StkJJ – User's Reference

E. Goldobin and A. Wallraff

The information in this document applies to StkJJ v3.66

Updated: December 11, 2008

Produced from stkjj2.tex by LaTeX

Contents

List of Tables

Chapter 1

Overview

1.1 Introduction

The latest version of this manual is available online [1] or from the author [2]. Please, feel free to report bugs, make suggestions and discuss with authors [2].

StkJJ is a program for simulation of the dynamics of Josephson phases in a system of $1\ldots3$ magnetically (inductively) coupled long Josephson junctions (LJJ) with arbitrary (not equal) parameters. It is based on the original code by P. Bodin, who is one of the developers of the theoretical model [3] which describes the dynamics of Josephson phases in magnetically coupled LJJs. The original code was almost completely rewritten and strongly extended in many directions. The current version of StkJJ is able to simulate the most experimentally relevant quantities and provides methods to analyze temporal and spatial variations of electromagnetic fields.

1.2 Basic features

StkJJ allows to calculate the profiles and evolution of Josephson phases and their derivatives in space and time [$\phi^{A,B}(x)$, $\phi_x^{A,B}(x)$, $\phi_t^{A,B}(x)$, $\phi_t^{A,B}(t)$], *i.e.*, phases, voltages and magnetic fields in arbitrary moment of time as well as integral characteristics of the system such as I–V characteristic (IVC); dependence of critical current $I_c(H)$ and maximum current of the step $I_{\max}(H)$ on magnetic field H; spectral characteristics of the output voltage $V(f)$; find eigenmodes of solutions and energy band structure of periodic solutions. The program is written in ANSI C and consists of about 3000 lines of code. The important feature of StkJJ is its portability — it can be compiled using any ANSI C compiler on computer with any CPU under any OS. To further increase portability StkJJ is designed with a command line interface. The program was tested on IBM PC compatible computer under MS DOS (or DOS box under MS Windows 3.x/95/NT/2000/XP), Apple Macintosh (in spite of difficulties with command line interface under Mac OS), IBM R50 and HP-7000 workstations as well as on a CRAY supercomputer under UNIX-like OS.

1.3 Command line interface

All input parameters are either parsed from the command line or read from a parameter file. All output is written to data files either in preprocessed or unprocessed form. This allows to run time consuming simulations in batch mode.

StkJJ is started from the command line when the system prompts the user to enter the command.

```
$ StkJJ [-opt] ResFile [-@OptFile] [-opt] SweepList [-opt]
```

There are two kinds of arguments to StkJJ, namely parameters and options. Options are arguments that are preceded by a "–" or "/" signs. All other arguments are considered as parameters. Parameters have to be specified in their proper sequence, whereas options can have arbitrary positions on the command line. The meaning and usage of different elements on the command line is the following:

- StkJJ — This is the call to the executable C program.
- `ResFile` — specifies the name of the file to which the calculated *I*-*V* curve or $I_c(H)$ curve will be written. It is generated in all operation modes of the program
- `-@OptFile` — is the name (if necessary including the path) of a separate file that contains standard options that the user might want to specify *i.e.* instead of putting many options into command line user may consider to put some of them into a `file` and use `-@file` option in command line to compel StkJJ to look for options in `file`. If the same option is specified in both options file and command line, the option in the command line overwrites the option from the option file.
- `-opt` — Any number of options can be specified on the command line. It is especially useful, if an option frequently changes between different runs of StkJJ . Please, note, that options can be specified anywhere and in any order in the command line, even between values in `SweepList`. If an option is specified more than ones in a command line or in an option file, the fist option value will be used.

  There are 3 kinds of options: options expecting boolen argument, integer number, and floating point number. Boolen options expect "+" or "−" or nothing after it. "+ " means "turn the option on", "−" means "turn the option off", no sign means "inverse the default behavior".

  Pitfall/bug: If some boolen option is used in both command line and options file and both times with no explicit meaning (*i.e.* no sign after an option), a double inversion will take place. As a result the boolen option will have its default value. Therefore, use boolen options with inversion only ones *i.e.* either in options file or in the command line.
- `SweepList` — The remaining parameters specify the sweep sequence for the normalized current γ or magnetic field H. `SweepList` has the following format:

$$\begin{array}{lll} \mathit{StartValue}_1 & & \mathit{StopValue}_1\ \mathit{Step}_1 \\ & [& \mathit{StopValue}_2\ \mathit{Step}_2 \\ & [& \ldots \\ & [& \mathit{StopValue}_N\ \mathit{Step}_N]\ldots]] \end{array}$$

  Here, several lines are used for clear explanation and in reality all items of `SweepList` should be typed in one line. The sign of Step_i is not important since StkJJ takes an absolute value of it and understands in which direction to go from current sweep value and $\mathit{StopValue}_i$ value.

  Pitfall/bug: When you need to specify a negative value in the `SweepList`, *e.g.* -1, StkJJ will erroneously misinterpet is as a switch since it starts with '−'. To avoid this use " -1" (open double-quote, space, -1, close double-quotes).

At the beginning of each run StkJJ creates a file `cmdline` which contains all command line parameters. It is useful to store this file together with `OptFile` and with simulation results [$V(I)$ or $I_c(H)$ plus may be phase (derivatives) profiles *etc.*]. 3 months later it will be still clear for you what the command line looked like and which options from `OptFile` were overwritten by command line options.

On exit, StkJJ produces the beep (simply by printing the character with the code '7' on the screen) which is very convinient notification when StkJJ runs in the background.

StkJJ provides a wide range of features to simulate different properties of the system. Different modes of operation as well as input and output formatting are briefly reviewed below.

1.4 Exit status

When StkJJ exits, it sets proper exit code which indicates whether the execution was successful or some problems occurred. This allows to make proper decisions in a batch file, shell script etc... The exit codes are summarized in Tab. 1.1.

Table 1.1: Exit codes

Code	**Explanation**
0	success
1	Wrong number of arguments. Usage hint was shown.
2	Memory allocation error (not enough memory).
3	File (opening/reading/writing) error. Disk full. File read-only.
4	User error: input wrong values, etc.
5	Internal design error in StkJJ. Contact developers.

Chapter 2

Physical and electrical parameters

2.1 Geometry

Before simulation can be started, the parameters of the system should be specifies: first of all number of junctions, topology (geometry) and other physical and electrical parameters. The options describing basic geometrical and electrical parameters are listed in Tabs. 2.1 and 2.2.

Table 2.1: Geometry

Option	Type	Default	Description
`-NumOfJJs:`N	int	2	sets the number of junctions in the system 1, 2 or 3.
`-3[+|-]`	bool	off	Obsolete since v3.65. Superseeded by `-NumOfJJs:3`.
`-Topology:`arg	int	0	geometry of the stack: $arg = 0$ — annular (default), $arg = 1$ — linear
`-JJ_Length:`ℓ	float	12.0	normalized length of LJJ ℓ (if we have more that 1 LJJ, then length of LJJ^A)

The switch `-NumOfJJs:` sets the number of junction in the system. The default value is 2. One can also simulate symmetric modes in 3 coupled junctions, exploiting the fact that in symmetric case ($\phi^A = \phi^C$) 3 equations degenerate to 2 equations similar to 2 LJJ case but with different coefficients. If you use `-NumOfJJs:3`, LJJ^A plays role of both LJJ^A and LJJ^C of 3-fold stack, while LJJ^B plays role of LJJ^B.

If you use `-NumOfJJs:1`, StkJJ becomes at least 2 times faster skipping essential part of calculations related to the LJJ^B. The output is still like for 2 LJJs but all values related to LJJ^B are equal to zero.

Table 2.2: Electrical parameters

Option	Type	Default	Description
`-Alfa0:`α^A	float	0.1	damping parameter α^A of JJ^A

2.2 Parameters for the stack of junctions

For a stack of coupled LJJs, *i.e.*, when `-NumOfJJs:` is equal to 2 or 3, one has to specify the coupling strength and the difference in the electrical and geometrical parameters of the LJJs, see

Tab. 2.3.

Table 2.3: Parameters for stack of LJJs

Option	Type	Default	Description
`-J:`J	float	1.0	ratio of critical currents j_c^A/j_c^B
`-R:`R	float	1.0	ratio of quasiparticle (subgap) resistivities R_J^A/R_J^B
`-C:`C	float	1.0	ratio of specific capacitances C^A/C^B
`-D:`D'	float	1.0	ratio of effective magnetic thicknesses d'^A/d'^B
`-L:`Λ	float	1.0	ratio of magnetic thicknesses Λ^A/Λ^B
`-S:`S	float	-0.3	coupling parameter S. StkJJ uses $-\|S\|$ for simulations since, according to the theory, S should be negative.

2.3 Space-dependent critical current $j_c(x)$

StkJJ can simulate structures with spacial-dependent critical current dencity $j_c(x)$. For this purpose $j_c(x)$ is represented as $j_c(x) = j_0\tilde{j}(x)$. All quantities are normalized to j_0 (i.e. $j_0 = 1$ in our normalized units) and $\tilde{j}(x)$ sets the spacial dependence of the critical current. The options which allow to specify different $\tilde{j}(x)$ are summarized in Tab. 2.4.

Table 2.4: Spacially dependent $j_c(x)$

Option	Type	Default	Description
-j_c(x)Type:*id*	int	0	Sets the type of $j_c(x)$ dependence: $id = 0:$ no dependence. $id = 1:$ rectangular dependence. $id = 2:$ sinusoidal dependence. $id = 3:$ ramp-like dependence. $id = 4:$ another ramp-like dependence. $id = 5:$ tanh-like dependence.
`-j_c(x)XShift:`Δx	float	0.0	Allows to shift the whole $j_c(x)$ dependence along x axis by Δx.
`-j_c(x)Shift:`Δj_c	float	0.0	Allows to shift the whole $j_c(x)$ dependence along j_c axis by Δj_c.
`-j_c(x)A:`A	float	1.0	Amplitude of $j_c(x)$.
`-j_c(x)L1:`L_1	float	diff.	Distance L_1
`-j_c(x)L2:`L_2	float	diff.	Distance L_2
`-j_c(x)L3:`L_3	float	diff.	Distance L_3
`-j_c(x)Period:`ΔL	float	2.0	Period ΔL for sinusoidal $j_c(x)$.

There are few predefined $j_c(x)$ dependences:

1. Rectangular. Along the distance L_1 $j_c = +A$, then along the distance L_2 $j_c = -A$, then along the distance L_3 $j_c = 0$. Then again along the distances L_1, L_2, L_3 $j_c = +A$, $-A$, 0, respectively. And so on along the whole LJJ with the period $L_1 + L_2 + L_3$. L_1, L_2, L_3 and A can be set using the options `-j_c(x)L1:`, `-j_c(x)L2:`, `-j_c(x)L3:`, `-j_c(x)A:`.

2. Sinusoidal.

$$j_c(x) = A \sin\left(\frac{2\pi x}{\Delta L}\right), \tag{2.1}$$

where the amplitude A and period ΔL can be specified using `-j_c(x)A:` and `-j_c(x)Period:` options.

3. Ramp. Along the distance L_1 the critical current ramps from $-A$to $+A$ linearly, then along the distance L_2 the critical current j_c ramps down from $+A$ to $-A$. Then again along the distances L_1 it goes from $-A$ to $+A$ and along the distance L_2 from $+A$ to $-A$, and so on along the whole LJJ with the period $L_1 + L_2$. L_1, L_2 and A can be set using the options `-j_c(x)L1:`, `-j_c(x)L2:`, `-j_c(x)A:`.

4. Ramp2. Along the dostance L_1 the critical current is $+A$, then along the distance L_2 it ramps linearly from $+A$ to $-A$, then along the distance L_3 it ramps back from $-A$ to $+A$, and then it stays at $+A$ up to the end of LJJ.

5. Tanh.
$$j_c(x) = A \tanh(\frac{x - L/2}{L_1}). \tag{2.2}$$

6. Custom. This is used internally inside StkJJ by higher-level modules e.g. by Hot Spot Scanner, see Sec. 6.4.

All dependences can be shifted along x or along j_c axis using the options `-j_c(x)XShift:` and `-j_c(x)Shift:`

2.4 0-π and 0-κ LJJs

StkJJ allows to model 0-π LJJ consisting of alternating facets of 0 and π JJs. This is modelled by introducing additional term $\theta_{xx}(x)$ into the sine-Gordon, *i.e.*,

$$\phi_{xx} - \phi_{tt} - \sin\phi = \alpha\phi_t - \gamma(x) - \theta_{xx}(x). \tag{2.3}$$

The function $\theta(x)$ is a step-like function: it is constant along the 0 or π part and makes a jump at the conjunction. $\theta_{xx}(x)$ can be represented as a sum of Heaviside step functions

$$\theta(x) = \sum_{i=1}^{N} \pi\kappa_i \mathcal{H}(x - x_i), \tag{2.4}$$

where κ_i can set the direction and/or amplitude of the jumps. Note that to model 0-π-LJJs one have to use only $\kappa_i = \pm 1$.

Since $\theta(x)$ is discontinuous, it is clear from Eq. (2.3) that the solution $\phi(x,t)$ will also be a discontinuous function of x at $x = x_i$. Therefore, x_i are called discontinuity points. In fact StkJJ can simulate any arbitrary $\pi\kappa$-discontinuities and the value of κ can be even swept.

The number and positions of $\pi\kappa$-discontinuity points can be set using options listed in Tab. 2.5. There are 2 major ways to set the positions of discontinuity points:

- either distribute them automatically along LJJ. The number of discontinuities is set by `-PiPts:`, polarity and positions by `-PiPtsOrder:`. The whole chain can be shifted using `-PiPtsShift:`.

- or specifying the positions (x-coordinates) of discontinuities as a list of values using `-PiPtsPos:`. Note that positive positions correspond to $+\pi\kappa$-discontinuity while negative positions to $-\pi\kappa$-discontinuity. For example: `-PiPtsPos:10,-12,14` corresponds to phase $\theta(x)$ jumping from 0 to $\pi\kappa$ at $x = 10$, from $\pi\kappa$ to 0 at $x = 12$, and from 0 to $\pi\kappa$ at $x = 14$, *i.e.*,

$$\theta(x) = \pi\kappa \left[\mathcal{H}(x - 10) - \mathcal{H}(x - 12) + \mathcal{H}(x - 14)\right]$$

Table 2.5: π-junctions & phase discontinuities

Option	**Type**	**Default**	**Description**
`-PiPtsPos:`$x_1, x_2, \ldots, x_n$	float		Sets the list of points in which the phase has π-discontinuity. This option overwrites `-PiPts:` if any. Positive value corresponds to $+\pi$-discontinuity, negative to $-\pi$-discontinuity.
`-PiPts:`N_c	int	0	Number of π-discontinuities (zigzag corners) equidistantly distributed along LJJ. Effective only if no `-PiPtsPos:` specified.
`-PiPtsOrder:`id	int	0	$id = 0$ — Equidistant, $a = L/N$, $b = a/2$. $L^0_\Sigma = L^\pi_\Sigma$ for any N_c. $id = 1$ — Equidistant, $a = b = L/(N+1)$. $L^0_\Sigma \neq L^\pi_\Sigma$ for even N_c. $id = 2$ — Equidistant around the center of LJJ. The a is set using `-PiPtsDist:` option. The `-PiPtsOrder:` option is effective only together with `-PiPts:`
`-PiPtsDist:`a	float	2.0	The distance a between π-discontinuity points. Effective only for `-PiPtsOrder:2`.
`-PiPtsShift:`Δx	float	0.0	Allows to shift the chain of phase discontinuities by Δx along LJJ. Effective only together with `-PiPts:`
`-Discontinuity:`κ	float	1.0	The jump at the discontinuity can be not only π but any other value $\pi\kappa$.

Note, that natural π-discontinuity can be emulated using two closely spaced δ-like injector and extractor of current. Changing the value of current one can control the value of the phase jump at the discontinuity. To simulate the dependence on the value of injector current and other discontinuity values which may appear in the future, there is a switch `-Discontinuity:`κ to specify the discontinuity of $\kappa\pi$.

When using saved states, you must specify the same positions and signs of the discontinuity points and the same number of trapped fluxons (and SFs???) to provide proper annular boundary conditions. For example, first, you get some state running StkJJ using the following command line:

```
StkJJ32  -@@ nul .01 1 2 -PXT:1 -FRAMES:200
         -InitCond:3 -NFlux0:1 -PiPtsPos:20,22,24,26
```

StkJJ creates a file `last.stt`. To continue simulations strting from this state, you have to restart StkJJ e.g. in the following way:

```
StkJJ32  -@@ nul .01 1 2 -PXT:1 -FRAMES:200
         -SavedState:last.stt -NFlux0:1 -PiPtsPos:20,22,24,26
```

The "rule of thumb" is to add the option `-SavedState` not changing anything else.

2.5 Current-phase relation

By default StkJJ uses sinusoidal current-phase relation (CPR):

$$j_s = j_c \sin(\phi). \tag{2.5}$$

Using options from Tab. 2.6 it is possible to set other types of CPR.

Table 2.6: Current-phase relation

Option	Type	Default	Description
`-CPR:`id	int	0	Type of CPR: 0: harmonic 1: ramp 2: rectangular
`-CPR_Harmonics:`n	int	1	The number of harmonics in CPR.
`-CPR_Is`k`:`I_s^k	float	0.0|1.0	The amplitude of k-th sin-harmonic, *i.e.*, critical current in front of $\sin(k\phi)$ term. Default value is 1.0 for $k = 1$, and 0.0 for all other k.
`-CPR_Ic`k`:`I_c^k	float	0.0	The amplitude of k-th cos-harmonic, *i.e.*, critical current in front of $\cos(k\phi)$ term.
`-CPR_RampMax:`ϕ_{max}	float	0.5	The phase at which the ramp reaches the summit (in units of π).
`-CPR_RectPhi0:`ϕ_0	float	0.5	The center phase of the rectangle (in units of π).
`-CPR_RectWidth:`w	float	1.0	The width of the rectangle (in units of π).

Chapter 3

Appying currents and fields

3.1 Applying the current through the structure

StkJJ has very flexible control of current passing through the junctions. The total current passing through each LJJ is given by:

$$\gamma_A(x,t) = B_A I[P_{\rm dc}(x) I_{\rm dc} + P_{\rm rf}(x) I_{\rm rf} \sin(\omega_{\rm rf} t)] + I_F \Gamma(t); \tag{3.1}$$

$$\gamma_B(x,t) = B_B I[P_{\rm dc}(x) I_{\rm dc} + P_{\rm rf}(x) I_{\rm rf} \sin(\omega_{\rm rf} t)] + I_F \Gamma(t), \tag{3.2}$$

where $B_{A,B}$ is used to specify disbalance of bias currents, I is used to specify common bias behavior, $I_{\rm dc}$ and $I_{\rm rf}$ are used to specify a dc and rf component of the current, respectively, and, finally, $P_{\rm dc}$ and $P_{\rm rf}$ are used to specify the spatial disctribution of dc and rf bias, respectively. This parameters can be specified using options listed in Tab. 3.1.

3.1.1 Spacial dc and rf bias current distribution

The current distribution in the thin superconducting film is given by the approximate formula

$$P(x) = \begin{cases} \dfrac{1}{\sqrt{1-\left(\frac{2x}{L}-1\right)^2}} & \text{for } \lambda_{\rm eff}/2 < x < L-\lambda_{\rm eff}/2; \\ \dfrac{\exp\left(\frac{L}{2}-\left|x-\frac{L}{2}\right|\right)}{\lambda_{\rm eff}\exp(-\frac{1}{2})\sqrt{1-\left(1-\frac{\lambda_{\rm eff}}{L}\right)^2}} & \text{for } x<\lambda_{\rm eff}/2 \text{ or } x > L-\lambda_{\rm eff}/2, \end{cases} \tag{3.3}$$

when current flows along the film in one direction. Here $\lambda_{\rm eff} = \lambda^2/d$ defines the distance from the edges of the LJJ where crossover between two dependences takes place. The denominator in the second expression is merely to join both pieces smoothly. If you plan to use current profiles 1, 2, 4, 5, you have to specify $\lambda_{\rm eff}$ using corresponding `-xxBiasProfileWidth:` option. Note that $\lambda_{\rm eff}$ should be given normalized to λ_J^A as all lengths in StkJJ.

Note, that the bias current distribution is not normalized and that it is rather sharp at the edges of LJJ so that at given value of discretization Δx, which is usually $\sim \lambda_J^A/20$, will result in rather rough approximation of the bias near to the edges. StkJJ performs normalization taking into account discrete nature of the procedure. StkJJ sums the absolute values of the current in each discrete point and then the current is divided by this value. To get more realistic simulation results one has to use smaller Δx of the order of λ which will result in a very time consuming simulations. We have to note also that the formula (3.3) is valid for $d \sim \lambda$ and $L \gg \lambda$.

In the film placed in magnetic field perpendicular to film's plane screening currents are induced. The distribution of such currents can be described by approximate formula[4]:

$$P(x) = \frac{1}{(\alpha(1-x^2)+\beta)^{\frac{5}{2}}}, \tag{3.4}$$

Table 3.1: Bias current injection

Option	Type	Default	Description
`-Bias0:`Bias0 `-Bias1:`Bias1	float	1.0	current disbalance as in Eqs. (3.1) and (3.2)
`-I:`I	float	0.0	value of normalized current. By defaut this perameter is swept (see `-Sweep1Var:`), and this option has no effect. **Note**, since v3.58 this option use capital "`-I:`".
`-I_DC:`$I_{\rm dc}$	float	1.0	amplitude of dc current as in Eqs. (3.1) and (3.2)
`-I_RF:`$I_{\rm rf}$	float	0.0	amplitude of rf current as in Eqs. (3.1) and (3.2)
`-FreqI_RF:`$\omega_{\rm rf}$	float	0.0	angular frequency of rf current as in Eqs. (3.1) and (3.2)

`-dcBiasProfile:`$P_{\rm dc}$`-id`
`-rfBiasProfile:`$P_{\rm rf}$`-id`
	int	0	ID of the spatial profile of dc ($P_{\rm dc}$) and rf ($P_{\rm rf}$) bias distributions, see Eqs. (3.1) and (3.2). Possible IDs are listed in Tab. 3.2.

`-dcBiasProfileWidth:`$\lambda_{\rm eff}$ `-rfBiasProfileWidth:`$\lambda_{\rm eff}$	float	10^{-3}	the width (crossover point) $\lambda_{\rm eff}$ of dc and rf bias distribution (3.3) expressed in units of λ_J^A. Has effect only if `-dcBiasProfile:1` or `-rfBiasProfile:1`.
`-dcBiasFile:`*name* `-rfBiasFile:`*name*	string	""	the file name with dc or rf bias profile. These options overwrite `-dcBiasProfile:` and `-rfBiasProfile:`
`-I_F:`I_F	float	0.0	amplitude (dispersion) of fluctuation current as in Eqs. (3.1) and (3.2). The probability distribution is given by Eq. (3.13)

where coefficients α and β depend on film thickness and can be approximately expessed as

$$\alpha \approx 0.25 - 0.63\tilde{\lambda}_{\rm eff}^{0.5} + 1.2\tilde{\lambda}_{\rm eff}^{0.8}; \tag{3.5}$$

$$\beta \approx 2/\pi\tilde{\lambda}_{\rm eff} + 4\tilde{\lambda}_{\rm eff}^{2}, \tag{3.6}$$

for thin films ($d < \lambda$)[4], and as

$$\alpha \approx 0.25; \tag{3.7}$$

$$\beta \approx 0.64\tilde{\lambda}_{\rm eff}, \tag{3.8}$$

for thick films ($d > \lambda$)[4].

Just in case we included mono-harmonic current distributions such as sin-like

$$P(x) = \sin\left(\pi\frac{x}{L}\right), \tag{3.9}$$

cos-like

$$P(x) = \cos\left(\pi\frac{x}{L}\right), \tag{3.10}$$

and twice faster sin-like which passes zero in the midle of LJJ

$$P(x) = \sin\left(2\pi\frac{x}{L}\right). \tag{3.11}$$

For some purposes it may be useful to have sign-like current distribution equal to +1 along the left half of LJJ and +1 along the right half.

$$P(x) = \mathrm{sgn}\left(x - \frac{L}{2}\right). \tag{3.12}$$

Table 3.2: Types of bias current distribution

P-id	**Current distribution is ...**	**Net Curr.**
0	constant	1.0
1	as in thin supercond. film when current flows in one direction, see (3.3)	1.0
2	is linear. Current density is -1 at $x = 0$ and +1 at $x = L$	0.0
3	as in supercond. film when current flows in opposite direction along the edges at $x = 0$ and $x = L$, see (3.4).	0.0
4	sin-like. At $x = 0$ and $x = L$ current is zero, see (3.9).	1.0
5	cos-like. At $x = L/2$ current is zero, see (3.10).	0.0
6	sin-like. At $x = 0$, $L/2$, L is zero, see (3.11).	0.0
7	as sign-like step function. At $x < L/2$ the bias is -1, at $x > L/2$ the bias is $+1$, see (3.12)	0.0

3.1.2 Fluctuations current

The I_F in Eqs. (3.1) and (3.2) is the intensity of fluctuations, and $\Gamma(t)$ is the noise with Gaussian distribution function

$$W(I) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{I^2}{2\sigma^2}\right), \tag{3.13}$$

where σ is taken equal to one. This means that $I_F\Gamma(t)$ will correspond to random process with Gaussian probability distribution (3.13) and dispersion $\sigma = I_F$. The physical meaning of i_F is the following:

$$i_F = \frac{\sqrt{\langle I_F^2 \rangle}}{I_c} = \frac{\sqrt{\langle j_F^2 \rangle}}{j_c} = \sqrt{\frac{k_B T}{2\pi r_n j_c^2 \Delta t w \Delta x}} = \sqrt{\frac{k_B T}{\Phi_0 I_c^{[w\times\lambda_J]}} \frac{\lambda_J \alpha}{\omega_p \Delta t \Delta x}} = \sqrt{\gamma_F \frac{\alpha}{\Delta\tilde{t}\Delta\tilde{x}}}, \tag{3.14}$$

where $\gamma_F = k_B T/\Phi_0 I_c^{[w\times\lambda_J]}$ is the parameter which characterizes relative strength of thermal fluctuations with respect to the Josephson energy per λ_J. For space dependent $w(x) = w_0\tilde{w}(x)$ and $j_c(x) = j_{c0}\tilde{j}_c(x)$ (and therefore $\alpha(x) = \alpha_0/\tilde{j}_c(x)$ and $\lambda_J(x) = \lambda_{J0}/\tilde{j}_c(x)$) we get:

$$i_F(x) = \sqrt{\gamma_F \frac{\alpha_0}{\tilde{j}_c(x)\tilde{w}(x)\Delta\tilde{t}\Delta\tilde{x}}}, \tag{3.15}$$

where γ_F is now defined as:

$$\gamma_F = \frac{k_B T}{\Phi_0 j_{c0} w_0 \lambda_{J0}}, \tag{3.16}$$

3.2 Applying field to linear LJJ

With StkJJ you can study the influence of applied rf radiation on the processes in the system. The rf electromagnetic wave is modelled as oscillating magnetic field at the left edge ($x = 0$) of LJJ's, so that total field at the left edge is:

$$H_\Sigma = H + H_{\rm rf}\sin(2\pi f_{H_{\rm rf}} t) \tag{3.17}$$

The options responsible for boundary conditions are summarized in Table 3.3. Note, that annular topology assumes periodic boundary conditions and do not need any option to specify.

3.3 Nonuniform magnetic field: $h(x)$-term

With StkJJ you can simulate what happens in non-uniform external magnetic field. When field is non-uniform the so-called $h_x(x)$-term appears in the sine-Gordon equation. Option for such a simulation are summarized in the Table 3.4.

Table 3.3: Boundary conditions

Option	Type	Default	Description
`-H:`H	float	0.0	value of normalized external magnetic field H. If StkJJ is simulating $I_{\max}(H)$ and "`-Sweep2Var:H`" is used (default), then this option has no effect. See also "`-HL:`" and "`-HR:`" below.
`-HL:`H_L	float	1.0	the factor on which the field H is multiplied on the left edge of the junction ($x = 0$) for linear case. The total filed applied to the left edge is $H \times H_L$. This allows to simulate LJJ with current injection into the edge.
`-HR:`H_R	float	1.0	the factor on which the field H is multiplied on the right edge of the junction ($x = \ell$) for linear case. The total filed applied to the right edge is $H \times H_R$. This allows to simulate LJJ with current injection into the edge. This option changed its meaning since v3.58!
`-H_RF:`$H_{\rm rf}$	float	0.0	$H_{\rm rf}$ is an amplitude of normalized externally applied ac magnetic field for linear case. The net field is $H_\Sigma = H + H_{\rm rf}\sin(2\pi f_{H_{\rm rf}} t)$
`-FreqH_RF:`$f_{H_{\rm rf}}$	float	0.0	frequency of externally applied ac magnetic field for linear case.

The presence of potential adds an extra term to the sine-Gordon equation.

$$\phi_{xx} - \phi_{tt} - \sin\phi = \alpha\phi_t - \gamma - h_x(x), \tag{3.18}$$

where

$$h(x) = \left(\tilde{\mathbf{B}} \cdot \tilde{\mathbf{n}}\right), \tag{3.19}$$

Since potential enters the Lagrangian, in sine-Gordon equation we will get its derivative over distance x *i.e.* $h_x(x)$. Here x is the coordinate directed along the perimeter of the structure.

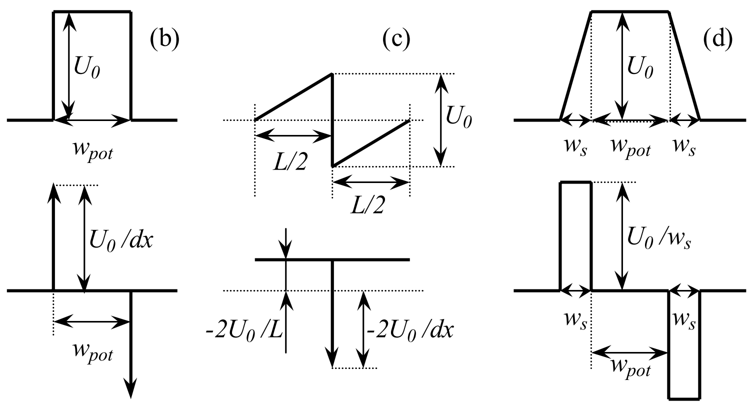

Figure 3.1: Various potential shapes.

The following potentials are implemented in StkJJ:

1. **Cosinusoidal (annular LJJ)** field. For usual cosinusoidal field (like in annular/circular LJJ)
$$h(x) = H_0 \cos\left(\frac{2\pi(x-x_0)}{P}\right), \tag{3.20}$$
where H_0 is the amplitude of the potential set by `-PotDepth:`, x_0 is the origin of the potential set by `-PotOrigin:` and P is the period of potential set by `-PotPeriod:`. The h_x term is sinusoidal:
$$h_x^{\sin}(x) = \underbrace{\frac{2\pi H_0}{P}}_{F_0} \sin\left(\frac{2\pi(x-x_0)}{P}\right), \tag{3.21}$$
The real force acting on a fluxon (in a perturbation theory limit i.e. assuming that fluxon shape does not change due to $h(x)$) is $2\pi F_0 \sin(2\pi x/L)$, which assumes that potential changes slowly in comparison with the fluxon size. For more exact expression see Ref. [5]. In this case the resonant frequency of fluxon in a well is:
$$\omega_0 = \pi\sqrt{\frac{F_0}{2L}}, \quad T_0 = 2\sqrt{\frac{2L}{F_0}}, \tag{3.22}$$
Accuracy of this formula is better than 5% for $L = 40$, than 10% for $L = 20$ and than 30% for $L = 10$.

2. **Rectangular potential** is shown in Fig. 3.1(b) and is given by the following expression:
$$h^{\text{rect}}(x) = U_0 \left\{ \begin{array}{ll} L - W_{\text{pot}} & \text{for } \frac{L}{2} - \frac{W_{\text{pot}}}{2} < x < \frac{L}{2} + \frac{W_{\text{pot}}}{2} \\ -W_{\text{pot}} & \text{for other } x \end{array} \right., \tag{3.23}$$
where W_{pot} is the relative width of the well given by "`-PotWidth:`". Therefore, the force density term will be given by:
$$h_x^{\text{rect}}(x) = U_0 \left\{ \begin{array}{ll} \dfrac{L}{dx} & \text{for } \frac{L}{2} - \frac{W_{\text{pot}}}{2} - \frac{dx}{2} < x < \frac{L}{2} - \frac{W_{\text{pot}}}{2} + \frac{dx}{2} \\ -\dfrac{L}{dx} & \text{for } \frac{L}{2} + \frac{W_{\text{pot}}}{2} - \frac{dx}{2} < x < \frac{L}{2} + \frac{W_{\text{pot}}}{2} + \frac{dx}{2} \\ 0 & \text{for other } x \end{array} \right., \tag{3.24}$$

3. **Asymmetric saw tooth** field is shown in Fig. 3.1(c) and is given by formula:
$$h^{\text{saw}}(x) = U_0 \left\{ \begin{array}{ll} \dfrac{2}{L-dx}x & \text{for } 0 < x < \dfrac{L}{2} - \dfrac{dx}{2} \\ -\dfrac{2}{dx}x + \dfrac{L}{dx} & \text{for } \dfrac{L}{2} - \dfrac{dx}{2} < x < \dfrac{L}{2} + \dfrac{dx}{2} \\ \dfrac{2}{L-dx}x - \dfrac{2L}{L-dx} & \text{for } \dfrac{L}{2} + \dfrac{dx}{2} < x < L \end{array} \right., \tag{3.25}$$
and corresponding force density is
$$h_x^{\text{saw}}(x) = U_0 \left\{ \begin{array}{ll} \dfrac{-2}{dx} & \text{for } \dfrac{L}{2} - \dfrac{dx}{2} < x < \dfrac{L}{2} + \dfrac{dx}{2} \\ \dfrac{2}{L-dx} & \text{for other } x \end{array} \right.. \tag{3.26}$$

4. **Trapezioidal** field is shown in Fig. 3.1(d) and is given by the following expression:
$$h^{\text{trap}}(x) = U_0 \left\{ \begin{array}{ll} 1 & \text{for } \frac{L}{2} - \frac{W_{\text{pot}}}{2} < x < \frac{L}{2} + \frac{W_{\text{pot}}}{2} \\ +\frac{1}{W_s}x + \ldots & \text{for } \frac{L}{2} - W_{\text{pot}} - W_s < x < \frac{L}{2} - W_{\text{pot}} \\ -\frac{1}{W_s}x + \ldots & \text{for } \frac{L}{2} - W_{\text{pot}} - W_s < x < \frac{L}{2} - W_{\text{pot}} \\ 0 & \text{for other } x \end{array} \right., \tag{3.27}$$

where W_{pot} is the relative width of the flat maximum given by "`-PotWidth:`", and W_s is the width of each slope given by "`-PotSlope:`". Therefore, the force density will be given by:

$$h_x^{\text{trap}}(x) = \gamma_h(x) = U_0 \begin{cases} 0 & \text{for } \frac{L}{2} - \frac{W_{\text{pot}}}{2} < x < \frac{L}{2} + \frac{W_{\text{pot}}}{2} \\ +\frac{1}{W_s} & \text{for } \frac{L}{2} - W_{\text{pot}} - W_s < x < \frac{L}{2} - W_{\text{pot}} \\ -\frac{1}{W_s} & \text{for } \frac{L}{2} - W_{\text{pot}} - W_s < x < \frac{L}{2} - W_{\text{pot}} \\ 0 & \text{for other } x \end{cases}, \tag{3.28}$$

5. **Weak saw-tooth** field is very similar to strong saw-tooth, but the steep slope has the width W_{pot} (set by `-PotWidth:`) instead of one discretization step dx:

$$h^{\text{saw}}(x) = U_0 \begin{cases} \dfrac{2}{L - W_{\text{pot}}} x & \text{for } 0 < x < \dfrac{L}{2} - \dfrac{W_{\text{pot}}}{2} \\ -\dfrac{2}{W_{\text{pot}}} x + \dfrac{L}{W_{\text{pot}}} & \text{for } \dfrac{L}{2} - \dfrac{W_{\text{pot}}}{2} < x < \dfrac{L}{2} + \dfrac{W_{\text{pot}}}{2} \\ \dfrac{2}{L - W_{\text{pot}}} x - \dfrac{2L}{L - W_{\text{pot}}} & \text{for } \dfrac{L}{2} + \dfrac{W_{\text{pot}}}{2} < x < L \end{cases}, \tag{3.29}$$

and corresponding force density is

$$h_x^{\text{ws}}(x) = U_0 \begin{cases} \dfrac{-2}{W_{\text{pot}}} & \text{for } \left|x - \frac{L}{2}\right| < \frac{W_{\text{pot}}}{2} \\ \dfrac{2}{L - W_{\text{pot}}} & \text{for other } x \end{cases}. \tag{3.30}$$

6. **Zigzag LJJ potential** simulates the non-uniform field distribution in the sigzag LJJs. It is assumed that each electrode of the LJJ has a zigzag ends and the zigzags of the top and bottom electrodes overlap a bit. By applying the magnetic field with the component perpendicular to the film (sample) plane, we induce screening currents. This is usually refeered as *field focusing effect*. For very general LJJ with coordinates from $x_{\min}$ to $x_{\max}$ the distribution of the field is given by the envelope

$$h_{\text{env}}(x) = \left(\frac{2}{L}\right)^{2p} \left[\left(\frac{L}{2}\right)^2 - (x - x_{\text{mid}})^2\right]^p = \left(\frac{2}{L}\right)^{2p} [x(L-x)]^p, \tag{3.31}$$

where $L = x_{\max} - x_{\min}$ is the length of the LJJ, and $x_{\text{mid}} = \frac{1}{2}(x_{\max} + x_{\min})$ is the coordinate of the middle point. Within each facet the modulation is given by similar formula

$$h_{\text{facet}}(x) = \frac{(r^2 - x_i^2)^p}{r^{2p}}, \tag{3.32}$$

where $r = L/(2N)$ is the "radius" (semi-length) of each facet, $x_i = \text{rmod}(x - x_{\min}, L/N) - r$ is the internal coordinate inside the facet ($x_i = -r, \ldots, +r$) The final $h(x)$ profile is given by the product of $h_{\text{facet}}(x) \cdot h_{\text{env}}(x)$.

7. **Step-like potential** in the center of JJ

$$h^{\text{st}}(x) = U_0 \begin{cases} 1 & \text{for } 0 < x < \dfrac{L}{2} - \dfrac{dx}{2} \\ -\dfrac{2}{dx} x + \dfrac{L}{dx} & \text{for } \dfrac{L}{2} - \dfrac{dx}{2} < x < \dfrac{L}{2} + \dfrac{dx}{2} \\ \dfrac{2}{L - dx} x - \dfrac{2L}{L - dx} & \text{for } \dfrac{L}{2} + \dfrac{dx}{2} < x < L \end{cases}, \tag{3.33}$$

8. **Injector potential** looks identical to trapezioidal potential described by Eqs. (3.27) and (3.28). Each trapezoidal potential corresponds to one pair of current injectors (non-ideal phase discontinuity). One can put as many injectors as possible using `-PotPos:` and set their individual amplitudes/polarities using `-PotAmp:`. Note that one can also simultaneously change the amplitudes of all of them by changing `-PotDepth:`.

Table 3.4: Nonuniform magnetic field: $h(x)$-term

Option	Type	Default	Description
`-PotType:`PoT	int	0	Type (profile shape) of the potential. 0 : no potential; 1 : sine (real LJJ); 2 : rectangular; 3 : saw tooth; 4 : Trapezoidal (I_{inj2} of finite size) 5 : Weak saw-tooth 6 : like in ZigZag LJJ 7 : 8 ”injector”: a set of injectors (trapezoidal potentials)
`-PotDepth:`U_0	float	0.0	normalized potential depth.
`-PotWidth:`W_{pot}	float	0.5	defines the size of some potential features. See the description of corresponding potential.
`-PotSlope:`W_{s}	float	0.5	defines the slope width of trapezoidal potential.
`-PotPeriod:`P	float	L	the period of a cos-like potential.
`-PotOrigin:`x_0	float	0.0	the origin of cos-like potential. Allows to shift the potential along LJJ.
`-PotPow:`p	float	0.25	For `-PotType:6` sets the power p in Eqs. (**??**) and (**??**).
`-PotFacets:`N	uint	4	For `-PotType:6` sets the number of facets.
`-PotPos:`$x_1, x_2, \ldots$	float	””	For `-PotType:8` sets the positions of injectors.
`-PotAmp:`$A_1, A_2, \ldots$	float	””	For `-PotType:8` sets the amplitudes (polarities) of injectors[1].
`-PotFile:`file-name	string	””	the name of the file where custom potential Ucust(x) profile is stored. This profile will be differentiated by StkJJ to get $F_{\text{cust}}(x)$. Note, that if the filename is not an empty string, this switch overwrites `-PotType:`

3.4 Width modulation along LJJ: $w(x)$-term

Using StkJJ one can also simulate the processes in LJJ with variable width $w(x)$ of superconducting electrodes. In this case there is additional term in sine-Gordon equation:

$$\phi_{xx} - \phi_{tt} - \sin\phi = \alpha\phi_t - \gamma + h_x(x) + \frac{w'(x)}{w(x)}\left(h(x) - \phi_x\right), \tag{3.34}$$

In case of fluxon solution, this w-term corresponds to an additional potential in which fluxon moves. In comparison with h-potential, w-potential does not depend on the fluxon polarity. Note also, that the presence of both h and w-terms, results in the appearence of the cross term $\frac{w'(x)}{w(x)}h(x)$.

Table 3.5 summarizes the options used to set $w(x)$ dependence. Various predefined w-potential types are as follows.

- no potential,

$$w(x) = w_0 = const, \quad \frac{w'}{w} = 0, \tag{3.35}$$

Table 3.5: LJJ with variable width; $w(x)$-potential

Option	**Type**	**Default**	**Description**

`-wPotType:`*id*	int	0	Type (shape) of the potential $w(x)$.
			0: no potential, see Eq. (3.35);
			1: sin-like, see Eq. (3.36);
			2: cos-like, see Eq. (3.37);
			3: linear width, see Eq. (3.38);
			4: exponential width, see Eq. (3.39);
			5: gaussian enwidthment, see Eq. (3.40);
			6: rect-like $w(x)$, see Eq. (3.41);

`-wPotDepth:`A	float	0.0	potential depth.
`-wPotPeriod:`P	float	L	pariod of periodic potentials (3.36), (3.37).
`-wPotWidth:`λ	float	0.5	defines the size of some potential features. See the description of corresponding potential.
`-wPotFile:`file-name	string	""	the name of the file where custom potential $w(x)$ profile is stored. This profile will be differentiated by StkJJ to get $w'(x)/w(x)$. Note, that if the filename is not an empty string, this switch cancels the effect of `-WPotType:`
`-$SaveWPot[+|-]`	bool	off	Saves $w(x)$ into `__w(x).dat`.

- sin-like width profile

$$w(x) = w_0\left[1 + A\sin\left(\frac{2\pi x}{P}\right)\right], \quad \frac{w'}{w} = \frac{2\pi}{P}\frac{A\cos(\ldots)}{1+A\sin(\ldots)}, \tag{3.36}$$

- cos-like width profile

$$w(x) = w_0\left[1 + A\cos\left(\frac{2\pi x}{P}\right)\right], \quad \frac{w'}{w} = \frac{2\pi}{P}\frac{-A\sin(\ldots)}{1+A\cos(\ldots)}, \tag{3.37}$$

- linearly increasing width

$$w(x) = w_0\left(1 + A\frac{x}{L}\right), \quad \frac{w'}{w} = \frac{A}{L+Ax}, \tag{3.38}$$

- exponentialy increasing width

$$w(x) = w_0\left[1 + A\exp\left(\frac{x}{\lambda}\right)\right], \quad \frac{w'}{w} = \frac{\frac{A}{\lambda}\exp(\ldots)}{1+A\exp(\ldots)}, \tag{3.39}$$

- gaussian enwidthment

$$w(x) = w_0\left[1 + A\exp\left(-\frac{x^2}{2\lambda^2}\right)\right], \quad \frac{w'}{w} = \frac{-\frac{A}{\lambda^2}\exp(\ldots)}{1+A\exp(\ldots)}, \tag{3.40}$$

- rectangular enwidthment

$$w(x) = \left\{\begin{array}{ll} w_0 + A, & L/2 - \lambda/2 < x < L/2 + \lambda/2; \\ w_0, & \text{otherwise}; \end{array}\right. \tag{3.41}$$

Chapter 4

Numerical aspects

4.1 Discretization

StkJJ uses space and time discretization for simulation. This parameters are given by switches described in Table 4.1.

Table 4.1: Discretization

Option	Type	Default	Description
`-dx`:Δx	float	0.05	discretization in space Δx
`-dt`:Δt_{rel}	float	0.25	Discretization in time relative to discretization in space *i.e.* real time step will be $\Delta t = \Delta t_{\text{rel}} \times \Delta x$. Note, that Δt_{rel} should be less than 1/2 to provide stability of numerical algorithm

Chapter 5

Initial and boundary conditions

5.1 Initial conditions

Simulations can be started with an initial phase distribution in each junction according to the settings in Tab. 5.1. The initial conditions can be specified in two ways: initial phase distributions from a state-file created during previous run of StkJJ or setting one of the predefined phase profiles in each JJ. Note, that when using predefined (nonuniform) phase profile (*i.e.* `-InitCond:3 or 4`) the fluxons shape is taken as in uncoupled LJJ. Therefore, if you simulate a coupled LJJs and an effort to create some fluxon configuration for relatively large $|S|$ fails, we recommend to go step by step from small to large $|S|$ saving the state each time. This saved state should be used on the next run for larger $|S|$. The procedure may be automated using UNIX shell scripts or DOS batch files. The examples of such scripts are provided with StkJJ.

You can also put positive (PSF) and negative (NSF) semi-fluxons (SF) into LJJ, which is useful when simulating LJJs with π-discontinuities. The corresponding options are described in Tab. 5.2. Note, that since SFs are static by nature (pinned to the discontinuity points), they are not affected by options like `-ICSpeed:` and `-ICSL:`.

It many cases it is natural to skip both `-SF_Pos:` and `-SF_Pol:` options. In this case the SFs will be placed at each π-point automatically. Then the most interesting options are `-SF_Order:0` and `-SF_Order:1`.

`-SF_Order:0` puts no SFs, *i.e.*, sets 0-π-0 phase jumps, and allows to study spontaneous formation of SFs.

`-SF_Order:1` puts PSF at each $-\pi$-discontinuity and NSF at $+\pi$-discontinuity. Thus phase jump $\pm\pi$ on the long scale is compensated by phase gradient due to SF. This also avoids global increase or decrease of of phase. The shape of PSF is given by (including $-\pi$-discontinuity):

$$\begin{aligned}\phi(x) &= +4\arctan\left[\tan\frac{\pi}{8}\exp(+x)\right], \quad x<0\\ \phi(x) &= -4\arctan\left[\tan\frac{\pi}{8}\exp(-x)\right], \quad x>0 \end{aligned} \tag{5.1}$$

5.2 Boundary conditions

Depending on the geometry, boundary condition can be very different.

- For linear (overlap) geometry, the boundary conditions are set by an external magnetic field, see Tab. 3.2.

- For annular or other loop-like geometry, we have to use periodic boundary conditions:

$$\phi(0) = \phi(L) + 2\pi N; \tag{5.2}$$

$$\phi_x(0) = \phi_x(L), \tag{5.3}$$

where N is the total number of fluxons trapped in corresponding LJJ, see options `-NFlux:`, `-NFlux0:` and `-NFlux1:` in Tab. 5.1. In this situation, external magnetic field enters as an additional term $h_x(x)$ in sin-Gordon equation and can be considered as a potential in which fluxon moves. Since such a potential can be created by different means, not only by external magnetic field, we separate the whole concept of the $h_x(x)$ term into a separate Sec. 3.3.

Thus the field set by `-H:` applies only in linear case and sets the boundary conditions. In annular case it has no effect.

Table 5.1: Initial conditions

Option	**Type**	**Default**	**Description**
`-InitCond:`id	int	0	defines type of initial conditions: $id = 0$ — for annular topology equivalent to `-InitCond:1`, for linear topology equivalent to `-InitCond:2`; $id = 1$ — uniform field (linear phase) distribution along the LJJ according to specified number of fluxons $N^{A,B}$; $id = 2$ — uniform field (linear phase) distribution according to specified value of H; $id = 3$ — N^A and N^B fluxons localized in the center of each junction, *i.e.* $2\pi N^{A,B}$ kinks in the center of corresponding LJJ; $id = 4$ — N^A, N^B fluxons distributed uniformly along corresponding LJJ
`-ICSpeed:`u	float	0.0	Initial Conditions Speed. Speed u of phase profile (fluxons) in both LJJs.
`-ICSL:`$u_{\max}$	float	$\bar{c}_+$	Initial Conditions Speed Limit. From $u_{\max}$ and u initial conditions routine "puts" properly Lorenz contracted moving phase profile (fluxons) into JJ ^{A,B}.
`-ICXShift0:`Δx^A	float	0.0	shift of phase profile (fluxons) in JJ^A to the right by Δx^A.
`-ICXShift1:`Δx^B	float	0.0	shift of phase profile (fluxons) in JJ^B to the right by Δx^B.
`-ICPS:`$\Delta\phi$	float	0.0	Obsolete, superseeded by `-ICRPS:`. Shifts the phase in JJ^B by $\Delta\phi$ *i.e.* $\phi^B{+}=\Delta\phi$.
`-ICAPS:`$\Delta\phi/\pi$	float	0.0	"absolute phase shift" — shifts the phase in all LJJs by $\Delta\phi$ given in units of π.
`-ICRPS:`$\Delta\phi/\pi$	float	0.0	"relative phase shift" — shifts the phase in LJJ^B by $\Delta\phi$ (given in units of π), i.e. $\phi^B{+}=\Delta\phi$.
`-NFlux:`N	int	1	number of fluxons $N = N^A = N^B$ in each LJJ.
`-NFlux0:`N^A	int	N	number of fluxons N^A in JJ^A. Overwrites N. Note that value of N^A from options file overwrites the value of N from command line.
`-NFlux1:`N^B	int	N	number of fluxons N^B in JJ^B. Overwrites N. Note that value of N^B from options file overwrites the value of N from command line.
-KinkPos$i$:$x_1, \ldots, x_n$	floats	""	Sets the arbitrary positions of the kinks (and anti-kinks) in i-th LJJ.
-KinkPol$i$:$p_1, \ldots, p_n$	ints	""	Sets the polarities of the kinks in i-th LJJ, e.g. -1,2,-1 means anti-kink, 2 kinks at the same pos, and one anti-kink.
`-SavedState:`*StateFile*	string	none	starts simulation with a phase distribution stored in the *StateFile*. All other initial conditions options have no effect. *StateFile* may be created using `-SaveState+` option on previous run of StkJJ.

Table 5.2: Positioning semifluxons

Option	**Type**	**Default**	**Description**
-SF_Pos:$x_1, \ldots, x_n$	floats		Sets the positions of SFs. If this list is empty the positions and number of SFs are the same as the positions and number of π-points.
-SF_Pol:$p_1, \ldots, p_n$	ints	””	Sets the polarities of the SFs, e.g. `-1` means NSF, 0 means no SF, +1 means PSF. If this list is empty, the polarities of SFs are set according to `-SF_Order:` switch.
-SF_Order:*ord*	int	0	Sets the polarities of SFs automatically in some order when they are not specified in `-SF_Pol:` option. $ord = 0$ — no SFs (all with zero polarity). $ord = 1$ — SFs compensates π-jumps. $ord = 2$ — Antiferromagnetic order: $\uparrow\downarrow\uparrow\downarrow \ldots \uparrow\downarrow$ $ord = 3$ — all SFs are positive: $\uparrow\uparrow \ldots \uparrow\uparrow$ $ord = 4$ — all SFs are negative: $\downarrow\downarrow \ldots \downarrow\downarrow$ $ord = 5$ — half of SFs are positive, half negative: $\uparrow\uparrow \ldots \uparrow\downarrow \ldots \downarrow\downarrow$

Chapter 6

Problem to solve

6.1 Current–Voltage Characteristic

By default, StkJJ calculates $V(I)$ of the system, sweeping I from Eqs. (3.1) and (3.2) according to the sweep list given in the command line. For this some parameters for averaging algorithm may be specified. These parameters are described in Table 6.1. Using `SweepList` consisting of more than one pass (3 elements) allows to trace hysteretic features of IVC as well as trace regions of interest with fine current step.

The averaging procedure works as follows:

1. simulate the phase dynamics for $T_{\rm init}$ time units to let the system relax.

2. simulate the phase dynamics for T_{aver} time units and calculate the average dc voltages $\bar{V}^{A,B}$ during this time interval as

$$\bar{V}^{A,B} = \frac{1}{T}\int_0^T \phi_t^{A,B}(t)\,dt = \frac{\phi^{A,B}(T) - \phi^{A,B}(0)}{T} \quad . \tag{6.1}$$

   For faster convergence, we use the fact that $\bar{V}^{A,B}$ do not depend on x and, therefore, we additionally use the spacial averaging of the phases $\phi^{A,B}$ in (6.1).

3. simulate the phase dynamics during $T_2^{\rm aver} = T_{\rm fac} \times T_1^{\rm aver}$ time units, calculate dc voltages for this new time interval and compare them with the previously calculated values.

4. repeat such iterations further increasing the averaging time interval $T_n^{\rm aver} = (T_{\rm fac})^{n-1} \times T_1^{\rm aver}$ by a factor $T_{\rm fac}$ until

   - either, the difference in dc voltages $\left| \bar{V}\Big|_{T_{\rm init}}^{T_{n+1}^{\rm aver}} - \bar{V}\Big|_{T_{\rm init}}^{T_n^{\rm aver}} \right|$
   - or, the difference in dc voltages $\left| \bar{V}\Big|_{T_n^{\rm aver}}^{T_{n+1}^{\rm aver}} - \bar{V}\Big|_{T_{n-1}^{\rm aver}}^{T_n^{\rm aver}} \right|$

   becomes less than a given accuracy δV.

Note, that if $f_{I_{\rm rf}} \neq 0$ and $I_{\rm rf} \neq 0$, then the averaging interval will be aways rounded to the integer number of the periods of $I_{\rm rf}$.

Note, that instead of voltage $\bar{v}$ StkJJ shows and outputs the value of "velocity" $\bar{v}\frac{\ell}{2\pi}$. This is done so because this is equal to the velocity u of a fluxon in the annular LJJ or at ZFS in linear LJJ. Thus, the first fluxon/zero-field step will have asymptotic "voltage" equal to 1, the second fluxon/zero-field step $u = 2$, *etc.*. We often call such a $\gamma(u)$ dependence a current-velocity characteristic. Note, that in this notation the McCumber line is $\gamma = \frac{2\pi}{\ell}\alpha u$. The Fiske steps have spacing $\Delta u = \frac{1}{2}$.

Table 6.1: IVC averaging options

Option	**Type**	**Default**	**Description**
`-IV_T_Init:`$T_{\rm init}$	float	10.0	initial relaxation time $T_{\rm init}$.
`-IV_T_Aver:`$T_1^{\rm aver}$	float	10.0	initial averaging time.
`-IV_V_Err:`$\Delta V_{\rm IVC}$	float	0.005	maximum acceptable difference in averaged voltage between two periods *i.e.* accuracy of convergence. Note that $\Delta V_{\rm IVC} = 0.001$ *does not* mean that averaged voltage value has this accuracy. Instead, it means that the difference between the last and prelast values of averaged voltage is less than 0.001.
`-IV_T_Fact:`$T_{\rm fact}$	float	1.2	The factor by which the averaging time interval is extended on every averaging iteration, if maximum error is still not achieved. Thus, the first time averaging takes place over $T_1^{\rm aver}$ time units, next time over $T_2^{\rm aver} = T_1^{\rm aver} \times T_{\rm fact}$ and obtained mean voltages are compared. If difference is larger than $\Delta V_{\rm IVC}$ than the mean voltage is calculated over the next $T_3^{\rm aver} = T_2^{\rm aver} \times T_{\rm fact}$ time units and compared with the mean voltage from previous step *i.e.* averaged over $T_2^{\rm aver}$ units of time *etc.* until convergence will be achieved or time $T_{\rm max}$ will be exceeded.
`-IV_T_Max:`$T_{\rm max}$	float	1000.0	Maximum amount of time after which averaging is stopped unconditionally. Note that averaging stops not immediately when simulator reaches $T_{\rm max}$ but when simulator finishes the calculation of average voltage over $T_i^{\rm aver}$ time interval. This last value of averaged voltage will be taken as an average voltage in a given point of IVC.
-dV_Stop:$\Delta V_{\rm stop}$	float	10^9	if in some point of IVC after averaging $\lvert V^A - V^B\rvert > \Delta V_{\rm stop}$, StkJJ exits immediately. Useful to save simulation time in batch runs or in $I_{\rm max}(H)$ mode.
`-IV_AvMode:`id	int	3	Averaging mode bit mask: $id = 1$ — old way (integral) $id = 2$ — new way (diff.) All specified modes will be used and convergence will be reached if at least one of them converges.
`-$IV_AvLog:`$level$	int	0	Log level to monitor the convergence of the averaging routine: $level = 0$ — do not log $level > 0$ — log t, V and $\delta V = V(t) - V(t - \Delta t)$ into the file `aver_new.log` and `aver_old.log` for the “new” and the “old” averaging algorithms, respectively.

6.2 Critical current vs. magnetic field

$I_c^{A,B}(H)$ [actually it is $I_{\max}^{A,B}(H)$] procedure can be applied to a linear or annular LJJ in external magnetic fields H applied in the plane of LJJs. It is not obligatory to start $I_c(H)$ procedure from the point on IVC close to the step for which we are going to measure $I_{\max}^{A,B}(H)$. You can start far away from this point. As soon as bias point on IVC enters the zone of interest $V_{\min} < V < V_{\max}$, $I_c(H)$ procedure starts " watching" looking for $I_{\max}$. If bias point on IVC never enters a zone of interest, StkJJ writes $I_{\max}(H) = 0$ to an output file.

Table 6.2: $I_c(H)$ options.

Option	**Type**	**Default**	**Description**
`-IcH[+|-]`	bool	off	Switches $I_c(H)$ mode on or off.
`-IcH_MinV:`$V_{\min}$	float	0.0	$V_{\min}$ defines the minimum voltage of the step where we are going to trace $I_c(H)$ or $I_{\max}(H)$. StkJJ assume that we are on the proper step while $V_{\min} < V < V_{\max}$
`-IcH_MaxV:`$V_{\max}$	float	1.0	$V_{\max}$ defines the maximum voltage of the step where we are going to trace $I_c(H)$ or $I_{\max}(H)$. StkJJ assume that we are on the proper step while $V_{\min} < V < V_{\max}$
`-IcH_FineStep:`$\Delta\gamma$	float	0.01	Accuracy of $I_c(H)$ or $I_{\max}(H)$ algorithm. First StkJJ makes rough $I_{\max}(H)$, then restore the state of the system in the last point before switching out of the step and approaches the switching point with very small steps $\Delta\gamma$ to determine $I_{\max}$ for current H with high accuracy.
`-IcH_MinH:`$H_{\min}$	float	0.0	obsolete, see `-Sweep2Seq:` on p. 27 Starting value of field for $I_{\max}(H)$ procedure. Note that StkJJ allows $H_{\min} > H_{\max}$.
`-IcH_MaxH:`$H_{\max}$	float	1.0	obsolete, see `-Sweep2Seq:` on p. 27 Ending value of field for $I_{\max}(H)$ procedure. Note that StkJJ allows $H_{\min} > H_{\max}$.
`-IcH_dH:`ΔH	float	0.1	obsolete, see `-Sweep2Seq:` on p. 27 Step in H for $I_{\max}(H)$ procedure. The sign is not important because StkJJ determine it from the values of $H_{\min}$ and $H_{\max}$.
`-IcH_Veps:`ΔV_{ICH}	float	0.01	Difference in voltages between two LJJ. If $\lvert V^A - V^B\rvert > \Delta V_{ICH}$, then StkJJ assumes that LJJ's and voltage delocked and the end of synchronous step is reached. This option makes effect only for `-IcH_VMode:3`
`-IcH_VMode:`*mode*	int	1	*mode* = 1 — make $I_{\max}(H)$ for JJ^A only; *mode* = 2 — make $I_{\max}(H)$ for JJ^B only; *mode* = 3 — make $I_{\max}(H)$ for both JJ^A and JJ ^{B} and take into account ΔV_{ICH}.

How large should be `-IcH_MaxH:`$H_{\max}$ to see several minima on $I_c(H)$? The normalization used in StkJJ is such that the first minimum in conventional rather long LJJ will be at $H = 2$. In short JJ ($L \lesssim 2$), the first minumum is at $H = 2\pi/L$.

Pitfall/Bug in command line processing routine: if you have one of the `-IcH_*` options in a command line or an options file before `-IcH` option, or you have `-IcH_*` options in a command line and `-IcH` only in an option file, StkJJ will erroneously interpret `-IcH_*` option as `-IcH` with illegal modifier "_". In this case StkJJ will print an error message and exit. To avoid this problem always put `-IcH+` *before* other `-IcH_*` options.

6.3 Sweeping arbitrary parameter

StkJJ allows to sweep not only bias current and magnetic field (in $I_{\max}(H)$) but also many other parameters. This is controlled by options described in the Table 6.3. The "name" can be one of the following: `I, Bias0, Bias1, H, HL, HR, J, C, R, D, L, PotDepth, PotWidth, H_RF, FreqH_RF, I_DC, I_RF, FreqI_RF, I_F, Discontinuity, kappa` . Note that `Discontinuity` and `kappa` have the same meaning, *i.e.*, they both control the strength of the two δ-injectors, *i.e.*, the phase jump at the discontinuity point.

Table 6.3: Sweeping arbitrary parameter

Option	Type	Default	Description
`-Sweep1Var:`name	string	"I"	says which parameter of the system to sweep instead of driving current in IVC.
`-Sweep2Var:`name	string	"H"	says which parameter of the system to sweep instead of magnetic field in $I_{\max}(H)$.
`-SweepH[+|-]`	bool	off	OBSOLETE since v3.58! USE `-Sweep1Var:` or `-Sweep2Var:` instead. This switch says what to sweep: current (`-SweepH-`) or field (`-SweepH+`).
`-Sweep2Seq:`⟨LIST⟩	string	""	⟨list⟩=from$_0$,to$_1$,step$_1$[,to$_2$,step$_2$,to$_3$,step$_3$,...] Sets the sweep2 sequence.

6.4 Simulating hot-spot scanning (LTSM)

With StkJJ you can simulate the scanning results of Low Temperature Scanning Microscope (LTSM), which can be based on e-beam or laser-beam (LTSEM and LTSLM). At this point no distinction is made between these two and the only effect considered is the local heating of the sample by δT. Current version of StkJJ implements only the change of $j_c(T)$ and $\alpha(T)$.

The parameters of the hot spot can be set using options from Tab. 6.4. To get the dependence of V or of I_c *vs.* hot spot position $x_{\rm hs}$, you have to set `-Sweep1Var:HotSpotPos` or `-Sweep2Var:HotSpotPos`, respectively. In case of $I_c(x_{\rm hs})$, you should also set `-IcH_MinH:` equal to the initial hot spot position, `-IcH_MaxH:` final hot spot position, and `-IcH_dH:` equal to the step $\Delta x_{\rm hs}$.

6.4.1 $\alpha(T)$ dependences

In the following we adopt the following notations: T_0 is the temperature of the sample; $\alpha_0 = \alpha(T_0)$ is the damping at this temperature which user sets using `-Alpha0:` option. The following $\alpha(T)$ dependences are implemented.

- Constant (no dependence)
$$\alpha(T) = \alpha_0, \tag{6.2}$$
- Power law dependence is given by the formula:
$$\alpha(T) = \alpha(0)\left(\frac{T}{T_R}\right)^n = \alpha_0\left(\frac{T}{T_0}\right)^n, \tag{6.3}$$

Table 6.4: Hot Spot options (LTSM)

Option	**Type**	**Default**	**Description**
`-HotSpotShape:`*id*	ID	no	Shape of the hot spot. 0"no": no hot spot 1"rect": rectangular hot spot 2"gauss": gaussian hot spot
`-HotSpotSize:`*size*	float	0.1	Size of the hot spot (in units of λ_J). For "rect" hot spot: radius For "gauss" hot spot: σ, aka the standard deviation.
`-HotSpotPos:`x_{hs}	float	0.0	Hot spot position.
`-HotSpotdT:`δT	float	0.0	Temperature increase by the hot spot
`-T:`T	float	0.7	Temperature of the sample
`-Tc:`T_c	float	1.0	T_c of the sample
`-HS_Jc(T):`*id*	ID	const	Type of $j_c(T)$ dependence 0"const":$j_c(T) = const$ 1"pow":$j_c(T) = [1-(T/T_c)^m]^n$
`-HS_Jc(T)m:`m `-HS_Jc(T)n:`n	double	1.0	m and n for `-HotSpotJc(T):1`. $m = n = 1$ represents linear slope useful for testing purposes. Real JJ dependence can be modelled by $m = 2$, $n = 0.5$
`-HS_alpha(T):`*id*	ID	0	Type of $\alpha(T)$ dependence 0"no" :no dependence, see Eq. (6.2) 1"exp":exponential dependence, see Eq. (6.4) 2"pow":power-law dependence, see Eq. (6.3)
`-HS_alphaPow_n:`n	double	1.0	Power n for power-like dependence $R_n(T)$ as in Eq. (6.3). Makes sense only for `-HS_alpha(T):1`
`-HS_alphaExp_Tr:`T_R	double	T_c	Typical temperature for which the damping increases by e times, in accordance with Eq. (6.4). Makes sense only for `-HS_alpha(T):2`

- Exponential dependence is given by the formula:

$$\alpha(T) = \alpha(0)\exp\left(\frac{T}{T_R}\right) = \alpha_0 \exp\left(\frac{T-T_0}{T_R}\right), \qquad (6.4)$$

Pitfall: Hot spot in annular system knows about periodicity and implemented as 3 hot spots at positions $x_{\mathrm{hs}} - L$, x_{hs} and $x_{\mathrm{hs}} + L$. Take care when L is comparable with the hot spot size. In the case of the Gaussian hot spot its shape deviates from the Gaussian distribution because it is a sum of 3 Gaussian peaks. In the case of rectangular hot spot and hot spot size $\geq L$, the hot spot will be infinitely large and span over the whole junction, thus just globally suppressing j_c and *alpha*.

Chapter 7

The results

7.1 Saving phase gradients, instant voltages, *etc.* in each point of IVC

A variety of space and time dependent fields such as $\phi(x)$, $\phi_x(x)$, $\phi_t(x)$, *etc.*. can be calculated using StkJJ . These fields are saved at every n-th point of the IVC (see `-every:`n option) after the averaging and FFT (if requested) are complete. The fields are saved into separate files that are numbered consequently. The options that specify what kind of output is required are listed in Tab. 7.1. The names and format of the files are given in Tab. 8.4. Tab. **??**

Table 7.1: Space and time dependent fields.

Option	Type	Default	Description
`-SaveState[+|-]`	bool	off	saves the state of the system (phase distribution) to a file `stat####.dat`
`-OutPhase[+|-]`	bool	off	saves the phases $\phi^{A,B}(x)$ to a file `p####.dat`
`-OutMuPhase[+|-]`	bool	off	saves phases $\mu^{A,B}(x)$ to a file `m####.dat`
`-OutJs[+|-]`	bool	off	saves supercurrent $\sin\phi^A(x)$ and $\sin\phi^B(x)/J$ to a file `js####.dat`
`-OutVoltage[+|-]`	bool	off	saves voltages $\phi_t^{A,B}(x)$ to a file `v####.dat`
`-OutField[+|-]`	bool	off	saves the phase gradients (magnetic fields) $\phi_x^{A,B}(x)$ to a file `h####.dat`
`-OutPhaseGrad:[+|-]`	bool	off	saves the phase gradients $\phi_x^{A,B}(x)$ to a file `phi_x####.dat`
`-OutKinkTraj:`$\Delta T_{\text{snapshot}}$	double	0.0	If $\Delta T_{\text{snapshot}} > 0$ saves the kink's trajectories to a file `KT####.dat`. See Sec. 7.5 on p. 34 for details.
`-FluxFile:`$\langle name \rangle$	string	""	Saves the magnetic flux Φ_i in the specified regions of the LJJ into file $\langle name \rangle$, see `-FluxPickupRange:` for details. The file format is `sweep` Φ_1^A Φ_1^B Φ_2^A Φ_2^B ...
`-FluxPickupRange:`$\langle x_1, x_2, x_3, x_4 \rangle$	floats	""	specifies several ranges along x, i.e. $[x_1, x_2]$, $[x_3, x_4]$, ..., where the flux should be calculated.

Table 7.2: Miscelaneous output options

Option	**Type**	**Default**	**Description**
`-CoordPresent:`	int	0	How to present coordinates in the output files: 0 : old way, i.e. $0, \Delta x, 2\Delta x, \dots, (n-1)\Delta x$. 1 : real, i.e. $\frac{\Delta x}{2}, \frac{\Delta x}{2} + dx, \frac{\Delta x}{2} + 2\Delta x, \dots, \frac{\Delta x}{2} + (n-1)\Delta x = L - \frac{\Delta x}{2}$.
`-n:`n	int	0	allows to start enumeration of file names such as `h####.dat` from any number n rather than from zero. This feature is especially useful when you restart simulation and use `-SavedState:file` switch.
`-every:`n	int	0	allows to save snapshots or states of the system in every n-th point of the sweep sequence rather than in every point. It is useful when you wanna sweep with very fine step but don't wanna create a huge number of state files or profile files.

7.2 Calculating Impedance of the LJJ in each point of IVC

If ac bias current is applied through the LJJ, StkJJ have possibility to calculate the impedance $Z = R + iX$ (R and X are resistance and reactance, respectively) of the system at the driving frequency. StkJJ calculates R and X for each sweep point in the following way. First StkJJ finds the instant "effective voltage" across LJJ as

$$V_{\text{eff}}(t) = \frac{P(t)}{I(t)} = \frac{\sum_{i=0}^{N} V_i(t) I_i(t)}{I_{\text{rf}} \sin(\omega t)}. \tag{7.1}$$

Then R_{ac} and X_{ac} are calculated as:

$$R_{dc}(I_{\text{dc}}) = \frac{2}{I_{\text{dc}}} \int_0^{T_{\text{imp}}} V_{\text{eff}}(t)\, dt; \tag{7.2}$$

$$R_{ac}(I_{\text{rf}}) = \frac{2}{I_{\text{rf}}} \int_0^{T_{\text{imp}}} V_{\text{eff}}(t) \sin \omega t\, dt; \tag{7.3}$$

$$X_{ac}(I_{\text{rf}}) = \frac{2}{I_{\text{rf}}} \int_0^{T_{\text{imp}}} V_{\text{eff}}(t) \cos \omega t\, dt. \tag{7.4}$$

If $I_{\text{dc}} = 0$ in Eq. (7.2) or $I_{\text{rf}} = 0$ in Eqs. (7.3) and (7.4) are equal to zero, StkJJ sets the resulting $R_{dc}(0)$, $R_{ac}(0)$ and $X_{ac}(0)$ equal to zero without any calculations.

The impedance related options are summarized in Tab. 7.3. Note, that impedance calculation is rather time consuming, so one can calculate impedance only in every n-th point of IVC by using `-every:`n.

Table 7.3: Impedance options.

Option	**Type**	**Default**	**Description**
`-OutImp[+|-]`	bool	off	Obsolete. Superseeded by `-ImpCalcTime:`. If `-OutImp` is on, StkJJ calculates R and X integrating over $T_{\text{imp}} = T_{\text{aver}}$ given by `-IV_T_Aver:` option rounded to the closest integer number of periods of ac drive (see `-Freq_RF:`).
`-ImpCalcTime:`T_{imp}	float	0.0	StkJJ calculates R and X integrating over T_{imp} rounded to the closest integer number of periods $2\pi/\omega$ of ac drive (see `-Freq_RF:`). If $T_{\text{imp}} = 0$, impedance is not calculated.
`-ImpFileName:`impfile.dat	string	imp.dat	StkJJ saves γ_{ac}, R_{dc}, R_{ac} and X_{ac} (in this order of columns) into file "impfile.dat".
`-ImpOutFmt:`%9G	string	%9.6f	StkJJ uses specified C notation to write the values of R_{dc}, R_{ac} and X_{ac} into file. The notation with fixed point is not always convinient esp. when later you would like to present data in log-scale (U will see artifactual discretization at low values).

7.3 Producing snapshots for animation

To produce snapshots for animation use one of the options listed in Tab. 7.4. Each of these options (except `-FRAMES:`) has one parameter Δt, which specifies the time interval between snapshots. The number of snapshots (frames) and files is given by the option `-FRAMES:`N_{fr}. Each file consists of the following columns:

x `value-LJJ`$_0$ `value-LJJ`$_1$ ...

After producing snapshots StkJJ saves the state of the system in the file `last.stt`, so that it can be used to continue simulation from the time when it was stopped.

Table 7.4: Snapshooting options

Option	**Type**	**Default**	**Description**
`-PXT:`ΔT	float	none	produces files `p####.dat` with $\phi(x)$.
`-MXT:`ΔT	float	none	produces files `m####.dat` with $\mu(x)$.
`-HXT:`ΔT	float	none	produces files `h####.dat` with $\phi_x(x)$.
`-VXT:`ΔT	float	none	produces files `vxt####.dat` with $\phi_t(x)$.
`-JXT:`ΔT	float	none	produces files `jxt####.dat` with $j_s(x)$.
`-PowXT:`ΔT	float	none	produces files `powx####.dat` with power dissipation snapshots.
`-EpXT:`ΔT	float	none	produces files `Ep####.dat` with potential energy $U(x) = \frac{1}{2}\phi_x^2 + 1 - \cos(\phi)$ snapshots.
`-LjXT:`ΔT	float	none	produces files `ljx####.dat` with Josephson inductance $L_j(x)$ snapshots.
`-FRAMES:`N_{fr}	int	200	the number of snapshots (frames) and files.
`-V(t)@x0:`Δt	float	-1.0	saves the voltages $V(t) = \phi_t(t)$ measured with the interval Δt at the point $x = x_0$ into file.
`-P(t)@x0:`Δt	float	-1.0	saves the phases $\phi(t)$ measured with the interval Δt at the point $x = x_0$ into file.
`-PickPt:`i	int	$N/2$	sets the phase/voltage pickup point, by default in the middle of LJJ.
`-E(t):`dt	float	?	saves the energies specified by `-E(t)Energies:` option into the output file.
`-E(t)Energies:`strList	strList	””	lists which energies to output, e.g. `-E(t)Energies:Ek,Ep,Ea`. See the energy output section 7.9 for the names of different energies.

7.4 Measuring $\phi(t)$, $V(t)$ at pickup point

Table 7.5: Options for measuring $\phi(t)$, $V(t)$ at pickup point

Option	Type	Default	Description
`-FRAMES:`N_{fr}	int	200	the number of snapshots (frames) and files.
`-PickPt:`i	int	$N/2$	sets the phase/voltage pickup point, by default in the middle of LJJ.
`-V(t)@x0:`Δt	float	-1.0	saves the voltages $V(t) = \phi_t(t)$ measured with the interval Δt at the point $x = x_0$ into file.
`-P(t)@x0:`Δt	float	-1.0	saves the phases $\phi(t)$ measured with the interval Δt at the point $x = x_0$ into file.

7.5 Tracking trajectories of vortices

To track trajectrories of fluxons it is enough to use only `-X(T):` switch. StkJJ creates a file (the file name is given in the command line, by the 1st non-switch parameter) with the coordinates of the kinks *vs.* time. Each row contains time, the coordinate x of a kink in LJJ^A, the coordinate x of antikink in LJJ^A, x of fluxon in LJJ^B, x of antifluxon in LJJ^B. If there are more the one (anti-)kink, StkJJ writes the second line with the same time value and coordinate of next (anti-)kink. If there is a kink, but no more anti-kinks in one of the LJJ's, StkJJ writes -1 as an (anti-)kink coordinate. Note, that if for a given moment of time the (anti-)kinks are absent in the system, the line corresponding to this time will not be written at all into the output file. NOTE, this behavior is valid for StkJJ v. > 3.62

If you would like to have more control, you can specify exactly the value of the phase in the so called Trajectory Tracking Point (TTP). Basically StkJJ outputs the trajectory of TTP. The phase at TTP is constructed as:

$$\phi_{TTP} = \mathcal{S} + \mathcal{P}\pi(2n+1),$$

where $\mathcal{P}$ is the period, *e.g.*, $\mathcal{P} = 1$ for fluxon, $\mathcal{P} = 0.5$ for semifluxon. The $\mathcal{S}$ is the shift. This shift can be set automatically to $\arcsin(\gamma)$ to compensate for the global phase shift due to dc bias current. This is accomplished by using `-TTPSAuto` option, in this case the value of `-TTPS:` is ignored. Note that `-TTPSAuto` works only if

- dc bias does not depend on x, $\gamma(x) = const$,
- no ac (rf) current applied,
- critical current does not depend on x, $j_c(x) = const$.

The option `-TTPSAuto` takes into account that currents passing through the junctions may have different values due to `-Bias0:` or `-Bias1:` switches, or may induce different phase shifts in different junctions due to `-J` option.

The summary of all options related to trajectory tracking is given in Tab. 7.6. After saving the file with trajectories StkJJ saves the state of the system in the file `last.stt`, so that it can be used to continue simulation from the time when it was stopped.

Table 7.6: Trajectory tracking options

Option	**Type**	**Default**	**Description**
`-X(T):`ΔT	float	none	Output $x(t)$ trajectroties of vortices. The positions are analyzed every ΔT time units, $N_{\rm fr}$ times.
`-FRAMES:`$N_{\rm fr}$	int	1000	number of times StkJJ will analyze the positions of vortices.
`-TTPP:`$\mathcal{P}$	float	1.0	Trajecttory Tracking Point Period (TTPP). Valid only together with `-X(T):`ΔT.
`-TTPS:`$\mathcal{S}$	float	0.0	Trajecttory Tracking Point Shift (TTPS). Valid only together with `-X(T):`ΔT.
`-TTPSAuto[+|-]`	bool	off	Auto sets `-TTPS:` to $\arcsin(\gamma)$, if γ does not depend on x.

7.6 Spectral characteristics

Fourier transforms performed by StkJJ use a special windowing technique discussed in detail in [6]. The FFT of the windowed data is performed using a NAG library routine. The specific options to use the FFT features of StkJJ are shown in Tab. 7.7.

Table 7.7: Fourier transform settings

Option	**Type**	**Default**	**Description**
`-OutFFT:`N_{FFT}	int	none	Switches on numerical FFT of N_{FFT} samples of the local voltage $V(t)$ at the arbitrary point of both junctions. The rate of samples in time is determined by the Nyquist frequency. Automatically analyze spectra to find maxima. Writes spectra and maxima to files. (filenames: fft####.dat, maxx###.dat)
`-FFTx:`x_0	float	0.0	Specifies the coordinate x_0 where data for FFT will be collected (since v 3.49)
`-FFTpt:`i	uint	0	The same as "`-FFTx:`x_0" but specifies the point number. Overwrites "`-FFTx:`x_0" (since v 3.49)
`-MaxFreq:`$f_{\max}$	float	10^9	sets the Nyquist frequency for sampling $V(t)$ signals for output and FFT. Decreasing this value one may increase resolution but produces the mirroring effect on the spectra at $f = f_{\max}$
`-OutVT:`N	float	none	saves local voltage $V(t)$ of both junctions for N subsequent time steps with a rate determined by the Nyquist frequency $f_{\max}$ to a file (filename: VT####.dat)

`-FFTWindowType:`*wtype*	int	0	the type of windowing function:
			wtype = 0 — rectangular (no window);
			wtype = 1 — Hamming;
			wtype = 2 — Parzen;
			wtype = 3 — Gaussian.

`-FFTWindowWidth:`WW	float	0.0447325	specifies the width of Gaussian window.
`-SEP[+|-]`	bool	on	skips extra points in FFT on output in order to save space on the disk. (filename: fftxx###.dat)
`-FindMaxThreshold:`ΔA	float	0.001	Threshold parameter for procedure which searches maxima in the spectrum. Only peaks higher than ΔA (rel. to neighbors) are considered as maxima.

7.7 Eigenvalues & eigenfrequencies, stability of solutions

7.7.1 Old Naive algorithm with $2N \times 2N$ non-symmetric matrix

Once you have got a stationary solution you can investigate its stability and find eigenfrequencies. You can do this in every point of the IVC.

Imagine that there is a solution $\mu_0(x)$ of perturbed sine-Gordon equation (may be obtained numerically). Then assume that there is a perturbation $\delta(x,t)$ to this solution such that

$$\mu(x,t) = \mu_0(x) + \delta(x,t) \tag{7.5}$$

is a new solution of sine-Gordon equation. The equation for perturbation is

$$\delta_{xx} - \delta_{tt} - j_c(x)\mathrm{CPR}'(\mu_0(x))\delta = \alpha\delta_t + \frac{w'}{w}\delta_x, \tag{7.6}$$

This 2nd order PDE is presented as a system of $N = L/dx$ ODE's for the variables $\delta_i = \delta(x_i)$ which can be written as

$$\delta_{i,tt} + \alpha\delta_{i,t} = \left(\frac{1}{\Delta x^2} - \frac{w_x}{w}\frac{1}{2\Delta x}\right)\delta_{i-1} - \left[\frac{2}{\Delta x^2} + \mathrm{CPR}'\phi_i\right]\delta_i + \left(\frac{1}{\Delta x^2} + \frac{w_x}{w}\frac{1}{2\Delta x}\right)\delta_{i+1}. \tag{7.7}$$

Further we reduce this to the $2N$ 1st order equations by introducing

$$y_{2i} = \delta_i; \tag{7.8a}$$
$$y_{2i+1} = \delta_{i,t}. \tag{7.8b}$$

Then Eq. (7.7) can be rewritten in a matrix form

$$\dot{\mathbf{y}} = \mathbf{A}\cdot\mathbf{y},$$

where $\mathbf{A}$ is the $2N \times 2N$ matrix which can be constructed from Eqs. (7.7) and (7.8). This matrix has some non-zero elements only along main 5 diagonals and it is not symmetric. To investigate the stability, we find the $2N$ eigenvalues of this matrix. All of them are complex because the matrix is non-symmetric.

7.7.2 New algorithm(s)

The old algorithm has a disadvantage that one should work with a huge $2N \times 2N$ non-symmetric matrix. The new algorithm described below need only $N \times N$ matrix and, in some simple cases, this can become tri-diagonal symmetric matrix for which eigenvalues can be calculated very fast (time $\propto N^2$ instead of N^3).

Imagine that there is a solution $\mu_0(x)$ of static perturbed sine-Gordon equation obtained numerically. Then assume that there is a perturbation $\delta(x,t) = \delta(x)e^{\lambda t}$ to this solution such that

$$\mu(x,t) = \mu_0(x) + \delta(x)e^{\lambda t} \tag{7.9}$$

is a new solution of sine-Gordon equation. The equation for perturbation is

$$\delta_{xx} - \frac{w_x}{w}\delta_x - j_c(x)\mathrm{CPR}'(\mu_0(x))\delta = \Lambda\delta, \tag{7.10}$$

where the CPR' means the derivative of the CPR with respect to the phase. Assuming that α is not a function of x, we introduce new eigenvalue variable Λ instead of λ:

$$\Lambda = \lambda^2 + \alpha\lambda. \tag{7.11}$$

If we will be able to find a spectrum Λ_i of Λ in Eq. (7.10), the corresponding λ_i are given by

$$\lambda_i^{\pm} = \frac{-\alpha \pm \sqrt{\alpha^2 + 4\Lambda_i}}{2}. \tag{7.12}$$

Table 7.8: Eigenvalues calculation.

Option	Type	Default	Description
`-OutEigenValues:[+|-]`	bool	off	Output all eigenvalues at each point of the IVC into a file `lambda????.dat`
`-EigenValFile:`filename	string	””	For each value of the sweep parameter writes several eigenvalues (with lowest frequencies) into the file filename. Valid only for `-OutEigenValues+`
`-EigenVals:`m	uint	3	How many eigenvalues (with lowest frequencies) write into the file specified by `-EigenValFile:`filename
`-EigenValSelect:`$\langle rule\rangle$	int	0	sets the rules for selection of eigenvalues with lowest frequencies: $rule = 0$ — no special selection $rule = 1$ — strict selection ($\Re(\lambda)$ must be equal to $-\alpha$) $rule = 2$ — not very strict selection ($\Im(\lambda)$ should not be zero and $\Re(\lambda)$ should not be positive)
`-EigenValAlpha:`α_0	float	α	damping which should be used to calculate eigenvalues.
`-EigenValProc:`id	id	auto	Allows to choose specific procedure/algorithm to calculate eigenvalues/functions: 0,auto: automatically use fastest algorithm 1,2Nx2N,old: general $2N \times 2N$ asymmetric matrix (old) algorithm 2,NxN: $N \times N$ asymmetric matrix (new) algorithm 3,NxN_Sym: $N \times N$ symmetric matrix algorithm ($w(x) = const$) 4,NxN_Sym_Diag,fast: $N \times N$ 3-diagonal symmetric matrix algorithm ($w(x) = const$, linear LJJ)
`-OutEigenFns:`n	int	0	saves eigenfunctions corresponding to n lowest eigenvalues to a file `psi????.dat`.
`-EffEscPotFile:`	str	“”	saves ω_0, M and F in the file for each value of sweep1. The calculation of eigenfunctions must be on.
`-$DebugEV`[+|−]	bool	0	Debug output from eigenvalue routine?

Note that a pair of complex conjugate or a pair of real $\lambda_i^{\pm}$ may correspond to to one Λ_i. If there is at least one $\Lambda_n > 0$, then $\lambda_n^+ > 0$ and the system is unstable. On the other hand, if all $\Lambda_i < -\alpha^2/4$, then all $\lambda_i^{\pm}$ are complex conjugate with $\Re(\lambda_i) = -\alpha/2 < 0$ and the system is stable. In the last case, when some of the Λ_i lay in the interval $-\alpha^2/4 < \Lambda_i < 0$ and the others have $\Lambda < -\alpha^2/4$, $\lambda_i^{\pm}$ are two real negative eigenvalues and the system is stable again. Thus, we conclude that the system is stable if all $\Lambda_i < 0$, and *the damping α does not affect stability* of our system. What *damping does affect* is the presence and *the number of the eigenfrequencies* $\omega_{0,i} = |\Im(\lambda_i)|$. If some of the Λ_i lay in the interval $-\alpha^2/4 < \Lambda_i < 0$, then the system will have no eigenfrequencies corresponding to these Λ_i in the presence of damping, but if the damping is switched off, eigenfrequencies will appear.

To solve Eq. (7.10) numerically we discretize it, presenting $\delta_{x,i}$ and $\delta_{xx,i}$ at the point x_i as

$$\delta_i = \delta(x_i) \quad , \quad \delta_x = \frac{\delta_{i+1} - \delta_{i-1}}{2\Delta x} \quad , \quad \delta_{xx} = \frac{\delta_{i+1} - 2\delta_i + \delta_{i-1}}{\Delta x^2}. \tag{7.13}$$

for all internal points of LJJ, i.e. for $i \neq 0$ and $i \neq N$. For the edge points, the boundary condition $\delta_x(0) = \delta_x(L) = 0$ should be taken into account. Thus, instead of Eq. (7.14) we get

$$\delta_{x,0} = \delta_{x,N-1} = 0 \quad , \quad \delta_{xx,0} = \frac{\delta_1 - \delta_0}{\Delta x^2} \quad , \quad \delta_{xx,N-1} = \frac{\delta_{N-2} - \delta_{N-1}}{\Delta x^2}. \tag{7.14}$$

Then the eigenvalue problem (7.10) can be written in a matrix form

$$\mathbf{A} \cdot \boldsymbol{\delta} = \Lambda \boldsymbol{\delta}, \tag{7.15}$$

where $\boldsymbol{\delta}$ is an N-dimensional vector with the components $\delta_i = \delta(x_i)$ and the $N \times N$ matrix $\mathbf{A}$ looks as follows.

$$\boldsymbol{A}_{\text{ann}} = \begin{bmatrix} \ddots & \ddots & 0 & 0 & e_0^+ \\ \ddots & \ddots & \ddots & 0 & 0 \\ 0 & e_i^+ & d_i & e_i^- & 0 \\ 0 & 0 & \ddots & \ddots & \ddots \\ e_N^- & 0 & 0 & \ddots & \ddots \end{bmatrix} \quad , \quad \boldsymbol{A}_{\text{lin}} = \begin{bmatrix} \ddots & \ddots & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & 0 & 0 \\ 0 & e_i^+ & d_i & e_i^- & 0 \\ 0 & 0 & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & \ddots & \ddots \end{bmatrix}, \tag{7.16}$$

for annular and linear LJJ accordingly. For annular case the off-diagonal elements are wrapped. For linear case the 0-th and $N-1$-st elements are expressed in a different way because of zero boundary conditions for $\delta_x(x)$. The diagonal and off-diagonal elements are given by the following formulas

$$d_i = -\frac{k_d}{\Delta x^2} + j_c(x_i)\text{CPR}'(\mu_0(x_i)); \tag{7.17}$$

$$e_i^\pm = \frac{1}{\Delta x^2} \pm \frac{k_e}{2\Delta x}\frac{w_x}{w}, \tag{7.18}$$

where $k_d = 2$, $k_e = 1$ for internal points $(0 < i < N-1)$ and $k_d = 1$, $k_e = 0$ for edge points $(i = 0, i = N-1)$.

To solve eigenvalue problem (7.15) we can use several numerical approaches.

- In the very general case (when $w_x \neq 0$) the $\boldsymbol{A}$ is non-symmetric and one has to use the most general eigenvalue routine for $N \times N$ matrices: reduction to Heisenberg form and searching for eigenvalues of Heisenberg matrix[7]. This numerical procedure corresponds to `-EigenValProc:2`
- If there is no width modulation $w_x \equiv 0$, then $\boldsymbol{A}$ is symmetric. It can be reduced to symmetric 3-diagonal matrix. Then eigenvalues and eigenfunctions can be found using very effective procedure described below. This numerical procedure corresponds to `-EigenValProc:3`
- If, in addition to $w_x \equiv 0$, the topology of the system is linear, then there are no parasitic wrapped off-diagonal elements and the matrix is 3-diagonal initially. The eigenvalues and eigenfunctions can be found using very effective procedure with working time $\propto N^2$ (if one does not need eigenvectors/eigenfunction) and $\propto N^3$ if one needs them. This numerical procedure corresponds to `-EigenValProc:4`

Normally StkJJ automatically choose the fastest numerical eigenvalue procedure for particular case. This happens when you do not use `-EigenValProc:` option or use `-EigenValProc:0`. Still, if you specify particular `-EigenValProc:` StkJJ checks whether requested procedure can be used for the particular case.

The options which allow to calculate eigenvalues and eigenfunctions are shown in Tab. 7.8.

Analytical results

The eigenfrequencies of a linear LJJ of finite L with the static state $\mu_0 = 0$ ($\gamma = 0$) are given by

$$\omega_n = \sqrt{1 + \left(\frac{\pi n}{L}\right)^2}, \quad n = 0, 1, 2, \ldots. \tag{7.19}$$

In an infinite LJJ the spectrum is continuous, but in finite length LJJ the wave vector k is quantized as $k_n = \pi n/L$, therefore the eigenfrequencies are quantized too.

The lowest eigenfrequency of a fractional vortex (κ-vortex) at $\gamma = 0$ in an infinite LJJ is given by

$$\omega_0(\kappa) = \sqrt{\frac{1}{2}\cos\frac{\kappa}{4}\left(\cos\frac{\kappa}{4} + \sqrt{4 - 3\cos^2\frac{\kappa}{4}}\right)}. \tag{7.20}$$

7.8 Plasmon Energy Bands in periodic structures

Table 7.9: Plasmon energy bands in periodic structures

Option	**Type**	**Default**	**Description**
`-OutBands:[+|-]`	bool	off	Output all eigenvalues at each point of the IVC into a file `lambda????.dat`
`-BandOmegaMin:`$\omega_{\min}$	float	0.0	The minimum ω of the scan interval
`-BandOmegaMax:`$\omega_{\max}$	float	2.0	The maximum ω of the scan interval
`-BandOmegaStep:`$\Delta\omega$	float	0.01	The step in ω within the scan interval
`-$FastMatExp`[+|−]	bool	0	Use explicit formula for matrix exponent
`-$BandVerbose`[+|−]	bool	0	Output progress info

The matrix A_n has the form

$$A_n = \begin{pmatrix} 0 & dx \\ y_n\, dx & 0 \end{pmatrix},$$

where $y_n = \mathrm{CPR}'\mu(x_n) - \omega^2$.

$$\exp(A) = \begin{pmatrix} \cosh(dx\sqrt{y}) & \frac{1}{\sqrt{y}}\sinh(dx\sqrt{y}) \\ \sqrt{y}\sinh(dx\sqrt{y}) & \cosh(dx\sqrt{y}) \end{pmatrix} = \begin{pmatrix} \cos(dx\sqrt{-y}) & \frac{1}{\sqrt{-y}}\sin(dx\sqrt{-y}) \\ -\sqrt{-y}\sin(dx\sqrt{-y}) & \cos(dx\sqrt{-y}) \end{pmatrix}. \tag{7.21}$$

The expression with hyperbolic functions is useful for $y \geq 0$ and the one with trigonometric functions for $y < 0$

7.9 Energy

Ones you have got a stationary solution at each point of the IVC you can save (potential, kinetic, Josephson, *etc.*) energy density in each LJJ. You can also ask StkJJ to create a file which contains the total energy of each LJJ at each point of the IVC.

At the moment the following energies and their densities can be calculated and output:

1. Josephson energy E_J and its density dE_J given by

$$E_J = \int_0^L dE_J(x)\,dx = \int_0^L w(x)\,\mathrm{IntCPR}(\mu+\theta)\,dx. \tag{7.22}$$

2. Inductive energy E_I and its density dE_I given by

$$E_I = \int_0^L dE_I(x)\,dx = \int_0^L w(x)\frac{1}{2}\mu_x^2\,dx. \tag{7.23}$$

3. Magnetic energy E_L and its density dE_L given by

$$E_L = \int_0^L dE_L(x)\,dx = \int_0^L w(x)\frac{[\mu_x - h(x)]^2}{2}\,dx. \tag{7.24}$$

4. Kinetic energy E_K and its density dE_K given by

$$E_K = \int_0^L dE_K(x)\,dx = \int_0^L w(x)\frac{1}{2}\mu_t^2\,dx. \tag{7.25}$$

5. Potential energy E_P and its density dE_P given by

$$E_P = E_I + E_J = \int_0^L [dE_I(x) + dE_J(x)]\,dx = \int_0^L dE_p(x)\,dx. \tag{7.26}$$

6. Dissipated instant power E_A and its density dE_A given by

$$E_A = \int_0^L dE_A(x)\,dx = \int_0^L w(x)\alpha(x)\frac{1}{2}\mu_t^2\,dx. \tag{7.27}$$

Table 7.10: Energy calculation options.

Option	**Type**	**Default**	**Description**
`-OutEnergy`$\langle$X$\rangle$`[+|-]`	bool	off	Output corresponding energy density denoted by $\langle X\rangle$ *vs.* x at each point of IVC into a file `Ep????.dat`. The $\langle X\rangle$ can be J,I,L,P,K,A, see Eqs. (7.22)–(7.27).
`-E`$\langle$x$\rangle$`File:`filename	string	""	Create a file filename with the total energy $E_{\langle x\rangle}$, see Eqs. (7.22)–(7.27), at each point of the IVC. $\langle$x$\rangle$ can be j,i,l,p,k,a. Works only if corresponding `-OutEnergy`$\langle$X$\rangle$`+`

The format of the file created by e.g. the `-OutEnergyP+` option is the following:

column 1	**column 2**	**column 3**	...	**column n**
x_0	$dE_p^1(x_0)$	$dE_p^2(x_0)$	...	$dE_p^n(x_0)$
x_1	$dE_p^1(x_1)$	$dE_p^2(x_1)$	...	$dE_p^n(x_1)$
...				
x_N	$dE_p^1(x_N)$	$dE_p^2(x_N)$	...	$dE_p^n(x_N)$

where $dE_p^n(x_i)$ is the energy density in n-th LJJ at the pont $x = x_i$.

The format of the file created by the `-EpFile:` option is the following:

column 1	**column 2**	**column 3**	**...**	**column n**
sweep1 value 0	E_p^1	E_p^2	...	E_p^n
sweep1 value 1	E_p^1	E_p^2	...	E_p^n
...				
sweep1 value N	E_p^1	E_p^2	...	E_p^n

where E_p^n is the energy of n-th LJJ.

Chapter 8

Tips

8.1 Accelerating your work

StkJJ has several features which allow to exit Sweep sequence by certain conditions, thus avoiding wasting time especially when StkJJ works in unattended mode. These features are summarized in Tab. 8.1.

The work of $I_c(H)$ algorithm at the last point can be considerably accelerated by the following trick. Note, that when the current exceeds the critical value, system switches to high voltage state (*e.g.* R-state) and StkJJ will follow this process of switching until the system relaxes and the voltage in a new state will be known with accuracy given by the switch "`-IV_V_Err:`". On the other hand, it is already clear much before, namely when the switching starts, that critical current reached and we can mark the next point on $I_c(H)$ dependence.

StkJJ contains two traps in the voltage averaging routine exactly for this case. The first trap just checks whether the mean voltage during the last averaging interval exceeds some value given by the switch "`-IcH_Vmax:`". If it is, then $I_c(H)$ is found and StkJJ proceeds to the next value of H.

The second trap is more sophisticated. It detects the switching dynamics, *i.e.*, strictly speaking, it detects the situation when the averaging routine starts converging to the voltage which is far away from the $[V_{\min}, V_{\max}]$ interval. The exact expression of the criterion is:

$$\begin{aligned}
&\text{we define } \delta V = \left|\bar{V}_n - \bar{V}_{n-1}\right|, \\
&\text{if } \left[\bar{V}_n - q\,\delta V, \bar{V}_n + q\,\delta V\right] \text{ doesn't intersect with } [V_{\min}, V_{\max}] \\
&\text{then Switching is happening!}
\end{aligned} \tag{8.1}$$

where q is the factor which defines the strictness of this trapping algorithm and can be set by "`-IcH_SwDet:`" switch. If q is large, the algorithm is more precise and saves less CPU time. The lower is q, the faster it detects the switch, but less accurately, so that it can also detect a switch when it does not take place. To disable this algorithm use $q < 0$.

The switch detection algorithm is automatically disabled for the first point of the IVC, *i.e.*, when the sweeping has just started. This is done because increasing H at $I = 0$ may result in internal change of configuration which may be detected as a switch. Since this is not desirable, switch detection logic is off for the 1st point of the IVC.

Both "`-IcH_SwDet:`" and "`-IcH_Vmax:`" respect "`-IcH_VMode:`" switch and work only with proper voltages.

Table 8.1: Conditional Sweep completion

Option	**Type**	**Default**	**Description**
`-V_max:`$V_{\max}$	float	1e9	If both voltages V_1 and V_2 exceed $V_{\max}$, current IVC stops and StkJJ exits.
`-dV_Stop:`δV_{stop}	float	1e9	If one of the voltage differences $\lvert V_{i+1} - V_i\rvert$ exceeds δV_{stop}, current IVC stops and StkJJ exits.
`-IcH_SwDet:`q	float	5.0	Parameter q which defines the sensitivity of the switch detection algorithm (8.1). If $q < 0$, the algorithm is off.
`-IcH_Vmax:`$V_{\max}$	float	10.0	$V_{\max}$ defines when to trap the escaping sequence of the average voltages converging far away from the region of interest (see text).

8.2 OS specific performance optimization

StkJJ includes several OS specific options which may help you to run StkJJ smoothly, see Tab. 8.2.

Table 8.2: OS specific performance optimization options

Option	Type	Default	Description
`-Idle[+|-]`	bool	off	make StkJJ run with idle priority *i.e.*, only when OS has free time. This option works only under Windows 9x/NT. Under UNIX using `nohup`, makes StkJJ rather nice.
`-NoFlush[+|-]`	bool	off	StkJJ will not flush stdout buffers after each screen update. If you are running in the background e.g. by using `nohup`, the stdout is redirected to a file `nohup.out`. Frequent flushing will degrade the performance, especially if `nohup.out` is on a network drive.
`-KeepFilesOpen[+|-]`	bool	off	Normally StkJJ opens output file(s) after each calculated point, writes to them (usually only one line), and closes them. This may hinder performance. `-KeepFilesOpen+` will keep the output file(s) open during the whole run time, thus avoiding many `open()` and `close()` calls and physically writing data in big 4kB chunks. Attention, at the moment, if you interrupt StkJJ, e.g. using Ctrl+C, the data, which were not written to disk, are lost.

8.3 Debuging

StkJJ includes several switches which allow to receive additional information while the program is running. Such information is very useful to find out the origin of some problems. The table 8.3 contains the list of related options and explanations.

Table 8.3: Debugging options.

Option	**Type**	**Default**	**Description**
`-$IV_AvLog:`*level*	int	0	If *level* $\neq 0$, switches on the logging in IV averaging routine. This creates 2 files `aver_old.log` and `aver_new.log` with V_i and δV_i corresponding to the old and new averaging algorithms.
`-$Log:`[+|−]	bool	off	Switches logging into `stkjj.log` on or off
`-$IVC/phi_x`	bool	off	Allows to see phase gradients after each averaging step in file `phi_x(t).dat`
`-$SaveK1` `...` `-$SaveK5`	bool	off	Allows to save the coresponding coefficient $k_n(x)$ of numerical scheme into the file `k`*n*`(x).dat`
`-$SaveAllK`	bool	off	Allows to save all the coefficients $k_1(x), \ldots, k_5(x)$ of numerical scheme into the files `k1(x).dat`...`k5(x).dat`. This option is equivalent to turning on all `-$SaveK`*n* options. If you specify "`-$SaveAllK+ -$SaveK2-`" all coeeficients except $k_2(x)$ except will be saved.
`-$SaveJ(x)`	bool	off	Allows to save the bias current distribution $\gamma(x)$ into file "`j(x).dat`"
`-$SaveJc(x)`	bool	off	Allows to save the critical current distribution $\tilde{j}(x)$ into file "`j_c(x).dat`"
`-$SaveRn(x)`	bool	off	Allows to save the damping distribution $\tilde{\alpha}(x)$ into file "`R_n(x).dat`"
`-$RandTests:`*n*	int	0	Allows to test Gaussian random number generator. Calculates and shows central moments $M_1 \ldots M_4$ and semi-invariants $K1 \ldots K_4$. For Gaussian distribution $K_1 = M_1 = \langle x \rangle = 0$,$K_2 = M_2 = \langle x^2 \rangle = \sigma^2$, $K_3 = K_4 = 0$
`-$RandSeqLen:`*N*	int	100000	The length N (number of samples) of the random sequence to test
`-$RandSeqSigma:`σ	float	2.0	The dispertion of the test random sequence.
`-$IcH_Debug`[+|−]	bool	0	Allows to show diagnostic messages of $I_c(X)$ routine.
`-$SavePot`[+|−]	bool	false	Allows to save potential profile $h(x)$ into file `__pot.dat`
`-$SavePotForce`[+|−]	bool	false	Allows to save potential force profile $h_x(x)$ into file `__pot_force.dat`
`-$SaveTheta`[+|−]	bool	0	Allows to save $\theta(x)$ profile into file `__theta(x).dat`

8.4 Output file formats

The output formats of files generated by StkJJ when the appropriate options are invoked are specified in Tab. 8.4. Additionally StkJJ saves a copy of the command line in the file `cmdline` and a log file in the file `stkjj.log` on each run of the program.

In the Tab. 8.4 we use the following notations:

$$\bar{V}^{A,B} = \left\langle \phi_{\tilde{x}}^{A,B}(\tilde{x},\tilde{t}) \right\rangle_{\tilde{x},\tilde{t}}, \tag{8.2}$$

Table 8.4: Format of output files.

file		column				
file name	kind of data	1	2	3	4	5
Main Result File	$\bar{V}(I)$	index	*sweepedparam.*	$\bar{V}^A$	$\bar{V}^B$	$T_{\text{av.done.}}$
Main Result File	$I_{\max}(H)$	H	$I_{\max}$	$< \phi_{\tilde{x}}^A(\tilde{x},\tilde{t}) >_{\tilde{x},\tilde{t}}$	$< \phi_{\tilde{x}}^B(\tilde{x},\tilde{t}) >_{\tilde{x},\tilde{t}}$	$< \tilde{t} >$
`stat####.dat`	$\phi_{n,g}^{0\ldots N}\,state$	ϕ_n^A	ϕ_g^A	ϕ_n^B	ϕ_g^B	
`p####.dat`	$\phi^{A,B}(\tilde{x},\tilde{t}_0)$	$\tilde{x}$	$\phi^A(\tilde{x},\tilde{t}_0)$	$\phi^B(\tilde{x},\tilde{t}_0)$		
`h####.dat`	$H(x)$	$\tilde{x}$	$\phi_{\tilde{x}}^A(\tilde{x},\tilde{t}_0)$	$\phi_{\tilde{x}}^A(\tilde{x},\tilde{t}_0)$		
`v####.dat`	$V(x)$	$\tilde{x}$	$\phi_{\tilde{t}}^A(\tilde{x},\tilde{t}_0)$	$\phi_{\tilde{t}}^B(\tilde{x},\tilde{t}_0)$		
`VT####.dat`	$V(t)$	$\tilde{t}$	$\phi_{\tilde{t}}^A(\tilde{x}_0,\tilde{t})$	$\phi_{\tilde{t}}^B(\tilde{x}_0,\tilde{t})$	$\phi_{\tilde{t}}^A + \phi_{\tilde{t}}^B$	
`fft####.dat`	$V(f)$	f	$V^A(f)$	$V^B(f)$	$V^{A+B}(f)$	
`fftxx###.dat`	sparse $V(f)$	f	$V^A(f)$	$V^B(f)$	$V^{A+B}(f)$	
`maxx###.dat`	maxs$(V(f))$	f	$V^A(f)$	$V^B(f)$	$V^{A+B}(f)$	
`KinkTraj.dat`	pos. of fl. & a-fl.	$x_{\text{fl}}^A(t)$	$x_{\text{af}}^A(t)$	$x_{\text{fl}}^B(t)$	$x_{\text{af}}^B(t)$	
`stkjj.log`	log file	log of all actions and files				
`cmdline`	command line	echo of arguments on command line on execution				

Bibliography

[1] E. Goldobin, A. Wallraff, *"StkJJ – User's Reference"*, [Online] Available http://www.geocities.com/SiliconValley/Heights/7318/StkJJ.htm, July 4, 1997. 1

[2] StkJJ support: stkjj@mail.ru; Edward Goldobin: gold@uni-tuebingen.de 1

[3] S. Sakai, P. Bodin, N. F. Pedersen, *"Fluxons in thin-film superconductor-insulator superlattices"*, J. Appl. Phys. **73** (5), 2411–2418 (1993). 1

[4] D. Yu. Vodolazov and I. L. Maksimov, Physica C **349** 125–138 (2001); see also cond-mat/0001035. 11, 12

[5] E. Goldobin, A. Sterck, and D. Koelle, *"Josephson vortex in a ratchet potential: Theory"*, Phys. Rev. E **63**, 031111 (2001). 15

[6] A. Wallraff, Diploma thesis (1997), [Online] Available http://www.physik.uni-erlangen.de/PI3/Ustinov/staff/a_wallraff/Academic.html, December 19, 1997. 35

[7] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, "Numerical Recipes in C++", Cambrige University Press, 2nd ed., 2002 38

[8] Avalable online at `http://www.nr.com`