH.

- `gfortran`
- `gnuplot`
- OpenMP
- Intel MKL
- HDF5

2.2 Installation

To install, first clone into the `nard` project (found here).

```
git clone git@github.com:JacobVandenberg/nard.git
```

`cd` into the created folder and make the tests to initialise nard. This is not necessary, but this will help to verify that the installation is correct. It will also compile many of the binaries which will be used when making a new reaction term. This compilation will be done on the first call of `nard new` is not sone here.

```
cd nard
make tests
```

It is also recommended to add `nard` to the PATH. This can be done by adding the following line to the end of your `.bashrc` file (found at `/.bashrc`). (replace '`path/to`' with the full path to the enclosing folder). Restart bash to take effect.

```
export PATH=path/to/nard:$PATH
```

3 Usage

`nard2` is the solver which utilises 2 spatial dimensions and has two options, `new` and `run`.

3.1 `nard2 new`

Usage:

```
nard2 new [function_file.f90] [reaction name]
```

Here `[function_file.f90]` is the path to the fortran file which contains the functions specifying the reaction term and the diffusion matrix. `[reaction name]` is the name the user assigns the reaction, and will be used by the user when running the reaction with a config. Best not to use special characters, especially space or any slash characters because this will be used as a filename. Numbers, letters and underscores should be ok. Using the same name as a previous reaction will overwrite the previous reaction.

3.2 `nard2 run`

Usage:

```
nard2 run [reaction name] [config.h5]
```

This will run nard with the reaction name (as specified when making the new reaction), and a config file (which is in the .h5) format. The specifications of the config file are given in 4

4 Config File Specifications

The config file is an h5 file, with a number of datasets, as follows. MATLAB code is provided for writing h5 config files. A config `struct` should be made with values associated with the same name and value as the corresponding h5 file.

5 User Function Specifications

To specify the functions in the reaction diffusion equation, as follows, we will provide them in a `fortran` file.

$$\frac{\partial \mathbf{u}}{\partial \mathbf{t}} = \mathsf{D}(\mathbf{u}, \mathbf{x}, t)\nabla^2 \mathbf{u} + \mathbf{F}(\mathbf{u}, \mathbf{x}, t). \tag{2}$$

Example files are provided in `nard/src/user_functions`. To specify a new reaction scheme you need to provide a single `fortran` file which defines a `module user_functions`. This module will `use precision`, and `contains` two subroutines. The subroutine `reaction_term` will specify the function F,

Table 1: Config File Specifications

Handle	Type	Explanation
`"x"`	1D vector of `float64`	Grid points along the x axis. Must be equispaced.
`"y"`	1D vector of `float64`	Grid points along the x axis. Must be equispaced.
`"diffusion_consts"`	1D vector of `float64`	If diffusion is autonomous, we can specify diffusion constants like this. Ensure that `iparams(7)` is set to 0 to take advantage of autonomous diffusion
`"user_params"`	1D vector of `float64`	User definable parameters which are passed into the diffusion and reaction functions.
`"rparams"`	Length 64 1D vector of `float64`	Assorted real parameters, as defined in 2
`"iparams"`	Length 64 1D vector of `int64`	Assorted integer parameters, as defined in 3
`"DCBx_plus"`	1D vector of `float64`	Vector of floats specifying the value of the fixed Dirichlet boundary condition on the positive x boundary
`"DCBx_minus"`	1D vector of `float64`	Vector of floats specifying the value of the fixed Dirichlet boundary condition on the negative x boundary
`"DCBy_plus"`	1D vector of `float64`	Vector of floats specifying the value of the fixed Dirichlet boundary condition on the positive y boundary
`"DCBy_minus"`	1D vector of `float64`	Vector of floats specifying the value of the fixed Dirichlet boundary condition on the negative y boundary
`"DCBx_plus_mask"`	1D vector of `int64` (0/1)	Vector of booleans specifying True if there is a fixed Dirichlet boundary condition on the positive x boundary
`"DCBx_minus_mask"`	1D vector of `int64` (0/1)	Vector of booleans specifying True if there is a fixed Dirichlet boundary condition on the negative x boundary

Handle	Type	Explanation
`"DCBy_plus_mask"`	1D vector of `int64` (0/1)	Vector of booleans specifying True if there is a fixed Dirichlet boundary condition on the positive y boundary
`"DCBy_minus_mask"`	1D vector of `int64` (0/1)	Vector of booleans specifying True if there is a fixed Dirichlet boundary condition on the negative y boundary
`"IC"`	2D matrix of `float64`	Matrix of floats specifying the initial conditions
`"savefilename"`	string	gives the path for the save file
`"plotfilename"`	string	gives the path for the plot png file

Table 2: `rparams` specifications

Parameter	Typical value	Explanation
`rparam(1)`	Depends on problem	dt: the timestep interval. Dictates how much time elapses between each time step.
`rparam(2)`	Depends on problem	Maximum time: changes upper bound of the time interval solved over. Should be significantly larger than dt
`rparam(3)`	10.0 - 60.0	Plot interval (seconds): how often (in real time) the solution is plotted.

Table 3: `iparams` specifications

Parameter	Typical value	Explanation
`iparams(1)`	100 - 1000	Number of saved timesteps. If this is larger than the total number of time steps, then all timesteps are saved.
`iparams(2)`	10^{10}	Maximum save size in bytes. Safety factor to prevent very large files.
`iparams(3)`	0	1 for periodic boundary conditions in x.
`iparams(4)`	0	1 for periodic boundary conditions in y.
`iparams(5)`	0	1 for periodic boundary conditions in z.
`iparams(6)`	1	Time stepping method. See Table 1.2.
`iparams(7)`	0	1 is for if diffusion is non-autonomous.

and the subroutine `diffusivity` implements the function D. The arguments of both functions are identical, both functions take 4 arguments, as follows.

- `u_in`: the concentration of each chemical at each gridpoint. Each column specifies a different chemical. So chemical 1 can be accessed with `u_in(:, 1)`. In general, the concentration of chemical j at gridpoint i is stored in `u_in(i, j)`.
- `x_in`: specifies the grid values. Each column is a flattened `meshgrid`. `x_in(:, 1)` contains the x coordinate of the j-th grid point. Correspondingly `x_in(:, 2)` are the y-coordinates.
- `t`: This is the time.
- `u_out`: This is the output variable. It has the same format as `u_in`.
- `user_params`: This is a constant 1D array of reals which is passed though from the config file, and is defined by the user.

6 MATLAB Functions

MATLAB functions can be found in `nard/bin/MATLAB`. Here the following 3 functions can be found.

- `h5Animation2D`
- `nard2`
- `write_config`

6.1 `h5Animation2D`

This function makes an animated heatmap plot for a given results file `fname`. Extra plotting parameters can be paeed in using a struct `extra_params`
Usage:
`h5Animation2D(fname, extra_params)`
Arguments:

`fname`: this if the filename of the h5 file which contains the results.

`extra_params`: this is a `struct` with the following possible entries.

extra_params.real_time (float): how long the animation should go for in real time in seconds [DEFAULT: 10.0 seconds]

extra_params.sim_interval	(float, size = (2,)): time intervals between which the result should be animated (inclusive). [DEFAULT: full range of t values]
extra_params.fps	(float): the frames per second of the output animation [DEFAULT: 20 fps]
extra_params.dpi	(int): the dots per inch of the output animation [DEFAULT: 200]
extra_params.interpolate_resolution	(int, size = (2,)): whether to interpolate the result to a finer mesh before plotting. Set to 0, or set one of the values to 0 to prevent interpolation. [DEFAULT: 0]
extra_params.range_max	(float, size =({# of chemical species},)): sets the scale of the plot. Each value is the maximum of the range for the respective chemical species.
extra_params.range_min	(float, size =({# of chemical species},)): sets the scale of the plot. Each value is the minimum of the range for the respective chemical species.

6.2 `nard2`

This is a wrapper for the `nard2 run` terminal command. Usage:
`nard2(reaction_name, config)`
Arguments:

reaction_name	(string): the name of the reaction name, which was given to `nard` when `nard2 new` was called.
config	(string): the filename of the config file to pass to `nard`.

6.3 `write_config`

This writes a config, specified using a MATLAB `struct` into an h5 file for use by `nard`. Usage:
`write_config(conf, filename, force)`
Arguments:

`conf`	(struct): The config struct object. An example of one of these can be found in
`filename`	(string): the filename of the config file to pass to `nard`.
`force`	(logical): Force the function to overwrite any existing config with the same filename.