GROMACS - Feature #1653

Feature # 1332 (In Progress): Supporting multiple end states instead of just A and B

Decide how to represent multiple lambda states in the .top file and how to parse them

12/04/2014 09:26 PM - Michael Shirts

Status:	New
Priority:	Normal
Assignee:	Michael Shirts
Category:	mdrun
Target version:	
Difficulty:	uncategorized

Description

As part of the discussion for having multiple lambda states, we need to decide how to have those lambda states represented in the .top files in as clear a way as possible.

History

#1 - 12/04/2014 09:28 PM - David van der Spoel

Gerrit Groenhof and Serrena Donnini have put a lot of work in this already!

#2 - 12/04/2014 09:31 PM - Michael Shirts

David van der Spoel wrote:

> Gerrit Groenhof and Serrena Donnini have put a lot of work in this already!

This is why I am posting this, so that we can get more discussion happening. Thomas Ullman and Carsten Kutzner have also been working on it, and it would be good to get the architecture out in the open to make sure that everyone is talking about the same thing and duplicated effort is minimized.

#3 - 12/04/2014 09:44 PM - Michael Shirts

I realize that other people have implemented solutions not in the main code already, but I thought I'd add my two cents, since I don't know them.

I'm going to argue that it would be best to have a solution that is global, but can be adapted to site-specific changes generally.

I think that the best way to do this in topologies is instead of stacking new parameters vertically repeat parameter sets

Instead of:

[atoms]
1 types A types B parameters A Parameters B
2 types A types B parameters A Parameters B

it becomes:

[atoms]
1 types A Parameters A
1 types B Parameters B
2 types A parameters A
2 types B parameters B

For example, in the old style, we would have

```
[ atoms ]
  ;   nr      type         resnr residue  atom   cgnr      charge       mass          typeB          chargeB
  massB
       1   opls_135       1       TRPB     CB       1       -0.1800     12.0110     opls_135m      0.0000    12.011
0
       2   opls_140       1       TRPB    HB1      1         0.0600     1.0080       opls_140m      0.0000      1.0
080
       3   opls_140       1       TRPB    HB2      1         0.0600     1.0080       opls_140m      0.0000      1.0
080
```

New style would be

```
[ atoms ]
  ;   nr      type        resnr residue  atom   cgnr      charge       mass
       1   opls_135       1      TRPB     CB       1      -0.1800    12.0110
       1   opls_135m    1      TRPB     CB       1        0.0000    12.0110
       2   opls_140       1      TRPB    HB1      1        0.0600     1.0080
       2   opls_140m    1      TRPB    HB1      1        0.0000     1.0080
       3   opls_140       1      TRPB    HB2      1        0.0600     1.0080
       3   opls_140m    1      TRPB    HB2      1        0.0000     1.0080
```

This will scale to as many end states are needed.

This can be clearly extended up to however many states are required.

However, this does not say which of the states move together. For example, we might want all of the 1st states to move together, all of the 2

Then either through index group functionality or through an extra command ([atom sites])? For example,

```
[ atom sites]
   1  1 2 3
   2  4 5 6
```

Would mean there are two "sites" defined through these indices.

Sites would be required to carry the same number of states for all atoms in the site. This would create a single topology structure; if one state would have a different number of atoms, then there would be dummy atoms in that place (which could potentially be created automatically).

This would mean that gromacs would create one lambda group for each site -- likely with N lambda vectors (bonded, coul, electrostatic, etc) for

So if there were 5 states for each of two sites, there would be 10 lambda vectors, two sets, with 5 states each. These could be set explicitly in the mdp or through some defaults.

I think this can express things cleanly and clearly.

Other people want to post what they are thinking about?

#4 - 12/06/2014 05:09 PM - Gerrit Groenhof

> Other people want to post what they are thinking about?

The suggestion of Michael could work for our constant pH lambda dynamics. We had considered this solution before, but at that time did not have constraints on the lambda's yet, which are essential to make it work I think.

With the proposed setup, we would handle titrating site with multiple end states, such as histidine as follows:

Feed the topologies in sequence either interwoven and atom after atom, as Michael suggests or simply as topology blocks, which I use for the example here:

HISA
HISB
HISH

In state A all interactions with the system would be off, while in state B all interactions would be on. Each topology block (which together represent the residue with multiple endstates) has a single lambda, so that we have three lambda coordinates in total for this residue. During the dynamics we constraint the sum of the lamda's to 1, so that all states can be visited, but no two end-states are fully interacting at the same time.

The proposed solution would simplify the input significantly, because one only needs to provide the A and B state per microscopic protonation state.

There are two problems for constant pH still.

The first is that we would need a way to figure out which blocks (and lambdas) belong together and describe a single titrating site with multiple endstates. I think this can be handled by additional input.

The second is that we don't know what to do with the coordinates of the non-interacting states during the dynamics. What can happen namely is that the non-interacting atoms move into a very unreasonable configuration, which would affect the samping in the lamdba subspace of that residue. In our current implementation we use additional columns with atom types, charges, parameters, etc. for the various microscopic protonation states. We thus only have one copy of each atom in the system, which avoids problems with the configurational dynamics of non-interacting states.

We have been discussing this among ourselves, but I agree that it is better done as open as possible.

#5 - 12/06/2014 06:56 PM - Thomas Ullmann

On Michael's suggestion for the topology:

With regard to the format of the .top file, I'm open to anything that works, but it would be very nice to keep backwards compatibility, so that people can still use the current topology format. The parsing of the topology file should not be the biggest problem.

I would like to have the possibility to define sites on the level of residue topology files, as proposed under issue #1332, and on the level of the system topology. The residue topology allows for an efficient treatment of many sites of the same kind in a system, like protonatable aminoacids.

For the system topology, as Gerrit already wrote, we need to have information on which lambda site and which form/state of the site an atom belongs to. This can be easily solved by adding two additional integer numbers for each atom: a site ID and a form ID. Global Lambdas, as needed by Michael would then simply realizable by giving all atoms involved the same site ID.

The chain and the molecule to which a site belongs is already defined by the position of the atoms in the topology. If one needs to connect parts of different chains/molecules into a site, one would either need some patch mechanism, as CHARMM has it, and/or to allow sites to span multiple chains/molecules in the topology from the outset.

Dual- and single-topology parts should both be possible. A third ID could indicate whether an atom belongs to the single-topology part (only parameters change, but the coordinates are the same for each atom) or to a dual-topology part (parameters and coordinates differ between forms. Atoms in the dual topology part would not be required to be present or filled in with dummy atoms in other forms also. How to attach a decoupled dual-topology part, or constrain groups of lambda values is a separate, almost fully independent issue. My implementation of the variables sites should allow easy attachment of custom variants, without having to care about the infrastructure around the lambda site. I will write a bit about the implementation that I'm working on under issue #1652.

This solution would be very general and allow to choose the best solution for any individual site/problem/method.

Where to put the lambda values

The lambda values are coordinates just as those of the atoms. Therefore, I think that it would be conceptually more sound not to put them in the .mdp/parameter file. It should be noted here, that the number of sites can be very large for realistic applications (thousands). A small protein of ca 100 aminoacids will typically already have about 30 protonatable sites. Separate lambda coordinate/trajectory files would perhaps be the easiest and cleanest solution, also in terms of easy implementation of a visualization solution, e.g., for VMD, which I also would like to have.

#6 - 12/08/2014 06:39 AM - Michael Shirts

> With regard to the format of the .top file, I'm open to anything that works, but it would be very nice to keep backwards compatibility, so that people can still use the current topology format. The parsing of the topology file should not be the biggest problem.

It should be easy to have code that can take the horizontally defined alternate parameters and read them in; it just needs to switch based on the number of entries.

> I would like to have the possibility to define sites on the level of residue topology files, as proposed under issue #1332, and on the level of the system topology. The residue topology allows for an efficient treatment of many sites of the same kind in a system, like protonatable aminoacids.

Agreed, and I think that repeated atom lines can happen in both cases.

> For the system topology, as Gerrit already wrote, we need to have information on which lambda site and which form/state of the site an atom belongs to. This can be easily solved by adding two additional integer numbers for each atom: a site ID and a form ID. Global Lambdas, as needed by Michael would then simply realizable by giving all atoms involved the same site ID.

Correct, my description included a site ID, i.e. molecules that move in lambda space together. Its just a question of where it would go. I think we are in complete agreement there.

Could you define what you mean by a form ID? We need to be more precise in definitions. I'm guessing it has to do with alternate variants of a sites (HIS, HID, HIE involving the same atoms?), but it's not clear.

> The chain and the molecule to which a site belongs is already defined by the position of the atoms in the topology. If one needs to connect parts of different chains/molecules into a site, one would either need some patch mechanism, as CHARMM has it, and/or to allow sites to span multiple chains/molecules in the topology from the outset.

If site ID's are also assigned in the index file, then this would allow spanning between molecules/chains. They wouldn't HAVE to be assigned in the index file, only one needed to span, and then there would just need to be a validity check that the two sets are compatible.

I think Berk has enabled some code that allows site-spanning indexing in the topology after the [system] command, which could also be called.

> Dual- and single-topology parts should both be possible.

I'm not entirely convinced that supporting the statistical mechanics for both parts is necessary. Just because there are two ways to solve a problem doesn't mean that one should do both of them. Code simplicity is very important.

> A third ID could indicate whether an atom belongs to the single-topology part (only parameters change, but the coordinates are the same for each atom) or to a dual-topology part (parameters and coordinates differ between forms).
> Atoms in the dual topology part would not be required to be present or filled in with dummy atoms in other forms also.

What are the statistical mechanical consequences of not having dummies for all atoms?

> How to attach a decoupled dual-topology part, or constrain groups of lambda values is a separate, almost fully independent issue.

Well, it's not <u>independent</u> issues, since it needs to be addressed for things to work. :) Measure twice, cut once.

My approach has been to always leave bonds on and turn other parameters off, since then you don't have atoms floating around. If chiral, then other bondeds may need to be left on (then would need gas phase calculation to cancel) Perhaps there's a better approach, though? There's a fundamentla problem of calculating a free energy difference between molecules with different numbers of atoms.

> My implementation of the variables sites should allow easy attachment of custom variants, without having to care about the infrastructure around the lambda site. I will write a bit about the implementation that I'm working on under issue #1652.

> This solution would be very general and allow to choose the best solution for any individual site/problem/method.

I'm not certain that this is always the case. Too many methods -> unsupportable code. We should try to be as general as possible without unduly increasing code complexity.

> The lambda values are coordinates just as those of the atoms.

Agreed.

> Therefore, I think that it would be conceptually more sound not to put them in the .mdp/parameter file.

Well, there are several "where" questions to answer. The .mdp, contains the initial values of the lambda.   The initial values have to be either in the .mpd, the .top, or the .gro file, or a new file type. the .mdp seems like the logical place.   The other where is where they should be output. I think they probably should be in the .trr; right now, they are really only in the .log.

> It should be noted here, that the number of sites can be very large for realistic applications (thousands).

Just to check, evolving independently, so there are 1000's of independent state variables?

> A small protein of ca 100 aminoacids will typically already have about 30 protonatable sites. Separate lambda coordinate/trajectory files would perhaps be the easiest and cleanest solution, also in terms of easy implementation of a visualization solution, e.g., for VMD, which I also would like to have.

I think that putting them in the .trr would be the cleanest -- all state information in the same place. If only one lambda, of course that section would be omitted.

#7 - 12/08/2014 06:45 AM - Michael Shirts

> Feed the topologies in sequence either interwoven and atom after atom, as Michael suggests or simply as topology blocks, which I use for the example here:

If atom by atom, the order doesn't really matter as long as the numbers line up.

> HISA
> HISB
> HISH
>
> In state A all interactions with the system would be off, while in state B all interactions would be on. Each topology block (which together represent the residue with multiple endstates) has a single lambda, so that we have three lambda coordinates in total for this residue.
>
> The proposed solution would simplify the input significantly, because one only needs to provide the A and B state per microscopic protonation state.

Or could write tools to "protonate" a .top, adding additional atoms to each protonizable residue so that logic can be abstracted to a different part of the code, rather than putting it all in the parser to handle multiple sites? Note that this could be in the .rtp as well, so that a HIS_PROT would contain all three.

> There are two problems for constant pH still.
>
> The first is that we would need a way to figure out which blocks (and lambdas) belong together and describe a single titrating site with multiple endstates. I think this can be handled by additional input.

I think this comes down to site ID's?

> The second is that we don't know what to do with the coordinates of the non-interacting states during the dynamics. What can happen namely is that the non-interacting atoms move into a very unreasonable configuration, which would affect the samping in the lamdba subspace of that residue. In our current implementation we use additional columns with atom types, charges, parameters, etc. for the various microscopic protonation states. We thus only have one copy of each atom in the system, which avoids problems with the configurational dynamics of non-interacting states.

If one can assume rigid rotor, isn't the solution just to leave bonded on?

#8 - 12/10/2014 01:20 AM - Thomas Ullmann

Could you define what you mean by a form ID? We need to be more precise in definitions. I'm guessing it has to do with alternate variants of a sites (HIS, HID, HIE involving the same atoms?), but it's not clear.

I guess you already had the right idea. The different forms of a site, like the different protonation forms of the histidine sidechain are simply numbered consecutively. The form ID is then simply a number. Implementation-wise I have also the higher-level IDs (a number for the molecule, a number for the chain and a number for the site) in the corresponding object for easy sorting of the lambda site objects in lists or for output.

I'm not entirely convinced that supporting the statistical mechanics for both parts is necessary. Just because there are two ways to solve a problem doesn't mean that one should do both of them. Code simplicity is very important.

Yes, but I think it is really not redundant to have both variants. As an example consider a relative binding free energy calculation for different small-molecule ligands in a receptor. The ligands may have a common, very rigid core structure but different substituents, with different interaction properties attached to different positions of the core structure. In this case it could be advantageous to treat the core in a single-topology setup, but the substituents in a dual topology setup. Exchange of an amino acid sidechain with a very different other sidechain, or one with different protonation is likely also such a case. I think, having this possibility might be very advantageous for sampling.

What are the statistical mechanical consequences of not having dummies for all atoms?

The number of atoms does not really change, does it? In the above example, the atoms of one ligand are decoupled from the receptor environment and the atoms of the other ligand are in turn coupled to the receptor environment, but all the atoms remain in the simulated system ...

>How to attach a decoupled dual-topology part, or constrain groups of lambda values is a separate, almost fully independent issue.
Well, it's not independent issues, since it needs to be addressed for things to work. :) Measure twice, cut once.

I already have an idea beyond restraint potentials, but I'm not yet sure whether it will work. But in any case, the problem is on the radar :-)

My approach has been to always leave bonds on and turn other parameters off, since then you don't have atoms floating around. If chiral, then other bondeds may need to be left on (then would need gas phase calculation to cancel) Perhaps there's a better approach, though? There's a fundamentla problem of calculating a free energy difference between molecules with different numbers of atoms.

Gerrit pointed out, I think correctly, that this approach is not entirely clean, because the decoupled part is not really fully decoupled with this solution and still exerts an influence on the rest of the system.

> My implementation of the variables sites should allow easy attachment of custom variants,
> without having to care about the infrastructure around the lambda site.
> I will write a bit about the implementation that I'm working on under issue #1652.
> This solution would be very general and allow to choose the best solution for any individual site/problem/method.
>
> I'm not certain that this is always the case. Too many methods -> unsupportable code. We should try to be as general as possible without unduly increasing code >complexity.

I put quite a bit of effort into ensuring maintainability with a letter/envelope + inheritance solution. Adding a new variant will not require any changes to or knowledge of the rest of the code. Any additional implementation has to fulfill a number of requirements to be accepted by the invariant rest of the code. That is, it has to provide all the necessary functions etc.

> Just to check, evolving independently, so there are 1000's of independent state variables?

Yes!, simulating such systems is an important goal for us. Especially bioenergetic systems and other biomolecular systems that a number of people here are highly interested in can be really large.

> I think that putting them in the .trr would be the cleanest -- all state information in the same place. If only one lambda, of course that section

> would be omitted.

Hm, but wouldn't that break standards/cause compatibility problems for all sorts of tools, for example VMD?

#9 - 12/10/2014 01:59 AM - Michael Shirts

Thomas Ullmann wrote:

> > I think that putting them in the .trr would be the cleanest -- all state information in the same place. If only one lambda, of course that section would be omitted.
>
> Hm, but wouldn't that break standards/cause compatibility problems for all sorts of tools, for example VMD?

OK, that's a good point. I guess there would have to be a separate file for now. It would be great to eventually have it as an optional section of a .trr eventually, because as you point out, it's just another coordinate for the system. .trr has a lambda (single variable) entry now. Then the patch for other codes will be very simple (check for presence of something, if there skip N entries). But a separate output file is a decent intermediate step.

> I guess you already had the right idea. The different forms of a site, like the different protonation forms of the histidine sidechain are simply numbered consecutively. The form ID is then simply a number. Implementation-wise I have also the higher-level IDs (a number for the molecule, a number for the chain and a number for the site) in the corresponding object for easy sorting of the lambda site objects in lists or for output.

OK, that makes sense. If an atom can have multiple states, some of these states are associated with each other.

10 atoms, each of which have three states.

5 of these atoms are site 1 (same site ID)
5 of these atoms are site 2 (same site ID)

Site one atoms have 4 possible variants that change together (each of these are a form ID)
Site two atoms have 4 possible variants that change together (each of these are a form ID)

Note that if each form has the same number of atoms (full atoms or dummies) then the form ID can be explicit; the order would be sufficient to specify the form. If they can have different numbers of atoms, then the form needs to be specified in some way.

Also, can you upload example files? I think that would be much clearer than descriptions in words (examples are always clearer than words).

The idea of the ID's for molecules is to allow different molecules with the same topology to change independently, correct?

> Yes, but I think it is really not redundant to have both variants. As an example consider a relative binding free energy calculation for different small-molecule ligands in a receptor. The ligands may have a common, very rigid core structure but different substituents, with different interaction properties attached to different positions of the core structure. In this case it could be advantageous to treat the core in a single-topology setup, but the substituents in a dual topology setup. Exchange of an amino acid sidechain with a very different other sidechain, or one with different protonation is likely also such a case. I think, having this possibility might be very advantageous for sampling.

I'll think about it, but I think we need to lay out the statistical mechanics carefully so that the same framework treats both cases with out too many conditionals. Perhaps we need to have white papers on implemetations as well as just this thread? I'm a strong believer in describing things before implementing things and discussing them among the people involved - it saves time in the long run. The more people see subtle statistical mechanics derivations, the more likely any problems can be identified.

Remember, someone is going to have to approve this in gerrit eventually anyway, so better to document it first now :)

> I already have an idea beyond restraint potentials, but I'm not yet sure whether it will work. But in any case, the problem is on the radar :-)
>
> Gerrit pointed out, I think correctly, that this approach is not entirely clean, because the decoupled part is not really fully decoupled with this solution and still exerts an influence on the rest of the system.

OK, then we need to have a clear description of the "right" way to do it.

> I put quite a bit of effort into ensuring maintainability with a letter/envelope + inheritance solution. Adding a new variant will not require any changes to or knowledge of the rest of the code. Any additional implementation has to fulfill a number of requirements to be accepted by the invariant rest of the code. That is, it has to provide all the necessary functions etc.

I think it's not just the code, it's the physical validity of the output. Two different integrators could be compatible in terms of the form of data (new coordinates) but be incompatible in terms of the distributions of energies/configurations produced. That's the scenario that I'm most concerned about, and the hardest thing to check (though we have some ensemble validity tools that will help.

> > Just to check, evolving independently, so there are 1000's of independent state variables?

> Yes!, simulating such systems is an important goal for us. Especially bioenergetic systems and other biomolecular systems that a number of people here are highly interested in can be really large.

Sounds good.

Internally, just having a large set of lambda arrays should be fine. The sophistication is in the setting up which lambda goes to which variable, which is set up by site ID's, form ID's etc. For MC, then we have a (static) list of values for each lambda.

it COULD be set up in a more object oriented way so that lambdas for each site are hidden, but then we lose some ability to quickly address and combine them all, which is likely to lead to some real code slowdown in the future -- I'd like the complicated object oriented stuff to be in the definitions of the indices.

That's actually a topic for the other thread, so I'll repost there.

#10 - 12/11/2014 06:46 PM - Carsten Kutzner

Couldn't we use the TNG format instead of a separate file / a .trr to store the system trajectory including the lambdas?

#11 - 12/11/2014 08:13 PM - Michael Shirts

Carsten Kutzner wrote:

> Couldn't we use the TNG format instead of a separate file / a .trr to store the system trajectory including the lambdas?

Sounds good to me - As long as the TNG support is in by then, then putting lambda coordinates in the new format is the better thing to do.

#12 - 12/11/2014 08:56 PM - Michael Shirts

> Sounds good to me - As long as the TNG support is in by then, then putting lambda coordinates in the new format is the better thing to do.

Note that in any program that includes lambda dynamics, there is some history dependent information. Most methods have associated weights with each lambda state (if descritized) or a weighting function (if continuous). This information is frequently history dependent if the program is adaptive (as methods like wang-landau are). It seems that there should be space in the TNG format for this history dependent weight information.

Note that the output should ideally support continuous and discrete states. It's not clear if just stating the value all lambdas is the best way to support discrete states, or if they should also have some number (the 9th state) or vector (1st state in 1st lambda, 4th state in 2nd lambda) instead/as well.

#13 - 12/11/2014 09:49 PM - Thomas Ullmann

For the output, I think that it would be most general to state the value of the lambda variable for each form of each state, even if that is a bit redundant for the discrete case where only one form per site has lambda != 0 at any given point of the simulation. In this way, one can have a single, simple solution for reading the output for further analysis.

Internally, there would be a state vector with one lambda variable for each form of each variable site.
I'm also not yet sure whether it would be better to have a single, large array for this and similar tasks, or a smaller array / vector for each site as member of the object representing the site. Both approaches can even be combined by
a) storing a pointer to the global array
b) a pointer to the relevant subsection of the global array or
c) a separate smaller array with the site-specific entries
as member of each object that represents a variable site.

As a memory saving and still fast solution, I already have data structures that encapsulate arrays (1,2,3,4-dimensional with different internal storage) with variable lengths of the dimensions, which are also ready for use on GPUs. In this way, one can have a different number of entries for each site. The two dimensional array could look like this, where the site index would run in dimension 1 and the form index in dimension 2. The different lengths in dimension 2 reflect the variable number of forms of each site:

```
-                   dim2
            ----------------->
          / xxxxxxxxxxxxxxxx
   dim1  / x
        / xxx
       / xxxxxx
      / xxx
     V xxxxxxxx
```

#14 - 12/12/2014 04:34 PM - Thomas Ullmann

Note that in any program that includes lambda dynamics, there is some history dependent information.
Most methods have associated weights with each lambda state (if >descritized) or a weighting function (if continuous).
This information is frequently history dependent if the program is adaptive (as methods like wang-landau are).

I'm not sure where to put best put the histogram and the current estimate of the density of states of a Wang-Landau simulation.
In the past, for GMCT, I had separate files for them and wrote them only at the end of the simulation on user request.
Is there a need for having a trajectory of the Wang-Landau data beyond checkpointing for restarting simulations?

Couldn't we use the TNG format instead of a separate file / a .trr to store the system trajectory including the lambdas?

Sounds good to me - As long as the TNG support is in by then, then putting lambda coordinates in the new format is the better thing to do.

I have no practical experience with the .tng file format yet, so what I write is based on the two corresponding papers
and a quick glance at the sources.
Adding additional data is possible by design and as judged from the comments in tngio.h also full-precision/lossless
trajectory writing is possible. So it could really be the best option to add the lambda values, the velocities in
lambda space and possible further data as additional trajectory data blocks to the .tng file.

#15 - 03/24/2015 03:29 PM - Magnus Lundborg

I haven't looked exactly at what to store, and the exact data structure, but I cannot foresee any problems storing it in a TNG file. I'm not sure exactly when I'll have time to look at it, though. If anyone else is very eager to add it I can probably be of some assistance at least.

#16 - 03/24/2015 03:42 PM - Thomas Ullmann

Sounds great, thanks for offering help! I will ask for your opinion on my solution as soon as I arrive at this point, which should not be too far in the future.

#17 - 03/24/2015 08:10 PM - Michael Shirts

I've been neglectful on commenting because I haven't had as much time to develop, but I hope to have more time soon. So I'd just echo the ideas that the sooner that people who have been working on similar ideas (myself, Berk, Gerrit, Serena) get a chance to look at the code, the less work it will be for everyone. I would urge putting it into the main repository so it can be tested and discussed easily. I can see there are certain circumstances that there are parts of the code might need to be kept private, but I expect that in most cases, the gain from getting many eyes looking at it will outweigh most other considerations.

The more the details of the implementation are accessible and made easy to understand, the more time you will be saving for yourself and others in the future. If your ideas are out there, then people will be more likely to incorporate them into their current efforts -- if not, you'll have to do the adoption yourself :)

#18 - 03/24/2015 09:40 PM - Thomas Ullmann

I'm planning to prepare some draft commits in the near future.
However, you can also have a look at the entire code code on our
own git server in Jülich at any time. Just send your ssh key(s)
to Ivo Kabadshow and you will have access.
Serena and I could already talk directly about the new code when
she was here in Göttingen with Gerrit. Recently, during their last
visit we also discussed the issue of how and if the PME can also
be used efficiently for lambda dynamics. Gerrit wanted to add that
as a Redmine issue too, but maybe he forgot. As far as I know,
Berk is already involved in trying that.

The first three drafts are planned to be:

1. Our current FMM - GROMACS interface.
2. generally usable data structures to be used in the other code parts
3. The current "lambda_site" module that treats the variable sites during
the simulation. The module will receive energies from the FMM (or a
PME workaround), perform the immediate lambda-dynamics related tasks
(calculation of forces in lambda space, coordinate transformations,
net charge constraints needed under infinite periodic boundary conditions,
time integration in lambda space)

Also a first version of the FMM could soon end up in a draft commit.

The new topology with local alternatives will take some more time.

#19 - 10/13/2015 08:52 PM - Gerrit Code Review Bot

Gerrit received a related patchset '1' for Issue #1653.
Uploader: Thomas Ullmann (thomas_ullmann@web.de)

Change-Id: I68814357b5959ea691bb508fd13aeb8eba4f1fff
Gerrit URL: https://gerrit.gromacs.org/5204

#20 - 10/27/2015 06:18 PM - Gerrit Code Review Bot

Gerrit received a related patchset '1' for Issue #1653.
Uploader: Thomas Ullmann (thomas_ullmann@web.de)
Change-Id: Ic80a3e9de7c54440a2734a6d798479c452383e5e
Gerrit URL: https://gerrit.gromacs.org/5250

#21 - 07/11/2016 08:06 PM - Mark Abraham

- Target version deleted (5.x)

#22 - 07/24/2017 12:11 PM - Gerrit Code Review Bot

Gerrit received a related patchset '5' for Issue #1653.
Uploader: Thomas Ullmann (thomas_ullmann@web.de)
Change-Id: gromacs~master~Ic80a3e9de7c54440a2734a6d798479c452383e5e
Gerrit URL: https://gerrit.gromacs.org/5250

#23 - 06/05/2018 03:58 PM - Gerrit Code Review Bot

Gerrit received a related patchset '1' for Issue #1653.
Uploader: Thomas Ullmann (thomas_ullmann@web.de)
Change-Id: gromacs~master~Ib042563a6df9b275a6e349a2b5c7434585ac32ac
Gerrit URL: https://gerrit.gromacs.org/7978