The JETSPIN User Manual

M. Lauricella*, G. Pontrelli*, I. Coluzza#, D. Pisignano°, S. Succi*

*Istituto per le Applicazioni del Calcolo CNR,
Via dei Taurini 19, 00185 Rome, Italy

#Faculty of Physics, University of Vienna,
Boltzmanngasse 5, 1090 Vienna, Austria

°Dipartimento di Matematica e Fisica "Ennio De Giorgi", University of Salento,
via Arnesano, 73100 Lecce, Italy

version 1.21 - July, 2016

Disclaimer

None of the authors, nor any contributor to the JETSPIN code or its derivatives guarantee that the software and associated documentation is free from error. Neither do they accept responsibility for any loss or damage that results from its use. The responsibility for ensuring that the software is fit for purpose lies entirely with the user.

JETSPIN Acknowledgements

The software development process has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013)/ERC Grant Agreement n. 306357 "NANO-JETS".

About JETSPIN

If results obtained with this code are published, an appropriate citation would be:

Marco Lauricella, Giuseppe Pontrelli, Ivan Coluzza, Dario Pisignano, Sauro Succi, JETSPIN: A specific-purpose open-source software for electrospinning simulations of nanofibers, *Computer Physics Communications*, 197 (2015), 227-238.

Contents

1 Introduction **5**
- 1.1 The JETSPIN Package . . . 5
- 1.2 Structure . . . 6
- 1.3 Parallelization . . . 8
- 1.4 Compiling the Source Code . . . 8
- 1.5 Running JETSPIN . . . 9
- 1.6 Restarting JETSPIN . . . 9

2 JETSPIN Model and Algorithms **11**
- 2.1 Equations of Motion . . . 11
- 2.2 Perturbations at the nozzle . . . 14
- 2.3 Jet insertion . . . 14
- 2.4 Dynamic refinement . . . 14
- 2.5 Dimensionless quantities . . . 15
- 2.6 Integration schemes . . . 17
- 2.7 Multiple Timestep Algorithm for Coulomb forces . . . 18

3 JETSPIN Data Files **19**
- 3.1 Input file . . . 19
- 3.2 Output files . . . 21

4 Example Simulations **23**
- 4.1 Test Case 1: 1-D case . . . 23
- 4.2 Test Case 2: 1-D case with bead insertion at the nozzle activated . . . 23
- 4.3 Test Case 3: Three-D simulation of electrospinning process . . . 23
- 4.4 Test Case 4: Three-D simulation in presence of a gas counterflow . . . 24
- 4.5 Test Case 5: Three-D simulation with dynamic refinement . . . 24
- 4.6 Test Case 6: Three-D simulation of a quasi-Newtonian fluid . . . 24

5 Tips **25**
- 5.1 How to visualize a PDB file in VMD software package . . . 25

Chapter 1

Introduction

Scope of Chapter

This chapter describes the design and directory structure of JETSPIN.

1.1 The JETSPIN Package

In the recent years, electrospun nanofibers have gained a considerable industrial interest due to many possible applications, such as tissue engineering, air and water filtration, drug delivery and regenerative medicine. In particular, the high surface-area ratio of the fibers offers an intriguing prospect for technological applications. As consequence, several studies were focused on the characterization and production of uni-dimensionally elongated nanostructures. A number of reviews [1, 2, 3, 4, 5] and books [6, 7, 8] concerning electrospinning have been published in the last decade.

Typically, electrospun nanofibers are produced at laboratory scale by the uniaxial stretching of a jet, which is ejected at the nozzle from a charged polymer solution. The initial elongation of a jet can be produced by applying an externally electrostatic field between the spinneret and a conductive collector. Electrospinning involves mainly two sequential stages in the uniaxial elongation of the extruded polymer jet: an initial quasi steady stage, in which the electric field stretches the jet in a straight path away from the nozzle of the ejecting apparatus, and a second stage characterized by a bending instability induced from small perturbations, which misalign the jet from its axis of elongation [9]. These small disturbances may originate from mechanical vibrations at the nozzle or from hydrodynamic-aerodynamic perturbations within the experimental apparatus. Such a misalignment provides an electrostatic-driven bending instability before the jet reaches the conductive collector, where the fibers are finally deposited. As a consequence, the jet path length between the nozzle and the collector increases, and the stream cross-section undergoes a further decrease. The ultimate goal of electrospinning process is to minimize the radius of the collected fibers. By a simple argument of mass conservation, this is tantamount to maximizing the jet length by the time it reaches the collecting plane. By the same argument, it is therefore of interest to minimize the length of the initial stable jet region. Consequently, the bending instability is a desirable effect, as it produces a higher surface-area-to-volume ratio of the jet, which is transferred to the resulting nanofibers [10].

Computational models provide a useful tool to elucidate the physical phenomenon and provide information which might be used for the design of electrospinning experiments. Numerical simulations can be used to improve the capability of predicting the key processing parameters and exert a better control on the resulting nanofiber structure. In recent years, with a renewed interest in nanotechnology, electrospinning studies attracted the attention of many resarchers both from modeling, computational and experimental point of view [11, 12, 13, 14, 15, 9]. Although some author uses dissipative particle dynamics (DPD) mesoscale simulation method into electrospinning study [16], most models usually treat the jet filament as a series of discrete elements (*beads*) obeying the equations of continuum mechanics [11, 12]. Each bead is subject to different types of interactions, such as long-range Coulomb repulsion, viscoelastic drag, the external electric field. The main aim of such models is to explain the complexity of the resulting dynamics and provide the

set of parameters driving the optimal process. The effect of fast-oscillating loads on the bending instability have been explored in an extensive modeling and computational study [17].

JETSPIN delivers a FORTRAN code especially designed to simulate the electrospinning process in a variety of different conditions and experimental settings. This comprehensive platform is devised such that different cases and input variables can be described and simulated. The framework is developed to exploit several computational architectures, both serial and parallel.

JETSPIN, as open-source software, can be used to carry out a systematic sensitivity analysis over a broad range of parameter values. The results of simulations provide valuable insight on the physics of the process and can be used to assess experimental procedures for an optimal design of the equipment and to control processing strategies of technological advanced nanofibers.

The JETSPIN code is freely downloadable at the web site:

http://www.nanojets.eu

1.2 Structure

JETSPIN is written in free format FORTRAN90, and it consists of approximately 160 subroutines. The source exploits the modular approach provided by the programming language. All the variables having in common description of certain features or method are grouped in modules. The convention of explicit type declaration is adopted, and all the arguments passed in calling sequences of functions or subroutines have defined intent. We use the PRIVATE and PUBLIC accessibility attributes in order to decrease error-proneness in programming.

The main routines have been gathered in the `main.f90` file, which drives all the CPU-intensive computations needed for the capabilities mentioned below. The variables describing the main features of nanofibers (position, velocity, etc.) are declared in the `nanojet_mod.f90` file, which also contains the main subroutines for the memory management of the fundamental data of the simulated system. Since the size system is strictly time-dependent, JETSPIN exploits the dynamic array allocation features of FORTRAN90 to assign the necessary array dimensions. In particular, the size system is modified by the routines `add_jetbead` and `erase_jetbead`, while the decision of the main array size, declared as `mxnpjet`, is handled by the routine `reallocate_jet`. The sizes of various service bookkeeping arrays are handled within a parallel implementation strategy, witch exploits few dedicated subroutines (see Sec 1.3). All the implemented time integrators are written in the `integrator_mod.f90` file, which contains the routine `driver_integrator` to select the proper integrator, as indicated in the input file. All the terms of equations of motion for the implemented model (see Sec 2.1) are computed by routines located in the `eom_mod.f90` file, which call other subroutines in the files `coulomb_force_mod.f90`, `viscoelastic_force_mod.f90` and `support_functions_mod.f90`. A summarizing scheme of the main JETSPIN program in the `main.f90` file has been sketched in Fig 1.1.

The user can carry out simulations of nanofibers without a detailed understanding of the structure of JETSPIN code. All the parameters governing the system can be defined in the input file (see Sec 3.1), which is read by routines located in the `io_mod.f90` and `parse_mod.f90` files. Instead, the user should be acquainted with the model described in Cap 2. The content of the output file is completely customizable by the input file as described in Sec 3.1, and it can report different time-averaged observables computed by routines of the module `statistic_mod` (see Sec 3.2). The routines in the `error_mod.f90` file can display various warning or error banners on computer terminal, so that the user can easily correct the most common mistakes in the input file.

JETSPIN is supplied as a single UNIX compressed (tarred and gzipped) directory with four sub-directories. All the source code files are contained in the *source* sub-directory. The *examples* sub-directory contains different test cases that can help the user to edit new input files. The *build* sub-directory stores a UNIX `makefile` that assembles the executable versions of the code both in serial and parallel version with different compilers, `which` should be copied (and eventually modified) into the *source* sub-directory in order to compile JETSPIN (see Section 1.4). Finally, after JETSPIN was compiled, the *execute* sub-directory contains the binary executable file, where it can be run.

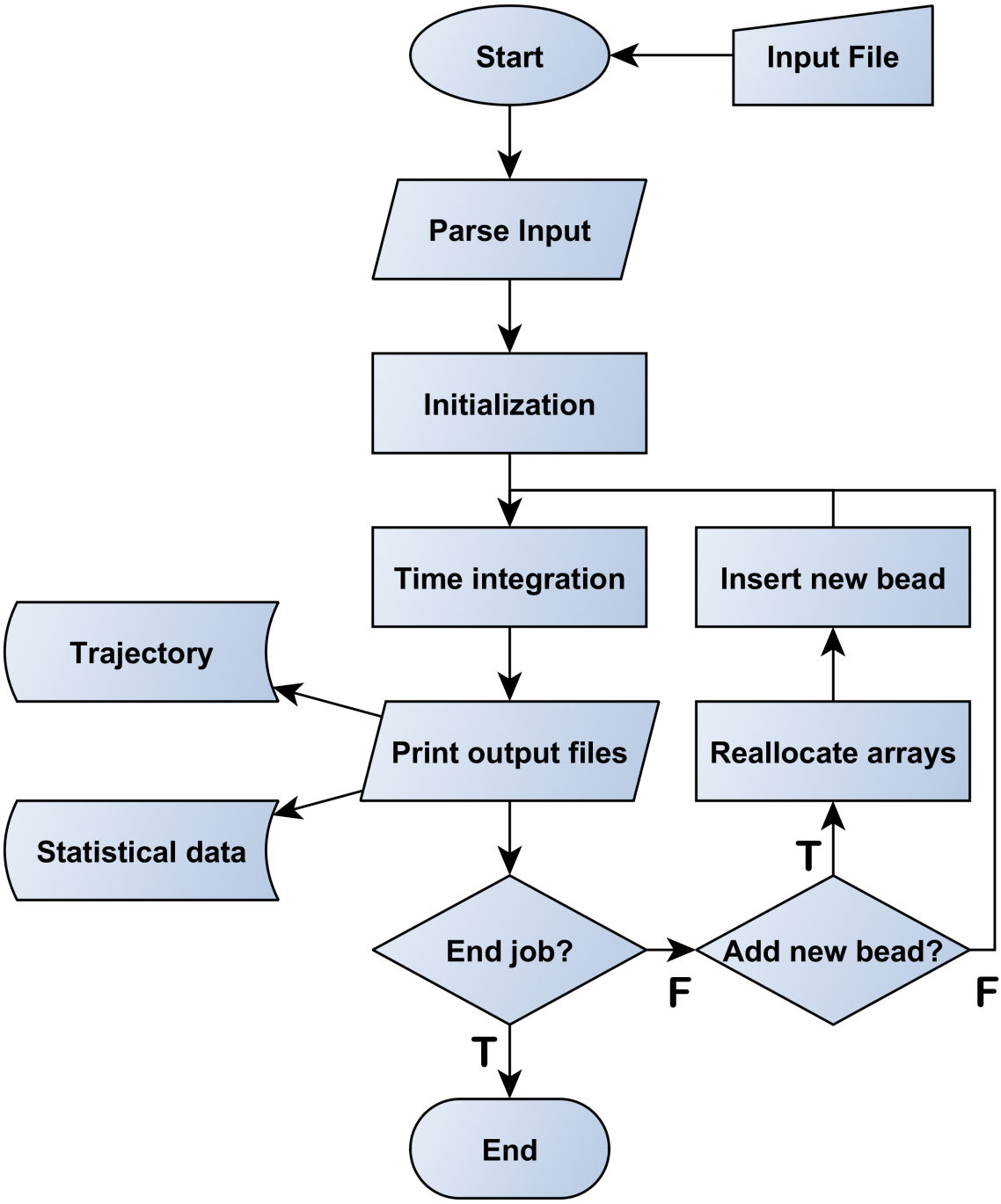

Figure 1.1: Structure of the main JETSPIN program.

1.3 Parallelization

The parallel infrastructure of JETSPIN incorporates the necessary data distribution and communication structures. The parallel strategy underlying JETSPIN is the Replicated Data (RD) scheme[18], where fundamental data of the simulated system are reproduced on all processing nodes. In simulations of nanofibers, the fundamental data consist of position, velocity, and viscoelastic stress arrays of each bead in which the nanojet is discretised (see below Cap 2). Further data defining mass and charge of each bead are also replicated. However, all auxiliary data are distributed in equal portion of data (as much as possible) for each processor. Despite being available other parallel strategies such as the Domain Decomposition[19], our experience has shown that such volume of data is by no means prohibitive on current parallel computers.

By the RD scheme, we adopt in JETSPIN a simple communication strategy of the communication between nodes, which is handled by global summation routines. The module `version_mod` located in the `parallel_version_mod.f90` file contains all the global communication routines which exploit the MPI (Message Passing Interface) library. Note that a FORTRAN90 compiler and an MPI implementation for the specific machine architecture are required in order to compile JETSPIN in parallel mode. An alternative version of the module `version_mod` is located in the `serial_version_mod.f90` file, and it can be easily selected by appropriate flags in the `makefile` at compile time. By selecting this version, JETSPIN can also be run on serial computers without modification, even though the code has been designed to run on parallel computers.

The size system is strictly time-dependent as mentioned in Sec 1.2 and, therefore, the memory of various service bookkeeping arrays is dynamically distributed over all the processing nodes. In particular, the bookkeeping array size, declared as `mxchunk`, is managed by the routine `set_mxchunk`. All the beads of the discretised nanofiber are assigned at every time step to a specific node by the routine `set_chunk`, and their temporary data are stored in bookkeeping arrays belonging to the assigned node. It is worth stressing that the communication latency makes the parallelization efficiency strictly dependent on the system size. Therefore, we only advice the use of JETSPIN in parallel mode whenever the user expects a system size with at least 50 beads for each node (further details in Ref [20]).

1.4 Compiling the Source Code

The *build* sub-directory stores a UNIX **makefile** that assembles the executable versions of the code both in serial and parallel version with different compilers. Note that JETSPIN may be compiled on any UNIX platform. The **makefile** should be copied (and eventually modified for several common workstations and parallel computers) into the *source* sub-directory, where the code is compiled and linked. Finally, the binary executable file can be run in the *execute* sub-directory, which isintended to be the 'working' directory from which jobs are submitted for execution and the data files manipulated. A list of targets for several common workstations and parallel computers can be used by the command

```
make target
```

where `target` is one of the options reported in Tab 1.1. On Windows system we advice the user to compile JETSPIN under the command-line interface Cygwin [21]. Finally, the binary executable file can be run in the *execute* sub-directory. Note that a FORTRAN 90 compiler is necessary in order to compile JETSPIN. Further, a Message Passing Interface Implementation should be also present, if the user wishes to compile JETSPIN in parallel mode. We recommend GFORTRAN as FORTRAN 90 compiler, but IFORT-Intel(R) Fortran Compiler or G95 can also be used. As Message Passing Interface Implementation, we suggest to use that provided by the Open MPI Project.

target:	meaning:
gfortran	compile in serial mode using the GFortran compiler.
gfortran-mpi	compile in parallel mode using the GFortran compiler and the Open Mpi library.
cygwin	compile in serial mode using the GFortran compiler under the command-line interface Cygwin for Windows.
cygwin-mpi	compile in parallel mode using the GFortran compiler and the Open Mpi library under the command-line interface Cygwin for Windows (note a precompiled package of the Open Mpi library is already available on Cygwin).
intel	compile in serial mode using the Intel compiler.
intel-mpi	compile in parallel mode using the Intel compiler and the Intel Mpi library.
intel-openmpi	compile in parallel mode using the Intel compiler and the Open Mpi library.
help	return the list of possible target choices

Table 1.1: List of targets for several common workstations and parallel computers, which can be used by the command "`make target`".

1.5 Running JETSPIN

In order to run the JETSPIN executable (main.x) the user should prepare the input file following the hints provided in Section 3.1. The test cases located in *examples* sub-directory can be used as starting point for editing the input file. As JETSPIN is executed, the program will look for the input file, named `input.dat`, in the same directory, where the executable was called. For instance, if JETSPIN is called in the *execute* sub-directory, the code will look for the file `input.dat`, which must have been put in the same sub-directory.

The JETSPIN code can be executed in serial mode by the command

```
./main.x
```

or in parallel mode by the command

```
mpirun -np $ncpu ./main.x
```

where `$ncpu` is the number of CPUs used in parallel (e.g. `mpirun -np 4 ./main.x` in order to run JETSPIN in parallel mode on 4 CPUs).

A successful run of JETSPIN will generate several output files, which appear in the directory, where the executable was called. The output files will be produced only if their writing was explicitly requested in input file by using proper directives (see Section 3.1). A detailed description of the generated output files is provided in Section 3.2. Briefly, the main output files are the files `statdat.dat`, `traj.xyz` and `frame%06d.xyz`, which are human readable. The contains a catalogue of instantaneous or blocked averaged values of several variables, which can be used for temporal or statistical analysis in post-processing. The `traj.xyz` file provides a time ordered sequence of configurations to facilitate further analysis of the jet dynamics, while the `frame%06d.xyz` file, reports the same information in splitted files, one for each registered time frame. Further, as JETSPIN is running, a series of lines will be printed in the terminal window, providing several information as requested by suitable directives in input file (see Section 3.1). Also present will be the `save.dat` file, which contains the accumulated data for computing statistical quantities and other quantities defining the current simulation. This file is intended to be used together with the input file for restarting a JETSPIN job (see Section 1.6), and it is *not* human readable.

1.6 Restarting JETSPIN

A simulation can be restarted after normal termination or error termination, if the last was due to inadequate time allocation. In order to restart a JETSPIN job the file `input.dat` is again required together with the file `restart.dat`, which is the exact copy of the `save.dat` file created by the previous job. In particular, the `save.dat` file was written by JETSPIN at regular time interval during the previous simulation. The

file `save.dat` is *not* human readable, and it should be renamed as `restart.dat`, and located in the same directory of input file. If the user attempts to restart JETSPIN without the `restart.dat` file, the job will immediately exit with error. Note that the directive "*restart yes*" needs to be added in input file in order to restart a previous JETSPIN job (see Section 3.1). Further, whenever a job is restarted, it is also possible to reset the accumulated statistical data, collected in the previous run, by adding the directive "*restart reset*" in input file. It is worth stressing that JETSPIN will append new data to the files `statdat.dat`, `traj.xyz` , while all the other output files will be **overwritten** (included the `save.dat` file). Thus, the user is advised to keep backup copies of the `restart.dat` file, noting the times it was written, to help you avoid going right back to the start of a simulation.

The same restarting procedure can be also used for extending a simulation beyond its initial allocation of timesteps. In this case, the directive *timesteps* should be adjusted in input file to the new total number of steps required for the simulation. For instance, if a 10000 step simulation needs to be extended by a further 5000 steps, the user should use the directive "*timesteps 15000*" in the input file and include the "*restart yes*" directive. Further, the restarting procedure can be useful in the event of machine failure. Here, the job can be restarted in the same way from the `save.dat` file, which is written at regular time intervals in order to meet just such an emergency. We stress that the procedure is not foolproof in the last event, since the job could be crash while the `save.dat` file is being written.

Chapter 2

JETSPIN Model and Algorithms

Scope of Chapter

This chapter describes the model and simulation algorithms incorporated into JETSPIN.

2.1 Equations of Motion

The model implemented in JETSPIN is an extension of the Lagrangian discrete model introduced by Reneker et al. [11] (in the following text referred as Ren model). The model provides a compromise of efficiency and accuracy by representing the filament as a series of n beads (jet beads) at mutual distance l connected by viscoelastic elements. The length l is typically larger than the cross-sectional radius of the filament, but smaller than the characteristic lengths of other observables of interest (e.g. curvature radius). Each $i-th$ bead has mass m_i and charge q_i (not necessarily equal for all the beads). The nanofiber is modeled as a viscoelastic fluid following the approach developed in Ref. [22], and the stress σ_i for the element connecting bead i with bead $i+1$ is given by the viscoelastic constitutive equation:

$$\frac{d\sigma_i}{dt} = -\frac{1}{t_0}\left(\sigma_i - \sigma_{HB}\right), \tag{2.1}$$

where t_0 is the relaxation time and σ_{HB} is the Herschel–Bulkley stress. Note that the relaxation time is give as

$$t_0 = \frac{\mu}{G}, \tag{2.2}$$

where G is the elastic modulus and μ the viscosity of the fluid jet, while σ_{HB} reads

$$\sigma_{HB} = \sigma_Y + K_{ve}\left(\frac{1}{l_i}\frac{dl_i}{dt}\right)^{n_{ve}}. \tag{2.3}$$

In the previous expression, K_{ve} is the flow consistency index with dimensions $\mathrm{gs}^{ne-2}\mathrm{cm}^{-1}$, σ_Y the yield stress, n_{ve} the flow behavior index, t the time, and l_i the length of the element connecting bead i with bead $i+1$. Note that it is possible to define an effective viscosity μ_{eff} as

$$\mu_{eff} = K_{ve}\left(\frac{1}{l_i}\frac{dl_i}{dt}\right)^{n_{ve}-1}. \tag{2.4}$$

It is worth stressing that the case $K_{ve} = \mu$, $n_{ve} = 1$ and $\sigma_Y = 0$ recovers the Maxwellian fluid model. Keeping the last case in mind, for convenience sake, we write

$$K_{ve} = k_{ve}\mu, \tag{2.5}$$

with k_{ve} the flow consistency index re-scaled by μ, and the viscoelastic constitutive equation is rearranged as

$$\frac{d\sigma_i}{dt} = -\frac{G}{\mu}\left[\sigma_i - \sigma_Y + k_{ve}\mu\left(\frac{1}{l_i}\frac{dl_i}{dt}\right)^{n_{ve}}\right], \tag{2.6}$$

The last constitutive equation allows the modeling of several fluids as, for example, Bingham fluids ($\sigma_Y > 0$), shear-thinning fluids ($n_{ve} > 1$), shear-thickening fluid ($n_{ve} < 1$) .

As alternative, it is possible to activate the Kelvin–Voigt model [23] (see Tab 3.1). In the last case, Eq 2.6 is replaced by

$$\frac{d\sigma_i}{dt} = G\frac{1}{l_i}\frac{dl_i}{dt} + \mu\frac{1}{l_i}\frac{dv_i}{dt}. \tag{2.7}$$

Given a_i the cross-sectional radius of the filament at the bead i, the viscoelastic force $\vec{\mathbf{f}}_{ve}$ pulling bead i back to $i-1$ and towards $i+1$ is

$$\vec{\mathbf{f}}_{ve,i} = -\pi a_i^2 \sigma_i \cdot \vec{\mathbf{t}}_i + \pi a_{i+1}^2 \sigma_{i+1} \cdot \vec{\mathbf{t}}_{i+1}, \tag{2.8}$$

where $\vec{\mathbf{t}}_i$ is the unit vector pointing bead i from bead $i-1$. The surface tension force $\vec{\mathbf{f}}_{st}$ for the $i-th$ bead is given by

$$\vec{\mathbf{f}}_{st,i} = k_i \cdot \pi \left(\frac{a_i + a_{i-1}}{2}\right)^2 \alpha \cdot \vec{\mathbf{c}}_i, \tag{2.9}$$

where α is the surface tension coefficient, k_i is the local curvature, and $\vec{\mathbf{c}}_i$ is the unit vector pointing the center of the local curvature from bead i. Note the force $\vec{\mathbf{f}}_{st}$ is acting to restore the rectilinear shape of the bending part of the jet.

In the electrospinning experimental configuration an intense external electric potential Φ_0 is applied between the spinneret and a conducting collector located at distance h from the injection point. As consequence, each $i-th$ bead endures the electric force

$$\vec{\mathbf{f}}_{el,i} = e_i \frac{\Phi_0}{h} \cdot \vec{\mathbf{x}}, \tag{2.10}$$

where $\vec{\mathbf{x}}$ is the unit vector pointing the collector from the spinneret and assumed along the x axis. Note that in Ren model the electric field Φ_0 is assumed to be static in order to avoid the computationally expensive integration of Poisson equation, whereas in reality Φ is depending on the net charge of the nanofiber so as to maintain constant the potential at the electrodes. The latter issue was elegantly addressed by Kowalewski et al. [24], and its implementation in JETSPIN will be planned. Further, it is possible to impose a time dependent external electric potential described by a rectangular waveform with amplitude Φ_0 and frequency given in input file (see Section 3.1).

The net Coulomb force $\vec{\mathbf{f}}_c$ acting on the $i-th$ bead from all the other beads is given by

$$\vec{\mathbf{f}}_{c,i} = \sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{q_i q_j}{R_{ij}^2 + a_j^2} \cdot \vec{\mathbf{u}}_{ij}, \tag{2.11}$$

where $R_{ij} = \left[(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2\right]^{1/2}$, and $\vec{\mathbf{u}}_{ij}$ is the unit vector pointing the $i-th$ bead from $j-th$ bead. Note that $j-th$ charge is assumed to induces a field from its outer shell of the fiber (a charged ring of radius a_j) and not from the center line (the particle-like point) as implemented in the original Ren model. Although the Ren model provides a reasonable description for the spiral motion of the jet, the last term $\vec{\mathbf{f}}_c$ introduces mathematical inconsistencies due to the discretization of the fiber into point-charges. Different approaches were developed to overcome this issue which usually imply strong approximations[25, 10, 26]. Other strategies use a less crude approximation by accounting for the actual electrostatic form factors between two interacting sections of a charged fiber[27] or involving more sophisticated methods which exploit the tree-code hierarchical force calculation algorithm[24]. In JETSPIN it is possible to activate the dynamic refinement algorithm to recover a finer discretization of the jet (see Sec 2.4).

Although usually much smaller than the other driving forces, the body force due to the gravity is computed in the model by the usual expression

$$\vec{\mathbf{f}}_{g,i} = m_i g \cdot \vec{\mathbf{x}}, \tag{2.12}$$

where g is the gravitational acceleration.

As shown in experimental evidences [28], the air drag force affects the dynamics of the nanofiber. The extended stochastic model developed in Refs [29, 30, 31] was included in JETSPIN to reproduce the aerodynamic effects. In particular, the air drag is modeled by adding two force terms: a random term and a dissipative term. The dissipative air drag term is usually dependent on the geometry of the jet, which changes in time, and it combines longitudinal and lateral components. Based on experimental results[28], the longitudinal component of the air drag dissipative force term acting on bead i is

$$\vec{\mathbf{f}}_{air,i} = -0.65\pi a_i l_i \rho_a \left(2a_i/\nu_a\right)^{-0.81} \upsilon_{tan,i}^{1+0.19} \cdot \vec{\mathbf{t}}_{i-1} \tag{2.13}$$

where ρ_a and ν_a are the air density and kinematic viscosity, respectively, $\vec{\mathbf{t}}_{i-1}$ is the unit vector pointing bead i from bead $i-1$, and $\upsilon_{tan,i} = \vec{\boldsymbol{\upsilon}}_i \cdot \vec{\mathbf{t}}_{i-1}$ is the projection of the vector velocity of bead i along the unit vector $\vec{\mathbf{t}}_{i-1}$. Further, it is possible to write $\vec{\mathbf{f}}_{air,i}$ as

$$\vec{\mathbf{f}}_{air,i} = -m_i \gamma_i \, l_i^{0.905} \upsilon_{t,i}^{1+0.19} \cdot \vec{\mathbf{t}}_{i-1} \tag{2.14}$$

if we introduced the dissipative coefficient γ_i equal to

$$\gamma_i = 0.65\pi^{0.905} \frac{\rho_a}{m_i} \left(\frac{2}{\nu_{air}}\right)^{-0.81} V_i^{0.095}, \tag{2.15}$$

where $V_i = \pi a_i^2 \, l_i$ is the jet volume represented by the $i-th$ bead.

Following the expression introduced by A. Yarin[32, 6], under a high-speed air drag the lateral component $\vec{\mathbf{f}}_{lift,i}$ of the aerodynamic dissipative force related to the flow speed is given in the linear approximation (for small bending perturbations) by

$$\vec{\mathbf{f}}_{lift,i} = -l_i \cdot k_i \rho_a \upsilon_i^2 \pi \left(\frac{a_i + a_{i-1}}{2}\right)^2 \cdot \vec{\mathbf{c}}_i, \tag{2.16}$$

where k_i is still the local curvature, and $\vec{\mathbf{c}}_i$ is the unit vector pointing the center of the local curvature from bead i. The combined action of such longitudinal and lateral components provide the dissipative force term acting on the $i-th$ bead

$$\vec{\mathbf{f}}_{diss,i} = \vec{\mathbf{f}}_{air,i} + \vec{\mathbf{f}}_{lift,i} \, . \tag{2.17}$$

Instead, the random force term for the $i-th$ bead has the form

$$\vec{\mathbf{f}}_{rand,i} = \sqrt{2m_i^2 D_\upsilon} \cdot \vec{\boldsymbol{\eta}}_i(t), \tag{2.18}$$

where D_υ denotes a generic diffusion coefficient in velocity space (which is assumed constant and equal for all the beads), and $\vec{\boldsymbol{\eta}}_i$ is a three-dimensional vector, whereof each component η is an independent stochastic process that is nowhere differentiable with $< \eta\left(t_1\right) \eta\left(t_2\right) >= \delta\left(|t_2 - t_1|\right)$, and $< \eta\left(t\right) >= 0$. Note that for the sake of simplicity we assume $\eta = d\omega(t)/dt$, where $\omega(t)$ is a Wiener process. Further details are available in Ref [33].

The combined action of these forces governs the elongation of the jet according to the Newton's equation providing a non-linear Langevin-like stochastic differential equation:

$$m_i \frac{d\vec{\boldsymbol{\upsilon}}_i}{dt} = \vec{\mathbf{f}}_{el,i} + \vec{\mathbf{f}}_{c,i} + \vec{\mathbf{f}}_{ve,i} + \vec{\mathbf{f}}_{st,i} + \vec{\mathbf{f}}_{g,i} + \vec{\mathbf{f}}_{diss,i} + \vec{\mathbf{f}}_{rand} \, , \tag{2.19}$$

where $\vec{\boldsymbol{\upsilon}}_i$ is the velocity of the $i-th$ bead. The velocity $\vec{\boldsymbol{\upsilon}}_i$ satisfies the kinematic relation:

$$\frac{d\vec{\mathbf{r}}_i}{dt} = \vec{\boldsymbol{\upsilon}}_i \tag{2.20}$$

where $\vec{\mathbf{r}}_i$ is the position vector of the $i-th$ bead, $\vec{\mathbf{r}}_i = x_i\vec{\mathbf{x}} + y_i\vec{\mathbf{y}} + z_i\vec{\mathbf{z}}$. The three Eqs 2.6, 2.19 and 2.20 form the set of equations of motion (EOM) governing the time evolution of system. It is worth stressing that Eq 2.19 recovers a deterministic EOM by imposing $\rho_a = 0$ and $D_v = 0$.

2.2 Perturbations at the nozzle

The spinneret nozzle is represented by a single mass-less point of charge $\bar{q}$ fixed at $x = 0$ (nozzle bead). Its charge $\bar{q}$ is assumed equal to the mean charge value of the jet beads. Such charged point can be also interpreted as a small portion of jet which is fixed to the nozzle. In JETSPIN it is possible to add small perturbations to the y_n and z_n coordinates of the nozzle bead in order to model fast mechanical oscillations of the spinneret[17]. Given the initial position of the nozzle

$$y_n = A \cdot \cos(\varphi) \tag{2.21a}$$

$$z_n = A \cdot \sin(\varphi), \tag{2.21b}$$

the equations of motion for the nozzle bead are

$$\frac{dy_n}{dt} = -\omega \cdot z_n \tag{2.22a}$$

$$\frac{dz_n}{dt} = \omega \cdot y_n, \tag{2.22b}$$

where A denotes the amplitude of the perturbation, while ω and φ are its frequency and initial phase, respectively.

2.3 Jet insertion

The jet insertion at the nozzle is modeled as follows. For the sake of simplicity, let us consider a simulation which starts with only two bodies: a single mass-less point fixed at $x = 0$ representing the spinneret nozzle, and a bead modeling an initial jet segment of mass m_i and charge e_i located at distance l_{step} from the nozzle along the x axis. Here, l_{step} denotes the length step used to discretize the jet in a sequence of beads. The starting jet bead is assumed to have a cross-sectional radius a_0, defined as the radius of the filament at the nozzle before the stretching process. Applying the condition that the volume of the jet is conserved, the relation $\pi a_i^2\, l_i = \pi a_0^2 l_{step}$ is valid for any $i-th$ bead. Furthermore, the starting jet bead has an initial velocity v_s along the x axis equal to the bulk fluid velocity in the syringe needle. Once this traveling jet bead is a distance of $2 \cdot l_{step}$ away from the nozzle, a new jet bead (third body) is placed at distance l_{step} from the nozzle along the straight line joining the two previous bodies. Let us now label $i-1$ the farthest bead from the nozzle, and i the last inserted bead. The $i-th$ bead is inserted with the initial velocity $v_i = v_s + v_d$, where v_d denotes the dragging velocity computed as

$$v_d = \frac{v_{i-i} - v_s}{2}. \tag{2.23}$$

Here, the dragging velocity should be interpreted as an extra term which accounts for the drag effect of the electrospun jet on the last inserted segment. Note that the actual dragging velocity definition was chosen in order to not alter the strain velocity term $(1/l_{i-1}) \cdot (dl_{i-1}/dt)$ of Eq 2.1 before and after the bead insertion.

2.4 Dynamic refinement

In the original Ren model, each bead represent a constant quantity V_c of jet volume, so that the relationship $\pi a_i^2\, l_i = \pi a_0^2 l_{step} = V_c$ is valid for any $i-th$ bead. Despite such relationship can be sometimes a useful advantage (as example, see Sec 2.5), the radius reduction a_i of the electrospun nanofiber provides a significant increase of the discretization length l_i value along the jet path (also greater than two order of magnitude). As

consequence, it was observed in literature [24] that the jet discretization close the collector becomes rather coarse to model efficiently the nanofiber. To get around this problem the dynamic refinement procedure developed in Ref [29] can be used in JETSPIN in order to maintain the the length of the elements l_i below a prescribed characteristic length max l_{max}. Whenever a bead length l_i is larger than l_{max}, the nanofiber is refined by discretizing uniformly the jet at the length step l_{step} value given in input file. The discretization is performed by Akima spline interpolations [34, 35] of the main quantities describing the jet beads (positions, fiber radius, stress, velocities). In order to perform the interpolation we proceed following three steps. First of all, we introduce the variable $\lambda \in [0,1]$ to parametrize the jet path, where $\lambda = 0$ identifies the nozzle, and $\lambda = 1$ the jet at the collector. For each bead we compute its respective λ_i value by using the formula

$$\lambda_i = \frac{1}{l_{path}} \sum_{k=1}^{i} l_k \tag{2.24}$$

where $l_{path} = \sum_{k=1}^{N_{beads}} l_k$ is the jet path length from the nozzle to the collector, and $i = 1$ and $i = N_{beads}$ denote the jet bead at the nozzle and collector, respectively ($i = 0$ denotes the nozzle). The set of λ_i values represent the mesh used to build the cubic spline. Given a generic quantity y, the data $y_i = y(\lambda_i)$ are tabulated and used in order to compute the coefficients of the *Akima spline*, following the algorithm reported in Ref [35]. Then, we enforce a uniform parametrization of the nanofiber by imposing all the lengths of the elements equal to l_{step}. The new mesh λ_i^* is defined as

$$\lambda_0^* = 0 \; \lambda_i^* = \frac{i}{N_{beads}^*}, \; i = 1, 2, \ldots, N_{beads}^*. \tag{2.25}$$

where $N_{beads}^* = l_{path}/l_{step}$ represents the number of jet beads in the new representation. Finally, the new values $y(\lambda_i^*)$ are computed for any $i-th$ bead by the *spline interpolation*. The procedure is repeated for positions, fiber radius, stress and velocities of the jet beads in order to provide the respective quantities in the new mesh λ_i^*. It is worth to dressing that the volume $V_i = \pi a_i^2 \, l_i$ represented by each bead is not anymore constant, and, consequently, $\pi a_i^2 \, l_i \neq \pi a_0^2 l_{step}$. It is also possible keep few beads in their positions before and after the refinement procedure in order to reduce possible errors introduced by the interpolation procedure. In other words, these anchor beads are manteined as knots of both the old and new meshes (see directive `dynam refin anchor` in Tab 3.1). In the following text we will refer to this emended version of the Ren model as DyRen model. Further details on the implemented dynamic refinement procedure are available in Ref [29].

2.5 Dimensionless quantities

In JETSPIN all the variables are automatically re-scaled and stored in dimensionless units. In order to adopt a dimensionless form of the equations of motion, we use the dimensionless scaling procedure proposed by Reneker et al.[11]. We define a characteristic length

$$L_0 = \sqrt{\frac{\bar{q}^2}{\pi a_0^2 G}} = l_{step} \sqrt{\frac{\pi a_0^2 \rho_V^2}{G}}, \tag{2.26}$$

where we write the charge q as $\pi a_0^2 l_{step} \rho_V$, denoting ρ_V the electric volume charge density of the filament. Further, we divide the time t and the stress σ by their respective characteristic scales reported in Tab2.1. Applying the condition that the volume of the jet is conserved $\pi a_i^2 \, l_i = V_i$ for any $i-th$ bead, we rewrite the EOM as:

$$\frac{d\vec{\bar{\mathbf{r}}}_i}{d\bar{t}} = \vec{\bar{v}}_i \tag{2.27a}$$

$$\frac{d\bar{\sigma}_i}{d\bar{t}} = -\left[\bar{\sigma}_i - \bar{\sigma}_Y + \bar{k}_{ve} \left(\frac{1}{\bar{l}_i} \frac{d\bar{l}_i}{d\bar{t}} \right)^{n_{ve}} \right] \tag{2.27b}$$

$$\begin{aligned}\frac{d\vec{\bar{v}}_i}{d\bar{t}} =&\Phi\cdot\vec{\mathbf{x}}+\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{Q_{ij}}{\bar{R}_{ij}^2+\bar{a}_j^2}\cdot\vec{\mathbf{u}}_{ij}-F_{ve,i}\bar{V}_i\frac{\bar{\sigma}_i}{\bar{l}_i}\cdot\vec{\mathbf{t}}_i+F_{ve,i+1}\bar{V}_{i+1}\frac{\bar{\sigma}_{i+1}}{\bar{l}_{i+1}}\cdot\vec{\mathbf{t}}_{i+1}\\ &+A_i\frac{\bar{k}_i}{4}\left(\frac{\sqrt{\bar{V}_i}}{\sqrt{\bar{l}_i}}+\frac{\sqrt{\bar{V}_{i-1}}}{\sqrt{\bar{l}_{i-1}}}\right)^2\cdot\vec{\mathbf{c}}_i+F_g\cdot\vec{\mathbf{x}}-\Gamma_i\,\bar{l}_i^{0.905}\bar{v}_{tan,i}^{1+0.19}\cdot\vec{\mathbf{t}}_{i-1}-\\ &-\Lambda_i\frac{\bar{k}_i\bar{v}_{tan,i}^2}{4}\left(\frac{\sqrt{\bar{V}_i}}{\sqrt{\bar{l}_i}}+\frac{\sqrt{\bar{V}_{i-1}}}{\sqrt{\bar{l}_{i-1}}}\right)^2\cdot\vec{\mathbf{c}}_i+\sqrt{2\Theta_v}\cdot\vec{\boldsymbol{\eta}}_i(\bar{t})\end{aligned} \tag{2.27c}$$

where we used the dimensionless derived variables and groups defined in Tab2.1. It is worth stressing that the viscoelastic and surface tension force terms are slightly different from the dimensionless form provided by Reneker et al. [11], since we are considering the DyRen model presented in Sec 2.4. In particular, we do not use the relation $\pi a_i^2\, l_i = \pi a_0^2 l_{step}$ since it is valid only for the Ren model (see discussion in Sec 2.4). Note that, whenever the Kelvin–Voigt model is activated, Eq 2.27b is replaced by

$$\frac{d\bar{\sigma}_i}{d\bar{t}}=\frac{1}{\bar{l}_i}\frac{d\bar{l}_i}{d\bar{t}}-\frac{1}{\bar{l}_i}\frac{d^2\bar{l}_i}{d\bar{t}^2}. \tag{2.28}$$

Finally, the dimensionless EOM of the nozzle are given as

$$\frac{d\bar{y}_i}{d\bar{t}}=-K_s\cdot\bar{z}_i \tag{2.29a}$$

$$\frac{d\bar{z}_i}{d\bar{t}}=K_s\cdot\bar{y}_i\,, \tag{2.29b}$$

with the dimensionless parameter K_s defined in Tab2.1.

Characteristic Scales	
$L_0 = l_{step}\sqrt{\frac{\pi a_0^2 \rho_V^2}{G}}$	$t_0 = \frac{\mu}{G}$
$\sigma_0 = G$	
Dimensionless Quantities	
$\bar{l}_i = \frac{l_i}{L_0}$	$\bar{R}_{ij} = \frac{R_{ij}}{L_0}$
$\bar{\sigma}_i = \frac{\sigma_i}{\sigma_0}$	$\bar{v}_i = v_i \frac{t_0}{L_0}$
$\bar{k}_i = k_i L_0$	$\bar{v}_{tan,i} = v_{tan,i} \frac{t_0}{L_0}$
$\bar{V}_i = \frac{V_i}{L_0^3}$	$\bar{k}_{ve} = \frac{k_{ve}}{\mu \tau^{(n_{ve}-1)}}$
$\bar{a}_i = \frac{a_i}{L_0}$	
Dimensionless Groups	
$\Phi_i = \frac{q_i \Phi_0 \mu^2}{m_i h\, L_0\, G^2}$	$Q_{ij} = \frac{q_i q_j \mu^2}{m_i L_0^3 G^2}$
$F_{ve,i} = \frac{L_0 \mu^2}{m_i G}$	$A_i = \frac{\alpha \mu^2}{m_i G^2}$
$F_g = \frac{g \mu^2}{L_0 G^2}$	$K_s = \omega \frac{\mu}{G}$
$H = \frac{h}{L_0}$	$L_{step} = \frac{l_{step}}{L_0}$
$\Gamma_i = \gamma_i \cdot t_0 L_0^{0.905} \left(\frac{L_0}{t_0}\right)^{0.19}$	$\Theta_v = D_v \cdot \frac{t_0^3}{L_0^2}$
$\Lambda_i = \frac{\rho_a L_0^3}{m_i}$	

Table 2.1: Definitions of the characteristic scales, dimensionless derived variables, and groups employed in the text.

2.6 Integration schemes

In order to integrate the homogeneous differential equations of motion we discretize time as $t_i = t_0 + i\Delta t$ with $i = 1, \ldots, n_{steps}$, where n_{steps} denotes the number of sub-intervals. In JETSPIN three different integration schemes: the first-order accurate Euler scheme, the second-order accurate Heun scheme (sometimes called second-order accurate Runge-Kutta), and the fourth-order accurate Runge-Kutta scheme[36]. The user can select a specific scheme by using appropriate keys in the input file, as described in Sec 3.1. If the airdrag is activated, the explicit strong order scheme by Platen [37, 38, 39] have to be selected, whereof the order of strong convergence was evaluated equal to 1.5.

This scheme avoids the use of derivatives by corresponding finite differences in the same way as Runge-Kutta schemes do for ODEs in a deterministic setting, and it is briefly summarized as follows.

Let us consider a Brownian motion vector process $\mathbf{X} = \{\mathbf{X}_t, t\}$ of d-dimensional satisfying the stochastic differential equation

$$\frac{d\mathbf{X}}{dt} = \mathbf{a}\left(t, X^1, \ldots, X^d\right) + \mathbf{b} d\Omega \tag{2.30}$$

where $\mathbf{a}$ and $\mathbf{b}$ are vectors of d-dimensional usually called drift and diffusion vector coefficients, and $\Omega(t)$ denotes a Wiener process. Denoted Y_t^k the approximation for the k-th component of the vector $\mathbf{X}$ at time t, the integrator has the following form:

$$Y_{t+\Delta t}^k = Y_t^k + b^k \Delta\Omega + \frac{1}{2\sqrt{\Delta t}}\left[a^k\left(\widetilde{\boldsymbol{\Upsilon}}_+\right) - a^k\left(\widetilde{\boldsymbol{\Upsilon}}_-\right)\right]\Delta\Psi + \frac{1}{4}\left[a^k\left(\widetilde{\boldsymbol{\Upsilon}}_+\right) + 2a^k + a^k\left(\widetilde{\boldsymbol{\Upsilon}}_-\right)\right]\Delta t, \tag{2.31}$$

with the vector supporting values

$$\begin{aligned}\widetilde{\boldsymbol{\Upsilon}}_{\pm} &= \mathbf{Y}_t + \mathbf{a}\Delta t \pm \mathbf{b}\sqrt{\Delta t} \\ \widetilde{\boldsymbol{\Phi}}_{\pm} &= \widetilde{\boldsymbol{\Upsilon}}_{+} \pm \mathbf{b}\left(\widetilde{\boldsymbol{\Upsilon}}_{+}\right)\sqrt{\Delta t}.\end{aligned} \tag{2.32}$$

Here, $\Delta\Omega$ and $\Delta\Psi$ indicate normally distributed random variables constructed from two independent $N(0,1)$ standard Gaussian distributed random variables (U_1, U_2) by means of the following linear transformation:

$$\begin{aligned}\Delta\Omega &= U_1\sqrt{\Delta t} \\ \Delta\Psi &= \frac{1}{2}\Delta t^{3/2}\left(U_1 + \frac{1}{\sqrt{3}}U_2\right).\end{aligned} \tag{2.33}$$

Note that in JETSPIN the time step Δt is automatically rescaled by the quantity τ, in accordance with the mentioned dimensionless scaling convention.

2.7 Multiple Timestep Algorithm for Coulomb forces

For simulations involving a large number of beads in the calculation of the long-range Coulomb interactions JETSPIN offers the possibility of using a multiple timestep algorithm which improves the efficiency. The method is based on that described by Streett *et al* in Ref [40] and modified for the special context of electrospinning process.

In the multiple timestep algorithm a primary cutoff r_{pcut} for the Coulomb pair interactions is given. The cutoff r_{pcut} is used to define a standard Verlet neighbor list [41]. Long-range Coulomb forces derived from beads in the neighbor list are usually larger than those due to remaining (far) beads not contained in the list. Therefore, it is possible to assess the Coulomb energy if the forces derived from the beads in the neighbor list are calculated at every timestep, while those from the remaining beads are calculated much less frequently, and are merely extrapolated over the interval. JETSPIN handles this procedure as follows: The Verlet neighbor list is updated at regular intervals n_{mult}. The interval between updates should be usually of the order of $\sim$50 timesteps n_{step}. Whenever the Verlet neighbor list is updated ($n_{step} = 1$), the force contributions from both the beads of the list and outside the list are calculated. Then, the Coulomb forces are computed in total in the subsequent timestep ($n_{step} = 2$), Hence, for $n_{step} > 2$ the Coulomb force acting on a single bead is assessed as sum of the forces derived from the beads in the neighbor list calculated explicitly, and those derived from the outer beads (outside the list) which are calculated by linear extrapolation of the exact forces obtained in the first two timesteps of the multi-step interval. As soon as the condition $n_{step} = n_{mult}$ is achieved, the procedure is restarted (the neighbor list is again updated). It is readily apparent how this scheme can lead to a significant saving in execution time.

Note that the accuracy of the algorithm is a function of the multiple step interval, and decreases as n_{mult} increases. Further, the neighbor list removes the need to scan over all the beads in the simulation at every timestep. It is worth to highlight that the algorithm is not time reversible.

We remark that it is also possible to define in input file a maximum displacement Δr_{pcut} in order to achieve a better control on the accuracy of the multiple timestep algorithm. The maximum displacement Δr_{pcut} works as follows: whenever a bead has traveled for a distance larger than Δr_{pcut} from its position at multi-step $n_{mult} = 1$, the Verlet neighbor list is updated (even if $n_{step} < n_{mult}$).

Chapter 3

JETSPIN Data Files

Scope of Chapter

This chapter describes all the input and output files of JETSPIN.

3.1 Input file

In order to run JETSPIN simulations an input file has to be prepared, which is a free-format and case-insensitive. The input file has to be named `input.dat`, and it contains the selection of the model system, integration scheme directives, specification of various parameters for the model, and output directives. The input file does not requires a specific order of key directives, and it is read by the input parsing module. Every line is treated as a command sentence (record). Records beginning with the symbol # (commented) and blank lines are not processed, and may be added to aid readability. Each record is read in words (directives and additional keywords and numbers), which are recognized as such by separation by one or more space characters.

As in the example given in `input test`, the last record is a `finish` directive, which marks the end of the input data. Before the `finish` directive, a wide list of directives may be inserted (see Tab 3.1). The key `systype` should be used to set the nanofiber model. In JETSPIN two models are available: 1) the one dimensional model similar to the model of Cap 2 but assuming the nanofiber to be straight along the $\vec{\mathbf{x}}$ axis, and, therefore, neglecting the surface tension force $\vec{\mathbf{f}}_{st}$; 2) the three dimensional model described in Cap 2. Internally these options are handled by the integer variable `systype`, which assumes the values explained in Tab 3.1. Further details of the 1-D model can be found in Refs [22, 42]. A series of variables are mandatory and have to be defined. For example, `timestep`, `final time`, `initial length`, etc. (see underlined directives in Tab 3.1). A missed definition of any mandatory variable will call an error banner on the terminal. The user can use two possible ways to define the initial geometry of a nanofiber both given the mandatory directive `initial length`: 1) the discretization step length of nanofiber is specified by the directive `resolution`, which causes automatically the setting of the jet segments number (the number of segment in which the jet is discretised); 2) the jet segments number is declared by the directive `points`, while the value of the discretization step length is automatically set by the program (as in Example **Test Case 3** in Sec 4.3). The directive `cutoff` indicates the length of the upper and lower proximal jet sections, which interact via Coulomb force on any bead. It is worth stressing that the length value is set at the nozzle, so that the effective cutoff increases along the simulation as the nanofiber is stretched. The directive `restart yes` can be used to restart a JETSPIN job (for further information see Section 1.6).

The user should pay special attention in choosing the time integration step given by the directive `timestep`, whose detailed considerations are provided in Ref [20]. The reader is referred to Tab 3.1 for a complete listing of all directives defining the electrospinning setup parameters. Not all these quantities are mandatory, but the user is informed that whenever a quantity is missed, it is usually assumed equal to zero by default (exceptions are stressed in Tab 3.1). **Note all the quantities have to be expressed in centimeter–gram–second unit system** (e.g. charge in statcoulomb, electric potential in statvolt, etc.).

directive:	meaning:
airdrag yes	active the inclusion of the airdrag force in the model (default: no) (note that if yes the directive "integrator 4" is mandatory)
airdrag airdensity f	mass density of the air surrounding the nanofiber (default: 0)
airdrag airviscosity f	kinematic viscosity of the air surrounding the nanofiber (default: 1)
airdrag airvelocity f	velocity of the air with respect to the z axis (default: 0)
airdrag amplitude f	amplitude of the randomic force rescaled by γ (default: 1) (note that the dissipation coefficient γ is automatically computed by JETSPIN using the input data)
collector distance f	distance of collector along the x axis from the nozzle (note the nozzle is assumed in the origin)
cutoff f	length of the proximal jet sections interacting by Coulomb force (default: equal to the collector distance)
density mass f	mass density of the nanofiber
density charge f	electric volume charge density of the nanofiber
dynam refin yes	active the dynamic refinement during the simultion
dynam refin threshold f	set the threshold beyond which the refinement is applied
dynam refin every f	the refinement is applied every f seconds
dynam refin start f	the refinement is applied only if the simulation time is greater than or equal to f seconds
dynam refin anchor f	during the refinement procedure all the anchor beads inserted at distance f preserve their positions (they are manteined as knots of both the old and new meshes)
elastic modulus f	elastic modulus of the nanofiber
external potential f	electric potential between the nozzle and collector
external potential type i	set the waweform describing the time evolution of the external electric potential. The *integer* can be equal to 0 for a static field, or 1 for a rectangular waveform (default: 0)
external potential freq f	frequency of the waveform describing the electric potential
final time f	set the end time of the simulation
finish	close the input file (last data record)
gravity yes	active the inclusion of the gravity force in the model (default: no)
hbfluid yes	allow the set of parameters needed to define the Herschel–Bulkley stress. Note that if it is set no, JETSPIN will use the simple Maxwellian fluid model (default: no)
hbfluid consistency f	flow consistency index rescaled by μ (in μunit) as defined in Eq 2.5 (Note you should also set: hbfluid yes) (default: 1)
hbfluid index f	flow behavior index (Note you should also set: hbfluid yes) (default: 1)
initial length f	length of the nanofiber at the initial time
integrator i	set the integration scheme. The *integer* can be '1' for Euler, '2' for Heun, '3' for Runge-Kutta, and '4' for the Platen scheme (only if airdrag force is activated).
inserting yes	inject new beads as explained in Subsec
kvfluid yes	activate the Kelvin–Voigt model which is used instead of Maxwell model (default: no) (Note it is actually implemented only for working with the 3d-model without airdrag ativated: system 3 is mandatory)
maximum displ f	set the maximum displacement of the neighbor list (see Sec 2.7)
multiple step yes	activate the multiple timestep algorithm for Coulomb forces (default: no)
multiple step every i	set the interval of multiple-steps (see Sec 2.7)
nozzle cross f	cross section radius of the nanofiber at the nozzle
nozzle stress f	viscoelastic stress of the nanofiber at the nozzle
nozzle velocity f	bulk fluid velocity of the nanofiber at the nozzle
perturb yes	active the periodic perturbation at the nozzle

perturb freq f	frequency of the perturbation at the nozzle
perturb ampl f	amplitude of the perturbation at the nozzle
points i	number of segment in which the jet is discretised
primary cutoff f	set the primary cutoff of the neighbor list (see Sec 2.7)
print list s_1 ...	print on terminal statistical data as indicated by symbolic strings (see Tab 3.2 for detailed informations)
print time f	print data on terminal and output file every f seconds
print pdb frame f	print the jet geometry in a single PDB file format in centimeters
print pdb rescalepdb f	print the PDB file format data rescaled by f (default: 1)
print xyz f	print the trajectory in XYZ file format in centimeters every f seconds
print xyz frame f	print the jet geometry in a single XYZ file format in centimeters every f seconds (default name of files: frame%06d.xyz)
print xyz maxnum i	print the XYZ file format with i beads starting from the nearest bead to the collector (default: 100)
print xyz rescale f	print the XYZ file format data rescaled by f (default: 1)
printstat list s_1 ...	print on output file statistical data as indicated by symbolic strings (see Tab 3.2 for detailed informations)
removing yes	remove beads at collector
restart yes	restart JETSPIN from a previous job (see Section 1.6)
restart reset	reset all the accumulated statistical data collected in the previous simulation when a job is restarted
resolution f	discretization step length of nanofiber
seed i	seed control to the internal random number generator
surface tension f	surface tension coefficient of the nanofiber
system i	set the nanofiber model. The *integer* can be equal to '1' for select the 1d-model, '3' for the 3d-model, and '4' for the 3d-model with airdrag ativated
timestep f	set the time step for the integration scheme
viscosity f	viscosity of the nanofiber
yield stress f	yield stress (default: 0)

Table 3.1: Here, we report the list of directives available in JETSPIN. Note i, f, and s denote an integer number, a floating point number, and a string, respectively. The underlined directives are mandatory. The default value is zero (exceptions are stressed in parentheses).

3.2 Output files

A series of specific directives causes the writing of output files (see Tab 3.1). In JETSPIN three output files can be written: 1) the file `statdat.dat` containing time-dependent statistical data of simulated process; 2) the file `traj.xyz` reporting the nanofiber trajectory in XYZ format file; 3) the files `frame%06d.xyz` reporting the nanofiber geometries in splitted XYZ format files (a single file for each frame); 4) the files `frame%06d.pdb` reporting the nanofiber geometries in splitted PDB (Protein Data Bank) format files (a single file for each frame). Note that in the last case a topological file `frame%06d.psf` in PSF (Protein Structure File) format will be printed alongside each pdb file. These two files (PDB and PSF) can be opened by VMD-Visual Molecular Dyanamics [43].

Various statistical data can be written on the file `statdat.dat` , and the user can select them in input using the directive `printstat list` followed by appropriate symbolic strings, whose the list with corresponding meanings is reported in Tab 3.2. The selected observables will be printed at time interval on the same line following the order specified in input file. In the same way, a list of statistical data can be printed on computer terminal using the directive `print list` followed by the symbolic strings of Tab 3.2.

The file `traj.xyz` is written as a continuous series of XYZ format frames taken at time interval, so that can be read by suitable visualization programs (e.g. VMD-Visual Molecular Dynamics [43], UCSF Chimera [44], etc.) to generate animations. The number of elements contained in the file is kept constant equal to the value specified by the directive `print xyz maxnum`, since few programs (e.g. VMD) do not manage a variable number of elements along the simulations. If the actual number of beads is lower than the given constant `maxnum`, the extra elements are printed in the origin point. On the other hand, the number of beads printed in the files `frame%06d.xyz` and `frame%06d.pdb` is exactly the actual number of beads used to discretized the jet at the given frame. Note that the unit of mesurement used in the files, `traj.xyz`, `frame%06d.xyz` and `frame%06d.pdb` is the centimeter multiplied by f, which was given in input file by the directve " *print xyz rescale f* " and " *print pdb rescalepdb f* " (see Table 3.1), respectively.

keys:	**meaning:**
t	unscaled time
ts	scaled time
x, y, z	unscaled coordinates of the farest bead
xs, ys, zs	scaled coordinates of the farest bead
st	unscaled stress of the farest bead
sts	scaled stress of the farest bead
vx, vy, vz	unscaled velocities of the farest bead
vxs, vys, vzs	scaled velocities of the farest bead
yz	unscaled normal distance from the x axis of the farest bead
yzs	scaled normal distance from the x axis of the farest bead
mass	last inserted mass at the nozzle
q	last inserted charge at the nozzle
cpu	time for every print interval
cpur	remaining time to the end
cpue	elapsed time
n	number of beads used to discretise the jet
f	index of the first bead
l	index of the last bead
curn	current at the nozzle
curc	current at the collector
vn	velocity modulus of jet at the nozzle
vc	velocity modulus of jet at the collector
svc	strain velocity at the collector
mfn	mass flux at the nozzle
mfc	mass flux at the collector
rc	radius of jet at the collector
rrr	radius reduction ratio of jet
lp	length path of jet
rlp	length path of jet divided by the collector distance
mxst	maximum stress along the jet
mxsx	x position of the jet maximum stress

Table 3.2: In JETSPIN a series of instantaneous and statistical data are available to be printed by selecting the appropriate key. Here, the list of symbolic string keys is reported with their corresponding meanings. Note by *scaled* we mean that the observable was rescaled by the characteristic values provided in Tab2.1. The underlined keys correspond to data which are averaged on the time interval given by the directive `print time` in input file. Note by *farest bead* we mean the farest bead from the nozzle (with greatest x value). Note all the quantities are expressed in centimeter–gram–second unit system, excepted the dimensionless scaled observables.

Chapter 4

Example Simulations

Scope of Chapter

This chapter describes the standard test cases for JETSPIN, the input and output files for which are in the *examples* sub-directory.

4.1 Test Case 1: 1-D case

This input file (located in the *input-1* sub-directory) provides the dimensionless parameter values $Q = 12$, $V = 2$ and $F_{ve} = 12$, which have been already used as reference case in Refs [11, 22].

4.2 Test Case 2: 1-D case with bead insertion at the nozzle activated

This input file (located in the *input-2* sub-directory) provides the dimensionless parameter values $Q = 12$, $V = 2$ and $F_{ve} = 12$, as in the previous test case, but here we have activated the injection of new beads by the directive `inserting yes` in the input file.

4.3 Test Case 3: Three-D simulation of electrospinning process

The electrospinning of polyvinylpyrrolidone (PVP) nanofibers is a prototypical process, which has been largely investigated in literature.[6, 8, 5] By the input located in the *input-3* sub-directory, we simulate the electrospinning process of PVP solutions. Then, the theoretical results predicted by the models are compared with the aforementioned experimental data. In particular, we reproduce an experiment in which a solution of PVP (molecular weight = 1300 kDa) is prepared by a mixture of ethanol and water (17:3 v:v), at a concentration of about 2.5 wt%. The applied voltage is in a range around 10 kV, and the collector is placed at distance 16 cm from the nozzle, which has radius 250 micron (further details are provided in Ref. [45, 20]). As rheological properties of such system we consider the zero-shear viscosity $\mu_0 = 0.2\,\mathrm{g/(cm \cdot s)}$ [46, 47], the elastic modulus $G = 5 \cdot 10^4\,\mathrm{g/(cm \cdot s^2)}$ [48], and the surface tension $\alpha = 21.1\,\mathrm{g/s^2}$ [46]. We use for the simulation a viscosity value μ which is two order of magnitude larger than the zero-shear viscosity μ_0 reported, since the strong longitudinal flows we are dealing with can lead to an increase of the extensional viscosity from μ_0, as already observed in literature.[11, 32] The mass density is equal to $0.84\,\mathrm{g/cm^3}$, while the charge density was estimated by experimental observations of the current measured at the nozzle. For convenience, all the simulation parameters are summarized in Tab 4.1.

ρ (kg/m^3)	ρ_q (C/L)	a_0 (cm)	υ_s (cm/s)	α (mN/m)	μ (Pa·s)	G (Pa)	V_0 (kV)	ω (s^{-1})	A (cm)
840	$2.8 \cdot 10^{-7}$	$5 \cdot 10^{-3}$	0.28	21.1	2.0	50000	9.0	10^4	10^{-3}

Table 4.1: Simulation parameters for the simulation of PVP nanofibers. The headings used are as follows: ρ: density, ρ_q: charge density, a_0: fiber radius at the nozzle, υ_s: bulk fluid velocity in the syringe needle, α: surface tension, μ: viscosity, G : elastic modulus, V_0 : applied voltage bias, ω: frequency of perturbation, A : amplitude of perturbation. The bulk fluid velocity υ_s was estimated considering that the solution was pumped at constant flow rate of 2 mL/h in a needle of radius 250 micron.

4.4 Test Case 4: Three-D simulation in presence of a gas counter-flow

4.5 Test Case 5: Three-D simulation with dynamic refinement

4.6 Test Case 6: Three-D simulation of a quasi-Newtonian fluid

Chapter 5

Tips

Scope of Chapter

This chapter describes few useful tips which can be used to enhance results obtained by JETSPIN.

5.1 How to visualize a PDB file in VMD software package

The PDB file written by JETSPIN can be easily visualized in VMD software package. The first step is to download and install VMD which can be found at website: http://www.ks.uiuc.edu/Research/vmd/. As VMD is installed, the PDB file can be opened using VMD: "vmd frame000001.pdb" in the terminal or click on "file" in the main white box and then on "load molecule" (see Fig 5.1), browse for your PDB file frame000001.pdb and load it. In order to load the topological structure of the jet, it is possible to load the PSF file written by JETSPIN alongside the PDF file: right click on the just uploaded file (e.g. "frame000001.pdb") in the white box in the middle of the "MD Main" window, and select "Load Coordinates into Molecule" pull-down menu option (see Fig 5.2). Hence, browse for the PSF file (frame000001.psf) defining the bonds for each pair of consecutive jet beads, and load it. Finally, the jet will be displayed in the graphics window.

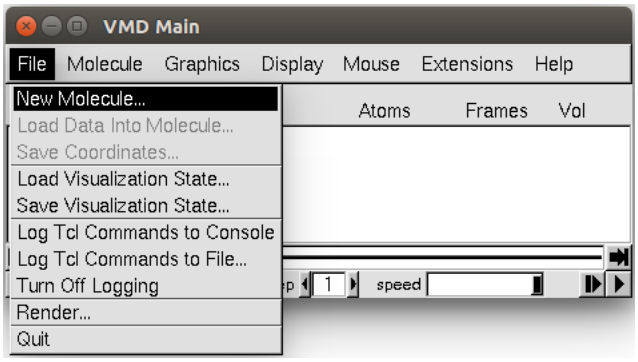

Figure 5.1: Snapshot of the main white box of VMD.

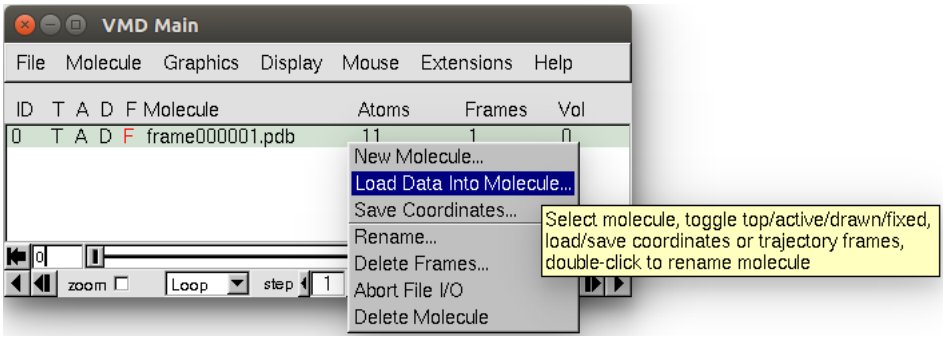

Figure 5.2: Snapshot of the "Load Coordinates into Molecule" selection in VMD.

Bibliography

[1] D. Li, Y. Wang, Y. Xia, Electrospinning nanofibers as uniaxially aligned arrays and layer-by-layer stacked films, Advanced Materials 16 (4) (2004) 361–366. 1.1

[2] A. Greiner, J. H. Wendorff, Electrospinning: a fascinating method for the preparation of ultrathin fibers, Angewandte Chemie International Edition 46 (30) (2007) 5670–5703. 1.1

[3] C. P. Carroll, E. Zhmayev, V. Kalra, Y. L. Joo, Nanofibers from electrically driven viscoelastic jets: modeling and experiments, Korea-Aust Rheol J 20 (2008) 153–164. 1.1

[4] Z.-M. Huang, Y.-Z. Zhang, M. Kotaki, S. Ramakrishna, A review on polymer nanofibers by electrospinning and their applications in nanocomposites, Composites science and technology 63 (15) (2003) 2223–2253. 1.1

[5] L. Persano, A. Camposeo, C. Tekmen, D. Pisignano, Industrial upscaling of electrospinning and applications of polymer nanofibers: a review, Macromolecular Materials and Engineering 298 (5) (2013) 504–520. 1.1, 4.3

[6] A. L. Yarin, B. Pourdeyhimi, S. Ramakrishna, Fundamentals and Applications of Micro and Nanofibers, Cambridge University Press, 2014. 1.1, 2.1, 4.3

[7] J. H. Wendorff, S. Agarwal, A. Greiner, Electrospinning: materials, processing, and applications, John Wiley & Sons, 2012. 1.1

[8] D. Pisignano, Polymer Nanofibers: Building Blocks for Nanotechnology, Royal Society of Chemistry, 2013. 1.1, 4.3

[9] Y. Zeng, Y. Wu, Z. Pei, C. Yu, Numerical approach to electrospinning, International Journal of Nonlinear Sciences and Numerical Simulation 7 (4) (2006) 385–388. 1.1

[10] J. Feng, Stretching of a straight electrically charged viscoelastic jet, Journal of Non-Newtonian Fluid Mechanics 116 (1) (2003) 55–70. 1.1, 2.1

[11] D. H. Reneker, A. L. Yarin, H. Fong, S. Koombhongse, Bending instability of electrically charged liquid jets of polymer solutions in electrospinning, Journal of Applied physics 87 (9) (2000) 4531–4547. 1.1, 2.1, 2.5, 2.5, 4.1, 4.3

[12] A. L. Yarin, S. Koombhongse, D. H. Reneker, Taylor cone and jetting from liquid droplets in electrospinning of nanofibers, Journal of Applied Physics 90 (9) (2001) 4836–4846. 1.1

[13] S. V. Fridrikh, H. Y. Jian, M. P. Brenner, G. C. Rutledge, Controlling the fiber diameter during electrospinning, Physical review letters 90 (14) (2003) 144502. 1.1

[14] S. Theron, E. Zussman, A. Yarin, Experimental investigation of the governing parameters in the electrospinning of polymer solutions, Polymer 45 (6) (2004) 2017–2030. 1.1

[15] C. Lu, P. Chen, J. Li, Y. Zhang, Computer simulation of electrospinning. part i. effect of solvent in electrospinning, Polymer 47 (3) (2006) 915–921. 1.1

[16] X. Wang, Y. Liu, C. Zhang, Y. An, X. He, W. Yang, Simulation on electrical field distribution and fiber falls in melt electrospinning, Journal of nanoscience and nanotechnology 13 (7) (2013) 4680–4685. 1.1

[17] I. Coluzza, D. Pisignano, D. Gentili, G. Pontrelli, S. Succi, Ultrathin fibers from electrospinning experiments under driven fast-oscillating perturbations, Physical Review Applied 2 (5) (2014) 054011. 1.1, 2.2

[18] W. Smith, Molecular dynamics on hypercube parallel computers, Computer Physics Communications 62 (2) (1991) 229–248. 1.3

[19] D. Brown, J. H. Clarke, M. Okuda, T. Yamazaki, A domain decomposition parallelization strategy for molecular dynamics simulations on distributed memory machines, Computer Physics Communications 74 (1) (1993) 67–80. 1.3

[20] M. Lauricella, G. Pontrelli, I. Coluzza, D. Pisignano, S. Succi, Jetspin: a specific-purpose open-source software for simulations of nanofiber electrospinning, Computer Physics Communications 197 (2015) 227–238. 1.3, 3.1, 4.3

[21] J. Racine, The cygwin tools: a gnu toolkit for windows, Journal of Applied Econometrics 15 (3) (2000) 331–341. 1.4

[22] G. Pontrelli, D. Gentili, I. Coluzza, D. Pisignano, S. Succi, Effects of non-linear rheology on the electrospinning process: a model study, Mechanics Research Communications 61 (2014) 41–46. 2.1, 3.1, 4.1

[23] H. A. Barnes, J. F. Hutton, K. Walters, An introduction to rheology, Vol. 3, Elsevier, 1989. 2.1

[24] T. A. Kowalewski, S. Barral, T. Kowalczyk, Modeling electrospinning of nanofibers, in: IUTAM symposium on modelling nanomaterials and nanosystems, Springer, 2009, pp. 279–292. 2.1, 2.1, 2.4

[25] J. Feng, The stretching of an electrified non-newtonian jet: A model for electrospinning, Physics of Fluids (1994-present) 14 (11) (2002) 3912–3926. 2.1

[26] M. M. Hohman, M. Shin, G. Rutledge, M. P. Brenner, Electrospinning and electrically forced jets. i. stability theory, Physics of Fluids (1994-present) 13 (8) (2001) 2201–2220. 2.1

[27] T. Kowalewski, S. NSKI, S. Barral, Experiments and modelling of electrospinning process, Technical Sciences 53 (4). 2.1

[28] A. Ziabicki, H. Kawai, High-Speed Fiber Spinning: Science and Engineering Aspects, Krieger Publishing Co, 1991. 2.1

[29] M. Lauricella, G. Pontrelli, D. Pisignano, S. Succi, Dynamic mesh refinement for discrete models of jet electro-hydrodynamics, Journal of Computational Science. 2.1, 2.4, 2.4

[30] M. Lauricella, G. Pontrelli, I. Coluzza, D. Pisignano, S. Succi, Different regimes of the uniaxial elongation of electrically charged viscoelastic jets due to dissipative air drag, Mechanics Research Communications 69 (2015) 97–102. 2.1

[31] M. Lauricella, G. Pontrelli, D. Pisignano, S. Succi, Nonlinear langevin model for the early-stage dynamics of electrospinning jets, Molecular Physics 113 (17-18) (2015) 2435–2441. 2.1

[32] A. L. Yarin, Free liquid jets and films: hydrodynamics and rheology, Longman Scientific & Technical Harlow, 1993. 2.1, 4.3

[33] M. Lauricella, D. Pisignano, S. Succi, Three-dimensional model for electrospinning processes in controlled gas counterflow, submitted to Journal of Physical Chemistry A. 2.1

[34] H. Akima, A new method of interpolation and smooth curve fitting based on local procedures, Journal of the ACM (JACM) 17 (4) (1970) 589–602. 2.4

[35] H. Akima, A method of univariate interpolation that has the accuracy of a third-degree polynomial, ACM Transactions on Mathematical Software (TOMS) 17 (3) (1991) 341–366. 2.4, 2.4

[36] W. H. Press, Numerical recipes 3rd edition: The art of scientific computing, Cambridge university press, 2007. 2.6

[37] E. Platen, Derivative free numerical methods for stochastic differential equations, in: Stochastic Differential Systems, Springer, 1987, pp. 187–193. 2.6

[38] P. E. Kloeden, E. Platen, Numerical solution of stochastic differential equations, Vol. 23, Springer, 1992. 2.6

[39] E. Platen, N. Bruti-Liberati, Numerical solution of stochastic differential equations with jumps in finance, Vol. 64, Springer, 2010. 2.6

[40] W. Streett, D. Tildesley, G. Saville, Multiple time-step methods in molecular dynamics, Molecular Physics 35 (3) (1978) 639–648. 2.7

[41] M. P. Allen, D. J. Tildesley, Computer simulation of liquids, Oxford university press, 1989. 2.7

[42] C. P. Carroll, Y. L. Joo, Discretized modeling of electrically driven viscoelastic jets in the initial stage of electrospinning, Journal of Applied Physics 109 (9) (2011) 094315. 3.1

[43] W. Humphrey, A. Dalke, K. Schulten, Vmd: visual molecular dynamics, Journal of molecular graphics 14 (1) (1996) 33–38. 3.2

[44] E. F. Pettersen, T. D. Goddard, C. C. Huang, G. S. Couch, D. M. Greenblatt, E. C. Meng, T. E. Ferrin, Ucsf chimera: a visualization system for exploratory research and analysis, Journal of computational chemistry 25 (13) (2004) 1605–1612. 3.2

[45] M. Montinaro, V. Fasano, M. Moffa, A. Camposeo, L. Persano, M. Lauricella, S. Succi, D. Pisignano, Sub-ms dynamics of the instability onset of electrospinning, Submitted to Soft Matter. 4.3

[46] N. Yuya, W. Kai, B.-S. Kim, I. Kim, Morphology controlled electrospun poly(vinyl pyrrolidone) fibers: effects of organic solvent and relative humidity, Journal of Materials Science and Engineering with Advanced Technology. 4.3

[47] V. Bühler, Polyvinylpyrrolidone excipients for pharmaceuticals: povidone, crospovidone and copovidone, Springer Science & Business Media, 2005. 4.3

[48] V. N. Morozov, A. Y. Mikheev, Water-soluble polyvinylpyrrolidone nanofilters manufactured by electrospray-neutralization technique, Journal of Membrane Science 403 (2012) 110–120. 4.3