BDpack

BDpack is a package to numerically calculate the configurational evolution of polymeric solution using Brownian dynamics simulation of bead-spring micro-mechanical model. The algorithm used in BDpack incorporates high-fidelity and computationally efficient calculation of hydrodynamic interactions (HI) and excluded volume (EV) forces. BDpack is written in parallel by employing message passing interface (MPI) for distributed-memory-architectures.

The aim of this project is to write the codes self descriptive, documented, yet very efficient to enable its applicability and development. To this end, the codes are written in modular fashion using Fortran.

Documentation

Installation and Running

In what follows, the terminology `< >` is used to specify the parts that should be filled by user.

Dependencies

- **MPI compiler**:

BDpack is parallelized using MPI and has been tested on several machines using OpenMPI MPI fortran compilers. Therefore, the user needs to make sure that the MPI libraries are installed on the machine. You can check whether you have it installed by typing:

```
$ mpif90 --version
```

It is possible that you have `MPI` libraries installed, but you need to `export` the address to the directory `bin` which contains the binary files of `MPI` implementation. Equivalently, the explicit address can be used when an MPI binary file is invoked, i.e., `<Address to the bin directory of MPI>/mpif90 --version`. The base level compiler of `mpif90` can be intel compiler `ifort` or GNU compiler `gfortran` or any other compiler. You can switch between the two by using `$ export OMPI_FC=<The desired compiler>`. For more recent Open MPI library versions, `mpiifort` is used for intel compiler and `mpif90` is considered for GNU compiler. Please consult Open MPI website to get the most updated information in this regard. In the rest of this documenation, it is assumed that `mpif90` is used as MPI compiler.

It is essential to compile BDpack with the same (or compatible) compiler which is used for making MPI libraries. Please consult Open MPI website if you wish to recompile the MPI libraries.

If there are more than one version of intel compiler on you machine and you wish to use a specific one, consider using `$ source <Address to intel>/bin/ifortvars.sh intel64`. The same result can be achieved for GNU compilers using `update-alternatives` command.

- **Intel math kernel library (MKL)**:

The BDpack program uses MKL, i.e., levels 1-3 of MKL_BLAS and MKL_LAPACK to perform the large scale linear algebra.

MKL can be accessed free of charge for individual use. Please consult the intel free software tools for more information. For students, the software tools contain MKL and intel compilers. Due to extensive application of MKL, the package works best if `mpif90` is used with intel fortran compiler and intel processors. However, gnu compiler can be used as well.

To install the package, the user has to make sure that MKL is installed properly. The root directory of MKL may be automatically saved in the macro `MKLROOT` which can be checked by typing `$ echo $MKLROOT`. If it is not detected, the address to MKL directory will be specified explicitly as explained in the step by step installation below.

It is essential to compile BDpack with the same (or compatible) compiler which is used for making MKL libraries `mkl_blas95_lp64.a` and `mkl_lapack95_lp64.a`. Otherwise, these two libraries should be recompiled using the appropriate source codes which are located in `interfaces/blas95` and `interfaces/lapack95` subdirectory of MKL root directory, respectively.

Moreover, due to dependency of fortran `.mod` files on compiler type and version, the files `blas95.mod`, `lapack95.mod`, 'f95_precision.mod' may need to be regenerated and transferred to `<MKL root>/include/intel64/lp64/`. This can be done by compiling `blas.f90` and `lapack.f90` files which can be accessed in `<MKL root>/include`.

Step by step installation

For complete installation of the package, the following steps should be performed:

1- Download the zip or tar file to the destination directory on your Linux machine.

2- Unzip or untar the package to a directory named BDpack. For instance, for tarball file:

```
$ mkdir ./BDpack | tar -xf <tar file> -C ./BDpack --strip-components=1
```

3- Make sure that `mpif90` is installed on your machine:

```
$ mpif90 --version
```

If you wish to use intel compiler, use:

```
$ export OMPI_FC=ifort
```

and for gnu compiler, use:

```
$ export OMPI_FC=gfortran
```

4- In any directory, type `$echo $MKLROOT`. If no address is returned for root directory of MKL, `MKLROOT` should be explicitly specified in the file `make.inc` which is located in `src` directory of the package. In other words, modify the `MKLROOT` line in `make.inc` file by uncommenting it and typing the address of MKL: i.e., `MKLROOT = <Address of MKL>`.

5- To compile the codes which are located in the `src` directory, simply type in the `src` directory:

```
$ make
```

6- The auxiliary codes (which are used in the [[Tutorials|https://github.com/amir-saadat/BDpack/wiki/Tutorials]]) are located in the `src/utils` directory. In the `src` directory, type:

```
$ make utils
```

Running BDpack projects

1- Upon proper installation of the program, the executable file `BDpack` is built in `<BDpack root>/bin/` directory.

2- The program can get started in any directory which contains `BDpack` and `input.dat` files and `data` directory by using

the standard run command of the OpenMPI:

```
$ mpirun -np x <BDpack root>/bin/BDpack input.dat
```

with x replaced by the number of processes.

3- For practice on how to use BDpack, several examples are provided in `<BDpack root>/projects/`. A sample form of `input.dat` can be found in given examples. The parameters are specified with their names followed by a separator `:` and their values:

```
driver        : dilute_bs
#---------------------------------------
# Configuration parameters
#---------------------------------------
nchain        : 2
nseg          : 10
tplgy         : Comb
Arms          : 2 2 4
nseg_ar       : 2
Rel-Model     : Rouse
...
```

4- All the output data will be provided in the directory `data`.

User Inputs

Description of different drivers

BDpack provides the tools to simulate polymers in infinitely dilute solutions using bead-spring model (**dilute_bs** driver) as well as interacting macromolecules in semi-dilue solutions (**semidilute_bs** driver). However, the codes for infinitely dilute solutions of bead-rod chains have also been implemented and will be provided in the later versions of the package.

In what follows, the parameter description of different drivers will be given in detail. In case the parameters are not specified by user, their default values will be set in the program. However, the parameters with U as their default value should be specified by user for proper behavior of the program. Note that the logical parameters are specified with **TRUE** or **FALSE**.

Infinitely dilute solution using bead-spring model (dilute_bs)

The governing equations and numerical algorithms for infinitely dilute solution which is used in BDpack can be found in our recent article; Saadat and Khomami (2014).

- **Configuration parameters**

Parameter	Description	Type	Default
nchain	total number of chains	integer	U
nseg	total number of segments of one chain	integer	U
tplgy	topology of the chain, currently **Linear** is implemented and tested. **Comb** topology is implemented but not tested.	character	Linear
Arms	(Only **Comb** topology) the first entry is the number of arms followed with the bead at which the arms are grafted. For example, 3 2 4 5 specifies a chain with 3 arms which are located at beads 2, 4, and 5	integer	U
nseg_ar	(Only **Comb** topology) number of segments in arms	integer	U
	the model which accounts for dimensionless longest relaxation time: **Rouse**,		

Rel-Model	**Zimm**, **Tanner**, and **Self** are the options. If **Self** is selected, its value should be provided as the next entry	character	Rouse

- **Stochastic differential equation (SDE) parameters**

Parameter	Description	Type	Default
initmode	denotes the initial state of the configuration, **st** is for starting from a known configuration (70% of maximum extension in equilibrium condition or from {q.st.dat, CoM.st.dat} files in non-equilibrium condition. **rst** is for restarting from {q.rst.dat, CoM.rst.dat} files. The files should be located in `data` directory	character	st
tend	end time in units of chain relaxation time	real	10.0
tss	steady state time in units of chain relaxation time	real	5.0
trst	(Only **rst** initmode) time prior to restarting the simulation in units of chain relaxation time	real	0.0
dt	the range of time step size, initial and final values and the method of spacing (**Linear** or **Log**)	real/character	0.01
tol	the convergence criteria for second corrector step of predictor corrector scheme	real	1.e-4
nroots	number of roots used in constructing look-up table	integer	10^6
PrScale	A factor >=1 which increases the precision of look-up table	integer	1
CoM	tracking center of mass	logical	FALSE
CoHR	tracking center of hydrodynamic resistance	logical	FALSE

- **Force parameters**

The force-extension relation is obtained based on flexibility of the macromolecules in solution. For flexible chains, Hookean,, FENE, ILCCP, and RWS are the alternatives. ILCCP stands for Cohen's Pade approximation to inverse Langevin function and RWS is the random walk spring model proposed by Underhill and Doyle (2004). For semi-flexible chains, approxiamtion models to real worm-like chain are used, e.g., WLC_MS, WLC_UD proposed by Marko and Siggia (1995) and Underhill and Doyle (2006), respectively.

Parameter	Description	Type	Default
SPR-Force	spring force-extension relation. Options are: **Hookean**, **FENE**, **ILCCP**, **RWS** for flexible chain and **WLC_MS** or **WLC_UD** for semi-flexible chains	character	Hookean
Truncation	The **FENE** and **ILCCP** force vs. extension can be truncated at specific value of relative extension. The first entry is the method for truncation which can be **None**, **Linear**, or **Cnst**. The second entry is the relative extension $0<qr<1$ after which the truncation is applied.	character/real	None
N_Ks	(Only **RWS** and **WLC_UD** models) number of Kuhn steps per spring	real	U
b	squared maximum dimensionless length of the springs	real	U
EXT-Force	if external force is applied to the ends of the chain (constant force ensemble) followed with the value of the force if the first entry is TRUE	logical/real	FALSE

- **(Non)equilibrium parameters**

Parameter	Description	Type	Default

Flow-Type	the type of external applied flow: **1**: equilibrium, **2**: shear, **3**: uniaxial extension, **4**: biaxial extension, **5**: planar extension	integer	1
nWi	total number of Weissenberg numbers (*Wi*) to be considered.	integer	1
Wi	the range of *Wi*, initial and final values and the method of spacing (**Linear** or **Log**)	real/character	0.0

- **Hydrodynamic interaction (HI) parameters**

Parameter	Description	Type	Default
hstar	*h** the dimensionless hydrodynamic interaction strength	real	0.0
HITens	(Only if *h** is nonzero) hydrodynamic interaction tensor. Options are **RPY** (Rotne-Prager-Yamakawa), **Zimm** (equilibrium pre-averaged Oseen), **OB** (Oseen-Burgers), and **RegOB** (regularized Oseen-Burgers given by Zylka and Öttinger (1989))	character	RPY
DecompMeth	(Only if *h** is nonzero) decomposition method. Options are **Cholesky**, **Lanczos** (Krylov subspace method), or **Chebyshev**	character	Cholesky
ncols	(Only if *h** is nonzero) the number of coloumns in a block of block decomposition methods	integer	1
m	(Only if *h** is nonzero and DecompMeth is **Lanczos**) the first two entries are initial value and upper bound of iteration number in (block)Lanczos method. The third entry specifies if the value of *m* is fixed at initial value in decomposition algorithm	integer/logical	3 15 FALSE
L	(Only if *h** is nonzero and DecompMeth is **Chebyshev**) the first two entries are initial value and upper bound of iteration number in (block)Chebyshev method. The third entry specifies if the value of *L* is fixed at initial value in decomposition algorithm	integer/logical	2 20 FALSE
AveIter-rep	(Only if *h** is nonzero and DecompMeth is **Lanczos** or **Chebyshev**) if the value of iteration needs to be recorded	logical	FALSE
errormin	(Only if *h** is nonzero and DecompMeth is **Lanczos** or **Chebyshev**) the minimum error used in iteration procedure	real	1.e-2
upfactr	(Only if *h** is nonzero and DecompMeth is **Lanczos** or **Chebyshev**) the update frequency of initial value of iteration number in units of number of columns (ncols)	integer	50

- **Excluded volume (EV) parameters**

Parameter	Description	Type	Default
EVForceLaw	the method for calculation of EV force. Options are **NoEV**, **Gauss**, and **LJ**. The latter two options stand for Gaussian potential proposed by Prakash and Öttinger (1999)	character	NoEV
zstar	(Only if EVForceLaw is **Gauss**) the EV potential strength for soft Gaussian potential	real	U
dstar	(Only if EVForceLaw is **Gauss**) the EV potential broadness for soft Gaussian potential. As the second entry, the method of calculating dstar is specified. If **Self** is chosen, the first entry is the value of dstar. If **Kumar** is selected, the first entry specifies parameter *K* in $d^* = Kz^{*1/5}$ proposed by	real/character	1.0 Kumar

	Kumar and Prakash (2003)		
LJ-Par	(Only if EVForceLaw is **LJ**) there are 4 entries which are dimensionless $\epsilon, \sigma,$ truncation and cutoff radii, respectively. Note that ϵ is nondimensionalized with $k_B T$	real	U
minNonBond	(Only if EVForceLaw is not **NoEV**) the minimum distance between the beads along the chain to count the EV potential	integer	1

- **Output parameters**

The program gives the user the option to extract configurational information, e.g., end-to-end distance, average segmental length, rheological material functions from the simulation. Also, the radius of gyration and the average cosine of the neighboring springs can also be obtained automatically using BDpack. Note that if SDE parameters CoM and CoHR are TRUE, the diffusivity of center of mass and center of hydrodynamic resistance are calculated in the program. Any further post processing is possible by using the dumped files {R.equil.dat, CoM.equil.dat} for equilibrium or {R.final.dat, CoM.final.dat} for nonequilibrium condition. The file which starts with R stores the bead to center of mass of all beads of all chains and the other file stores the center of mass position vectors of all chains. The user also has the option to dump configuration of the chains at specific strains.

Parameter	Description	Type	Default
frm-rt-rep	the rate of reporting the time passed in units of relaxation time	real	0.1
frm-rt-pp	the rate of recording data for post processing in the program in units of relaxation time	real	0.002
frm-rt-rst	the rate of recording restart files in units of relaxation time	real	0.1
frm-rt-dmp	the rate of dumping data in units of relaxation time	real	0.1
Conf-anal	if the configurational information is desired	logical	TRUE
Timer-rep	if timing of the program is desired	logical	FALSE
Rg	if the calculation of the radius of gyration is desired	logical	FALSE
cosTh	if the calculation of the cosine of the neighboring springs is desired	logical	FALSE
Dumpstr	if dumping configurational info at specific strains is desired. If the first entry is TRUE, the next entry is the number of strains followed by the value of strains	logical/integer/real	FALSE

Tutorials

This tutorial, provides practical examples in Brownian dynamics simulation of polymeric solutions. It is aimed to clearly show the steps of the simulations using the package. Moreover, it will be shown that BDpack results are in excellent agreement with analytical formulations as well as BD simulations of flexible and semi-flexible chains available in the literature.

Example 1: Equilibrium distribution of segmental length and end-to-end distance:

In this example, an ensemble of 1000 bead-spring chains with 20 springs under equilibrium condition is simulated. The chains are in *theta*-solvent (EV is not present) and are under free-draining condition (HI is not present). The entropic force between the beads is considered to follow *Hookean* force-law. The input file for this example is provided in `BDpack/projects/example_1/`. In this directory, type:

```
$mpirun -np <No. processes> ../../bin/BDpack
```

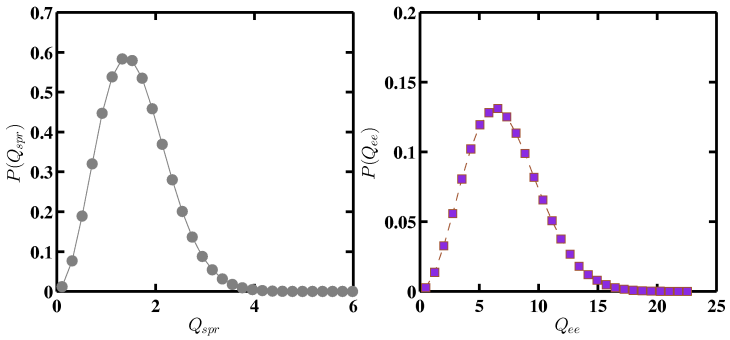
Note that, the number of chains divided by the number of processes should be an even number. Upon successful running of the program, the output files will be generated in the directory `data`. Moreover, for a 20 spring chain with Hookean springs and under theta-solvent and free-draining condition, the theoretical values of end-to-end distance and radius of gyration are known:

File name	Content	Theoretical value
QeeSq	Time averaged value of "ensemble averaged squared end-to-end distance"	60
QsprSq	Time averaged value of "ensemble averaged squared spring connectivity vector"	3
RgSq	Time averaged value of "ensemble averaged squared radius of gyration"	10.48

The probability distribution functions of end-to-end distance and segmental length are commonly used to interpret the simulation results. In order to evaluate pdf, BDpack provides an auxiliary tool (the user compile the utilities before this step as instructed in Installation and Running part). The information regarding springs connectivity vector is stored in `data/q.equil.dat`. In the directory `data` type:

```
$../../../bin/pdf --file q.equil.dat --nCh 200000 --nSeg 20 --nbinSeg 30 --nbinCh 30
```

The PDF results will be stored in `PQee.dat` and `PQspr.dat` for end-to-end distance and spring length, respectively.



As expected a Gaussian distribution is obtained. Moreover, the PDF for segmental length is equivalent to the results of Fig. 1 reported by Herrchen and Öttinger (1997).

Example 2: Simulations of Hookean dumbells in the presence of HI:

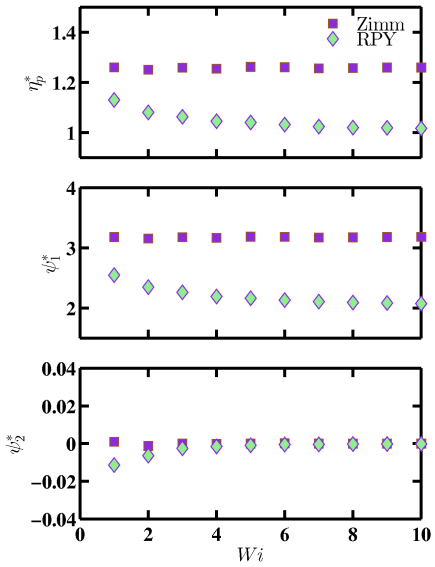
In this example, various material functions for an ensemble of 50000 Hookean dumbells with hydrodynamic interaction in theta-solvent under steady shear flow are predicted. The pre-averaged Oseen-Burgers (Zimm) tensor and Rotne-Prager-Yamakawa (RPY) tensor with h^*=0.15 are assumed. The input file for this example is provided in `BDpack/projects/example_2/`. In this directory, type:

```
$mpirun -np <No. processes> ../../bin/BDpack
```

The table below summarizes the file names and contents which is obtained in this example:

File name	Content
EtavsWi	polymer contribution to viscosity as a function of *Wi*
Psi1vsWi	first normal stress coefficient as a function of *Wi*
Psi2vsWi	second normal stress coefficient as a function of *Wi*

Note that all material functions are in nondimensional form [Saadat and Khomami (2014)]. The exact same simulations were performed by Zylka and Öttinger (1989). BDpack results are in excellent agreement with Figs. 1, 2, and 3 of their article.



Example 3: Fractional extension of DNA in planar elongational flow:

In this example, the transient fractional extension and the steady state value of fractional extension at different flow strength is evaluated and compared with experimental and simulation results reported by Schroeder *et al.* (2004).

The table below summarizes the parameters which were chosen by Schroeder *et al.* (2004) to simulate 7-lambda DNA. Note that the nondimensional value of relaxation time was obtained based on characteristic time of a Hookean spring Sunthar and Prakash (2005).

Parameter	value
λ_1^*	90.9845
*z**	1.52415
*d**	0.5774

First, an initial configuration is generated for an ensemble of 300 chains with 28 springs. In the directory `data` type:

```
$../../../bin/cnfgen --file q.st.dat --nCh 300 --nSeg 28
```

In the next step, the configuration of the chains is further equilibriated for about 8 relaxation times. In `BDpack/projects/example_3/ directory:

```
$mpirun -np <No. processes> ../../bin/BDpack inp1.dat
```

The final configuration is used as the initial configuration for evaluating the transient and steady state fractional extension, In the `BDpack/projects/example_3/ directory:

```
$cp ./data/q.rst.dat ./data/q.st.dat
```

Finally, the `inp2.dat` and `inp3.dat` are used to obtain the transient fractional extension:

```
$mpirun -np <No. processes> ../../bin/BDpack inp2.dat
```

and steady state fractional extension:

```
$mpirun -np <No. processes> ../../bin/BDpack inp3.dat
```