Notes on using the PARFEDVR GPE solver

Tapio Simula (tapio.simula@gmail.com)

Aarhus, August 2007

1 Quick and dirty start

- have suitable versions of MPI and Fortran 90 installed
- `gtar -xvf PARFEDVR.tar`
- `mkdir DATA`
- `make`

This creates a 4 process executable called `proggis`, which can be tested on a single processor machine with `mpirun -np 4 proggis`. On the LTC cluster `farfar`, the script `paja4` can be submitted to the que using `qsub paja4`. The program finds an approximate ground state for a trapped and interacting BEC by propagating the 3D GPE in imaginary time, starting from a constant initial wavefunction. Output data is dumped in the directory `DATA`.

2 Usage

First read the article by Schneider *et al.* PRE **73**, 036708 (2006), which contains all relevant information about the physics behind the method implemented by this program.

- `globaali.f90` contains (almost) all parameters which need to be altered to run the program for different systems. The variables and their function are commented in the file. The complex time step parameter `dt` should either be chosen real (for real time propagation) or imaginary (for imaginary time propagation). Set `io=.FALSE.` if you do not want to write wavefunctions to file. If you change the number of processes (prosessors) to be employed, you must change the dimensions of the cartesian communicator, `dims`, accordingly (by default $2\times2=4$). The computational space and grid spacing is determined by the variables below ==== setup X grid===== etc.
- `GPfemdvr.f90` is the main driver containing mainly subroutine calls.

- `quadrature.f90, lagrange.f90, kanta.f90 and ptswts.f90` set up the FEDVR basis.
- `ke_lohkot.f90 and mom_lohkot.f90` set up the kinetic energy and momentum operator block matrices.
- `ke_props.f90 and mom_props.f90` set up the kinetic energy and momentum operator block propagators.
- `prop_so2.f90, prop_so4.f90 and propagate.f90` control the split-operator time-propagation.
- `p_kinetic_p.f90,p_kinetic_q.f90, kin_propagation.f90, mom_propagation.f90 and ang_propagation.f90` apply the propagators, corresponding to the kinetic energy and angular momentum operators, to the wavefunction and implement the associated message passing between processes after each propagator is applied.
- `pot_propagation.f90` constructs the external (diagonal) potential and the corresponding propagator, and applies it to the wavefunction. If you wish to change the form of the external potential, this is the file you need to modify. By default only the trap and the nonlinear term are included.
- `normalize.f90` normalizes the wavefunction. This is needed during the imaginary-time propagation.
- `virhenormi.f90` implements the computation of various expectation values and computes the error norm $\sqrt{\int |(H-\mu)\psi|^2 dr}/\langle H \rangle$ which provides a measure of the absolute accuracy/convergence of the ground state solution.
- `seiv_slice.f90 and seiv_wfn.f90` write a column density, a central slice and the whole wavefunction to files every nth time-step. Change n directly in files `prop_so2.f90, prop_so4.f90` (or better, add n to `globaali.f90`).

The 3D vector (wavefunction) is divided into equal areas (one for each process) in the x-y plane and a 2D cartesian communicator is employed for message passing between the processes (processors). The relative nearest neighbour processes are referenced by `top,bot,lft, and rgt` variables.

3 Output and post processing

- Wavefunctions are written to files in 2 ascii column format which, respectively, contain the real and the imaginary part of the wavefunction at each spatial point. The quadrature points and weights are written in separate files `pwx.txt, pwy.txt and` `pwz.txt` such that first column in each file contain points and the second the corresponding weights.

- In Matlab the wavefunctions may be read in, for instance, using the provided scripts `read3D.m` and `read2D.m`. Notice that the wavefunctions are returned on the quadrature grid.
- In imaginary time propagation is used, a control file `DE_wf_abc.txt` is written every nth step containing 2 columns: the first is the error norm and the second one is the chemical potential.
- Additionally, a file `TLP_wf_abc.txt` is written with 3 columns: time, $\langle$angular momentum$\rangle$, $\langle$linear momentum$\rangle$
- There is also some output diagnostics written to the stdout during the execution of the program.

4 Miscellaneous points

- Currently the program needs to be compiled every time parameters are changed.
- The "$\Omega \cdot L$" term, while producing a vortex lattice, has not been tested yet and is not guaranteed to work properly.
- Finding an optimal spatial grid and corresponding propagation time-step for a given problem can sometimes be quite tricky.
- The efficiency of the adaptive time stepping in imaginary time propagation is HIGHLY dependent on the system parameters, and especially so on the variable `tol` (which also depends on n). If unsure, set `tol=0` in order to keep the imaginary time-step constant.
- Loads of disk space could be saved by writing the wavefunctions in binary instead of ascii.
- Real time propagation could be made faster by merging the potential propagation in the beginning and the end of each time step in the split-step method.