A Finite Volume Method for Stochastic Integrate and Fire Models

# DOCUMENTATION

# CONTENTS

# 1 Overview of the Code

Given two LIF neurons,V and W governed by the following Langevin equations :

$$V = f(V,W) + I_V(t)\,; \quad I_V(t) = \mu_V + \sqrt{1-c}\xi_V(t) + \sqrt{c}\xi_c(t)$$
$$W = g(W,V) + I_W(t)\,; \quad I_W(t) = \mu_W + \sqrt{1-c}\xi_W(t) + \sqrt{c}\xi_c(t).$$

receiving both independent inputs $\xi_V(t), \xi_W(t)$ and shared uncorrelated Gaussian inputs $\xi_c(t)$ , the Finite Volume Code computes the joint probability density (whose evolution is determined by the corresponding Fokker Planck Equation) as well as other quantities derived from this density.
In Particular , the code allows for the computation of the flux across the 2 boundaries as well the cross intensity function for both constant as well as time dependent parameters.

## 2 Installing the Code

On machines running LINUX ,to install the Finite Volume Code ,follow these steps :

1. Download the file *neuro2d.tar.gz* .
2. From the shell ,change the current directory to the one in which the file *neuro2d.tar.gz* was placed by using the *cd* command .
3. Decompress the file by using :

   *$ tar -zxvf neuro2d.tar.gz*

4. This will create a folder called *neuro2d* . *cd* to this folder from the shell.
5. Then type :

   $ *make cclean*

6. And finally,to compile the code, type:

   $ *./arrow*

The software is now installed .

## 3 Folders

In the primary folder *neuro2d*, the folder *ktest* contains a number of folders titled *test 1* ,*test 2* . Each of these folders ,(apart from other files) contains 2 files :

1. **PARAM.DAT**
2. **DOM.DAT**

The file **PARAM.DAT** contains the parameters used in the simulation as is explained in detail below . And, the file **DOM.DAT** contains the Mesh .

# 4 Running the Code

In order to run the code, follow these steps :

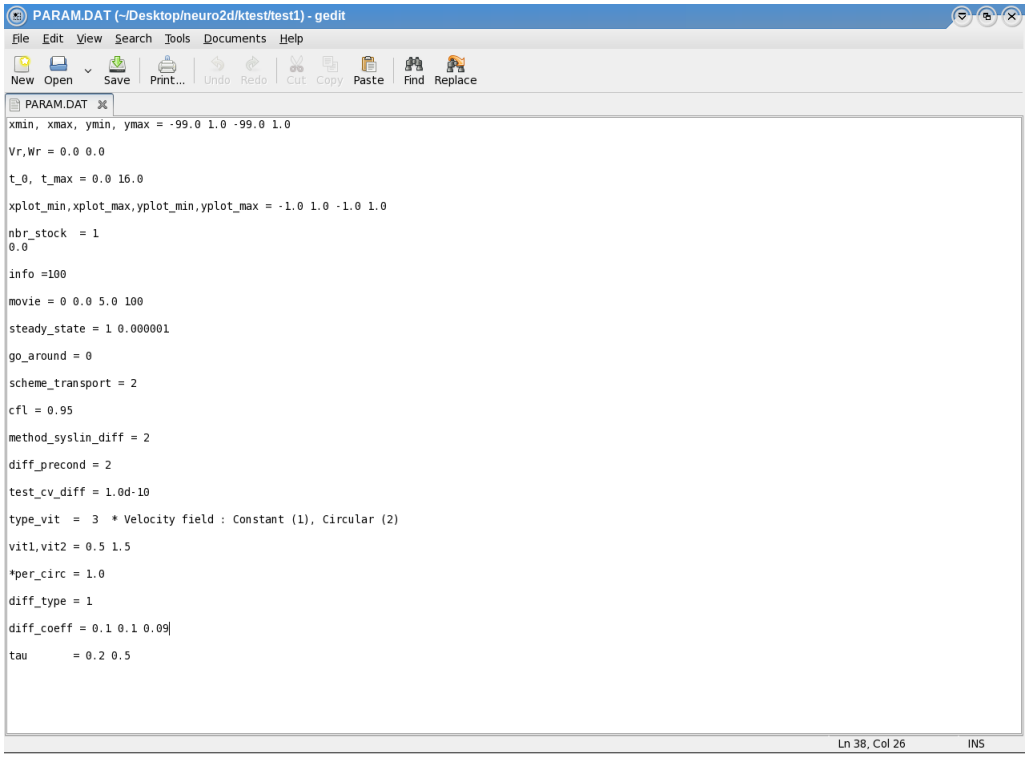
1. From the *neuro2d* directory,change the working directory to one of the test folders i.e. test 1, test 2 etc . For example, to work from test folder 1 , type :

$$\$ \; cd \; ktest/test1$$

2. Then, to run the code, type

$$\$ \; ./sabor < RUN.DAT$$

# 5 Modifying the Parameters for the simulation




The file **PARAM.DAT** contains the parameters. A line by line description of this file follows :

1. **Domain of solution**

   **xmin, xmax, ymin, ymax = a b c d**

   $[a, b] \times [c, d]$ is the domain over which the Fokker Planck Equation is solved.As an example , to solve the equation over the region $[-1, 1] \times [-1, 1]$, this line would be modified to :

   **xmin, xmax, ymin, ymax = -1.0 1.0 -1.0 1.0**

   This line also incorporates the threshold as xmax and ymax are the thresholds for Neuron 1 and Neuron 2 respectively .

2. **Reset Potentials**

   **Vr,Wr =** $V_{reset}$ $W_{reset}$

   Reset Potentials for the two neurons ( V and W) . For example, to change the reset potential of Neuron 1 to 0.1 and that of Neuron 2 to -0.1 , the line above would be changed to :

   **Vr,Wr = 0.1 -0.1**

3. **Time over which the solution is computed**

   **t_0, t_max =** $t_{initial}$ $t_{final}$

   $t_{initial}$ is initial time and $t_{final}$ is the maximum time till which the Fokker Planck equation is solved. It is not always the case that the equation is solved till $t_{final}$ . If a steady state solution is sought (to be explained) and the density converges before $t_{final}$ ,then $t_{final}$ is ignored.

4. **Region over which the solution is plotted**

   **xplot_min,xplot_max,yplot_min,yplot_max = a b c d**

   $[a, b] \times [c, d]$ is the domain over which the solution is to be plotted.

5. **Saving Results at specified points of time**

num_stock=n
$time_1$
$time_2$

n is the number of times the solution is saved, and the values below **nbr_stock** indicate the times at which the solution is to be saved. Suppose we wanted to save the solution at 5 points of times starting from 0.1 and ending at 0.5 , this line would be modified as :

**nbr_stock=5**
**0.1**
**0.2**
**0.3**
**0.4**
**0.5**

6. **Creating a Movie**

**movie** = *option* $t_{initial}$ $t_{final}$ $N$

option is a binary variable either 1 or 0, where 1 indicates that a movie is to be created and 0 indicates that a movie is not to be created. $t_{initial}$ is the starting time for the movie and $t_{final}$ is the ending time. $N$ is the number of frames . So , if we wanted to create a movie starting at time $t_{initial} = 0.5$ ,ending at $t_{final} = 10.0$ and consisting of 200 frames , the appropriate statement would be :

**movie = 1 0.5 10.0 200**

7. **Searching for a Steady State**

**steady_state** = *option convergence_criterion*

where *option* is a binary variable - either 1 or 0 , with 1 indicating that the code will search for a steady state solution and 0 indicating that the code will not look for a steady state solution . *convergence_criterion* is the ..

8. **Go Around**

9. **Transport Scheme**

**scheme_transport=n**

n=1 is for 1st order Murman ( Godunov ) scheme . And , n=2 is for a high-resolution Murman (Godunov) scheme with flux limiters( recommended because a lot more accurate)

10. **Setting the Courant-Friedrich-Levy ratio**

**cfl = x**

Select x to be a number between 0 and 1 .

11. **Setting the iterative method for solving linear systems**

**method_syslin_diff=n**

Use n=1 for Conjugate Gradient
Use n=2 for Bi-Conjugate Gradient with LU Preconditioner
Use n=3 for GMRES (minimal residual) method

12. **Diffusion Preconditioner**

**diff_precond=n**

13. **Convergence Criterion for solving the Linear System**

**test_cv_diff=$\epsilon$**

14. **Specifying the type of drift**

**type_vit=n**

Choose n=3 for constant drift
Choose n=4 for time dependent drift

15. **Specifying the Drift**

$$\boxed{\textbf{vit1,vit2=}\mu_1\ \mu_2}$$

$\mu_1$ is the input current to the 1st Neuron(V) and $\mu_2$ is the input current to the 2nd Neuron(W). As an example , if we wanted $\mu_1$ to be 1.5 and $\mu_2$ to be 0.5 , we would set :

**vit1,vit2=1.5 0.5**

16. **Diff**

$$\boxed{\textbf{diff\_type= n}}$$

17. **Specifying the Diffusion Coefficient**

$$\boxed{\textbf{diff\_coeff }=D\ D\ Dc}$$

The first 2 entries are the diffusion coefficients for the 2 neurons and the last entry is the product of the diffusion coefficient and the correlation between the 2 neurons . As an example , suppose we want the diffusion coefficient to be 0.1 for both the neurons and the correlation between the 2 neurons to be 0.5 , then this line would be modified to :

**diff_coeff=0.1 0.1 0.05**

18. **Specifying the refractory periods**

$$\boxed{\textbf{tau }=\tau_1\ \tau_2}$$

$\tau_1$ is the refractory period for the first neuron and $\tau_2$ is the refractory period for the 2nd Neuron. So if we wanted the refractory period for the 1st neuron to be 0.2 and that of the 2nd neuron to be 0.5 , we would set :

**tau=0.2 0.5**

## 6 Visualizing the Results

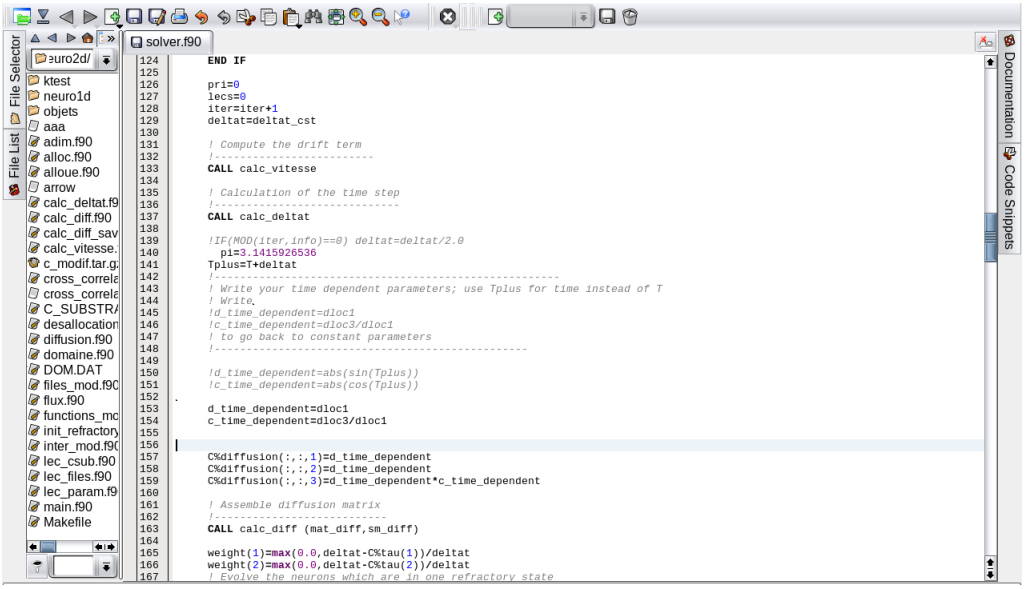
The results of the computations are stored in the test folder from which the code was run . If the movie option was set to 1 , then the specified number of movie frames would have been created . The names of these files are listed in **result.visit** Each of these movie frames is a text file named **result_time.vtk** , for example : **result_1.20.vtk** is the joint density at time 1.20 . If the steady state option was selected , then another file , named **Steady_state.vtk** will be created. This will contain the joint density when the steady state is attained. The results can be plotted by using the freeware VISIT

## 7 Computing Fluxes Across the boundaries

If the movie option is set to 1 , then the flux across the boundaries are computed at each specified point of time . The fluxes are stored as **cross1_time** for the flux across the boundary of Neuron 1 and **cross2_time** for the flux across the boundary of neuron 2 .The times at which the fluxes were computed are stored in the files : **list1_cross** and **list2_cross** both of which are in the same folder that the tests were run from . The first entry in these files is a number which denotes the number of times the fluxes across the boundary was computed. For example **cross1_1.10** is the flux across the boundary of Neuron V at time 1.10 and **cross2_2.50** is the flux across the boundary of Neuron W at time 2.50.

## 8 Time Dependent Parameters

Time dependent parameters can be encoded by modifying 2 files both in the primary neuro2d folder . To encode a time dependent c or D :




1. Open the file solver.f90
2. To change D so that it is now a function of time ,remove the ! preceeding line number 150 and then modify it using Tplus as the time variable. Likewise , to change c , remove the ! preceeding line number 151 and then modify it using Tplus as the time variable.
3. Comment line numbers 153 and 154 using ! mark .
4. Recompile the code by going back to the *neuro2d* folder and typing

$$\$make\ cclean$$
$$\$./arrow$$

5. cd back to the test folder and run the code by typing :

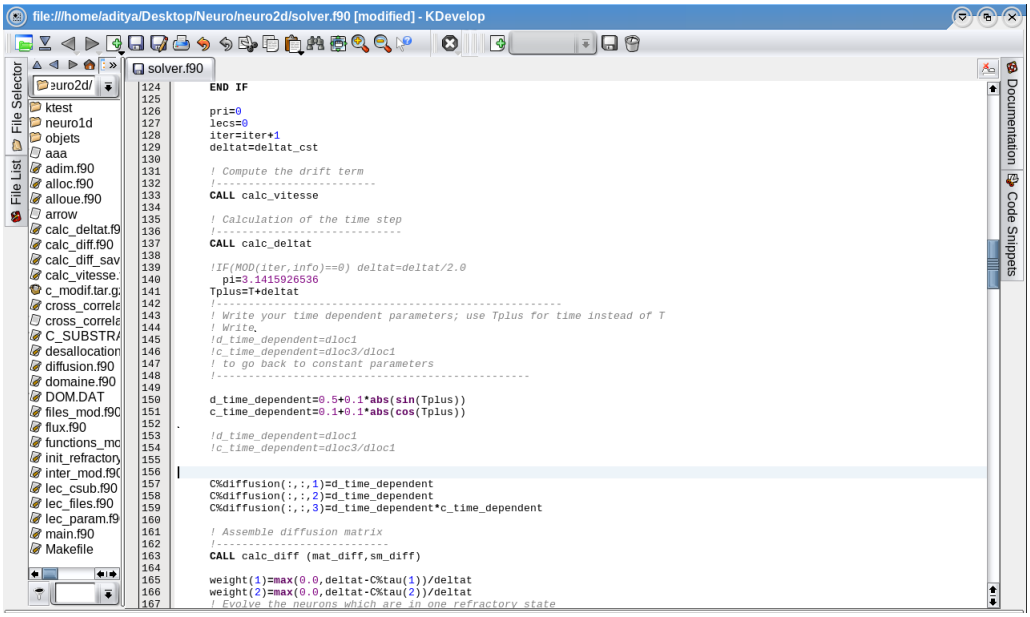
$$\$./sabor < RUN.DAT$$

For example , suppose we wanted $D(t) = 0.5 + 0.1 * |sin(t)|$ and $c(t) = 0.1 + 0.1 * |cos(t)|$ then line number 150 should be changed to :

$$d\_time\_dependent = 0.5 + 0.1 * abs(sin(Tplus))$$

and line number 151 should be changed to :

$$c\_time\_dependent = 0.1 + 0.1 * abs(cos(Tplus))$$




To encode step changes in either of these 2 parameters the fortran function *sign* should be used.

To encode a time dependent $\mu_1$ or $\mu_2$

1. Open the file calc_vitesse.f90
2. To change $\mu_1$ modify line 80 by writing the time dependent $\mu_1(t)$ after the + sign with the time variable as T .
3. To change $\mu_2$ modify line 87 by writing the time dependent $\mu_2(t)$ after the + sign with the time variable as T
4. Recompile the code by going back to the *neuro2d* folder and typing

$$\$make\ cclean$$
$$\$./arrow$$

5. cd back to the test folder and run the code by typing :

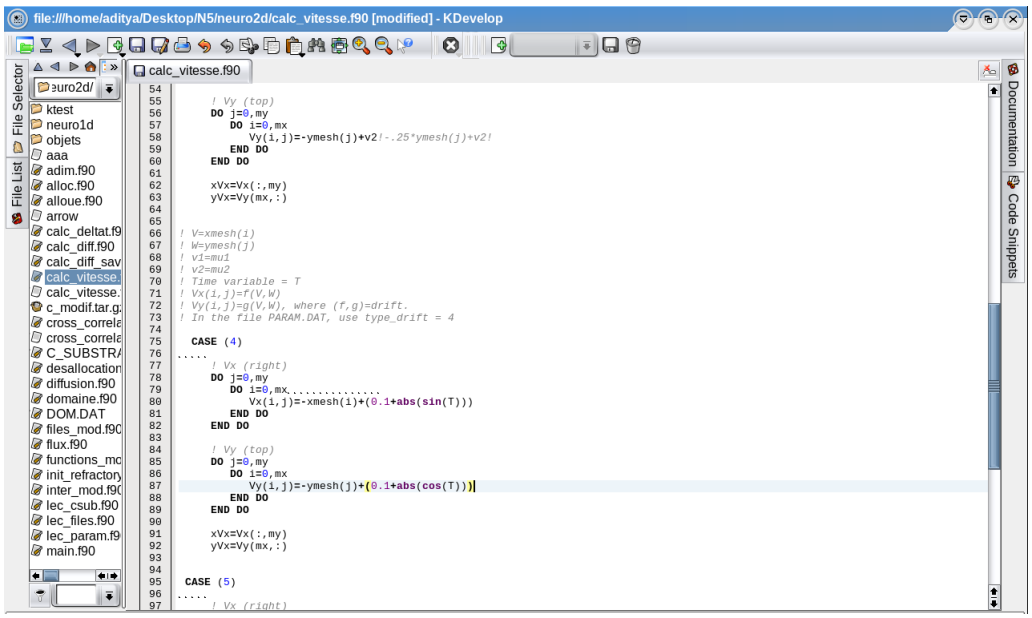
$$\$./sabor < RUN.DAT$$

For example suppose we wanted $\mu_1(t) = 0.1 + |sin(t)|$ and $\mu_2(t) = 0.1 + |cos(t)|$,line 80 would be modified to :

Vx(i,j)=-xmesh(i)+0.1+abs(sin(T)

and line 87 would be modified to :

Vx(i,j)=-ymesh(j)+0.1+abs(cos(T)



## 9 Computing Cross Intensity Functions

The files **list1_cross** and **list2_cross** contain the times at which the fluxes across the boundaries were recorded.There are 2 options to compute the cross intensity functions :

1. The cross intensity functions can be computed at each of these times .

2. The cross intensity funcions can be computed at a subset of these times. For example , the fluxes across the boundary were computed every 0.1 unit of time over a time interval of 10 units leading to 101 entries in the files **list1_cross** and **list2_cross**. If it is desired to compute the cross intensity at just 2 times say 1.00 and 2.00 , then the follwing steps need to be taken :

   Open **list1_cross** and **list2_cross** and change the first entry in the file to 2.

   Delete all the other entries in the file **list1_cross** ,*except* for **cross1_1.00** and **cross1_2.00** . Similarly , delete all the other entries in **list2_cross** *except* for **cross2_1.00** and **cross2_2.00**

Once this is done , the cross intensity functions can be computed by typing the following in the shell:

*$ ./cross*

Then type *1* to compute $H_{21}(t)$ .
Similarly , to compute $H_{12}(t)$ , type : *./cross* and then enter *2*.
The Cross Intensities at the different times for $H_{21}(t)$ are stored in the files *cross_correlation_1* and *cross_correlation_2* . In these files , rows represent time and columns represent space.