A USER GUIDE FOR OLIM3D PACKAGE FOR COMPUTING THE QUASIPOTENTIAL

SHUO YANG, SAMUEL POTTER, AND MARIA CAMERON

Contents

The package contains two C source files implementing the ordered line integral method with the midpoint quadrature rule for computing the quasipotential for nongradient SDEs in 3D on regular rectangular meshes, and a MATLAB code for visualization:

- `olim3D.c`, a C source code, the version without local factoring,
- `olim3DLFac.c`, a C source code, the version with local factoring,
- `olim3Dvisualize.m`, a MATLAB script visualizing the quasipotential and the MAP.

The SDE should be of the form,

$$d\mathbf{x} = \mathbf{b}(\mathbf{x})dt + \sqrt{\epsilon}dW, \quad \mathbf{x} \in \Omega \subset \mathbb{R}^3, \tag{1}$$

where $\mathbf{b}(\mathbf{x})$ is a continuously differentiable vector field, ϵ is a small parameter, and dW is the standard 3D Brownian motion. The detailed description of the solver is found in [1]. The codes `olim3D.c` and `olim3DLFac.c` are are written so that the computation of the quasipotential is initialized near an asymptotically stable equilibrium $\mathbf{x}^*$ of the

corresponding ODE $\dot{\mathbf{x}} = \mathbf{b}(\mathbf{x})$. The computational domain must be set up so that $\mathbf{x}^*$ is a mesh point.

1. Compiling and running

Any C compiler should be applicable for running `olim3D.c` and `olim3DLFac.c`. At the command line, type

```
\$ clang olim3D.c -lm -O3
\$ ./a.out
```

If the program runs normally with the default settings, you should see something like:

```
chfield = d
x_ipoint: Iindex = 67502848, -1.0000e+00, 0.0000e+00, 0.0000e+00
in param()
in ipoint
in olim()
The boundary is reached:
34308742 accepted points,(511,247,247) is accepted, g=9.9889e-01
NX = 513,NY = 513, NZ = 513, K = 14
cputime of olim() = 5944.45
# of Accepted points = 34308769, umax = 9.9962e-01
Per mesh point: <#1ptupdates> = 1820.87, <#2ptupdates> = 17.81
<#3ptupdates> = 10.26, <#2call> = 49.14, <#3call> = 25.88
ErrMax = 1.5688e-03, ERMS = 4.2508e-04
 Normalized ErrMax = 1.5694e-03, Normalized ERMS = 6.3319e-04
MAP, the initial point: -2.0508e-02, 0.0000e+00, 0.0000e+00
 distance to equilibrium: 9.7949e-01
The last point: -9.9360e-01, 2.1284e-03, 2.1154e-03
 distance to equilibrium: 7.0691e-03
Npath = 12933
```

Here, `Umax` is the maximal value of the quasipotential among Accepted and Accepted Front points. `NX`, `NY` and `NZ` indicate the size of the computational domain; `K` is the update factor; `ErrMax` is the maximal absolute error; `ERMS` is the root mean square error; `Npath` is the number of points in the minimum action path (MAP) from the asymptotically stable equilibrium $\mathbf{x}^*$ to the user-specified point `struct myvector x_ShootMAP`.

2. Available options

The codes `olim3D.c` and `olim3dLFac.c` come with several options for choosing the vector field $\mathbf{b}(\mathbf{x})$. The settings in `olim3DLFac.c` and `olim3D.c` are all the same except that `olim3DLFac.c` contains one extra setting: the radius of the ball for local factoring is assigned in line 37. From now on, we will refer only to `olim3D.c`, and all line numbers below will be given for `olim3D.c`.

The choice of the vector field is done by the variable `char chfield` whose value is assigned at its declaration in line 182. Choose its value out of {`'l','s','r','a','t','u','d'`}. The corresponding vector fields are defined in the function `struct myvector myfield(struct myvector x)` starting on line 252.

- `char chfield = 'a'` defines the linear vector field in Example 1 in [1].
- `char chfield = 'l'` and `char chfield = 's'` define the linear vector fields in Examples 2 and 3 in [1] respectively.
- `char chfield = 't'` defines Tao's nonlinear vector field in Example 6 in [1].
- `char chfield = 'u'` defines Tao's nonlinear vector field in Example 7 in [1].
- `char chfield = 'd'` defines the nonlinear vector fields in Examples 4 and 5 in [1]: the variable `fac` in line 303 is the variable ρ in Examples 4 and 5.

`olim3D.c` and `olim3DLFac.c` have three running options defined in line 37.

- Set `#define CH_SHOOT_MAP 'y'` if you want to compute the quasipotential and shoot a MAP right after that.
- Set `#define CH_SHOOT_MAP 'n'` if you want to compute the quasipotential but do not want to shoot a MAP. You will be able to do it later.
- Set `#define CH_SHOOT_MAP 's'` if you have already computed the quasipotential and want to shoot a MAP now.

3. Output and visualization

Depending on the running option `CH_SHOOT_MAP` there will be two or three output files, with the quasipotential, with the parameters for visualization in MATLAB, and with the MAP. The names of these files are set up in lines 248–250:

```
const char *f_qpot_name = "Qpot.txt"; // output file with the quasipotential
const char *f_MAP_name = "MAP.txt"; // output file with the MAP
const char *f_par_name = "parameters.txt"; // output file with parameters
```

If `CH_SHOOT_MAP` is `s`, the input file with the quasipotential having the same name as the one defined in line 248 is required. Otherwise, no input file is necessary.

The quasipotential and the MAP can be visualized by MATLAB. Once `olim3D.c` produced the three output files above, run the MATLAB script `olim3Dvisualize.m` to plot the level sets of the quasipotential and the MAP.

4. Changing the SDE, the parameters, and the computational domain

The mesh size `NX`×`NY`×`NZ` and the update factor `K` are specified in lines 29–33. We recommend to choose the value of `K` according to a guideline in Table 2 in [1] also written in commented lines 6–10 for your convenience.

The computational domain is set up in the function `void param()` starting on line 389.

The index of the asymptotically stable equilibrium $\mathbf{x}^*$ with respect to which the quasipotential should be computed is specified by the variable `long Iindex` whose value is assigned in `void param()`.

The point from which you want to shoot a MAP back to the equilibrium is specified by the variable `struct myvector x_ShootMAP` whose value is assigned in `void param()`.

If you want to add your own vector field, please do the following steps:

(1) Pick a character to denote your vector field, e.g. `'x'`, and change the value of the variable `chfield` in line 182 to `'x'`:
```
char chfield = 'x';
```
(2) Add `case 'x'` to the operator `switch` in the function `struct myvector myfield(struct myvector x)` starting on line 252 and define your field.
(3) If the quasipotential is available analytically, add `case 'x'` to the operator `switch` in the function `double exact_solution(struct myvector x)` starting on line 318 and define the exact solution. Also, add
```
|| chfield == 'x'
```
right before `)` in line 392.
(4) Calculate the Jacobian of your field and evaluate it at the asymptotically stable equilibrium with respect to which you want to compute the quasipotential. Let `J` be the matrix that you obtained. If its quasipotential decomposition is not apparent, use MATLAB to find the quasipotential matrix `Q`:
```
format long
```

`Q = inv(sylvester(J,J',-2*eye(size(J))))`

The matrix `Q` is defined in `olim3D.c` in the variable `Qmatrix` of type `struct matrixS3`. Since it is symmetric, it suffices to record it entries with $j > i$. Add `case 'x':` to the operator `switch` in the function `void makeQmatrix(char chfield)` starting on line 337 and define `Qmatrix`.

(5) Add `case 'x'` to the operator `switch` in the function `void param()` starting on line 389. You will need to define the computational domain which is a box `[XMIN,XMAX]`$\times$`[YMIN,YMAX]`$\times$`[ZMIN,ZMAX]`, and the index `Iindex` of the mesh point at which the asymptotically stable equilibrium $\mathbf{x}^*$ is located. If you want to shoot a MAP, define the starting point for shooting it in the variable `x_ShootMAP` of type `struct myvector`.

Now you are ready to compile and run the program.

References

[1] S. Yang, S. Potter, and M. Cameron, Computing the quasipotential for non-gradient SDEs in 3D, *submitted to* Journal of Computational Physics, 2018, arXiv: