I've encountered a problem in thermodynamic integration with GROMACS. I'm interested in calculating of solvation free energy differences between holbenzenes and benzene. To calculate this values I use thermodynamic integration approach (5 point formula).

I use GAFF forcefield with TIP3P water models. To obtain GROMACS topology and coordinate files I convert AMBER topologies (generated with tleap) with amb2gmx.pl script and this topologies look fine.

The result seems to differ in different GROMACS versions. (With the same *.top, *.gro and *.mdp files - attached)

AMBER11 : -0.27 +- 0.16 kcal/mol (consistent with literature)
GROMACS 4.6.1: -0.146 +- 0.09 kcal/mol - good and consistent with AMBER and literature.
GROMACS 4.6.5: 1.333 +- 0.15 kcal/mol - overestimated energy.

I've played with some options (tried adding soft-core potentials and turned on/off couple-intramol option, cut-off VdW algorithm). The results for 4.6.5 are:

no_sc no_intramol 1.27 +- 0.18
no_sc intramol 1.20 +- 0.20
sc no_intramol 1.35 +- 0.21
sc intramol 1.22 +- 0.19

Some observations:
The biggest difference is located at lambda = 0.95 (95% benzene and 5% brombenzene).
At this point ee analysis shows extremely large correlation times (near 1 ns). Don't know whether it's a property of the system or a bug signal.
Contrary to my belief, this has not been fixed by the recent free energy modifications in 4.6/5.0 branches. I'll try to run a git bisect on it tomorrow to find the commit that changed it.

#4 - 06/21/2014 10:49 PM - Michael Shirts
- File test.vac.mdp added

Erik Lindahl wrote:

Contrary to my belief, this has not been fixed by the recent free energy modifications in 4.6/5.0 branches. I'll try to run a git bisect on it tomorrow to find the commit that changed it.

Some diagnosis here (no solution yet):

4.6.1 is problematic, and if it's giving correct answers, it's giving them for incorrect reasons.

So, something is definitely wrong with dhdl in 4.6.1 (not to say that every thing is right with 4.6.5).

First, potential energy differences between all states are the same in 4.6.5 and 4.6.1, so the differences appear to be entirely in dhdl.

With 4.6.1, then dH/dl (coul-lambda) is 2.7688036, whereas the change in energy from coul-lambda=0 to coul-lambda=1 is 5.6992810. In 4.6.5, these two numbers are both the same (5.6992810), as they should be.

In this case, there are no softcore terms because there are no dummy points, so we also have linearity in the vdw interactions. With 4.6.1, then dH/dl (vdw-lambda) is -20.736193, whereas the change in energy from lambda=0 to lambda=1 is -9.8461055. In 4.6.5, these are both the same (-9.8461055), as they should be.

Splitting the components into mass, coul, vdw, and bonded, which means setting mass-lambdas, coul-lambdas, vdw-lambdas, bonded-lambdas, we get:

4.6.1
Running with init-lambda

Total dhdl = -1220.4625

Running with init-lambda-state and setting lambda vector

dhdl
dU (Final-Initial)
together: -1220.4625 -500.18359
	 dhdl(mass) dhdl(coul) dhdl(vdw) dhdl(bond) dU(1-0)
separate: 0.0000000 2.7688036 -20.736193 -1202.4951 -643.52339

Sum of sep: -1220.4624894 (agrees)
Also: -500.18359 does not equal -643.52339

4.6.6
Running with init-lambda

Total dhdl = -1286.9460

Running with init-lambda-state and setting lambda vector

dhdl
dU (1-0)
together: -1286.9460 -723.82746
	 dhdl(mass) dhdl(coul) dhdl(vdw) dhdl(bond) dU(1-0)
separate: -80.304077 5.6992817 -9.8461113 -1202.4951 -723.82746

sum of separate terms: -1286.9460066 (agreement)

Potentials properly agree.
So at this point, we can’t really use 4.6.1 as a reference for what is right, rather why 4.6.5 might not match the expected values, which is problem with a larger scope.

One thing that might help is to try performing the calculation with masses fixed, as it would remove any weird effects that might be improperly being left out (though the bug was filed using single vector, which does include the masses).

One issue that popped up with 4.6.5 is that when separating the interactions, when the masses are changed but the mass-lambdas is not explicitly stated, then the mass term is never printed. There should be some catch for this, but it’s harder. Probably the best way to avoid this is to always print the remainder term (fep-lambdas) if it is nonzero, which is not done right now. However, the mass change still gets included when using just a single lambda. However, then there’s the problem that we don’t know what to put into the header until after generating data. Maybe it should just always be printed out. The alternative would be iterating over all of the atoms and seeing if any of them change mass, which wouldn’t be that hard to do.

I’ve included the mdp I used below for the tests, with various alternatives commented.

#5 - 06/22/2014 07:08 PM - Erik Lindahl
- Assignee changed from Erik Lindahl to Michael Shirts

Michael: If we focus on reproducing experimental values first, exactly what setup would you use (type of calculation, number of lambda-points, separate coul/vdw stages, etc)?

#6 - 06/22/2014 07:37 PM - Michael Shirts
Erik Lindahl wrote:

Michael: If we focus on reproducing experimental values first, exactly what setup would you use (type of calculation, number of lambda-points, separate coul/vdw stages, etc)?

Reproducing experimental values is definitely not what we want to do; we could have serious errors, and get the experimental values right by accident.

I would want to run through alchemical-gromacs.py so that we can compare TI and MBAR. The posted calculation, if it ran with calc-lambda-neighbors would have given different answers with TI and MBAR in 4.6.1, whereas in 4.6.5 those should have agreed (at least, would not have had the SAME errors!)

I think the best thing to do would be to try to get cycle completion in the free energy difference between benzene and bromobenzene. We should get the same free energy differences by mutating in vacuum and in solvent OR in decoupling the two molecule from solvent.

I would try to do it first without the mass changing. I suspect there might be something off in the handling of masses. Since the mass component of the free energy change is separable, a solvation calc and vacuum calc will always have the mass-dependent term cancel out, so the total result will be the same regardless. So first one would get the no-change-in-mass calc working, and then add the mass and see if the solvation and vacuum calculation increased by the same amount.

I would try first with separate coul,vdw,bonded components in the lambda vector. The analysis I did seemed to indicated that in 4.6.5 the result would be the same whether or not it was separated, but separation allows one to make sure that the calculation is being done correctly. Note that agreement can be checked pretty quickly on short configurations, as the dhdl and the dE’s should agree.

Not entirely sure on number of lambda points -- I’d think that 10 would probably be enough (generally is for good configuration overlap), but I don’t know for sure.
Between the attached 4.6.1 and 4.6.5 only the dvdl output at step 0 differs, and it differs significantly. When I generate a 4.5 tpr and run it in 4.5 and 4.6.6 I get a difference of 4-5 kJ/mol in LJ-14, Coul-1-4, LJ and Coul(SR) and dvdl is off by more than 100. LJ and Coul could be noise, but the 14's are certainly off. I am not 100% that 4.5 is correct for this setup.

This system produces very different dvdl results with different Gromacs versions on 1 MPI rank:

- 4.5.7: 1.56925e+02
- 4.6.5: 5.90438e+00
- 4.6.6: 1.04675e+01

With 16 MPI ranks all 4.6 versions are off compared to 1 rank:

- 4.5.7: 1.58925e+02
- 4.6.5: -1.98460e+00
- 4.6.6: -1.98220e+00

I assume the difference between 4.6.5 and 4.6.6 is the switch/shift FE fix. There is clearly a bug in MPI summing dvdl contributions in 4.6. And we should find out why 4.5.7 is different from 4.6.

#8 - 06/23/2014 04:38 PM - Michael Shirts

Berk Hess wrote:

Between the attached 4.6.1 and 4.6.5 only the dvdl output at step 0 differs, and it differs significantly.

Yes, I posted an analysis of this below. 4.6.1 had incorrect summing, and the commit for my fix for this explained how dhdl was being added twice because of an incorrect merge. That's not to say 4.6.5 is right, just that 4.6.1 was wrong in a certain way that 4.6.5 fixed.

When I generate a 4.5 tpr and run it in 4.5 and 4.6.6 I get a difference of 4-5 kJ/mol in LJ-14, Coul-1-4, LJ and Coul(SR) and dvdl is off by more than 100. LJ and Coul could be noise, but the 14's are certainly off. I am not 100% that 4.5 is correct for this setup.

I only looked at 4.6.5, 4.6.1, and release-4-6 with one thread, and 4.6.5 and release-4-6 were the same. I didn't check multithread, and I'll do that now.

I haven't checked differences from 4.5.7 yet either. That may illuminate same.

I assume the difference between 4.6.5 and 4.6.6 is the switch/shift FE fix.

That shouldn't affect the vacuum calc I don't think -- which calculation are you running?

There is clearly a bug in MPI summing dvdl contributions in 4.6.

Seems like it, I will confirm.
I'm getting similar dhdl results between release-4-6 with 1 and 2 MPI threads, i.e. first step is the same and subsequent steps differ as one would expect chaotically. The numbers are consistent with separations.

Note that I am using md to avoid any sort of random number issues with sd (the 4.6.1 errors were independent of md vs sd).

One thread:

```
s0 legend "dH/dx[ff] \times [ff] 0.0000"
s1 legend "xD[ff][H \times [ff]] 0.0000"
@ s2 legend "xD[ff][H \times [ff]] 1.0000"
0.0000 -1286.9460 0.0000000 -723.82746
0.1000 -683.67236 0.0000000 66.933665
0.2000 -320.14240 0.0000000 541.70676
0.3000 123.87421 0.0000000 1114.3770
```

2 threads:

```
s0 legend "dH/dx[ff] \times [ff] 0.0000"
s1 legend "xD[ff][H \times [ff]] 0.0000"
@ s2 legend "xD[ff][H \times [ff]] 1.0000"
0.0000 -1286.9460 0.0000000 -723.82745
0.1000 -683.67236 0.0000000 66.933661
0.2000 -320.14996 0.0000000 541.69703
0.3000 123.83945 0.0000000 1114.3342
```

These were all with the vacuum calculations.

---

I just found out that the parallel differences are due to the SD random numbers (in combination with the short tau_t). So luckily there is no parallel issue!

So the question is now if Erik's recent switch/shift fix also fixes this issue. My results clearly show that 4.6.6 produces results quite different from 4.6.5 (and that's again different from 4.5.7). If someone has time to rerun this system with the current 4-6-release branch, please do so!
#11 - 06/23/2014 05:46 PM - Berk Hess
PS my results are all for the solvated system at lam=0.95, input file: 4.6.5/phbr.prod.0.95308.tpr

#12 - 06/23/2014 06:02 PM - Berk Hess
I just doubled checked and now also 4.6.5 and the lastest 4-6-release match.
So then the only discrepancy is a difference at t=0 in dvdl of about 100 between 4.5 and 4.6.5/6.

#13 - 06/23/2014 09:22 PM - Erik Lindahl
As I wrote above, this is unfortunately not affected by my free energy fixes at all. Maybe not surprising, since those only relate to nonbonded interactions, and I suspect this has to do with the bondeds.

#14 - 06/23/2014 11:30 PM - Michael Shirts
- File solv.test.mdp added

I'm getting very similar results between 4.6.5 and 4.6.6 (release-4-6). This is with the attached mdp, using solvent. It works with sd as well.

4.6.6

```text
@ subtitle "T = 0 (K) \xdl\f{} = 0.9500"
0.0000 -3.1571808
0.1000 -25.723724
0.2000 167.95674
0.3000 1.0883408
```

4.6.5

```text
@ subtitle "T = 0 (K) \xdl\f{} = 0.9500"
0.0000 -3.1573105
0.1000 -25.722805
0.2000 167.94453
0.3000 1.0697937
```

If I use cutoff instead of PME for coulomb, I get the same results out to 300 fs. So it appears that any slight difference is in the nonbonded modifications, but it's really very small.

#15 - 06/24/2014 02:04 PM - Berk Hess
Version 4.6.1 calculates correct dvdl contributions, but the total dVremain/dl is incorrect.
In version 4.6 the kinetic energy contribution to dh/dl ends up in dVremain/dl, which is confusing.
In 4.5 \( \frac{dE_{\text{kin}}}{dl} = -0.5 \left( m_B - m_A \right) v^2 \), which seems to have the wrong sign.
In 4.6, the kinetic contribution to \( \frac{dV_{\text{remain}}}{dl} = 0.5 \left( m_B - m_A \right) v^2 \). This makes 4.6.6 \( \frac{dH_{\text{dl}}}{dl} \) differ by \( 2 \frac{dE_{\text{kin}}}{dl} \) from gmx 4.5.
But in the manual it says that the contribution to \( DG \) is \( -1.5kT \ln(m_B/m_A) \), which seems to have the opposite sign of the formula in 4.6. I don't understand where this comes from. Or should we write the kinetic energy in terms of the momenta: \( p^2/2m \), then the kinetic contribution to \( dH_{\text{dl}} \) will look quite different.

But the \( E_{\text{kin}} \) contribution should drop out in a proper cycle. So then 4.5.7 and 4.6.5 should give the same results and 4.6.1 is wrong. But that doesn't explain the difference of 1.5 kcal/mol observed above. There can be many explanations for this difference though.

#16 - 06/24/2014 02:45 PM - Berk Hess
Filed #1527 for the kinetic energy contribution bug in 4.6 (but I don't expect this had an effect on the final number here).

#17 - 06/28/2014 08:33 AM - Erik Lindahl
- Status changed from In Progress to Feedback wanted
- Priority changed from High to Normal
- Target version changed from 5.0 to 5.x

Oleg - see comments from Michael with suggestions/requests for things to test; right now we are not certain this is a bug!

#18 - 07/01/2014 02:51 AM - Oleg Titov
OK.
As I understood 2 tests are possible:
1. Free energy cycle completion.
2. alchemical-gromacs.py to compare TI and MBAR

I'll try to run both with latest GROMACS with increased number of points for TI (guess 9 would be enough). Did I miss something?

#19 - 07/01/2014 03:01 AM - Michael Shirts
So, it appears the likely problem is that the mass contribution to the free energy change was incorrect -- mass transformations had not been tested as much as other transformations, since the mass-dependent term can always be calculated analytically. If that was the problem, then the 5.0 code will fix the problem and the previous test should give a correct answer.

Oleg Titov wrote:
OK.
As I understood 2 tests are possible:
1. Free energy cycle completion.
2. alchemical-gromacs.py to compare TI and MBAR

I'll try to run both with latest GROMACS with increased number of points for TI (guess 9 would be enough). Did I miss something?
I recalculated the same value in 5.0 version. The result is 1.3 +- 0.1 kcal/mol, so nothing has changed.

Strange that we made a change that should make a difference (the mass term is changed), but the answer didn't change.

Can you recalculate with the bromine / hydrogen not changing in mass? The analytical term cancels out of the cycle anyway. It would be interesting to see if that changes.

Strange that we made a change that should make a difference (the mass term is changed), but the answer didn't change.

I take that back. The error that we fixed should have canceled out in the cycle.

Note that 4.6.1 is NOT an appropriate code base to validate against, as there was a known bug. The question is, is there something about the Amber calculation that is different than the Gromacs calculation? It would be useful to track down those simulation parameters.

Strange that we made a change that should make a difference (the mass term is changed), but the answer didn't change.

I take that back. The error that we fixed should have canceled out in the cycle.

Note that 4.6.1 is NOT an appropriate code base to validate against, as there was a known bug. The question is, is there something about the Amber calculation that is different than the Gromacs calculation? It would be useful to track down those simulation parameters.

I've found a minor error in my topology. Masses for carbon and hydrogen were slightly incorrect (12.00 instead of 12.01 etc), but fixing this had no effect, since this term cancels in full cycle.

OK, I'll try to setup GROMACS to use the same algorithms Amber uses, but unlikely it will change something. The main difference is that AMBER has $H(l) = l^*H_a + (1-l)^*H_b$, not the complicated formula implemented in GROMACS. So it's hard to compare...
intermediate results. I'll recalculate this free energy difference in both AMBER and GROMACS with increased number of points to make sure dH/dl curve is well sampled.

#24 - 07/04/2014 03:59 AM - Michael Shirts

I've found a minor error in my topology. Masses for carbon and hydrogen were slightly incorrect (12.00 instead of 12.01 etc), but fixing this had no effect, since this term cancels in full cycle.

Correct, this is irrelevant to the answers.

OK, I'll try to setup GROMACS to use the same algorithms Amber uses, but unlikely it will change something.

I'm more interested in things like Ewald parameters, cutoffs, etc., not the free energy parameters.

The main difference is that AMBER has $H(l) = l^*H_a + (1-l)^*H_b$, not the complicated formula implemented in GROMACS. So it's hard to compare intermediate results.

If atoms are changing identity, then Gromacs linearly interpolates parameters. Gromacs only uses softcore when atoms are going to endpoints that have zero energy (i.e. changing to dummies).

But one of the reasons that GROMACS uses more complicated formulas is that simpler formulas can introduce non-negligible error at the endpoints. How many lambda points are you using in AMBER? Are you using the endpoints? Have other AMBER calculations been performed that can be compared to this? Even if no-one has done the transformation, I'm sure someone has done the two solvation calculations, whose difference should equal your calculation.

I agree it's not necessary or useful to compare intermediate results between the two programs.

I'll recalculate this free energy difference in both AMBER and GROMACS with increased number of points to make sure dH/dl curve is well sampled.

It looks like you only used 5 points with AMBER -- I would definitely recommend using more points to see if it gets closer. With AMBER, it's particularly important to include additional points, because the linear functional form changes quite a bit at the endpoints.

If you had a chance, computing the full thermodynamic cycle (difference of the solvation of the two molecules) would also help identify any issues.
It took a little more time than I expected. Sorry for long pause.

I'm more interested in things like Ewald parameters, cutoffs, etc., not the free energy parameters.

I've made simulation parameters as equal as possible in both programs. Using 9 points (6 ns simulation per point) I have:

AMBER 11 : -0.1 +- 0.1 kcal/mol
GROMACS 5.0 : 1.4 +- 0.2 kcal/mol

More details (note that AMBER data is in kcal/mol while GROMACS is in kJ/mol):

solvated:

<table>
<thead>
<tr>
<th>lambda</th>
<th>AMBER</th>
<th>GROMACS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01592</td>
<td>0.08198</td>
<td>0.19331</td>
</tr>
<tr>
<td>0.19331</td>
<td>0.33787</td>
<td>0.5</td>
</tr>
<tr>
<td>0.5</td>
<td>0.66213</td>
<td>0.80669</td>
</tr>
<tr>
<td>0.80669</td>
<td>0.91802</td>
<td>0.98408</td>
</tr>
<tr>
<td>0.98408</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

vacuum:

<table>
<thead>
<tr>
<th>lambda</th>
<th>AMBER</th>
<th>GROMACS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01592</td>
<td>0.08198</td>
<td>0.19331</td>
</tr>
<tr>
<td>0.19331</td>
<td>0.33787</td>
<td>0.5</td>
</tr>
<tr>
<td>0.5</td>
<td>0.66213</td>
<td>0.80669</td>
</tr>
<tr>
<td>0.80669</td>
<td>0.91802</td>
<td>0.98408</td>
</tr>
<tr>
<td>0.98408</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Amber and Gromacs inputs attached.

But one of the reasons that GROMACS uses more complicated formulas is that simpler formulas can introduce non-negligible error at the endpoints. How many lambda points are you using in AMBER? Are you using the endpoints? Have other AMBER calculations been performed that can be compared to this? Even if no-one has done the transformation, I'm sure someone has done the two solvation calculations, whose difference should equal your calculation.

I used J Comput Chem 32: 2564–2574, 2011 as reference. Author used only 3 points (500 ps each) per transformation and got results comparable to what I obtain with AMBER.

If you had a chance, computing the full thermodynamic cycle (difference of the solvation of the two molecules) would also help identify any issues.

I guess, It's time for full cycle...
I've computed the full thermodynamic cycle. Again I'm using 9 point formula (6 ns production for each point), decoupling both electrostatics and VdW. Softcore for both EL and VdW.

\[
dG_{solv}(PhBr) = -0.9 \pm 0.3 \text{ kcal/mol}
\]
\[
dG_{solv}(PhH) = -0.7 \pm 0.2 \text{ kcal/mol}
\]
\[
ddG = 0.2 \pm 0.4 \text{ kcal/mol}
\]

Note: direct transformation result : 1.4 \pm 0.2 \text{ kcal/mol}
AMBER direct transformation result : -0.1 \pm 0.1 \text{ kcal/mol}

Michael: Did you get anywhere on this?

Update:

I was experimenting with another thermodynamic cycles and found something interesting.
1. PhBr->PhH and PhH->PhBr transformations do give the same dG estimation and this is correct.
2. I've constructed cycle a) PhBr->PhH, b) PhBr->PhF, c) PhH->PhF. For this cycle dG(a) + dG(c) - dG(b) = 0.5 \pm 0.2 \text{ kcal/mol} , and this difference comes from solvated systems (~0.05 kcal/mol for vacuum cycle and ~0.5 for solvated cycle ). So the problem lies in nonbonded interactions.

For the next step I planned to separate energy components: integrate only change of charges, then only VdW, then bonded and angular terms. I created topologies with all parameters taken from PhBr but different charges (from PhBr, PhH and PhF). However this runs failed! When I start the simulation mdrun terminates with segfault a couple of seconds after initialization finishes. This happens only for solvated system. Vacuum simulation runs fine.

That's weird, but that's something to start with. Should I file a separate bug report? Topologies and coordinates attached.

Oleg Titov wrote:

Update:

I was experimenting with another thermodynamic cycles and found something interesting.
1. PhBr->PhH and PhH->PhBr transformations do give the same dG estimation and this is correct.
2. I've constructed cycle a) PhBr->PhH, b) PhBr->PhF, c) PhH->PhF. For this cycle dG(a) + dG(c) - dG(b) = 0.5 \pm 0.2 \text{ kcal/mol} , and this difference comes from solvated systems (~0.05 kcal/mol for vacuum cycle and ~0.5 for solvated cycle ). So the problem lies in nonbonded interactions.

For the next step I planned to separate energy components: integrate only change of charges, then only VdW, then bonded and angular terms. I
created topologies with all parameters taken from PhBr but different charges (from PhBr, PhH and PhF). However this runs failed! When I start the simulation mdrun terminates with segfault a couple of seconds after initialization finishes. This happens only for solvated system. Vacuum simulation runs fine.

That's weird, but that's something to start with. Should I file a separate bug report? Topologies and coordinates attached.

Only if you can produce it with 5.0.1 - so that we know that the cause of the new(?) issue is not various bugs fixed in 5.0.1.

---

#30 - 09/16/2014 12:09 AM - Oleg Titov

Mark Abraham wrote:

Oleg Titov wrote:

Update:

I was experimenting with another thermodynamic cycles and found something interesting.
1. PhBr->PhH and PhH->PhBr transformations do give the same dG estimation and this is correct.
2. I've constructed cycle a) PhBr->PhH, b) PhBr->PhF, c) PhH->PhF. For this cycle dG(a) + dG(c) - dG(b) = 0.5 +- 0.2 kcal/mol , and this difference comes from solvated systems (~0.05 kcal/mol for vacuum cycle and ~0.5 for solvated cycle). So the problem lies in nonbonded interactions.

For the next step I planned to separate energy components: integrate only change of charges, then only VdW, then bonded and angular terms. I created topologies with all parameters taken from PhBr but different charges (from PhBr, PhH and PhF). However this runs failed!

When I start the simulation mdrun terminates with segfault a couple of seconds after initialization finishes. This happens only for solvated system. Vacuum simulation runs fine.

That's weird, but that's something to start with. Should I file a separate bug report? Topologies and coordinates attached.

Only if you can produce it with 5.0.1 - so that we know that the cause of the new(?) issue is not various bugs fixed in 5.0.1.

Yes, I can produce it with 5.0.1. Filed Issue #1596.

BTW There are no news about 5.0.1 release on the website. :)

11/25/2015
That's weird, but that's something to start with. Should I file a separate bug report? Topologies and coordinates attached.

Just to double check - Did you minimize the simulation energy before running? Could be that something happened that put vdw centers on top of each other.

---

Michael Shirts wrote:

That's weird, but that's something to start with. Should I file a separate bug report? Topologies and coordinates attached.

Just to double check - Did you minimize the simulation energy before running? Could be that something happened that put vdw centers on top of each other.

---

No, I started with PhBr->PhH system equilibrated at lambda = 0.5. All the systems (PhBr->PhH, PhH->PhF, PhBr->PhF) were started from this geometry and ran fine for normal transformation (first 500 ps of trajectories were not used for free energy estimation).

After your question I did a simple check. I ran the same simulation but for PhH molecule with charges taken from PhBr and PhF (both with the same starting geometry and with ex-bromine hydrogen moved to equilibrium distance). The result is:

```
... starting mdrun 'PhBr (resp) in 10A water'
6500000 steps, 6500.0 ps.
Segmentation fault (core dumped)
```

However when I change atom type of "bromine" in molB to something different from molA the simulation runs fine:

```
... starting mdrun 'PhBr (resp) in 10A water'
6500000 steps, 6500.0 ps.
```

NOTE: Turning on dynamic load balancing

...
Nice news!
I've put a stub for issue #1596 to investigate electrostatic and VdW contributions separately.

In file src/gromacs/mdlib/force.c
lines 540-557

ewald_LRcorrection(fr->excl_load[t], fr->excl_load[t+1],
    cr, t, fr,
    md->chargeA,
    md->nChargePerturbed ? md->chargeB : NULL,
    md->sqrt_c6A,
    md->nTypePerturbed ? md->sqrt_c6B : NULL,
    md->sigmaA,
    md->nTypePerturbed ? md->sigmaB : NULL,
    md->sigma3A,
    md->nTypePerturbed ? md->sigma3B : NULL,
    ir->cutoff_scheme != ecutsVERLET,
    excl, x, bSB ? boxs : box, mu_tot,
    ir->ewald_geometry,
    ir->epsilon_surface,
    fnv, *vir_q, *vir_lj,
    Vcorrt_q, Vcorrt_lj,
    lambda[efptCOUL], lambda[efptVDW],
    dvdlt_q, dvdlt_lj);

replaced with

ewald_LRcorrection(fr->excl_load[t], fr->excl_load[t+1],
    cr, t, fr,
    md->chargeA,
    (md->nChargePerturbed | md->nTypePerturbed) ? md->chargeB : NULL,
    md->sqrt_c6A,
    (md->nChargePerturbed | md->nTypePerturbed) ? md->sqrt_c6B : NULL,
    md->sigmaA,
    (md->nChargePerturbed | md->nTypePerturbed) ? md->sigmaB : NULL,
    md->sigma3A,
    (md->nChargePerturbed | md->nTypePerturbed) ? md->sigma3B : NULL,
    ir->cutoff_scheme != ecutsVERLET,
    excl, x, bSB ? boxs : box, mu_tot,
    ir->ewald_geometry,
    ir->epsilon_surface,
    fnv, *vir_q, *vir_lj,
    Vcorrt_q, Vcorrt_lj,
    lambda[efptCOUL], lambda[efptVDW],
    dvdlt_q, dvdlt_lj);

With this modification I finally get 0.0 + 0.2 for all free energy cycles.

ddG,kcal/mol
phbr->phf(full)  1.065 +- 0.104
phbr->phf(el)   -0.131 +- 0.004
phbr->phf(vdw)   1.373 +- 0.007
phbr->phh(full)  1.413 +- 0.170
phbr->phh(el)   -1.382 +- 0.010
phbr->phh(vdw)   1.060 +- 0.015
phh->phf(full)  -0.364 +- 0.091
phh->phf(el)   1.255 +- 0.012
phh->phf(vdw)   0.317 +- 0.006

However, the ddG estimate for PhBr -> PhH is still 1.4 +- 0.2 kcal/mol which is in disagreement with AMBER results.

Is it possible that there are more issues with this pointer logic?
Oleg Titov wrote:

BTW There are no news about 5.0.1 release on the website. :)

Fixed, thanks.

Oleg Titov wrote:

Nice news!

I've put a stub for issue #1596 to investigate electrostatic and VdW contributions separately.

In file src/gromacs/mdlib/force.c
lines 540-557
[...]
replaced with
[...]

With this modification I finally get 0.0 + 0.2 for all free energy cycles.

[...]

However, the ddG estimate for PhBr -> PhH is still 1.4 +- 0.2 kcal/mol which is in disagreement with AMBER results.

Is it possible that there are more issues with this pointer logic?

Oh yes. :) git grep n."Perturbed shows lots of possibilities.

I'd also suggest trying cutoff-scheme = Verlet, to probe whether (remaining) differences exist in non-bonded implementations, or elsewhere.
Mark Abraham wrote:

I'd also suggest trying cutoff-scheme = Verlet, to probe whether (remaining) differences exist in non-bonded implementations, or elsewhere.

The differences DO exist.

I switched to 5.0.2 version. "Fix" for issue #1596 was applied as before.

<table>
<thead>
<tr>
<th>ddG, kcal/mol</th>
</tr>
</thead>
<tbody>
<tr>
<td>phbr-phf(group)</td>
</tr>
<tr>
<td>phbr-phh(group)</td>
</tr>
<tr>
<td>phh-phf(group)</td>
</tr>
<tr>
<td>phbr-phf(Verlet_cpu)</td>
</tr>
<tr>
<td>phbr-phh(Verlet_cpu)</td>
</tr>
<tr>
<td>phh-phf(Verlet_cpu)</td>
</tr>
<tr>
<td>phbr-phf(Verlet_gpu)</td>
</tr>
<tr>
<td>phbr-phh(Verlet_gpu)</td>
</tr>
<tr>
<td>phh-phf(Verlet_gpu)</td>
</tr>
</tbody>
</table>

I've got significant performance loss for Verlet scheme on CPU because of plain C kernels (16 h of calculation on 8 core CPU resulted in 2.5 ns trajectory for such a tiny system). I'll try to fix this issue and calculate longer version of trajectories for better estimations. GPU calculations are still in process.

#37 - 10/12/2014 03:56 PM - Oleg Titov

GPU calculations are complete.

The results are:

<table>
<thead>
<tr>
<th>ddG, kcal/mol</th>
</tr>
</thead>
<tbody>
<tr>
<td>phbr-phf(Verlet_gpu)</td>
</tr>
<tr>
<td>phbr-phh(Verlet_gpu)</td>
</tr>
<tr>
<td>phh-phf(Verlet_gpu)</td>
</tr>
</tbody>
</table>
I'm still experiencing performance issues with Verlet CPU code, but it seems to give the same estimations as GPU.

So, for now, for PhBr->PhH transformation I have 3 estimations:

- AMBER11: -0.1 ± 0.1 kcal/mol
- GROMACS with group cutoff-scheme: 1.4 ± 0.2 kcal/mol
- GROMACS with Verlet cutoff-scheme: -0.4 ± 0.2 kcal/mol

May be I want too much, and 1st and 3rd estimations differ only due to some numerical issues and differences in implementation?

#38 - 10/12/2014 05:28 PM - Michael Shirts

May be I want too much, and 1st and 3rd estimations differ only due to some numerical issues and differences in implementation?

The first and third calculations are almost certainly statistically indistinguishable. The fact that verlet and group calculations give results that are this far off is problematic. I'm handling a newborn baby, so I won't be able to look at these in detail right now. My first guess would be something with the dispersion interactions, but I can't tell for sure. At least we should be able to tell the difference between group and verlet in the same code.

If you start from the same configuration, with the only difference in the cutoff type? what do the .dhdl files and .edr files look like over the first 10 output points?

#39 - 10/12/2014 08:01 PM - Oleg Titov

Yes, the only difference was the cutoff type.

Group:

\[
\begin{align*}
0.0000 & \quad 3.6820323 \\
0.1000 & \quad -60.792847 \\
0.2000 & \quad -68.538551 \\
0.3000 & \quad 67.601639 \\
0.4000 & \quad -108.946340 \\
0.5000 & \quad 78.883484 \\
0.6000 & \quad -12.955036 \\
0.7000 & \quad -22.846668 \\
\end{align*}
\]
### Verlet:

<table>
<thead>
<tr>
<th>Time (ps)</th>
<th>Potential</th>
<th>Kinetic En.</th>
<th>Total Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0000</td>
<td>5234.904785</td>
<td>171.204041</td>
<td>-1585.685547</td>
</tr>
<tr>
<td>0.1000</td>
<td>4417.827148</td>
<td>172.22896484</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.2000</td>
<td>5109.294434</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.3000</td>
<td>4795.032227</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.4000</td>
<td>5150.375000</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.5000</td>
<td>5131.946289</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.6000</td>
<td>5162.003906</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.7000</td>
<td>4911.013672</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.8000</td>
<td>4892.047363</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.9000</td>
<td>4933.041016</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>1.0000</td>
<td>5045.150879</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
</tbody>
</table>

### dhdl:

<table>
<thead>
<tr>
<th>Time (ps)</th>
<th>Potential</th>
<th>Kinetic En.</th>
<th>Total Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0000</td>
<td>5.1170769</td>
<td>97.674988</td>
<td>-1220.524414</td>
</tr>
<tr>
<td>0.1000</td>
<td>-11.776363</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.2000</td>
<td>-63.417797</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.3000</td>
<td>19.785820</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.4000</td>
<td>41.371532</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.5000</td>
<td>1.1633410</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.6000</td>
<td>104.64513</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.7000</td>
<td>-27.479208</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.8000</td>
<td>20.021889</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.9000</td>
<td>20.021889</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>1.0000</td>
<td>48.864159</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
</tbody>
</table>

### edr:

<table>
<thead>
<tr>
<th>Time (ps)</th>
<th>Potential</th>
<th>Kinetic En.</th>
<th>Total Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0000</td>
<td>-13201.57129</td>
<td>-2332.405781</td>
<td>-1626.365479</td>
</tr>
<tr>
<td>0.1000</td>
<td>-12678.53418</td>
<td>-2326.631836</td>
<td>-1623.436646</td>
</tr>
<tr>
<td>0.2000</td>
<td>-12271.642578</td>
<td>-2331.041992</td>
<td>-1620.280273</td>
</tr>
<tr>
<td>0.3000</td>
<td>-11828.53418</td>
<td>-2335.119141</td>
<td>-1621.891357</td>
</tr>
<tr>
<td>0.4000</td>
<td>-11827.109375</td>
<td>-2340.092773</td>
<td>-1625.506958</td>
</tr>
<tr>
<td>0.5000</td>
<td>-12094.136719</td>
<td>-2327.474609</td>
<td>-1630.506958</td>
</tr>
<tr>
<td>0.6000</td>
<td>-12848.688477</td>
<td>-2340.705078</td>
<td>-1632.126709</td>
</tr>
<tr>
<td>0.7000</td>
<td>-12935.219727</td>
<td>-2336.125977</td>
<td>-1631.546631</td>
</tr>
<tr>
<td>0.8000</td>
<td>-11566.465820</td>
<td>-2343.884766</td>
<td>-1628.037231</td>
</tr>
<tr>
<td>0.9000</td>
<td>-12103.160156</td>
<td>-2336.125977</td>
<td>-1628.356445</td>
</tr>
<tr>
<td>1.0000</td>
<td>-1826.365479</td>
<td>-2332.405781</td>
<td>-1626.365479</td>
</tr>
</tbody>
</table>

### eedr:

<table>
<thead>
<tr>
<th>Time (ps)</th>
<th>Potential</th>
<th>Kinetic En.</th>
<th>Total Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0000</td>
<td>5.1170769</td>
<td>97.674988</td>
<td>-1220.524414</td>
</tr>
<tr>
<td>0.1000</td>
<td>-11.776363</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.2000</td>
<td>-63.417797</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.3000</td>
<td>19.785820</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.4000</td>
<td>41.371532</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.5000</td>
<td>1.1633410</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.6000</td>
<td>104.64513</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.7000</td>
<td>-27.479208</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.8000</td>
<td>20.021889</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.9000</td>
<td>20.021889</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>1.0000</td>
<td>48.864159</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
</tbody>
</table>

### dhdl:

<table>
<thead>
<tr>
<th>Time (ps)</th>
<th>Potential</th>
<th>Kinetic En.</th>
<th>Total Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0000</td>
<td>5.1170769</td>
<td>97.674988</td>
<td>-1220.524414</td>
</tr>
<tr>
<td>0.1000</td>
<td>-11.776363</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.2000</td>
<td>-63.417797</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.3000</td>
<td>19.785820</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.4000</td>
<td>41.371532</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.5000</td>
<td>1.1633410</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.6000</td>
<td>104.64513</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.7000</td>
<td>-27.479208</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.8000</td>
<td>20.021889</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>0.9000</td>
<td>20.021889</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
<tr>
<td>1.0000</td>
<td>48.864159</td>
<td>172.72289648</td>
<td>-1856.532959</td>
</tr>
</tbody>
</table>
Is it ok that total energy is nearly two times different for the same system?

#40 - 10/13/2014 04:54 AM - Michael Shirts
That seems very strange that the potential energies are so different from even the initial configuration. Is this after any equilibration, or literally starting from the same frame for both simulations?

Oleg Titov wrote:

Yes, the only difference was the cutoff type.

Group:
dhdl:
[...]
edr:
[...]

Verlet:
dhdl:
[...]
edr:
[...]

Is it ok that total energy is nearly two times different for the same system?

#41 - 10/13/2014 09:33 AM - Oleg Titov
This is first 10 output points (output each 100 steps with 1 fs step) of simulation started with the same .top and .gro. .mdp differs only by cutoff-scheme. .gro contains system Phh->PhBr, previously equilibrated at lambda = 0.5 (and lambda = 0.5 for this simulation) for 500 ps in some older version of GROMACS (4.6.4 or 5.0).

#42 - 10/14/2014 02:15 PM - Mark Abraham
Can you upload some tarballs of your latest grompp inputs, .tpr and resulting .log files, please Oleg?

#43 - 10/14/2014 07:10 PM - Oleg Titov
- File latest.tar.bz2 added

Mark Abraham wrote:

Can you upload some tarballs of your latest grompp inputs, .tpr and resulting .log files, please Oleg?
Sure. Uploaded.

Files

<table>
<thead>
<tr>
<th>File Name</th>
<th>Size</th>
<th>Date/Time</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>test.7z</td>
<td>1.52 MB</td>
<td>02/20/2014</td>
<td>Oleg Titov</td>
</tr>
<tr>
<td>test.vac.mdp</td>
<td>1.73 KB</td>
<td>06/21/2014</td>
<td>Michael Shirts</td>
</tr>
<tr>
<td>solv.test.mdp</td>
<td>1.36 KB</td>
<td>06/23/2014</td>
<td>Michael Shirts</td>
</tr>
<tr>
<td>prod.vac.templ.mdp</td>
<td>1.43 KB</td>
<td>07/15/2014</td>
<td>Oleg Titov</td>
</tr>
<tr>
<td>prod.templ.mdp</td>
<td>1.41 KB</td>
<td>07/15/2014</td>
<td>Oleg Titov</td>
</tr>
<tr>
<td>prod.vac.templ.mdin</td>
<td>276 Bytes</td>
<td>07/15/2014</td>
<td>Oleg Titov</td>
</tr>
<tr>
<td>prod.templ.mdin</td>
<td>288 Bytes</td>
<td>07/15/2014</td>
<td>Oleg Titov</td>
</tr>
<tr>
<td>phbr.equil.segfault.gro</td>
<td>111 KB</td>
<td>09/15/2014</td>
<td>Oleg Titov</td>
</tr>
<tr>
<td>phbr-phf.segfault.top</td>
<td>10.7 KB</td>
<td>09/15/2014</td>
<td>Oleg Titov</td>
</tr>
<tr>
<td>prod.0.5.segfault.mdp</td>
<td>1.41 KB</td>
<td>09/15/2014</td>
<td>Oleg Titov</td>
</tr>
<tr>
<td>latest.tar.bz2</td>
<td>5.47 MB</td>
<td>10/14/2014</td>
<td>Oleg Titov</td>
</tr>
</tbody>
</table>