

# DXSpiral Version 1.0 : User's Guide

D. Barkley, V. N. Biktashev, I. V. Biktasheva, G. V. Bordyugov, A. J. Foulkes

September 17, 2010

## Abstract

DXSpiral is a collection of programs dealing with spiral waves on a polar grid in a disk, including finding their response functions. It is a public release version of the code used for obtaining results reported in [1, 2, 3, 4, 5], light-weighted by removing the most specialized bits.

## 1 Purpose

DXSpiral is a toolkit for calculating solutions of the following problems:

Reaction-diffusion system of equations in a disk of radius  $R$ :

$$\partial_t \mathbf{u} = \mathbf{f}(\mathbf{u}, \mathbf{p}) + \mathbf{D} \nabla^2 \mathbf{u}, \quad |\vec{r}| < R, \quad \vec{r} \cdot \mathbf{D} \nabla \mathbf{u} = 0, \quad |\vec{r}| = R, \quad (1)$$

where  $\mathbf{u}(\vec{r}, t) = (u_1, \dots, u_\ell)^T$  is a column-vector of the reagent concentrations,  $\mathbf{f}(\mathbf{u}; \mathbf{p}) = (f_1, \dots, f_\ell)^T$  is a column-vector of the reaction rates depending also on a vector of parameters  $\mathbf{p} \in \mathbb{R}^k$ ,  $\mathbf{D}$  is a diagonal  $\ell \times \ell$  matrix of diffusion coefficients, and  $\vec{r} \in \mathbb{R}^2$  is the vector of coordinates in the plane. Functions  $\mathbf{f}$  are for three selected models: FitzHugh-Nagumo and Barkley models (both for  $\ell = 2$ ,  $k = 3$ )

Nonlinear problem

$$\begin{aligned} \mathbf{f}(\mathbf{U}, \mathbf{p}) - \omega \mathbf{U}_\theta + \mathbf{D} \nabla^2 \mathbf{U} &= 0, \quad |\vec{r}| < R, \quad \vec{r} \cdot \mathbf{D} \nabla \mathbf{U} = 0, \quad |\vec{r}| = R, \\ P(\mathbf{U}(\vec{r}_P)) &= 0. \end{aligned} \quad (2)$$

The unknowns in this problem are function  $\mathbf{U}(\vec{r}) \in \mathbb{R}^\ell$ , the *spiral wave solution*, and scalar  $\omega$ , the *rotation frequency*. The vector PDE is for a stationary solution of (1) considered in a frame of reference rotating with angular frequency  $\omega$ , clockwise for  $\omega > 0$ . The finite scalar equation is the *pinning condition*: without it, the solution of (2) would be non-unique because of rotational symmetry. Currently the pinning condition is ‘hard-wired’ as  $P(\mathbf{U}) = U_2 - u_*$ ,  $u_* = 0.1$ ,  $|\vec{r}_P| = R$ , whereas direction of  $\vec{r}_P$  is arbitrary, depending on initial approximation. **The details of the pinning condition should be made parameters and/or model-specific in future releases.**

Linear problems

$$\begin{aligned} (\mathbf{D} \nabla^2 - \omega \partial_\theta + \partial_{\mathbf{u}} \mathbf{f}(\mathbf{U})) \mathbf{V}^{(n)} &= (i\omega n + \mu_n) \mathbf{V}^{(n)}, \quad \left[ \vec{r} \cdot \mathbf{D} \nabla \mathbf{V}^{(n)} \right]_{|\vec{r}|=R} = 0, \\ (\mathbf{D} \nabla^2 + \omega \partial_\theta + \partial_{\mathbf{u}} \mathbf{f}(\mathbf{U})^T) \mathbf{W}^{(n)} &= (-i\omega n + \nu_n) \mathbf{W}^{(n)}, \quad \left[ \vec{r} \cdot \mathbf{D} \nabla \mathbf{W}^{(n)} \right]_{|\vec{r}|=R} = 0, \end{aligned} \quad (3)$$

for  $n = 0, \pm 1$  (obviously the case of  $n = -1$  is the complex conjugate to that of  $n = 1$ ), where  $\mathbf{U}$  is a solution of (2). Here the unknowns are the Goldstone Modes  $\mathbf{V}^{(n)} \in \mathbb{C}^\ell$ , the Response Functions  $\mathbf{W}^{(n)} \in \mathbb{C}^\ell$  and the eigenvalues  $\mu_n, \nu_n \in \mathbb{C}$  which are expected to be small when  $R$  is large. In the same limit, the solutions for  $\mathbf{V}^{(n)}$  can be obtained explicitly via derivatives of  $\mathbf{U}$  (“analytical solution”). In the formulation (3), the solutions for  $\mathbf{V}^{(n)}$  and  $\mathbf{W}^{(n)}$  are non-unique due to linearity, so *normalization conditions* are required. We use normalization conditions based on expected properties of inner products of  $\mathbf{V}^{(n)}$  and  $\mathbf{W}^{(n)}$  for large  $R$ ; see [1] for details.

## 2 Tests

### 2.1 1: Barkley model, starting from EZSPIRAL data

This test uses file `fc.dat` in subdirectory `EZSPIRAL`. The file is the result of an interactive run of `EZSPIRAL` with the task file placed in the same subdirectory. The interactive run was done in such a way that to ensure the spiral wave rotates around the centre of the box, by switching tip tracing on and off and using arrow keys to move the spiral.

This DXSpiral test can be run using command

```
make test1
```

which in turn will lead to execution of individual sub-tests, such as

```
make test1-2dx
make test1-omg
make test1-non
make test1-lin
make test1-int
```

etc (see `Makefile`). The sub-tests can be called in any order, provided the pre-requisites already exist, as the targets of all test rules are phony. This is done so that to avoid the necessity of doing all tests from the beginning after an insignificant correction of a dependency. All the intermediate and final results of this test are placed in subdirectory `test1`; if it does not exist, it will be created (similarly for other tests).

If you have `X11` graphics on, the tests will be accompanied by graphical illustrations of results obtained; otherwise, you will only see the messages in the terminal window and result files in subdirectory `fhn`.

The individual steps (subtests) do the following:

- **test1-2dx**: converts the `EZSPIRAL` data into DXSpiral data. This assumes that the centre of rotation of spiral is in the centre of the box, and uses the maximal inscribed disk of the box. The  $\omega$  component of the resulting DXSpiral solution is zero, as there is no information about  $\omega$  in the `EZSPIRAL` data.
- **test1-omg**: estimates  $\omega$  for the result of the previous step.
- **test1-non**: finds the spiral wave solution of eigen/boundary value problem (2) using the result of the previous steps as an initial guess. No parametric continuation is done in this simplest example. Note that we know the size of the disk that was produced by `ez2dx`: it is half the box size of `EZSPIRAL/task.dat`.
- **test1-lin**: Solves the linear problems (3) for the solution obtained in the previous steps.
- **test1-int**: calculate some of the integrals using solutions of the linear and nonlinear problems.

### 2.2 2: FitzHugh-Nagumo model, starting from EZRide data

This test uses files `fhn1/fc.dat` and `quot.dat` in subdirectory `EZRide/fhn1`. The file is the result of a consecutive fully automatic run of `EZRide` with the task files `fhn0.task` and `fhn1.task` placed in `EZRide` subdirectory (see `EZRide/Makefile`). This has produced a spiral wave which is not centered but whose center of rotation and angular velocity can be found based on data in `quot.dat`.

The test can be run using command

```
make test2
```

and the individual steps do the following:

- **test2-2dx**: convert the **EZRide** data into the **DXSpiral** data. Program **ez2dx** calculates the position of the centre of rotation of the spiral in the **EZRide** box and choses the **DXSpiral** disk as the maximal inscribed disk centered at that point. Hence the disk radius is not known *a priori* but is a result of calculations made at this step. It happens to be  $R = 14.0282$  (see `test2/ez2dx.log`).
- **test2-non**: solves the nonlinear problem. Here we pretend that we need the solution in a disk of a particular radius,  $R = 12.8$ , so **dxnon** continues the resulting spiral wave solution to the required disk radius.
- **test2-lin**: solves the linear problem.
- **test2-int**: calculates some of the integrals.

### 2.3 3: FitzHugh-Nagumo model, starting from scratch

This test uses **DXSpiral**'s own timestepper **dxtime** to obtain an initial approximation.

- **test3-arch**: Creates a “cross-field” initial condition for a spiral wave in a disk of radius  $R = 12.8$ . The cross-field values are specified in file **fhn1.rec**: columns correspond to dynamic variables and rows to records equidistant along a period of a periodic plane wave solution. In our case, the values are chosen by hand and there is four of them; the corresponding values are assigned to the four 90°-sectors of the disk. The “cross-field” is a bit unusual in that the lines separating different values of dynamic variables are not straight, but Archimedean spirals, with the pitch  $\lambda = 10$ .
- **test3-time1**: Starts from this initial condition and calculates a spiral wave by running forward in time, *i.e.* solving (1).
- **test3-time2**: Continues time run, now gradually decreasing disk radius down to  $R = 4.8$ . The purpose of this and the next steps is to force the spiral to rotate around the centre of the disk, with sufficient accuracy.
- **test3-time3**: Continues to run further in the same disk with  $R = 4.8$ , to let the spiral to settle to a circular motion around the disk centre.
- **test3-omega**: Endows the previously obtained spiral wave solution with an estimate of its  $\omega$ .
- **test3-non1**: solves the nonlinear problem in the disk  $R = 4.8$  using the result of the previous step as an initial approximation, and then continues it to a bigger disk  $R = 12.8$ .
- **test3-non2**: interpolates the previous result to a finer grid, from **nr** = 48 to **nr** = 160, and solves the nonlinear problem again (no further parameter continuation is done).
- **test3-lin**: solves the linear problem.
- **test3-int**: calculates some integrals.

Compare the results of tests 2 and 3. They are for the same problem but obtained from different initial approximations and to a different radial resolution. The spiral wave solutions are not close, as their angular position is arbitrary (depends on initial approximation). However the results of integration are comparable, with the exception of the  $x$  and  $y$  components of the resonant drift velocity (which depends on the choice of the angular position of the spiral wave). Naturally, the finer resolution of test 3 makes its results more accurate (compare with [1]).

### 3 Data formats

First we describe the format of data with which all the individual programs operate. The data represent the solution of one of the above problems, real- or complex-valued, on a polar grid with regular radial and angular spacings. Its syntax stems from EZ-software: first several lines describe some parameters in ASCII, one parameter per line, with the rest of the line treated as a comment. After that go the data in the said format, in a certain order described below.

The parameter lines go like this:

- Line 1: **model\_name** - character string. Currently recognized values are: “FitzHugh-Nagumo” or “FHN”, “Barkley” or “bkl”, and “Beeler-Reuter-M”.
- Line 2: **np** - integer, number of model parameters. Currently FitzHugh-Nagumo depends on 3 parameters, Barkley model depends on 3 parameters and Beeler-Reuter depends on 5 parameters. The number specified here must match the number expected for the selected model.
- Line 3: **nv** - integer, number of dynamic variables (components) in the model. For both FitzHugh-Nagumo and Barkley the value is 2, for Beeler-Reuter it is 7. The number specified here must match the number expected for the selected model.
- Lines 4...**np** + 3 : reals, the values of the **np** model parameters.
- Lines **np** + 4...**np** + **nv** + 3 : reals, the values of the diffusion coefficients for every of the **nv** components of the model.
- Line **nv** + **np** + 4: **nt** - integer, the number of discretization points in the angular direction.
- Line **nv** + **np** + 5: **nr** - integer, the number of discretization points in the radial direction, not including the centre.
- Line **nv** + **np** + 6: only the first character in this line is important. This must be either ‘A’ or ‘B’, which states whether the subsequent data will be in Ascii or Binary formats respectively.

The order of data values:

- The value of  $\omega$ .
- The point record of the solution at the origin (the centre of the disk), comprising
  - The **nv** values of the dynamic variables
- The **nv** “rings”, each comprising
  - The radius of the ring, i.e. its distance from the origin,
  - The **nt** point records on that ring, each comprising
    - \* The **nv** values of the dynamic variables

For complex data produced by **dxlin**, the dynamic variable data double up: each entry presents first the real part and then the imaginary part of the corresponding element.

Apart from data files, some of the programs require “task” files. Their formats are described below together with the corresponding programs.

## 4 Individual programs in the kit

### Common features

The programs use input data (some don't), task files (some don't) and put the results into files with fixed names in an output directory. The output directory is specified on the command line. If the output directory does not exist it will be created. If files with the same names as output files already exist in the output directory, they will be overwritten. The messages are issued to the standard output and copied to file called `<name>.log` where `<name>` is the name of the program.

If you have `X11` on your computer and its use is enabled in the `Makefile`, then the results of calculations will be illustrated graphically, including the final results and sometimes intermediate results. Each colour component (red/green/blue) represents distribution of one of the dynamic variables, e.g. red corresponds to the activator  $x$ . The values of the variables are suitably scaled so bigger luminosity of colour corresponds to bigger value of the variable and vice versa. The scaling method differs for solutions of the linear problems (`dxlin`) and nonlinear problems (all the rest that produce graphics). For linear problems, zero is mapped to 50% luminosity, and either 0% or 100% luminosity is achieved at least at one point. For nonlinear problem, the values corresponding 0% and 100% luminosity are “hard-wired” in the code, model-specific. Note that solutions of linear problems are complex, so each is shown twice: for the real and for the imaginary part.

If you have `libjpeg` installed on your computer and its use is enabled in the `Makefile`, then the output directories will contain JPEG images. The JPEG images are all grayscale and correspond to all, rather than selected three, dynamic variables (this comment obviously is not essential for FHN and Barkley model, but will be for models with  $nv > 3$ ). Solutions of linear problems are complex and their real and imaginary parts are written in separate files, which is reflected in the files' names. Each JPEG image `<name>.jpg` corresponding to a linear solution accompanied by the corresponding file `<name>.max` which is a plane text file containing a positive real number  $A$ . This number defines the value scale for the image: the white of the image corresponds to value  $A$  of the corresponding variable, and the black of the image corresponds to value  $-A$ .

### Groups of programs

We shall now proceed to description of individual programs, by groups:

- first the most essential (`dxnon`, `dxlin` and `dxint`),
- then convenience programs used in the tests (`dxarch`, `dxtime`, `dxomega`, `ez2dx`),
- and finally other programs which are not used in the tests but which the user might find convenient in some circumstances (`dxflip`, `dxreport`).

## dxnon — Solve the nonlinear problem

### Call:

`dxnon <task file> <input data file> <output directory>`

### Purpose:

This program solves the nonlinear problem (2) by Newton iterations and, if asked, parameter continuation. The parameter continuation is via straightforward division of the parameter range to a specified number of parameter steps, and then solving the nonlinear problem for each step, taking previous solution as the initial approximation for the next step, without anything sophisticated like prediction-correction. The continuation is done along a segment of a straight line in the  $\mathbf{np} + \mathbf{nv} + 1$ -dimensional parametric space ( $\mathbf{np}$  kinetic parameters,  $\mathbf{nv}$  diffusion coefficients and disk radius  $R$ ). The beginning of this segment corresponds to the parameters in the input data file, and the end of this segment corresponds to the parameters in the task file. If the grid in the task file and in the input data files do not coincide, then interpolation (bi-linear in polar coordinates) to the task grid is done first. If parameter  $R$  (the disk radius) changes during parameter continuation, the data are not interpolated from the old to the new grid on every step, instead the grid itself is considered expanded or compressed (thus change of  $R$  should never be done in too large steps).

The Newton iteration are also done in a straightforward manner, controlled by the norm of the residual, which is Euclidean norm of a vector of differences between left- and right-hand sides of the discretized system of equations. The only deviation is: if a given Newton iteration would lead to an increase rather than a decrease of the residual norm, then only half of the Newton step is tried, and if that is still too much, then a quarter is tried, and so on, until the resulting residual falls below the previous value, in which case we accept the step and proceed to the next iteration, or the fractional steps becomes unreasonably small, in which case we admit defeat and stop the program with an error message and nonzero exit code.

### The task file

specifies a number of compulsory parameters and a number of optional parameters. If an optional parameter is not specified, a default value is taken.

The compulsory parameters are in the top lines of the file:

- Line 1: model name
- Line 2:  $\mathbf{np}$ , the number of model parameters
- Lines 3... $\mathbf{np} + 2$ : the  $\mathbf{np}$  values of model parameters
- Line  $\mathbf{np} + 3$ :  $\mathbf{nv}$ , the number of model dynamic variables
- Lines  $\mathbf{np} + 4$ ... $\mathbf{np} + \mathbf{nv} + 3$ : the  $\mathbf{nv}$  values of diffusion coefficients
- Line  $\mathbf{np} + \mathbf{nv} + 4$ :  $\mathbf{nt}$ , the number of angular discretization intervals
- Line  $\mathbf{np} + \mathbf{nv} + 5$ :  $\mathbf{nr}$ , the number of radial discretization intervals
- Line  $\mathbf{np} + \mathbf{nv} + 6$ :  $R$ , the disk radius.

The optional parameters are given in subsequent lines in the following order (we give the default values in parentheses):

- $\mathbf{nsteps}$  (0): the number of parameter continuation steps. If  $\mathbf{nsteps} = 0$ , no parameter continuation is done, parameters given in the input data file are ignored and the problem is solved straight at the parameter values specified in the task. If  $\mathbf{nsteps} = 1$ , then the problem is solved first at the parameters given in the input data file, then the result is used as an initial approximation for the problem at the output data file, etc.

- **binary** (1): if zero, resulting data file is written in ascii, otherwise binary.
- **verbose** (1): 0 for none except fatal error messages, 1 for very brief, 2 for more detailed etc; values above 5 make no further difference.
- **newt\_necc** ( $10^{-3}$ ): the convergence is considered failed unless the residual falls below this level before the maximal number of iterations (see below) is done.
- **newt\_suff** ( $10^{-11}$ ): the convergence will is considered achieved as soon as the residual falls below this level.
- **maxit** (25): the convergence is considered failed if the residue will not fall below **newt\_necc** after this many iterations.

If a line is empty or value in it cannot be recognized, the parameter is given the default value. If there are too few lines, the parameters that would be in the missing lines are given the default values,

### Output:

- **spiral.dat** — the resulting solution data file,
- **spiral\*.jpg** — JPEG images of the components of the solution,
- **spiral-ravg.dat** — ASCII file with dependence of the components of the solution on the radius (angle-averaged absolute values of the dynamic variables).

Also, depending on the value of **verbose**, optional data that may be useful for understanding convergence problems:

- **newton.dat** — result of last Newton iteration.
- **newton-\*\*-\*.jpg** — JPEG images of Newton iterations (first star replaced by the step number, the second with the iteration number).
- **read.jpg** — picture of the very first initial approximation obtained from the input data file.
- **interpolated.jpg** — same, after interpolation (if task grid is different from input grid)
- **initial.jpg** — same, after rotating the initial approximation into the standard position according to the pinning condition.

## **dxlin — Solve the linear problem**

### **Call:**

`dxlin <task file> <output directory>`

The output directory must exist, and the input data file, representing solution of the nonlinear problem (2), is looked for in that directory, under the name `spiral.dat` (as it would be created by `dxnon`).

### **Purpose:**

This program solves the linear problems (3), that is finds both Goldstone Modes (eigenfunctions of the linearized operator) and Response Functions (eigenfunctions of the adjoint linearized operator). The Goldstone Modes are found in two ways: by numerical differentiation of the spiral wave solution, and by solving the eigenvalue problem by the same method as used for finding Response Functions, and which is described in the Programmers Guide and in the journal publication [1].

### **The task file:**

contains no compulsory parameters. The optional parameters are:

- `binary (1)`: same meaning as in `dxnon`.
- `verbose (1)`: same meaning as in `dxnon`.
- `kryl_dim (10)`: Krylov subspace dimensionality (see the discussion in the Programmer's Guide and in [1])

### **Output:**

- `GM<n>.dat`  $n = 0, 1, 2$ : the Goldstone Modes  $\mathbf{V}^{(k)}$ , calculated by differentiation of the given spiral wave solution, where  $k = n - 1$ .
- `EF<n>.dat`: same, calculated by solving the eigenvalue problem.
- `RF<n>.dat`,  $n = 0, 1, 2$ : Response Functions  $\mathbf{W}^{(k)}$ ,  $k = n - 1$ .
- `<pref><n>-<part>-<comp>.jpg`, for `pref=gm,ef,rf`,  $n = 0, 1, 2$ , `part=re,im`, `<comp>=0..nv-1`: corresponding JPEG images.



## dxint — Calculate the simpler integrals of response functions

### Call:

`dxint <output directory>`

The output directory must exist and contain files `spiral.dat`, `GM<n>.dat`, `EF<n>.dat`, `RF<n>.dat`,  $n = 0, 1, 2$ , produced previously by `dxnon` and `dxlin`.

### Purpose:

This program uses the results produced previously by `dxnon` (file `spiral.dat`) and `dxlin` (files `GM<n>.dat`, `EF<n>.dat`, `RF<n>.dat`,  $n = 0, 1, 2$ ) to calculate several integrals, used for prediction of the spiral wave drift, as described in [1] and [3], namely:

$$\langle \mathbf{W}^{(i)} | \mathbf{V}^{(j)} \rangle = \delta_{i,j}, \quad i, j = 0, \pm 1, \quad (4)$$

$$\langle \mathbf{W}^{(1)} | \mathbf{D}\mathbf{V}^{(1)} \rangle = b_2 + ic_3, \quad (5)$$

$$\langle \mathbf{W}^{(0)} | \mathbf{D}\partial_\theta \mathbf{V}^{(0)} \rangle = a_0, \quad (6)$$

$$\langle \mathbf{W}^{(0)} | \mathbf{D}\mathbf{V}^{(0)} \rangle = b_1, \quad (7)$$

$$\langle \mathbf{W}^{(1)} | \mathbf{P}_k \mathbf{V}^{(1)} \rangle, \quad k = 1 \dots \text{nv}, \quad (8)$$

$$\langle \mathbf{W}^{(i)} | \frac{1}{2} \mathbf{e}_k \rangle, \quad i = 0, \pm 1, \quad k = 1 \dots \text{nv}, \quad (9)$$

$$\langle \mathbf{W}^{(1)} | \frac{1}{2} \rho e^{-i\theta} \frac{\partial \mathbf{U}}{\partial p_\ell} \rangle, \quad i = 0, \pm 1, \quad \ell = 1 \dots \text{np}, \quad (10)$$

$$(11)$$

where

$$\langle \mathbf{w} | \mathbf{v} \rangle = \int_{\mathcal{D}} \mathbf{w}^\dagger(\vec{r}) \mathbf{v}(\vec{r}) d^2 \vec{r}, \quad (12)$$

vector  $\mathbf{e}_k$  is a unit column-vector with  $k$ -th component equal to one and the rest zero, and  $\mathbf{P}_k = \mathbf{e}_k \mathbf{e}_k^T$ . Integrals (4)–(8) are calculated with  $\mathbf{V}^{(j)}$  both “analytical” (obtained by differentiation of  $\mathbf{U}$ ) and “numerical” (obtained by solving (3)).

### The task file:

Not used.

### Output:

The values of integrals (4)–(10) are printed as comments to standard output and `dxint.log`. Note that in the present version, in the symbolical notations for the integrals in the program output, the dynamic variables and parameters are enumerated starting from 0, whereas in formulas above they are enumerated by indices  $k$  and  $\ell$  starting from 1.

## ez2dx — Convert EZSPIRAL or EZRide data to DXSpiral data

### Call:

```
ez2dx <task file> <EZSPIRAL data file> <output directory>
    or
ez2dx <task file> <EZRide data file> <EZRide quotient file> <output directory>
```

### Purpose:

Creates initial approximation for `dxnon` from results of calculations with `EZSpiral` ([http://www.warwick.ac.uk/~masax/Software/ez\\_software.html](http://www.warwick.ac.uk/~masax/Software/ez_software.html), [6]) or `EZRide` (<http://www.maths.liv.ac.uk/~vadim/software/EZRide/>, [7]).

The programs distinguishes between the two cases by the number of command-line arguments.

In principle, the result of conversion can be also used as initial condition for `dxtime`.

### The EZSPIRAL data file

is what EZSPIRAL itself would deal with as `ic.dat` or `fc.dat` file. It can be ASCII or binary. See EZSRIRAL documentation for the format of the file. We used version 3.2; if another EZSPIRAL version uses a different format, then `ez2dx.c` may need to be adjusted. Note that use of EZSPIRAL data requires special care: the user is expected to arrange that the spiral wave rotates around the centre of the box. Failure to do that is likely to produce unsuitable initial approximation fo `dxnon` (but makes no big difference to `dxtime`).

### The EZRide data file

For the purposes of `DXSpiral`, the format of this file is identical to that of EZSPIRAL except it reports in its first line the name of the model, which can be either `Barkley` or `FitzHugh-Nagumo` (synonym `FHN`). All comments from the previous paragraph apply for this case, except the one about centre of rotation. The current version of `ez2dx` is compatible with `EZRide` version 1.0.

### The EZRide quotient file

The advantage of `EZRide` calculations in “riding mode” is that it generates the spiral wave solution in a fixed position (tip of the spiral at the centre of the box and in a fixed orientation) and also the linear and angular velocity of the tip, as it would move if the riding mode is suddenly switched off. This allows one to determine the instant rotation centre for the spiral wave, so the manual adjustments of the spiral position, which was required for EZSPIRAL, is not required for `EZRide`. The determination of the centre may be not very precise, but it is automatic. For a `DXSpiral` user, it is only important to understand that the quotient file is a plain text file containing four coulmnns of numbers, and `ez2dx` only uses the last line of this file. Hence it is important that the last line is complete and contains four numbers.

### Output:

- `ez2dx.dat`, the result of conversion,
- `ez2dx-<n>.jpg`,  $n = 1 \dots nv$ , the corresponding images.

## **dxarch — Archimedean spiral initial condition**

### **Call:**

`dxarch <task file> <1D pulse profile> <output directory>`

### **Purpose:**

Creates initial conditions for `dxtime` using given 1D pulse profile and Archimedean phase distribution.

### **The task file:**

Compulsory parameters:

- Line 1: model name
- Line 2: `np`, the number of model parameters
- Lines 3...`np` + 2: the `np` values of model parameters
- Line `np` + 3: `nv`, the number of model dynamic variables
- Lines `np` + 4...`np` + `nv` + 3: the `nv` values of diffusion coefficients
- Line `np` + `nv` + 4: `nt`, the number of angular discretization intervals
- Line `np` + `nv` + 5: `nr`, the number of radial discretization intervals
- Line `np` + `nv` + 6: `R`, the disk radius.

Optional parameters:

- `1/λ` (0.0) where  $\lambda$  is the asymptotic wavelength of the Archimedean spiral. Positive value corresponds to spiral turning clockwise as it moves away from the centre.
- `binary` (1): as usual.
- `verbose` (1): as usual.

### **The 1D pulse profile:**

This file should contain the values of the dynamic variables within a period of a plane wave, in ASCII. The values should be arranged in `nv` columns. The lines will be interpreted as values of the variables at equidistant moments in time throughout the period of the wave. These values will be assigned to the points in the output “initial condition” data file depending on their polar coordinates: direction from top to bottom in the profile corresponds to counter-clockwise direction. The resulting data file represents a piece-wise constant function on the disk, where the number of pieces equals the number of lines in the 1D pulse profile.

The contents of 1D pulse profile may be obtained by a simulation code or written by hand. A case of four lines in the 1D pulse profile and  $1/\lambda = 0$  in the task file can represent to the classical “cross-field” stimulation, similar e.g. to that used in EZSPIRAL. This is what is done in the example profile `fhn1.rec`, only  $1/\lambda$  is taken nonzero.

### **Output:**

File `arch.dat` in the output directory is the generated data file that can be used as an initial condition for `dxtime`.

## dxtime — Time run

### Call:

`dxtime <task file> <input data file> <output directory>`

### Purpose:

This solves Cauchy problem for (1). The problem is described in the task file, and the initial conditions are taken from the input data file. If the model parameter values and/or value of  $R$  given in the task file differ from those given in the initial conditions, these parameters will be linearly changed in time, from the initial conditions parameters in the beginning to the task parameters in the end. The timestepping method is operator splitting, with reaction kinetics and radial derivatives treated explicitly and angular derivatives treated implicitly, similar to that used in [8].

### The task file:

Compulsory parameters:

- Line 1: model name
- Line 2: `np`, the number of model parameters
- Lines 3...`np` + 2: the `np` values of model parameters
- Line `np` + 3: `nv`, the number of model dynamic variables
- Lines `np` + 4...`np` + `nv` + 3: the `nv` values of diffusion coefficients
- Line `np` + `nv` + 4: `nt`, the number of angular discretization intervals
- Line `np` + `nv` + 5: `nr`, the number of radial discretization intervals
- Line `np` + `nv` + 6:  $R$ , the disk radius.

Optional parameters:

- `nsteps` (1000): number of time steps to be done
- `ts` (0.1): the time step
- `n_out` (1): if nonzero, the intermediate results will be output after every `n_out`'th timestep.
- `binary` (1): as usual.
- `verbose` (1): as usual.

### Output:

- `time.dat`: the final condition achieved so far, updated every `n_out` steps. After completion, contains the final condition.
- `time-<k>.jpg`, for  $k = 0 \dots nv - 1$ : the images corresponding to `time.dat`.
- `time-<i>-<k>.jpg`: images of intermediate results saved in separate files, rather than overwriting the same file over and over again — this happens when `verbose` > 4. This can be useful for making movies.
- `read.jpg` — picture of the very first initial approximation obtained from the input data file.
- `interpolated.jpg` — same, after interpolation (if task grid is different from input grid).

## **dxomega — Estimate $\omega$ for a solution**

### **Call:**

`dxomega <task file> <input data file> <output directory>`

### **Purpose:**

The nonlinear problem (2) differs from the dynamic equations (1), in particular, in that it has an extra unknown, the angular rotation frequency  $\omega$ . To proceed from solving time-dependent problem (1) to the boundary-value eigenvalue problem (2), one can use the final condition of (1) as an initial approximation for (2), but then one also needs an initial approximation for  $\omega$ . Program `dxnon` provides such an initial approximation. Let function  $\mathbf{u}(\vec{r})$ ,  $|\vec{r}| \leq R$ , be this final condition of (1), which is also the initial approximation of (2) and which is given in the input data file. The idea is to calculate  $\partial_t \mathbf{u}$  according to (1), and then the required estimate is

$$\omega = \frac{\int_{\mathcal{D}} (\partial_t \mathbf{u})^+ \partial_\theta \mathbf{u} d^2 \vec{r}}{\int_{\mathcal{D}} (\partial_\theta \mathbf{u})^+ \partial_\theta \mathbf{u} d^2 \vec{r}}.$$

(see [9] for why this is a reasonable estimate).

A further observation is that the Laplacian term in the right-hand side of (1) contributes nothing to the value of the numerator, as  $\oint (\mathbf{u}_{\theta\theta})^+ \mathbf{u}_\theta d\theta = 0$ , hence in the above formula we take  $\partial_t \mathbf{u} = \mathbf{f}(\mathbf{u})$ .

### **The task file:**

not taken. The value `verbose = 4` is “hard-wired”, and the output file will be written binary if and only if the input file was binary.

### **The input data:**

contains the solution  $\mathbf{u}$  for which  $\omega$  is to be estimated.

### **Output:**

`spiral-omega.dat`: A copy of the input data file with the  $\omega$  as estimated.

## dxflip — Flip the solution

### **Call:**

`dxflip <input data file> <output directory>`

### **Purpose:**

Invert the given solution about  $x$ -axis, that is,  $\theta \rightarrow -\theta$  or equivalently  $y \rightarrow -y$ , and also invert angular velocity,  $\omega \rightarrow -\omega$ . This may be needed if an initial approximation or initial condition is available in which the spiral rotates wrong way.

### **The task file:**

not taken. The value `verbose = 1` is “hard-wired”, and the output file will be written binary if and only if the input file was binary.

### **Output:**

- `flipped.dat`, the output data file
- `flipped-<n>.jpg`,  $n = 1 \dots nv$ , the corresponding images.

## **dxreport — Report on a solution**

### **Call:**

`dxreport <input data file> <output directory>`

### **Purpose:**

Print out the values of parameters of the solution in the given data file, including those that may be hidden in a binary format (namely  $\omega$  and  $R$ ), and visualize the solution.

### **The task file:**

not taken any.

### **Output:**

only report on the parameters to stdout with a copy to `dxreport.log`, and the images of the solution.

## Acknowledgement

Development of this software was supported in part by EPSRC grants EP/D074789/1 and EP/D074746/1.

## References

- [1] I. V. Biktasheva, D. Barkley, V. N. Biktashev, G. V. Bordyugov, and A. J. Foulkes. Computation of the response functions of spiral waves in active media. *Phys. Rev. E*, 79(5):056702, 2009.
- [2] V. N. Biktashev, D. Barkley, and I. V. Biktasheva. Orbital motion of spiral waves in excitable media. *Phys. Rev. Lett.*, 104(5):058302, 2010.
- [3] I. V. Biktasheva, D. Barkley, V. N. Biktashev, and A. J. Foulkes. Computation of the drift velocity of spiral waves using response functions. *Phys. Rev. E*, 81(6):066202, 2010.
- [4] A. J. Foulkes, D. Barkley, V. N. Biktashev, and I. V. Biktasheva. Alternative stable scroll waves and conversion of autowave turbulence. arXiv:1006.5650v3 [nlin.PS].
- [5] V. N. Biktashev, I. V. Biktasheva, and N. A. Sarvazyan. Evolution of spiral and scroll waves of excitation in a mathematical model of ischaemic border zone. arXiv:1006.5846v1 [q-bio.TO].
- [6] D. Barkley. A model for fast computer simulation of waves in excitable media. *Physica*, 49D:61–70, 1991.
- [7] A. J. Foulkes and V. N. Biktashev. Riding a spiral wave: Numerical simulation of spiral waves in a co-moving frame of reference. *Phys. Rev. E*, 81(4):046702, 2010.
- [8] E. V. Nikolaev, V. N. Biktashev, and A. V. Holden. On feedback resonant drift and interaction with the boundaries in circular and annular excitable media. *Chaos Solitons & Fractals*, 9(3):363–376, 1998.
- [9] I. V. Biktasheva, A. V. Holden, and V. N. Biktashev. Localization of response functions of spiral waves in the FitzHugh-Nagumo system. *Int. J. of Bifurcation and Chaos*, 16(5):1547–1555, 2006.