GROMACS - Feature #1849

expanded ensemble -- Adaptive Integration Method

11/03/2015 12:10 AM - Christopher Mirabzadeh

Status:	New
Priority:	Normal
Assignee:	Christopher Mirabzadeh
Category:	analysis tools
Target version:	
Difficulty:	uncategorized

Description

I am attempting to include the Adaptive Integration Method (DOI: http://dx.doi.org/10.1103/PhysRevE.69.056704) to the expanded ensemble functions. I have included the files as of version 5.0.4 that I have edited. I'm at the point where my code seems to be working. I would like some feedback on the internal terms that I've chosen to use based on the calculations needed to be made. See my questions below.

Edits I have made thus far:

Names.c, Enums.h -- Added "aim" as an mdp option in the lmc move names.

State.h -- added definitions to the df_history_t structure

Expanded.c -- added AIMChooseNewLambda() method

Typedefs.c -- edited this to init the arrays I created

These are the mdp options I have created that make aim selectable, along with expanded ensemble options:

Lmc-move = aim
Nstdhdl = 1
Nstexpanded = 1

For my AIMChooseNewLambda() method I have borrowed heavily from the ChooseNewLambda() method available in expanded.c.

Algorithm:

1 Store the current potential energy -- dfhist->store_fepot[fep_state] = enerd->term[F_EPOT]

2 Randomly choose a direction +/- lambda -- using metropolis sampler as found in ChooseNewLambda() method

3 fep_state is the current/old configuration of the system

4 lamtrial is the new configuration of the system

5 Get the energy difference between fep_state and lamtrial de = U(lamtria) - U(fep_state)
de = (double)dfhist->store_fepot[lamtrial]-(double)dfhist->store_fepot[fep_state];

6 Trapezoidal rule df = integral from lamtrial to fep_state
df = 0.5*(double)(lamtrial-fep_state)*(1.0/(double)(nlim-1.0))*(dfhist->dfavg[lamtrial]+dfhist->dfavg[fep_state]);

7 If exp(-beta*(de - df)) is greater than random(0,1), then accept and update count

8 Update fep_state

9 Calculate running average
delta = enerd->term[F_DVDL]-dfhist->dfavg[fep_state];

10 Update Free energy estimates
dfhist->dfavg[fep_state] += delta/dfhist->aim_at_lam[fep_state];

Questions:
Am I using the correct term, enerd->term[F_POT], to store the potential energy of the system at fep_state?

Is there another term that has the potential energy difference between lambda states?
Am I using the correct derivative, enerd->term[F_DVDL], for the derivative of the potential energy between lambda states? -- I have already asked this question before. I'm just looking for additional confirmation.
What if someone chooses vdw-lambdas instead of fep-lambdas? Will the code need to use a different term for the derivative?

Thanks

History

#1 - 12/15/2015 06:30 PM - Christopher Mirabzadeh

- File expanded.c added

The AIM function is now working, but it's slow. It's slow because of the output of dhdl.xvg. I need a way to suppress the output of dhdl but still have access to enerd->term[F_DVDL]. The F_DVDL term needs to be calculated every step.

I have attached the new expanded.c code with the AIM function.

#2 - 07/11/2016 08:01 PM - Mark Abraham

- Target version deleted (5.x)

Files

expanded.c	50.9 KB	11/02/2015	Christopher Mirabzadeh
state.h	14.1 KB	11/02/2015	Christopher Mirabzadeh
names.c	8.08 KB	11/02/2015	Christopher Mirabzadeh
typedefs.c	26.6 KB	11/02/2015	Christopher Mirabzadeh
enums.h	13.7 KB	11/02/2015	Christopher Mirabzadeh
expanded.c	50.3 KB	12/15/2015	Christopher Mirabzadeh