GROMACS - Task #2898

Naming common variables

03/19/2019 01:18 PM - Artem Zhmurov

Status:	New	
Priority:	Low	
Assignee:		
Category:	documentation	
Target version:		
Difficulty:	simple	

Description

One of the most complicated problem to a programmer is to figure out how to name a variable, a function, a class or a method. Renaming afterwards can also be rather frustrating. To make the process of naming easier, below are the list of the common variables that anyone can encounter in code they are working at. We can add this list to developers documentation, once it will be filled.

```
Example variables
-----------------

* numAtoms -- total number of atoms in the system.
* numSomething -- total number of something.
* computeVirial -- whether or not virial should be computed.
* virialScaled -- walue of the scaled virial tensor.
* compute_dHdLambda -- whether or not free energy perturbation is being
  evaluated.
* dHdLambda -- free energy perturbation value.
* numIterations -- number of iterations in LINCS projection correction.
* expansionOrder -- order of expansion used to compute inverse matrix in LINCS.
```

History

#1 - 03/19/2019 02:58 PM - Kevin Boyd

One suggestion - I'd naively interpret computeVirial and compte_dHdLambda as a verb, so that seems to me more like a function call than a bool. Maybe something like shouldComputeVirial

#2 - 03/19/2019 04:07 PM - Artem Zhmurov

Kevin Boyd wrote:

> One suggestion - I'd naively interpret computeVirial and compte_dHdLambda as a verb, so that seems to me more like a function call than a bool. Maybe something like shouldComputeVirial

I think it should be clear from the fact that these variables are boolean. The 'b' prefix was abandoned due to the same reasons. Besides, adding 'should' will make already long variable names even longer.

#3 - 03/20/2019 08:58 AM - Eric Irrgang

Is the idea that these are variable names that frequently get reused and should therefore have documented conventions or that these are data that don't have a clear owner yet? Or is this a to-do list of variables with clear ownership in the code that are currently insufficiently documented in the headers?

#4 - 03/20/2019 11:21 AM - Mark Abraham

Naming conventions serve the purpose of helping humans understand the code that they read. The original Hungarian notation was for things like "positionX" and "sizeX" which later got warped into Microsoft-style "dwIndex" which everyone agrees is much less useful.

Artem Zhmurov wrote:

> Kevin Boyd wrote:
>
> > One suggestion - I'd naively interpret computeVirial and compte_dHdLambda as a verb, so that seems to me more like a function call than a bool. Maybe something like shouldComputeVirial

> I think it should be clear from the fact that these variables are boolean. The 'b' prefix was abandoned due to the same reasons. Besides, adding 'should' will make already long variable names even longer.

In some places we use e.g. doCoulomb and doCptPullCoordHist. "do" is brief, clear, and a verb which we could reserve for booleans (but that is yet another something unique for this project for devs to remember). That leads us back to bComputeVirial, for which I see no defects and clear benefits (see above comment about Hungarian notation).

I would suggest we regard "derivative of energy with lamda" as something which has a well known abbreviation, like others things such as the Message Passing Interface, Particle-Mesh Ewald, and Open Compute Language which also have abbreviations that have internal capitalizations (MPI, PME, OpenCL). As we already have the style of g_usingMpi, numPmeDomains, and writeOclBuildLog for those, computeDhdl or bComputeDhdl are usable, clear, and consistent.

A list of such examples is probably good to add to the developer guide, but I suggest we transform Artem's "Common variables" into "example variables"

#5 - 03/20/2019 01:43 PM - Artem Zhmurov

- Description updated

Eric Irrgang wrote:

> Is the idea that these are variable names that frequently get reused and should therefore have documented conventions or that these are data that don't have a clear owner yet? Or is this a to-do list of variables with clear ownership in the code that are currently insufficiently documented in the headers?

The idea was to have a list of names (e.g. virialScaled) or conventions (e.g. numSomething) that frequently appear in the code.

Mark Abraham wrote:

> A list of such examples is probably good to add to the developer guide, but I suggest we transform Artem's "Common variables" into "example variables"

Done. I intentionally made the part of the issue into pre-formated text, so we can edit it on the go.

#6 - 03/26/2019 11:46 PM - Erik Lindahl

Just my $0.02, but I strongly agree with the points that we should optimize variable names to aid understanding of the algorithm the first time someone reads the code.

- LtsBVryRstrictWthAbrv (let's be very restrictive with abbreviations). We read code 50-100x more than we write it, and complex expressions will anyway benefit from being split into multiple lines. We will need a few abbreviations (such as "number of"), so let's pick e.g. "num" - but that means we should never use "n" or "number".

- In particular for settings and function parameters, I think the solution to the boolean question is to replace them with enum classes in most cases. The second version here is much clearer, and as a bonus it's impossible to swap parameters by mistake:

```
callMyFunction(true,true,false);
callMyFunction(energy::evaluate,force::evaluate,virial::dontEvaluate);
```

- Let's have a clear decision whether we put adjectives before (scaledVirial) or after (virialScaled). I have a slight preference to just follow natural language, but the most important thing is to strive for consistency.

... and in particular: I don't want us to necessarily enforce this for all new external code (because we already have way too many hard conditions that raises the barrier for people to contribute), but it would be helpful to at least have a more extensive reference of what perfect code should look like (then we can be tough on each other internally :-)