UAlg
UNIVERSIDADE DO ALGARVE

Optimization of Molecular Dynamics Simulation Code and Applications to Biomolecular Systems

David M. Bowman

Advisor: Dr. Paulo Martel, Faculty of Science and Technology,
University of the Algarve

Dissertation for the Degree of Doctor in Bioinformatics through the Centre for Biomedical Research, University of the Algarve, FCT

Faro, December 2015

University of the Algarve
Faculty of Science and Technology

Optimization of Molecular Dynamics Simulation Code and Applications to Biomolecular Systems

David M. Bowman

Advisor: Dr. Paulo Martel, Faculty of Science and Technology,
University of the Algarve

Dissertation for the Degree of Doctor in Bioinformatics through the Centre for Biomedical Research, University of the Algarve, FCT

Faro, December 2015

“Imagination is more important than knowledge.”

- *Albert Einstein*

“Doing nothing is infinitely faster than doing something.”

- *Unknown*

Contents

Optimization of Molecular Dynamics Simulation Code and Applications to Biomolecular Systems

Declaração de autoria do trabalho

Declaro ser o autor deste trabalho, que é original e inédito. Autores e trabalhos consultados estão devidamente citados no texto e constam da listagem de referências bibliográficas incluída.

Acknowledgements

The author would like to express his thanks to Dr. Paulo Martel, Centre for Biomedical Research, and the University of the Algarve, Faro, Portugal for his confidence that this project was feasible and for his support during the project.

The author is also grateful to members of the Computational Systems Biology Group, Center for Neuroscience and Cell Biology, UC-Biotech, University of Coimbra led by Dr. Armindo Salvador; the Biological Chemistry Group, Department of Chemistry, University of Coimbra led by Dr. Maria Joao Moreno; and Dr. Claudio Soares, New University of Lisbon, Lisbon, Portugal for stimulating discussions about the project and suggestions on approaches to validation of the methodology.

The Biological Chemistry Group, Department of Chemistry, University of Coimbra also provided substantial computational resources that made the free energy studies possible. The author would like to thank Virtual Strategy, Inc., Boston, MA for funding this research and supporting the patent application.

Dedication

I dedicate my dissertation work to my children, Jonathan, Hannah, and Luis. You are the great joys of my life.

General Introduction

Biomolecular simulation methods are a powerful tool to study the physico-chemical properties of biomolecules and their mechanisms and to describe extremely short-lived molecular phenomena otherwise difficult to describe.[1, 2] We have come a long way since the first protein simulations in the early 70's, both in terms of theoretical methods and the computer hardware used. Simulation of the molecular dynamics (MD) of solvated proteins with atomic detail requires the use of systems with the order of tens to hundreds of thousands of atoms. Particle simulations with detailed molecular potentials can be extremely heavy in systems of this size, and until recently only relatively short simulation times (10-100 ns) were accessible with most computer systems available. Improvements in computer algorithms and hardware have made possible, in a few cases, to reach timescales of the order of the microsecond. However, many fundamental biomolecular processes, like protein folding, may take place over periods of a millisecond or more, meaning that speedups in computer time of over three orders of magnitude are still required.

The advent of parallel computing and massively parallel hardware has had significant impact in the attainable simulation times[3]. Single thread, single core and single processor code is the fundamental unit of execution in whatever environment it is executed whether it be single processor, OpenMP, MPI or GPUs. The key objective of this project is to improve performance of MD simulations at this level to effectively raise the performance bar on all simulations in multiple system architectures and topologies. The software suite GROMACS[4] is widely regarded has one of the fastest single-processor MD simulation engines available. This is mostly due to several clever optimization techniques including hard-coded assembly code for different machine architectures and specialized inner loops for different types of non-bonded interactions.[5,6] Even with these optimizations, simulation times of the order of the micro- to millisecond are still prohibitive for all but the smallest of molecular systems running on the most powerful computer architectures available[7]. The present proposal aims to enhance the GROMACS software by a combination of different procedures, including the use of binary operations and incremental computation techniques adapted from compiler optimization theory. Using these methods, we expect to speed up the GROMACS code that is used to calculate forces by a factor of 2-3 times. This will mean being able to simulate biomolecular processes for both longer periods of time and for larger simulations. It will also benefit investigators with less expensive, less powerful computers to perform simulations that were previously beyond the capabilities of their computational environments. Projects may also be done with less expensive computer systems that use substantially less electricity.

Theory and Methods

An initial analysis of GROMACS using the available development tools was performed on Intel based platform and as expected the primary performance bottleneck is in the calculation of nonbonded interactions typically accounting for over 90% of total computation time. This code has already been rewritten by hand in assembly language using the latest Intel computer instructions. An examination of both the single precision and double precision assembly language implementations revealed that there was only *one* computer instruction in the double precision implementation that was not required in the routines that process Lennard-Jones[8] and reaction field nonbonded interactions. This created a fundamental challenge for this project. The code to compute these forces had already been highly optimized for many years by top researchers in the world. Where would the major improvements come from? The only possibility was to avoid computation at the core of GROMACS.

Application of Compiler Optimization Generation Techniques

One of the fundamental assumptions of this project was that it was possible to apply advanced complier optimization and code generation techniques at the application and simulation level to the GROMACS application to substantially increase performance.

The principle object of compiler optimization techniques and indeed high performance computing is to avoid as much computation as possible. This requires knowledge of the application and the simulation involved and this analysis can only be fully done at runtime. Optimization techniques frequently move calculations outside of internal processing loops to perform calculations only one/few times. This provides two benefits: the calculations are simply not performed inside inner loops and the results that are calculated before entering the innermost loops highly cacheable.

In addition to avoiding computation or computing results only once in many cases binary or integer instructions may be used within inner processing loops to replace sequences of floating point computation. These instructions in many cases run in less than one CPU cycle. This project takes full advantage of this technique.

This project targeted the Intel instruction set but the conversion to integer math and the simulation of intermediate code optimization will be generally applicable to other platforms and high performance computing (HPC) applications.

Trading Memory for Calculations

The core methodology used in this project is the substitution of small multi-layer caches with partial results to be assembled with non-floating point or few floating point instructions at runtime. This effectively allows the replacement of 'hot spots' of computation with small lookup tables with direct lookups and assembly or incremental calculation. This allows calculations to be computed upon entry into the application based on the simulation definition and reused when needed in the future.

Implementation Approaches

General Purpose vs. Specific Purpose

Software programs, libraries and even CPUs, GPUs are designed to solve a general class of problem. They are designed to support a range of user requirements. ANSI and ISO standards exist for general purpose computing for data formats, computer languages, database, communications and other functionality. Additionally industry and academic organizations have formed groups to establish de facto standards. These standards are required to solve a broad range of problems across many disciplines. For example the IEEE 754 floating point standard[9] supports applications with a very large number of significant digits and 3-4 digit exponents.
The programming languages such as C and C++ are built upon the underlying hardware standards and have their own general purpose standards. Both these hardware and computer language standards are not designed for a specific problem or even a class of problems. They are designed to be used to solve all possible problems

that might be needed in any problem domain, industry or organization. They are general purpose tools for constructing software and problem domain solutions.

It is almost never a requirement to support exponents in the range of -308 to +308 with 15 significant digits or even greater ranges. Nothing in reality corresponds to these values. Molecular dynamics and other applications use a limited portion of the IEEE 754 standard and the capabilities of processing these data types with ANSI C or C++. If 1.0 is 1 nanometer then a distance of 10^{-308} or 10^{308} nanometers is never useful to users of molecular dynamics software. Users of molecular dynamics software and other applications use only a small fraction of the standard format. Molecular dynamics simulations typically use the smallest boxes and fewest atoms possible. This constrains the space and time in which the simulation runs to some number of nanometers and some number of time steps. Time steps are generally in the range of 1 femtosecond to 1 millisecond. Similarly the experimental and theoretical constants that are available (or meaningful) have a limited range and precision. Thus the range of values within the floating point standard that are actually used is relatively small.

This observation is true for virtually every class of computer application that exists. The largest floating point format of the IEEE 754 standard supports values that if 1.0 = 1 meter then the maximum value is greater than the size of the universe and the smallest value does not correspond to anything known. Application developers do not use the entire range of values in the same application. This means that for a specific problem within a specific problem class, such as molecular dynamics optimizations to avoid many floating point calculations are possible by trading memory for calculations and incrementally assembling the results or performing very few calculations at runtime. This is the approach taken in this project.

Alternative 1: Manual Modification of MD Software to Exploit Incremental Non-Computational Optimization

In order for an application program to be optimized using these techniques it must have already been analyzed for computational 'hot spots' using the standard tools such a gprof or Visual Studio. All coding, algorithmic and design must have been optimized using both developer and user expertise. If this has not been done, it should. This is just good software development practice.

When no further optimizations are possible use incremental calculation techniques to pre-calculate and cache partial results for a high percentage of the key functions and equations.

A key objective of the implementation process is to enable the incremental noncomputational method to self-adapt to the problem being solved. For molecular dynamics software this is done for the SPECIFIC simulation and the hardware environment where it is being run. This requires the identification of the data ranges for key variables used in calculations at the application/simulation/run level. The incremental noncomputational method *does not support* the full range of floating point numbers. In order to minimize memory use for the method the developer must identify runtime variables that are actually invariant at runtime for the SPECIFIC simulation or function based on what is being run. This enables both the elimination of computation and reduces the amount of runtime assembly or incremental calculation required. This also reduces greatly the number of computer instructions and moves much of the complexity of the algorithm, equation or function to a table or application cache initialization. Using the data profile/signature of the simulation at runtime and information about ranges and other application specific information can enable the use of integer, logical and memory instructions instead of floating point computations. At program start or on first use, the incremental caches/tables are initialized for future use. Values are calculated once and retrieved on subsequent accesses.

Generally the application developer must do profiling of the data that is actually used to understand the subset of the IEEE 754 standard that the software actually needs. This combination of user expertise and runtime data profiling enables the precision of the data to be reduced and the range of data to be limited significantly reducing the size of the data tables used in incremental computation.

The incremental non-computation method also requires the software to know how much CPU cache is available in order to perform well. If too high a percentage is used then cache misses will occur and performance will be degraded rather than improved.

Performance decreases of three or more times the computed performance have been observed when the limits of the available CPU cache is not respected.

After the ‘hot spots’ are converted to use the developed algorithm, the application must be tested to show that the results are ‘equivalent’ and still meet user requirements. This process was done statistically for the inner non-bonded kernel routines for the GROMACS software running on one thread/core/processor.

Alternative 2: Profile Guided or Automatic Optimization

Recent advances in software optimization techniques, such as profile guided optimizers (PGO) include the ability to produce a profile or runtime application signature for the execution of a program automatically on a given platform with a specific set of input files for use with the application. These application profiles/signatures can then be used to compile/build a higher performance version of the application based on actual runtime knowledge (Figure 1).

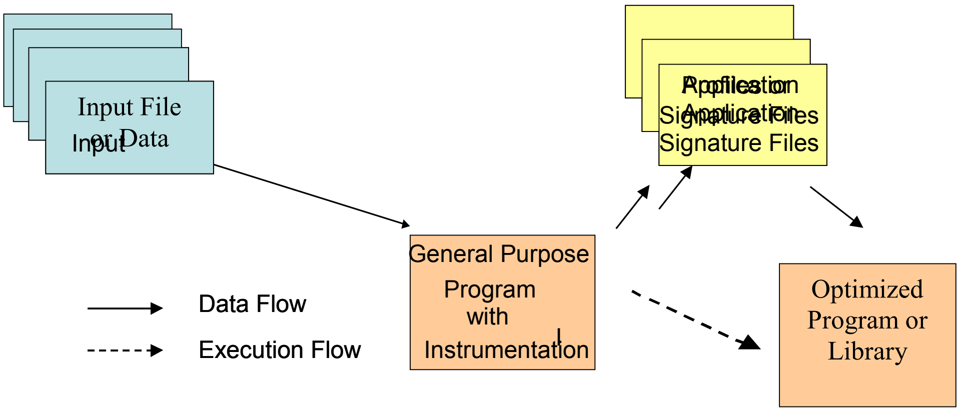


Figure 1 Profile Guided Optimization (PGO)

This approach requires knowledge of the data range and precision required for each variable that is used to build precalculated results and that the precalculated results for the used to support the various subexpressions/expressions be small enough to be highly cacheable. The method is highly dependent on the data map of the application and the degree to which the application contains subexpressions or expressions within the inner loops that are suitable for the developed algorithm. In theory it is possible to develop an application development tool based on an existing program profiler that could track the data ranges and precisions of the data values used in optimized subexpressions and expressions during an actual execution run of the program (perhaps a short execution). The tool could then suggest to the developer areas that could benefit from the developed algorithm providing suggested data ranges and precisions. The developer could then select the suggestions that were of interest and modify the data ranges and precisions if needed. These suggested and modified changes could then be used by the application development tool to automatically generate modified application source code to define and initialize the tables for the precomputed results and to update the source code for the inner loop to minimize computation within the inner computational loops by retrieving the pre-calculated results from one of the tables. Currently, Microsoft[10], Intel[11] and GNU[12] have program guided optimization (PGO) functionality that creates instrumented versions of an application and allows the application to be 'trained' to produce higher performance code. The developed methodology and algorithm has not yet been incorporated into any software development tool or PGO option.

It is also possible that optimizations could be made dynamically *during* execution in an instrumented application. This could be performed without a detailed understanding of the application or library function using an incremental noncomputational method with a failsafe fall through to calculate the result of the subexpression or expression if the precalculated lookup was not within range. This approach would need to be hardware platform and operating environment aware since there is substantial difference in performance between CPU manufacturers, CPU families, cache memory available, memory architecture and other aspects of the operating environment. Operating system support would also likely be required.

Abstract

The performance of molecular dynamics (MD) software such as GROMACS is limited by the software's ability to perform force calculations. The largest part of this is for nonbonded interactions such as between water molecules and water molecules and solute. The determination of nonbonded interactions may account for over 90% of the total computation and real time of a simulation. The objective of this project is to greatly improve the performance of force calculations for nonbonded on a single core/processor. By doing this it is possible to raise the bar on all simulations that can be performed by GROMACS (single, multi-core or MPI). The resulting modifications need to then be verified to determine that the software still works. That it is still 'good enough' for performing molecular dynamics simulations. Figure 2 shows the magnitude of the problem.[13]

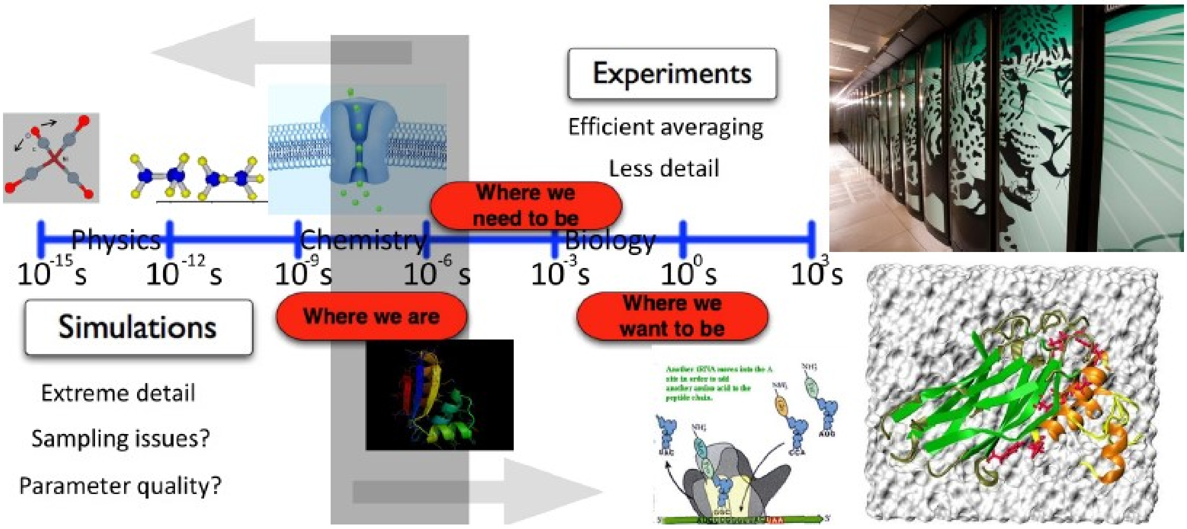


Figure 2 How can molecular simulation reach the exascale? Challenge in performance and parallelism

Adapted from Roland Schulz and Erik Lindhal

The magnitude of this task is large due to the large number of solvent molecules in the typical simulation and the number of time steps needed simulation duration. The number of time steps between femtoseconds and milliseconds also is very large.

Most of the computational overhead is in the processing of the solvent, usually water. The simulation of water is generally calculated between pairs of molecules with nine (9) interactions calculated for each pair (OO, OH, HH).

The 1/sqrt function, Lennard-Jones equation and Reaction Field terms are typically calculated. (This project does not address PME but the technical approach can be used to optimize it). There are also other solvent models where the solvent overhead is more. Water to other atoms are the second most common type of interaction. *If the processing of the solvent cannot be made significantly faster, then the simulation will not run faster.*

Limitations on Scalability

Studies have shown that even using multiple processors/cores/clusters that there are fundamental limitations on the scalability of MD simulations. This limitation is due to limitations on how atoms may be distributed across cores/processors and network node. As the number of atoms/core decreases the amount of time in communications increases. It is not possible to run one atom per core. Studies have shown that ≈500-1000 atoms/core is approximately the lower limit in most system environments. With recent developments in supercomputer environments based on Intel Sandy Bridge and Ivy Bridge processors, multiple NVIDIA GPU coprocessors, hybrid OpenMP and MPI, Infiniband networks and Verlet cutoff schemes as few as ≈100-140 atoms/core have been achieved. Figure 3 shows the results of the study of the peak performance of MD simulations by Gruber and Pleiss based on number of atoms and cores.

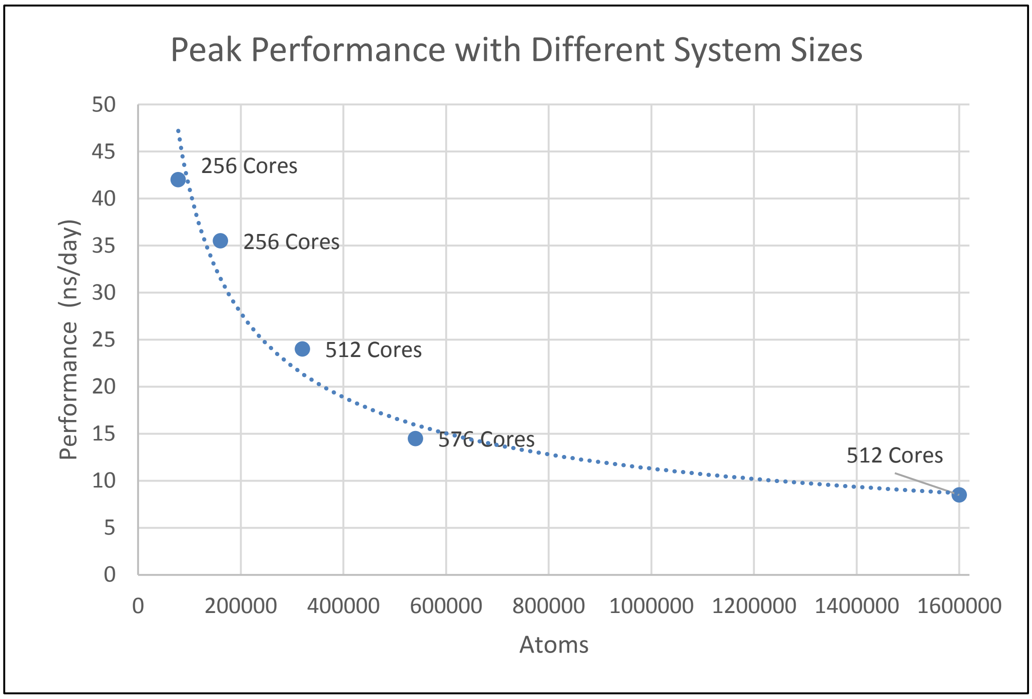


Figure 3 Peak performance by system size. Adapted from Gruber and Pleiss

References

[1] Karplus, M & McCammon, JA 2002, 'Molecular dynamics simulations of biomolecules'. *Nature Structural Biology*, 9(9): p. 646-652.
[2] Karplus, M & Kuriyan J 2005, '*Molecular dynamics and protein function'*. Proceedings of the National Academy of Sciences of the United States of America, 102(19): p. 6679-6685.
[3] Pande, V.S., et al. 2003, 'Atomistic protein folding simulations on the submillisecond time scale using worldwide distributed computing'. *Biopolymers*, 68(1): p. 91-109
[4] Lindahl, E, Hess, B and van der Spoel, D 2001, 'GROMACS 3.0: a package for molecular simulation and trajectory analysis.' *Journal of Molecular Modeling*, 7(8): p. 306-317.
[5] Hess, B, et al., 2008, 'GROMACS 4: Algorithms for highly efficient, load-balanced, and scalable molecular simulation.' *Journal of Chemical Theory and Computation*, 4(3): p. 435-447.
[6] D. van der Spoel, E. Lindahl, B. Hess, A. R. van Buuren, E. Apol, P. J. Meulenhoff, D. P. Tieleman, A. L. T. M. Sijbers, K. A. Feenstra, R. van Drunen and H. J. C. Berendsen, Gromacs User Manual version 4.5.6, www.gromacs.org (2010)
[7] Duan, Y. & Kollman, P.A, 1998, 'Pathways to a protein folding intermediate observed in a 1-microsecond simulation in aqueous solution' Science, 282(5389): p. 740-744.
[8] Darden, T., York, D., Pedersen, L. 1993. "Particle mesh Ewald: An N•log(N) method for Ewald sums in large systems." *Journal of Chemical Physics*. 98:10089–10092,

[9] IEEE Computer Society (August 29, 2008). "IEEE Standard for Floating-Point Arithmetic". IEEE. doi:10.1109/IEEE STD.2008.4610935. ISBN 978-0-7381-5753-5. IEEE Std 754-2008

[10] https://msdn.microsoft.com/en-us/library/e7k32f4k.aspx (retrieved July, 23, 2015)

[11] https://wiki.scinet.utoronto.ca/wiki/images/2/2d/Compiler_qrg12.pdf (retrieved July 23, 2015)

[12] https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html (Retrieved July 23, 2015)

[13] Roland Schulz & Erik Lindahl, "*How can molecular simulation reach the exascale? Challenges in performance and parallelism",* (retrieved July 29, 2015), http://www.csm.ornl.gov/workshops/biomolecular/documents/MDFuture.pdf

List of Abbreviations and Symbols

Aminoacids

Ala	A	Alanine	Arg	R	Arginine
Asn	N	Asparagine	Asp	D	Aspartate
Cys	C	Cysteine	Gln	Q	Glutamine
Glu	E	Glutamate	Gly	G	Glycine
His	H	Histidine	Ile	I	Isoleucine
Leu	L	Leucine	Lys	K	Lysine
Met	M	Methionine	Phe	F	Phenylalanine
Pro	P	Proline	Ser	S	Serine
Thr	T	Threonine	Trp	W	Tryptophan
Tyr	Y	Tyrosine	Val	V	Valine

Abbreviations

AES	Intel Advanced Encryption Instructions
AVX	SIMD Advanced Vector Instructions
AVX2	SIMD Extensions to AVX
AVX512	SIMD Extensions to AVX for the Intel Phi Processor
CUDA	Compute Unified Device Architecture
FF	Force field
FMA	Fused Multiple Add Instructions (multiple versions exist)
H	Hamiltonian operator
HPC	High Performance Computing
L1	Level 1 Cache. Small caches for Instructions and data
L2	Level 2 Secondary Cache. Larger cache than Level 1
L3	Level 3 Third level cache. Larger cache than Level 2
L4	Level 4 Fourth level cache. Larger cache than Level 3
LJ	Lennard-Jones
LxA	Level x Application cache paralleling L2, L3, L4 or a shared memory level in GPUs
MC	Monte Carlo
MD	Molecular dynamics
MPI	Message Passing Interface
NPT	Isothermal-isobaric ensemble (constant pressure and temperature)
NMR	Nuclear Magnetic Resonance
NVT	Canonical ensemble (constant volume and temperature)
OpenCL	Open Computing Language
OpenMP	Open Multi-Processing
PGO	Program Guided Optimization
PME	Particle Mesh Ewald
QM	Quantum mechanics
RF	Reaction Field
RMS	Root mean squared

RMSD	Root mean squared deviation.
SIMD	Single Instruction Multiple Data
SoC	System on a Chip
SPC	Single Point Charge Water Model
SPCE	Single Point Charge Water Model with Average Polarity Adjustment
SSE	Instructions set for SIMD on Intel and AMD
SSE2	SIMD Instruction set extensions to SSE
SSE4.1	Intel Instruction extensions to SSE2 not on all AMD CPUs
TIP3P	Transferable Intermolecular 3-point Water Model
vdW	Van der Waals terms

µops	Micro-operations (CPU)

T	Temperature.
V	Potential energy.
W_{ij}	Electrostatic interaction between site i and j.
e	Protonic charge.
$k_{\mathbf{B}}$	Boltzmann constant.
k^{b}	Bond force constant.
q	Point charge.
r_{ij}	Interatomic distance between atoms i and j.
$\mathbf{r}_{p}$	Coordinates at point P.

Greek Letters

ΔG	Hydration Free Energy Change
ΔH	Hydration Enthalpy Change

List of Tables

List of Figures

Outline

This dissertation is organized into an introduction, abstract, three main chapters and a closing general discussion and conclusion. It also contains supplementary materials for chapter 3. Chapter 1 discusses the developed incremental noncomputational algorithm and the associated variable precision version of the IEEE 754 floating standard that was developed to support the algorithm. It also discusses the general application of the methodology to object interaction based high performance computing (HPC) applications for different problem domains. Chapter 2 presents the detailed approach and results of applying the algorithm to the molecular dynamics application GROMACS. It shows the effect of various spatial granularities represented by variable precision floating point mathematics on the calculation of nonbonded Lennard-Jones and reaction field interactions. It also assesses the impact of the methodology on Lennard-Jones only interactions. It presents both performance results and the impact on the energy and forces involved in the simulation for water boxes, argon and proteins. Chapter 3 presents an extensive study of the hydration free energies of five amino acid analogues used to validate the developed algorithm with experimental results and the work of other research. The study uses approximately 170 microseconds of simulation time to obtain a statistically significant amount of data. Statistical equivalence methods were used to validate the developed algorithm against the three widely used builds of GROMACS 4.5.3 (single precision C code, single precision SSE, and double precision SSE2). A final general discussion and conclusion is provided after Chapter 3. Appendix 1 contains the detailed statistical data and results for chapter three.

Chapter 1 – Optimizing Applications in HPC Environments Using Incremental and Noncomputational Methods

Optimizing Applications in HPC Environments Using Incremental and Noncomputational Methods

Portions of chapter 1 of this thesis are based on a provisional US Patent application #62213053 that is the intellectual property of Virtual Strategy, Inc. and are used by permission.[14]

1.1 Abstract

The performance of most HPC applications has *already* been fully optimized using optimizing compilers, off-loaded to GPUs, run in distributed environments and using the latest and most efficient algorithms. When the code to be optimized cannot be made faster a way must be found to avoid computations completely or the application will not run any faster. This approach should be used only after all other optimization techniques have been done.

Supercomputers are generally used to address two major classes of problems: 1) problems with large amounts of data that has relatively few/no interdependencies and can be processed as many streams and 2) problems that are computationally intensive because they have large numbers of interactions between many objects. These problems may be in many diverse areas such as: weather forecasting, oceanography, climate change, the evolution of galaxies, development of stars and clusters, black holes, particle physics, molecular dynamics, protein folding, fluid dynamics, economics among other applications. The first class of applications can be easily distributed over an unlimited number of processors or cores by dividing the data stream. The second class of applications generally have a computationally intensive code section at their core (frequently to calculate forces) and a large number of object interactions. Software simulating large numbers of interactions uses algorithms (e.g. lattice summation or spherical cutoffs) to reduce the number of interactions from $O(N^2)$ to O(NlogN) or O(N). [15,16] They also exploit the latest processor architectures, OpenMP, MPI, vector instruction sets such as AVX, AVX512 and FMA and offload work to GPU coprocessors using NVIDIA CUDA or OpenCL. Even with these

techniques there are limits to both the number of object interactions that can be processed and the amount of time that these interactions may be simulated independent of the problem being solved.

Problems involving a few 10s of thousands of objects will not run significantly faster on a supercomputer than on a 64 core server. The developed algorithm provides a ‘noncomputational’ approach based on the definition of the problem followed by incremental computation. It exploits a ‘variable precision’ numeric format based on the IEEE 754 standard. Performance on a single thread/physical core is the fundamental building block of performance whether code runs on a CPU or GPU and it is independent of whether or not the application is running in a distributed environment or not. This study reports single thread/physical core improvements in the performance for solving multiple force equations between atomic level objects on the Intel Core i7 ‘Sandy Bridge’ of 14-15 times that of the existing hand coded assembly language builds for the GROMACS application(SSE, SSE2). The method was also used to develop simple math functions and compared to the standard C library functions with performance speedups in the range of 11 to 125 times faster using gcc 4.7. The developed method may be applied to simple functions, equations or the simultaneous solution of multiple equations in a broad set of applications.

1.2 Introduction

1.2.1 HPC Applications Supporting Object Interactions

HPC applications supporting object interactions are critically important for many problem domains. At their core is an inner processing routine based on the object parameters that is calculated every time there is an object interaction. Typically the object interactions are calculated at time step intervals whether they are fine grain atomic level simulations or very coarse grain weather forecasting applications. If this core processing routine cannot be made faster on a single thread/physical core then the software will not run faster. Incremental and noncomputational methods address the need to improve the performance of already fully optimized software by avoiding computation at runtime and performing incremental computation based on precomputed results that are application and execution specific. The fundamental performance unit is how fast the algorithm or functions performs on a single thread/physical core whether it is on a CPU or GPU. By improving performance at this level both larger and longer simulations may be run.

Software tools and development methodologies and even CPU hardware have been highly optimized for performance for over 20 years. Global optimizers available in the compilers in general use generate code so well that it is rarely necessary to code critical portions of software in assembly code. Access to instruction sets such as SSE2, AVX, AVX2, AVX512, and FMA are available through the use of intrinsics or assembly code directives and may be used in a high level language for development. If the developer uses these instructions in a high level language through intrinsics the assembly instructions are exposed to the global optimizer of the compiler. CPUs from the major manufacturers make use of branch prediction, pipelines, instruction reordering, and multiple levels of cache. It is usually a waste of time to second guess a compiler global optimizer and to try to guess what the processor will do at runtime. The cycles/instruction vary greatly based on where the data resides (register, L1, L2, L3, L4 cache or main memory). Cycles/instruction also vary based on what other instructions are near the instruction. Architectural differences between Intel, AMD and other CPU vendors vary greatly between each other as do differences between GPU processor vendors. There are also significantly different performance characteristics between the ‘generations’ or ‘families’ within a manufacturer’s product line. Other important factors that impact application performance in HPC environments include physical memory speed, network bandwidth, and CPU-GPU bandwidth.

GPU support has been added to HPC environments in the last few years using NVDIA’s CUDA[17] to offload some portions of a simulation to a high performance GPU.[18] There are however underlying limitations for GPU based computing including the limited bandwidth between the CPU and GPU and limitations in GPU hardware as a generalized coprocessor.[19] Recent advances in GPU memory architectures that include a larger number of cores and GBs of memory as well as the addition of GPU cache memory has greatly enhanced the ability to GPUs as general purpose computational engines. NVIDIA CUDA and OpenCL[20] now make it possible to have TFLOPs available on the desktop or in servers.

Many object interaction based applications share some common performance problems. The 1/sqrt function is the largest computational contributor for force-based object interactions that perform force calculations. These applications already benefit from domain decomposition schemes, MPI, OpenMP and vector based instructions and highly optimized code and algorithms. Interaction-based problems have probably benefited the most from specialized SIMD (single instruction multiple data) instructions and fused multiple and add instructions (FMA). For many years most applications that address interaction based problems have exploited the single instruction multiple data (SIMD) instruction sets SSE (single precision), SSE2 (double precision) and recently added support for AVX[21] and AMD's FMA[22] instructions for fused multiply and add. Interaction based applications such as molecular dynamics software have achieved very high performance on a single processor/core by using hand coded assembly language to process 4 single precision values simultaneously. With the support of the AVX and AVX2[23] instructions 8 single precision values may be processed simultaneously. The Intel Xeon Phi processor supports the AVX512 instruction set and can process 16 single precision values simultaneously.

1.2.2 Performance Limiting Factors for Interaction Based Problems

Processing speed has been limited primarily by CPU clock speed (cycle/second GHz) and the amount of data than can be processed in one cycle. In 1965 Gordon Moore, Intel co-founder, predicted that processing power would double approximately every 2 years[24] but in 2005 Moore declared that his law was 'dead'.[25] This was largely due to the limitations on CPU clock speed, heat dissipation on the chip and fabrication costs due to the on chip density. With current technology this limits the performance on a single core/processor primarily based on CPU clock speed (GHZ). There is a direct relationship between clock speed, power consumption and temperature.

From Figure 4 one can see that CPU clock speed has flattened since about 2003. There are more cores/chip and more transistors/chip to support them. Techniques such as multiple cores on chip can reduce the communication costs between threads but CPU clock speed is not substantially increasing and is a fundamental barrier to the performance of object interaction solutions. Over-clocking of CPUs can be done but is limited by the amount of power consumed and heat generated. Over-clocking can also result in computational errors.[26]

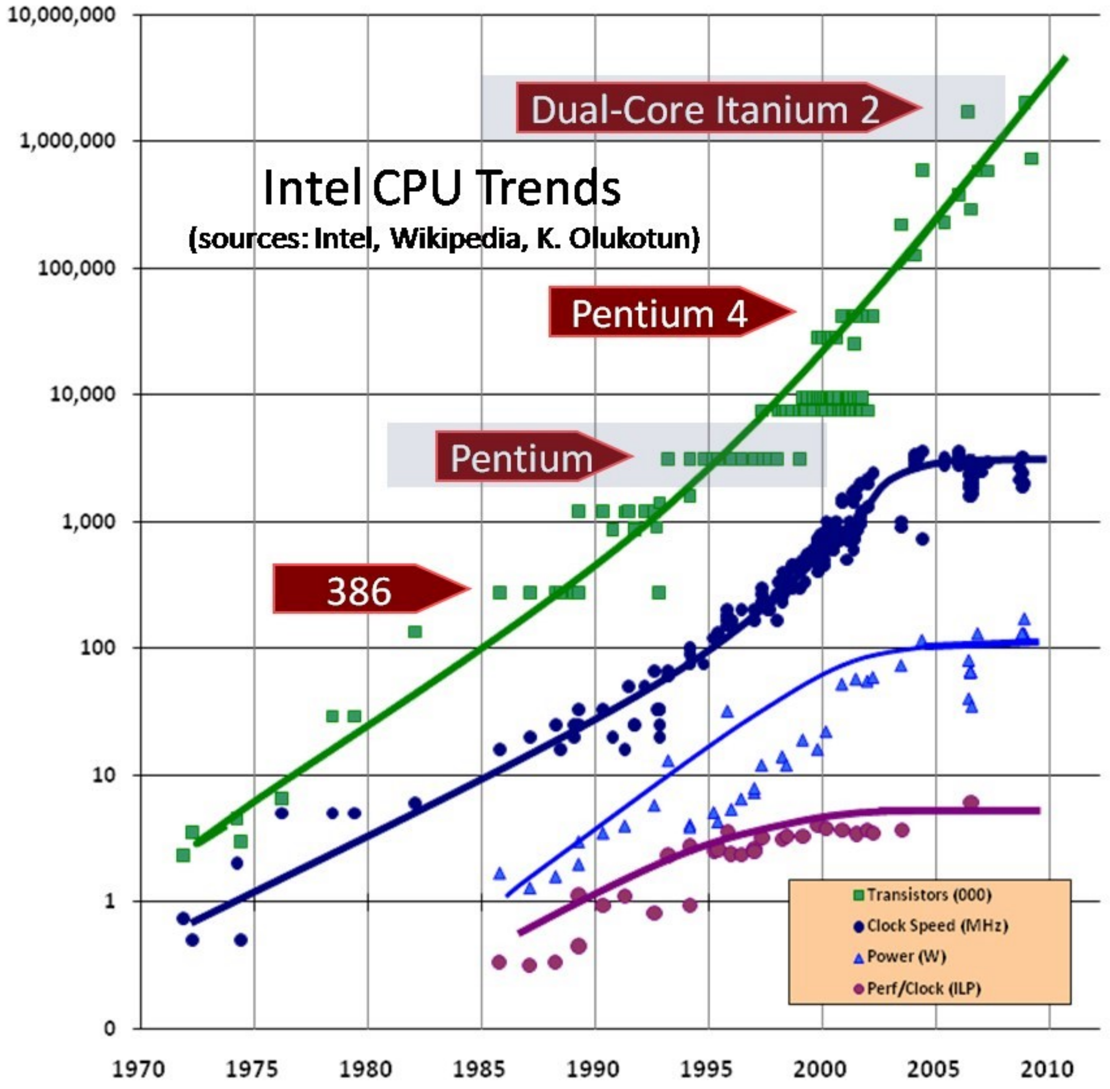


Figure 4 Intel CPU Trends – Limitations on Performance
© Herb Sutter, Used with permission

Studies have also shown that there are limits to the scalability of a simulation based on the number of objects in the simulation.[27] There exists a minimum number of objects that can be processed per processor/core whatever those objects are before the communications costs and real time delays are greater than the real time performance gains. For simulations in the range of a few 10s of thousands of objects this is usually less than 64 cores. This implies that a simulation in this size range will not run significantly faster on a system with 1000 cores. In fact if a simulation were forced to be split into too few objects per core the communication cost could consume most of the real time and slow the real time to process the simulation.

Atomic level interaction in molecular dynamics software is typical of object interaction solutions. Figure 5 shows the results of the study by Gruber and Pleiss[28] in 2010 that investigated the peak performance that can be achieved with molecular dynamics simulations at the atomic level for different system sizes. They tested object simulations distributing force calculations workload between objects (atoms) across up to 1000 cores. Simulation of atomic level forces between objects differs from other object interaction applications primarily in the following areas: 1) the definition of the object, 2) the equations used to calculate the interactions (usually forces), and 3) the granularity of the objects. Larger objects are built from smaller objects from the atomic level up but with a loss/change in the nature of the object details. In the process of defining higher level objects, additional parameters may change or the equations may vary but the fundamental problem of having many objects interacting with many others remains the same. Fortunately applications do not need atomic level granularity to model weather or the economy.

There are therefore fundamental limitations in simulation performance that cannot be overcome by improvements to programming: CPU clock speed, size and speed of L1, L2, L3, L4 cache memory and speed of main memory, transistor chip density, heat, and ultimately the speed of light. Distances off chip to CPU blocks, blades or server nodes

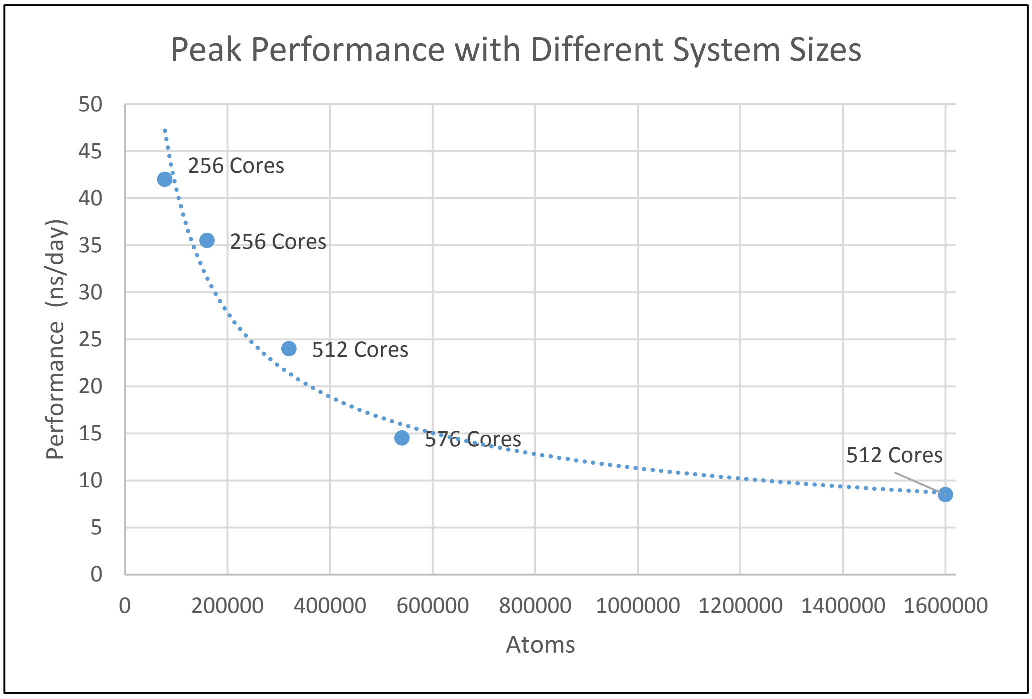


Figure 5 Peak performance by system size. Adapted from Gruber and Pleiss

have significant delays. Distance is a fundamental problem. Light travels about 30cm per nanosecond in a vacuum. Sending data and receiving an acknowledgement in a vacuum is limited to only 15cm.

Other fundamental limitations exist to performance improvements such as those imposed by Amdahl's law.[29, 30] The performance of a program that can be improved is limited by the percentage of the code that can be improved. The speedup of a program using multiple processors in parallel computing or improved algorithms is limited by this sequential fraction of the program. For example, if 95% of the performance of program can be parallelized or improved by code changes, the theoretical maximum speed up would be 20 × as shown in Figure 6[31], no matter how many processors or how good the programming changes are assuming that the portions of the program that can be improved may be parallelized completely or that the performance of the code being changed may be reduced to zero percent of the total time for the program. Problems where the data streams can be split suffer the same limitations but due to the nature of the data being processed Amdahl's Law is generally of little consequence. Interactions between objects, however limits the ability to split computation between cores/processors and nodes.

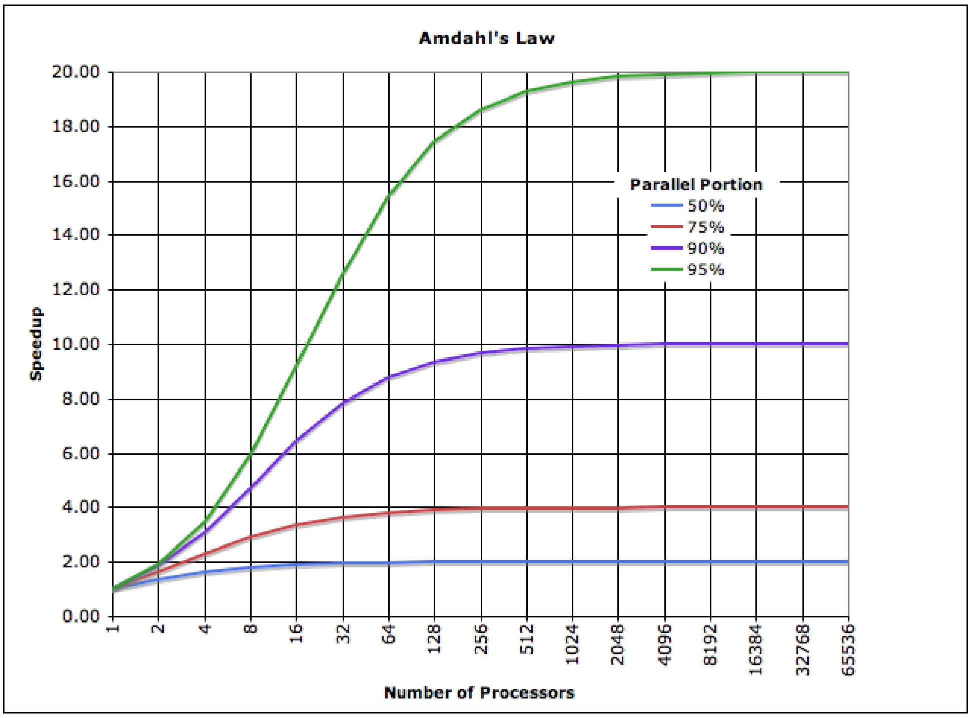


Figure 6 Performance Limitations - Amdahl's Law

The processing of object interactions has a fundamental limit based on the number of objects that can be processed per core. The minimum number of objects per core is application specific depending on the nature of the application and problem being solved. For force based atomic level object interactions this limit is currently in the range of ≈100-1000 objects per thread/physical core depending on hardware and application requirements.

Fortunately the percentage of CPU time and clock time for interaction processing is usually concentrated in core processing routines where these calculations are performed. The performance improvement that may be obtained for processing the object interactions has some limitations based on the percentage of the application that is spent calculating object interactions. For a large number of applications in this class the percentage of time processing these interactions is extremely high (over 90%) because forces, velocities, and distances are being calculated. Whether the objects are particles, atoms, stars, galaxies, or coarse grain objects in weather, climate or economic models there is a define set of processing functions. The ability of the incremental noncomputational methodology to improved object interaction performance is based on the number of types of objects and not just the number of objects. In the example used in this study of atomic level molecular dynamics simulations, the dominant object interactions are water molecules interacting with other water molecules, followed by water to solute interactions and then solute to solute interactions. There is a relatively small number of different types of objects.

1.3 Methodology

1.3.1 Noncomputation vs. Faster Computation

The fundamental question is if performance cannot be substantially increased by faster CPU clock times, offloading to GPUs or other processors in a network, where will the breakthroughs in performance occur? This study has developed an algorithm that eliminates most of the computation inside the inner processing loops for object based interactions. It can effectively raise the scalability bar for all single core/processor and multiprocessor environments where large numbers of objects and object interactions occur.

1.3.2 Optimization and Incremental Computation

For more than 30 years compliers and code generators have performed various levels of performance and memory minimization optimizations for software developed in high level languages such as C, C++, FORTRAN, and others. In 1986, *Compilers: Principles, Techniques, and Tools* the 'Dragon Book'[32] was published and became the definitive reference guide for compiler development and has remained so to this day. It was updated in 2006 to include additional details on new machine architectures, parallel processing, JIT compiling, data flow analysis and directed translation.[33] High level language optimization operates at several levels. The developed method uses an approach similar to that of a compiler optimizer's analysis of the code flow[34] and data flow as represented by the compiler's intermediate code and symbol table prior to code generation. After the compiler generates intermediate code and a data map for the program from the source files usually an optimization phase occurs within the compiler prior to code generation to identify opportunities for performance improvement or to reduce the amount of memory required. The optimization of the intermediate representation of the program looks for common expressions within the 'scope' that the optimizer can see to minimize the number of times that the code is executed. In many cases this means that the expressions and intermediate code is restructured so that portions of the code may be executed once while other parts of the code must be executed every time. The optimizer also creates temporary variables for use in storing intermediate results shared between expressions and subexpressions. Another key optimization technique is to move code execution outside of inner loops so that fewer instructions are executed in the computationally intensive loops. Compiler optimization is limited because it is performed within a scope. These scopes are associated with the program's source files. An optimizer may use the local source file being compiled including the merged files or in some cases it may use all of the source code for the entire application. The later takes more time to build and relate the data maps and code flows for the sources files of the application but can result in valuable interprocedural optimizations. In highly optimized HPC applications this is part of the standard software build process by specifying compile time optimization levels and has been done for many years. Developers have also moved as much computation out of the inner computational loops as possible based on user knowledge of the application.

The developed algorithm optimizes the performance of the computationally intensive inner loop code by using *runtime* knowledge at the application level and knowledge of the equations and functions used in the inner loop. This permits the algorithm to be used to replace most computation with the retrieval of precalculated subexpression and expression results (not computed within the inner loops) followed by a few computational instructions (incremental computation) to provide either subexpression or final results. The precalculated subexpression or expression results are calculated once when the program or function is first entered or when the results are first required.

This approach requires knowledge of the data range and precision required for each variable that is used to build precalculated results and that the precalculated results for the used to support the various subexpressions/expressions be small enough to be highly cacheable. The method is highly dependent on the data map of the application and the degree to which the application contains subexpressions or expressions within the inner loops that are suitable for the developed algorithm. In theory it is possible to develop an application development tool based on an existing program profiler that could track the data ranges and precisions of the data values used in optimized subexpressions and expressions during an actual execution run of the program (perhaps a short execution). The tool could then suggest to the developer areas that could benefit from the developed algorithm providing suggested data ranges and precisions. The developer could then select the suggestions that were of interest and modify the data ranges and precisions if needed. These suggested and modified changes could then be used by the application development tool to automatically generate modified application source code to define and initialize the tables for the precomputed results and to update the source code for the inner loop to minimize computation within the inner computational loops by retrieving the pre-calculated results from one of the tables. Currently, Microsoft[35], Intel[36] and GNU[37] have program guided optimization (PGO) functionality that creates instrumented versions of an application and allows the application to be 'trained' to produce higher performance code. The developed methodology and algorithm has not yet been incorporated into any software development tool or PGO option.

In addition to avoiding computation or computing results only once in many cases binary or integer instructions may be used within inner processing loops to replace sequences of floating point computation. These instructions in many cases run in less than one CPU cycle. This project takes full advantage of this technique.

The algorithm also exploits the fact that many variables within the inner object processing routines are used as constants at *runtime* and are not known to be constants when the software is compiled. A compiler optimizer's constant folding and propagation algorithm reduces constant expressions and subexpressions at compile time to a compile time precalculated value so that constant expressions and subexpressions are never calculated at runtime. The developed algorithm performs a similar function when the precalculated tables are initialized and all expressions are reduced to constant expressions with one or two variable over a range of values.

The developed methodology and algorithms in both the C language and hand coded assembler routines provide a 'noncomputational' approach based on the definition of the simulation and the incremental computation of pre-calculated results to obtain the final result. The method can return multiple intermediate results that are then assembled with non-floating point instructions or incrementally calculated to produce a final result for each 'hot spot' in a computationally intensive application.

In incremental computation and compilation models data elements and their associated code are identified and only the data elements and code that needs to be recalculated is recalculated when a change is made. The best known examples of this are spreadsheets such as Microsoft Excel™ where only the cells impacted by a change in data values or formulas are recalculated and the rest of the spreadsheet remains unchanged. The same process also occurs in incremental programming language development environments where there is a demand for very fast compilation speeds. In 1998 IBM introduced VisualAge C++ Professional for AIX Version 4.0, a fully incremental compiler[38]. Microsoft 'Roslyn' is using this approach in Visual Studio 2015.[39, 40] In interactive development environments only the data and code that is impacted by a change as the change is being made is recompiled. Using the developed methodology precalculated intermediate or final results provides an opportunity to reduce the data and code interdependency so that runtime performance is faster.

These incremental results are each stored in a table and accessed with memory, integer and bit manipulation instructions only. These instructions can be efficiently executed in the CPU pipeline and many of these instructions execute in less than one clock cycle.

No floating point instructions are required except where pre-calculated results cannot be used for intermediate results or final results that are used external to the function being optimized. For force base object interaction problems such as molecular dynamics this approach effectively produces a 'coarse grain' space for the simulation to run in using the distance squared (r^2). The algorithm however cannot be used to reduce the cost of processing distance calculations associated with object interactions at the computational core of an application. Fused multiple and add (FMA) instructions can help reduce the overhead of the distance component of force based object interactions.

The objective of this research is to address the approximately ≈70-90% of the CPU cost of most object interaction applications and to greatly increase the performance of core math functions (e.g. C library functions) and equation solutions used by HPC applications in general. In order to improve the performance of these equation solutions it is necessary to eliminate or reduce substantially the computer instructions and CPU cycles required to solve them. Fortunately at runtime, simulation model and application specific values are loaded into the program, based the requirements of the application, and used as constants. An HPC application also almost always uses only a portion of the IEEE 754 data format both in terms of exponents and mantissa bits.

In order for the developed methodology to be memory efficient and fit within CPU caches it is critically important that the developer and the user understand the data being used in core processing routines. The algorithm assumes that a reduced precision is acceptable and that only a portion of the range of IEEE 754 exponents are used. The range of exponents and the number of significant digits required are directly related to the size of the tables used to hold each of the incremental or function results. The methodology however *does not* require that all of the exponent ranges or ranges be contiguous or that each range support the same number of significant digits (bits of the mantissa).

In order to minimize the amount of CPU cache memory used, a new mathematical and floating point format model has also been developed, providing a variable precision floating point calculation model based on the IEEE 754 standard to reduce the size of the pre-calculated tables. This variable precision format parallels the IEEE standard without the need for additional tables or runtime operations. This algorithm may be used in any application that is computationally intensive. There are no approximations other than those reflected in the reduced precision using a subset of the IEEE floating point definitions for both the number of significant digits required and the exponents required.

All calculations that were used for the incremental calculations are performed in double precision and stored as single precision when the tables are initialized unless the application requires a double precision result. Other numeric formats could be similarly processed performing calculations in a form most appropriate to the application and storing the incremental results in a form that would not require a conversion at runtime.

Applications may use a mixture of single or double precision tables for storing incremental results depending on the requirements of what is being optimized. When the tables containing the incremental pre-calculated results are generated the variable precision algorithm adjusts the floating point representation to parallel the IEEE 754 standard but at reduced precision. The algorithm uses a simulated 'guard bit' to avoid creating a very small cumulative divergent error.[41,42] Processor manufactures incorporate 'guard bits' to avoid rounding errors that would create a cumulative error. These 'guard bits' are used to create a forward backward offsetting error correction across the range of floating point values. When the tables are created the exponent portion of the floating point representation of the variable used to determine the noncomputational result (e.g. distance squared in the case of most force based object interactions) *may* be masked with a binary value to alter or create one or more ranges of exponents and/or eliminate the sign bit of the floating point representation. The precision of the resulting value is reduced shifting the floating point value in binary form to the right reducing the mantissa. A right shift effectively divides the representation by a power of two and reduces the size of the tables required. After this is performed, a simulated 'guard' bit is added enabling the calculation of the values stored in the table to be treated as a scaled base 2 value. This provides a mixed stability model with a very small offsetting forward and backward error correction paralleling the IEEE 754 standard. This is necessary because dividing an integer by 2 causes a truncation on odd values creating a cumulative error. Using a right shift to divide an integer by two gives the correct value for all even numbers. For odd integer values it is necessary to create an alternating forward and backward error correction. This is performed at table initialization time. A floating point to integer conversion method is not used because this creates a divergent error relative to the IEEE standard by eliminating the ‘floating’ property of IEEE 754 compliant CPUs. If the GROMACS[43] nonbonded routines used a floating point to integer conversion technique to create a table index such as was added as an option to CHARMM, this would create an increasingly large offsetting corrective error as the distance squared decreases. This has an undesirable effect on the results for the force values as the distance squared becomes smaller. At some point in a simple floating point to integer conversion method cutoffs are reached because of the scaling factor.

The developed algorithm is designed to scale using single instruction multiple data (SIMD) extensions to instruction sets such as SSE, SSE2, SSE4.1, AVX, AVX2 and AVX512 Intel Xeon Phi[44]. In these instructions sets it is possible to process 128, 256 or 512 bits of data in parallel. This may be in the form of 4, 8, or 16 single precision values or 2, 4, or 8 double precision values. The Intel Xeon Phi coprocessor processes 512 bits but requires special programming when used as a coprocessor. The molecular dynamics simulation program GROMACS 4.5.3 supports single and double precision C language code, and hand coded single precision (SSE) and double precision (SSE2) assembly code. It has been highly optimized for calculating atomic level force interactions for over 20 years. The variations of the developed algorithm were evaluated against the standard build options for GROMACS 4.5.3 for performance and equivalence of functionality.

The developed methodology could also benefit from new computer instructions that could reduce the number of instructions from three to two in an Intel or AMD architecture and support the allocation and management of a portion of L2 and L3 CPU caches. They could also be implemented by other CPU or GPU vendors.

In order to test the developed algorithm a C language test program was developed that reads a file with 10^8 atomic level object interactions output from the GROMACS software for use in force interactions. The performance of the test application using this dataset compared the C version of GROMACS, and the hand coded highly optimized SSE and SSE2 assembly language code with the SSE2, SSE4.1 and AVX2 versions of the developed algorithm.

The same test program and dataset also evaluated the performance of the developed algorithm in C versus the performance of the gcc C library functions for sin, cos, tan, sqrt, log, and pow over the limited range of values in the dataset. Included in this testing was the C version of the Newton-Raphson 1/sqrt that is included in the C version of GROMACS 4.5.3 and Lennart Nilsson's floating point integer conversion table lookup method[45].

The developed algorithm has 22 variations reflecting the number of significant digits that may be expressed by reducing the number of bits in the mantissa of the IEEE 754 single precision representation. The full IEEE 754 single precision format supports ≈7.22 significant digits with 23 explicit bits and 1 implicit bit for the mantissa. In the testing with GROMACS the fundamental value used to calculate object (nonbonded forces) interactions is distance squared and the object type. The type of object includes attributes such as charge and other constants. The developed algorithm effectively reduced the spatial and computational granularities using a variable precision floating point representation. This was possible because the application uses limited range of the IEEE 754 format. In the application a single precision value of 1.0 is used to represent 1.0 nm^2. Given this application assumption it is unnecessary to use all ≈7.22 significant digits of the IEEE format because that would support a spatial granularity representing the size of a subatomic particle. This is not meaningful to the application and using the full IEEE 754 format would defeat the algorithm's ability to improve performance because of the amount of CPU cache memory required.

The algorithm's forward and backward error correction provides stable computational results with no divergence for each reduction of the mantissa bits but the requirements of application determine what number of significant digits is required. The algorithm supports the reduction or elimination of program code if this is known. The IEEE 754 standard avoids this problem by making the range of exponents and number of significant digits so large that the application developer does not need to be too concerned. In many cases the results returned by the developed algorithm are used with experimentally determined data. For example, ocean or atmospheric temperatures are measured in terms of a very limited range of significant digits and only a limited exponent range. It is unnecessary to believe that ocean temperatures are measured to an accuracy of 10^{-7} degrees. A few significant digits and a few exponents are all that is required for terrestrial ocean and air temperatures. It is also necessary to validate an application using the incremental noncomputational methodology with a reduced precision to determine if the results are 'good enough'. Using the developed methodology in our study of atomic level simulations using GROMACS we found that three significant digits and 7 base 2 exponents from the IEEE 754 format were sufficient to have simulation results that were within the deviations between the C, single precision SSE and double precision SSE2 builds of unmodified version of GROMACS that are all considered 'good enough' for general use.

1.3.3 Evaluation of the Impact of Reducing the Floating Point Precision

The analysis in Figure 7 represents the results of reproducing the precision on a water box simulation. This should be typical of force based object simulations. The results were generated from a series of simulations using the unmodified application in using the SSE, C builds and comparing them to the application optimized with of the developed algorithm from ≈7.22 significant digits to ≈2.11 significant digits. Each of the values between ≈7.22 and ≈2.11 represent the number of base 10 significant digits that may supported by reducing the mantissa portion of the IEEE 754 format by one bit starting from the full IEEE single precision format that supports ≈7.22 significant digits.

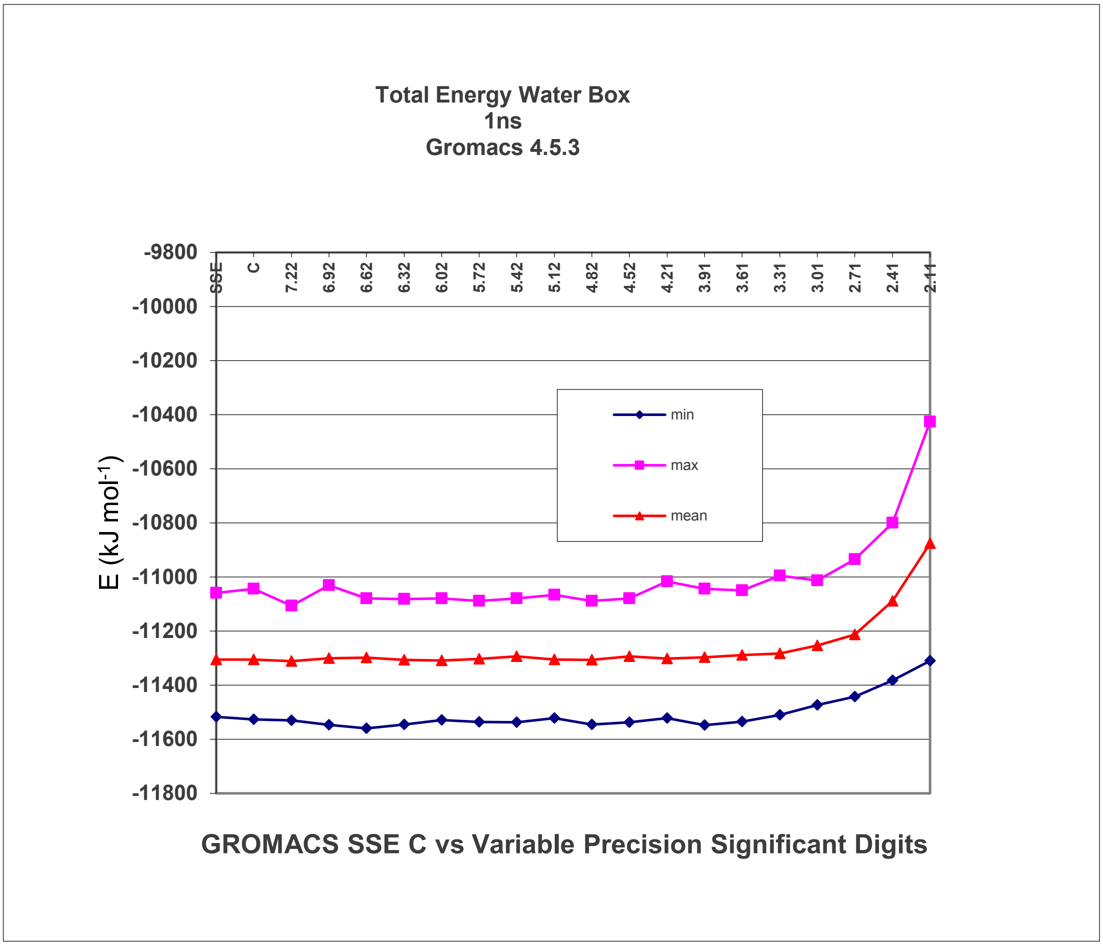
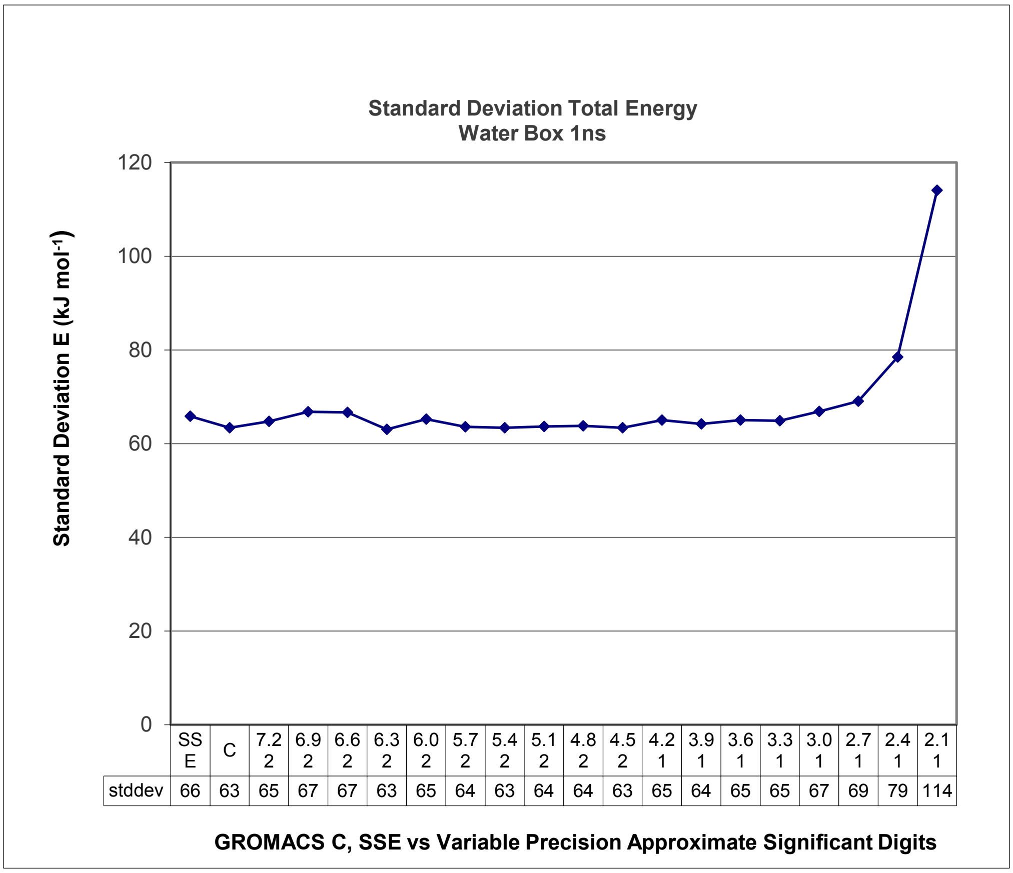


Figure 7 Increase in Total Energy as Computational Precision Decreases

As can be seen total system energy increases after the spatial granularity is reduced to less than ≈3.01 significant digits. Tests were attempted using variable precision of less than ≈2.11 significant digits but the system blew up!

Figure 7 shows the variation in mean, max and minimum values for the total energy for the same water box simulation. It can be seen that the values change significantly as the number of significant digits is reduced beyond ≈3.31 and ≈3.01 significant digits. Figure 8 shows increasingly large changes in the standard deviation for total energy values for the water box simulation with a reduction of precision of greater than ≈3.01 significant digits indicating greater instability in system energy. Starting at ≈2.71 significant digits the standard deviation for total energy starts to increase rapidly until at ≈2.11 significant digits the standard deviation is 114 kJ mol^{-1}.



	SSE	C	7.22	6.92	6.62	6.32	6.02	5.72	5.42	5.12	4.82	4.52	4.21	3.91	3.61	3.31	3.01	2.71	2.41	2.11
stddev	66	63	65	67	67	63	65	64	63	64	64	63	65	64	65	65	67	69	79	114

Figure 8 Standard Deviation Increasing with Reduced Precision

The following figure 9 shows the degree to which the standard deviation of the variable precision algorithm lies between the standard deviations of the unmodified application C and SSE builds. Once again after ≈3.31 or ≈3.01 significant digits there are increases in the standard deviations as the 'spatial' granularity is reduced. Variable precision formats within or close to the deviation between the unmodified builds of the application are considered to likely be ‘good enough’ for the intended use.

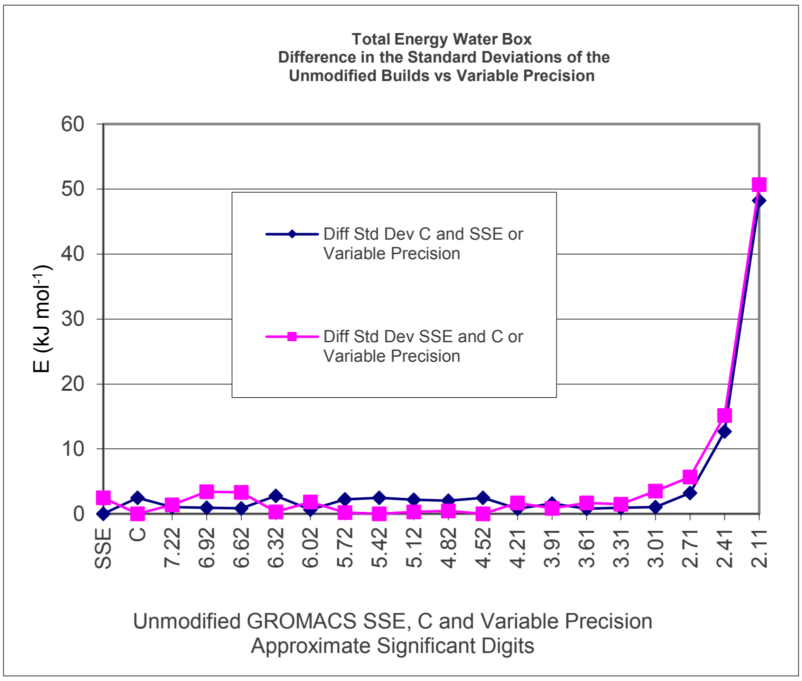


Figure 9 Standard Deviation Between GROMACS C and Single Precision SSE as compared to Variable Precision

1.3.4 Test Environments

Performance testing was performed on the following systems: AMD Opteron 6272 2.1 GHZ 2MB CPU cache (Bulldozer) 64 core server, Intel Xeon 5650 2.67 GHZ 12MB Cache 6 cores (Westmere-EP 32nm), Intel Core i7 (2630QM) 2.0 GHZ 6 MB Cache (Sandy Bridge), Intel Core i7 (930) 2.8 GHZ, 8 MB cache (Nehalem), Intel Core 2 Duo 2.24 GHZ 3 MB cache, Intel Core 2 Quad Core 2.2 GHZ 3 MB cache, AMD Athlon 3800+ 2.4 GHZ 512KB cache, AMD Athlon X2 4400+ 2.3 GHZ 512KB cache and Intel Core i5 1.7 GHZ 3MB cache (4201U (Haswell) ULT).

All testing was performed on Ubuntu 14.04. The Intel Software Development Emulator (SDE) was used for the initial development and testing of the AVX2 instruction set version (Intel Haswell architecture). The gcc 4.8 C compiler was used for development.

1.3.5 A ‘Noncomputational’ Incremental Algorithm

The algorithm exploits the internal structure of the IEEE 754 floating point standard’s representation of single and double precision numerical values. Once these values are processed they are used to index tables of pre-calculated results and then to incrementally compute additional intermediate or final results. The objective of the methodology is to avoid computation at the core of the nonbonded routines and to move this computation so that the pre-computed tables are generated only on first use. Compiler global optimizers have been generating results once and performing incremental computation as needed since the 1980s.

Figure 10 illustrates the differences between the methodologies. A computer program may be considered to be sequences of instructions and functions that have a start and an end with branches representing calls to the functions of the program. The branches for computing the object interactions use the greatest amount of CPU time and real clock time. In the computational model every function calculates everything every time the function is executed. If the algorithms in these portions of the program have already been coded optimally the only alternative is to not perform the calculations. The sections in red indicate the computationally intensive parts of the code that have *already* been fully optimized computationally.

In the noncomputational incremental method when a program is run to process object interactions the input files and parameters are analyzed for a number of key application specific parameters that comprise the definition of what is being modeled based on the problem domain. In the case of our molecular dynamics application these are: type of forces, water model definitions, cutoff schemes, type of box (cubic, dodecahedron), etc. This information is used to construct a series of small memory caches of pre-computed results the first time the results are used that can be

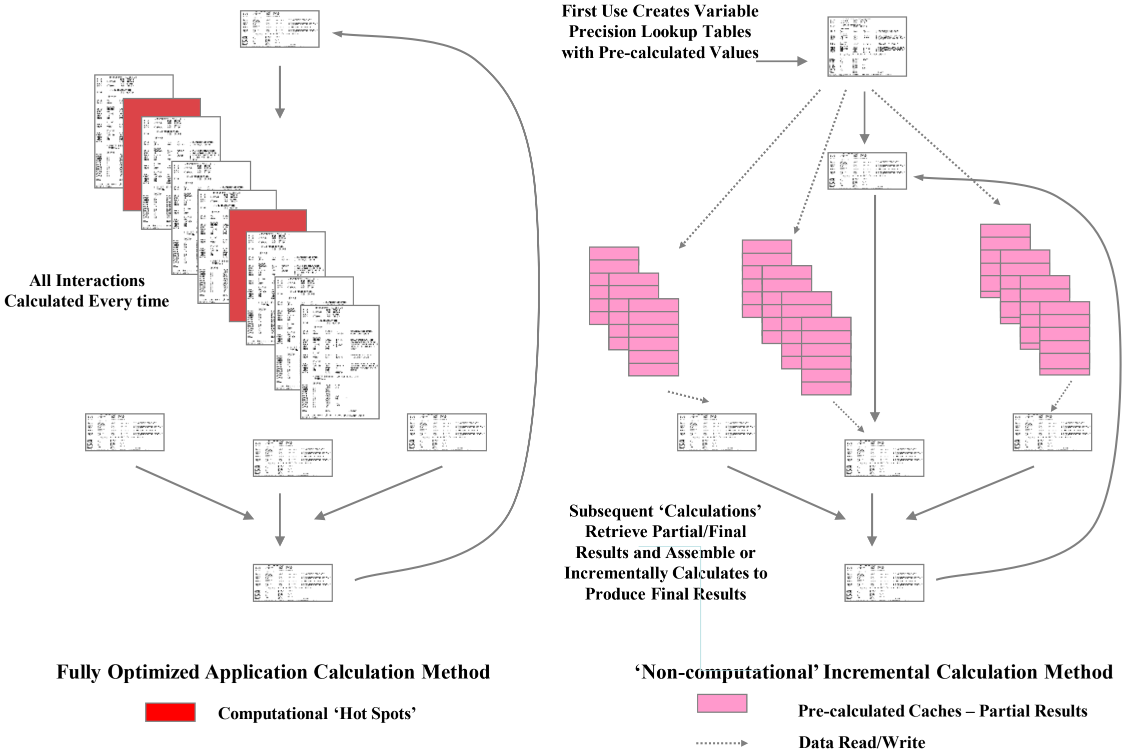

First Use Creates Variable Precision Lookup Tables with Pre-calculated Values
All Interactions Calculated Every time
Subsequent 'Calculations' Retrieve Partial/Final Results and Assemble or Incrementally Calculates to Produce Final Results
Fully Optimized Application Calculation Method
Computational 'Hot Spots'
'Non-computational' Incremental Calculation Method
Pre-calculated Caches – Partial Results
Data Read/Write

Figure 10 Computation-based vs. Noncomputational Incremental Method

assembled or incrementally calculated at runtime for all subsequent processing eliminating the need to recomputed the entire equation or function.

If too high a percentage of the CPU cache is used because the tables are too large performance gains decrease and can result in a loss of performance. In the testing performed it was found that if over approximately fifty percent (50%) of the CPU cache was used the performance could vary substantially between multiple executions of the test program.

1.4 Background – Floating Point Representations

1.4.1 IEEE 754 Floating Point Standard

The IEEE 754 floating point standard defines digital representations for ranges of real numbers. The single precision real number format is represented using 32 bits and

IEEE 754 Representations			
	Width	Range	Precision[a]
Single Precision	32 bits	$\pm 1.18\times 10^{-38}$ to $\pm 3.4\times 10^{38}$	≈ 7.2 digits
Double Precision	64 bits	$\pm 2.23\times 10^{-308}$ to $\pm 1.80\times 10^{308}$	≈ 15.9 digits
[a] Decimal digits precision is mantissa bits * log10(2)			

Table 1 IEEE 754 Format Ranges

the double precision format is represented in 64 bits. The standard also supports other formats such as half precision (16 bits) and quad-precision (128 bits). All binary representations have three components: a fraction (mantissa), an exponent and a sign. The differences between the formats are in the number of bits used to represent the exponents and mantissas. A single precision value is represented in a 32 bit binary format. Figure 11[46] shows the single precision values there is a sign bit, 23 bits plus an implicit 24th bit for the mantissa, and an 8 bit exponent that is biased by 127.

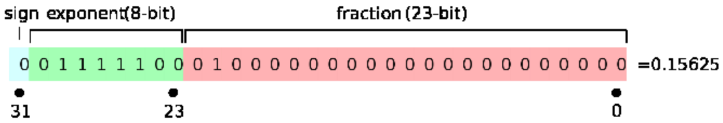


Figure 11 IEEE 754 Single Precision Format

Equation 1 is used to convert the binary 32 bit representation of a single precision value to its base 10 format where *i* is the first bit of the mantissa to the maximum supported by the floating point format. Thus the value represented in figure 11 is 0.15625.

$$x = (-1)^{sign}\left(1 + \sum_{i=1}^{23} \left\lceil b_{23-i} \right\rceil 2^{-i}\right) \times 2^{(e-127)}$$

Equation 1 IEEE 754 Single Precision Value

Figure 12 shows the 64 bit double precision representation that has additional bits for both the exponent and mantissa.[47]

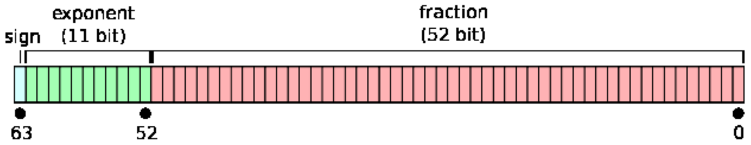


Figure 12 IEEE 754 Double Precision Format

The binary representation of a double precision value is converted to the base 10 format using the following formula. This formula varies from the single precision formula only in the number of bits that are used for the exponent and mantissa.

$$x = (-1)^{sign}(1 + \sum_{i=1}^{52} \lceil b_{52-i} \rceil 2^{-i}) \times 2^{(e-1023)}$$

Formula 2 IEEE 754 Double Precision Value

1.4.2 Understanding the Purpose of the Application

The objective of HPC application software should be to make the *function being performed* run as fast as possible not to just be a general purpose tool. Most software uses data formats and processing instructions whose functionality is defined by standards organizations such as IEEE, ISO or ANSI. These formats and instructions support a broad set of general purpose capabilities but because of their digital implementation they have a large but defined set of limitations. Users of applications accept that the supported ranges are ‘good enough’ for their general use. The CPU vendors accept that their products address enough of the total market requirements that they will be competitive. Applications use constants or calculated values that may be experimentally determined. Data is subject to the limitations of measurement, assumptions of the application, and by equations or algorithms. Almost all applications use objects that are coarse grained at some level.

1.4.3 Developed Variable Precision Floating Point Based on IEEE 754

By manipulating the binary format of the floating point representation of data it is possible to vary the precision of a floating point value. These reduced precision representations and knowledge of the equation or function can be used to create indices. These indices can then be used for accessing pre-computed results that are part of function, equation or application solution eliminating runtime computation by one or more table lookups. The multiple results returned can be 'assembled' or incrementally calculated to product the final results. Due to CPU cache limitations variable precision can operate only over one or more small ranges of the IEEE format.

If variable precision floating point representations are used computationally intensive portions of functions, equations, or solutions can be completely avoided without violating the 'floating' characteristics of the IEEE 754 standard. The computationally intensive portions of the solution become 'noncomputational' at runtime. Use of variable precision requires application and solution specific knowledge that must be provided either by the developer or obtainable at runtime. This information needed by the application to create the tables for use with the methodology is frequently available to the application at runtime based on input files, parameters, or through processes that occur when the applications starts. The methodology also requires knowledge of the properties of the CPU and especially the cache size available to process.

With almost no exceptions the full IEEE 754 single/double precision is therefore unnecessary for such things as distance, force and economic calculations in a single application. Problem domains operate using granular objects. Applications performing force related object interactions with few exceptions do not model quarks and galaxies at the same time.

Physical object interactions operate in real space and time and are simulated by applications to run in virtual real space and time that maps to a range of computational values. This real space and computational space can be represented by fewer bits for both the exponent and mantissa portions of the IEEE 754 representation used for general computation because of the nature of the problem being solved.

Object interaction software should be able to operate on a reduced precision form of the IEEE format.

Reduced granularity uses fewer of the mantissa bits and exponent bits from the entire IEEE range. The following example is for a contiguous range.

$$x = (-1)^{sign}(1+\sum_{i=1}^{\leq 23}\lceil b_{23-i}\rceil 2^{-i})\times 2^{(e\in\{1,2,...254\})-127}$$

Equation 3 Variable Precision Formats for Contiguous Base 2 Ranges

One or more base 2 ranges of the IEEE format may be used per result to optimize the use of cache memory, remembering that the IEEE 754 exponents are biased by -127. Noncontiguous base 2 exponent ranges may be used. For example $R_1 \in \{90, 91, 92\}$, $R_2 \in \{100,101,102,103\}$, $R_l \in \{121,122,123...254\}$. This is useful when a function or equation returns values that are predominately in specific ranges and do not frequently exist in other ranges.

$$x = (-1)^{sign}(1+\sum_{i=1}^{\leq 23}\lceil b_{23-i}\rceil 2^{-i})\times 2^{(e\in\{R_1\cup R_2...\cup R_l\})-127}$$

Equation 4 Variable Precision Formats for Multiple Base 2 Ranges

Partial ranges may also be used by changing the lower bound used for cache table creation to a value that is not a power of two. This may be used to reduce the amount of memory used for each result. This technique may also be employed at table allocation time to reduce the number of entries in the high end of the table. The same techniques for adjusting lower and higher bounds that are used for a single contiguous table may also be applied to a model using multiple ranges. Cyclical functions over many powers of two may benefit from a small amount of code to select table portions or to perform short calculations to use less memory. Some functions or equations may have special requirements for use with the developed algorithm based on boundary conditions, discontinuity or values that would result in an imaginary number, NAN or ±∞. Special coding for table initialization is required for these situations.

The simplest form of a variable precision data format is a constant such as π that is not initialized to the full number of significant digits supported by the IEEE 754 standard. It is usually used as a constant that may be thought of as a table with one entry where the value is coded by the application programmer. No one would think of writing the following code to calculate the value 3.1415.

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right)$$

Equation 5 Formula for π

In a water box test simulation that was used to provide the base for reproducibility for this study the following data profile was observed during a 1 nanosecond simulation where *r* represents the distance between interacting molecules. The maximum distance used to calculate force interactions was 1.4 nm. This value was set in an input file to the software at runtime. The minimum distance has real world physical limits based on the forces and velocities involved in the interactions. At room temperature water molecules can only approach each other at a distance greater than the hydrogen to hydrogen bond length (74pm).

This implies that a variable form of the IEEE 754 format could be used to generate the results for nonbonded interactions using relatively small amounts of CPU cache memory for pre-computed tables. Values representing the ± 1 pm granular space may be represented as a reduced precision IEEE 754 format of three significant digits.

This is accomplished by reducing the number of bits *i* in the mantissa. The following defines the variable precision format for ≈3.01 significant digits over a limited range of exponents. The value of *i* represents the number of bits for the mantissa and the *e* must be in the set of biased values representing the range of powers of 2 that the application requires.

$$x = (-1)^{sign} (1 + \sum_{i=1}^{9} \lceil b_{23-i} \rceil 2^{-i}) \times 2^{(e \in \{121,122,...128\})-127}$$

Equation 6 Variable Precision ≈3.01 Significant Digits, 7 Base 2 Exponents Ranges

If the application represents distance using 1.0 as 1.0 nm^2 and the minimum distance is 74 pm and the maximum distance is 1.4 nm then the application uses only 7 base 2 exponent ranges from the biased IEEE 754 single precision exponent. It was theorized that the number of significant digits used to represent space in the application (molecular dynamics) is approximately 3 digits.

There are therefore four reduced precision formats representing approximately 3 significant digits requiring 9-12 mantissa bits. These formats are memory efficient enough to be useful in currently available CPUs. This may be represented as follows:

$$x = (-1)^{sign}(1 + \sum_{i=1}^{9,10,11,12} \lceil b_{23-i} \rceil 2^{-i}) \times 2^{(e \in \{120,121,...128\})-127}$$

Equation 7 Variable Precision formats for 3 significant digits for test dataset

The most memory efficient alternative supports ≈3.01 significant digits and uses 28,672 bytes of storage for each single precision pre-calculated value (7 base 2 exponent ranges * 1024 entries/power of two * 4 bytes/single precision entry). The distance squared (r^2) and the type of interaction is used to determine what intermediate or final results are created and are available for use during the execution of the simulation. The other alternatives of interest are ≈3.31, ≈3.61, and ≈3.91 significant digits requiring 57,344, 114,688 and 229,376 bytes of storage respectively. Any of these fits easily within the CPU L2/L3 caches. The following shows the memory requirements for variable precision lookup tables where 1.0 equals 1 nm^2 for all the object interactions and intermediate results for a pair of water molecules calculating the non-bonded forces.

Mantissa Bits	Approx Base 10 Significant Digits	Table Entries Per Base 2 Power	Distance Squared (r^2)	Entries 7 Base 2 Powers	Memory Required Per Result	W3A-W3A Interactions LJ + RF (bytes)
9	2.71	512	> 1 pm	3,584	14,336	100,352
10	3.01	1,024	1 pm	7,168	28,672	200,704
11	3.31	2,048	1 pm	14,336	57,344	401,408
12	3.61	4,096	1 pm	28,672	114,688	802,816
13	3.91	8,192	1 pm	57,344	229,376	1,605,632
14	4.21	16,384	0.1 pm	114,688	458,752	3,211,264
15	4.52	32,768	0.1 pm	229,376	917,504	6,422,528
16	4.82	65,536	0.1 pm	458,752	1,835,008	12,845,056
17	5.12	131,072	0.01 pm	917,504	3,670,016	25,690,112

Table 2 Memory Requirements for Variable Precision Format

In our test object interaction application (GROMACS) this permits the use of the algorithm on solvent-solvent, solvent-solute interactions and some solute-solute interactions depending on what is being simulated and the amount of CPU cache available. Simulating the object interactions between water molecules in our test used only a fraction of the CPU cache available and water interactions account for most of the performance cost of a typical molecular dynamics simulation, as much as 90%.

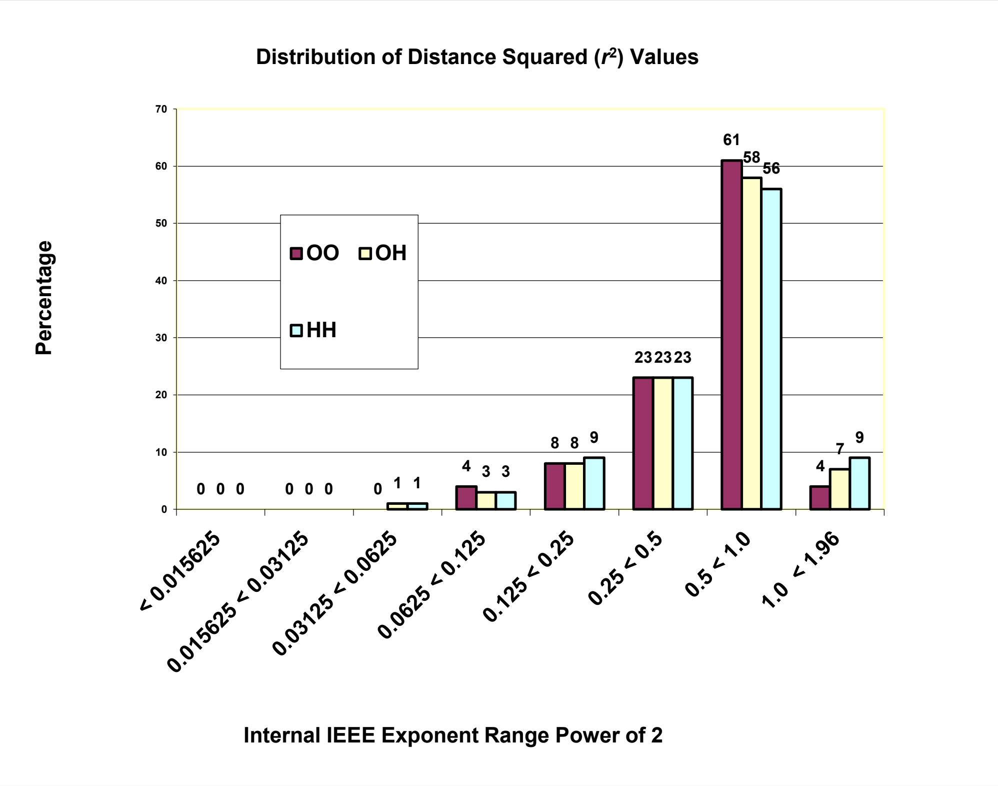


Figure 13 Distribution of Distance Squared (r^2) Values

The requirements of the object interaction application and even a specific execution of that application determine the number bits of the IEEE exponent (base 2 powers) and number of mantissa bits of the IEEE representation that is required. Application developers frequently use the value of 1.0 within the problem domain whether it is one nm, m, km, light year, time step or other value.

Application and execution specific runtime ‘constants’ not known at compile time also may be useful in reducing the number and size of tables used for the developed method. These runtime constants can be used to reduce or eliminate calculations that are performed at runtime. In our example there are a number of variables that are known at runtime and used as constants by the inner most computational routines.

The noncomputational incremental algorithm for use in the water box test can exploit the following[48]:

- Distance squared is always within a limited range and is always a positive value.
- 1/sqrt is not required

  precalculated results are based on distance squared this is the largest part of force based object interactions

- Calculations of numerous intermediate results are not needed.
  - powers of the inverse square root of the distance
  - Intermediate variables associated with equation calculating Lennard-Jones values
  - Constants that are used in the equations known at runtime only
  - Charge, force field and other constants that are specific to the object types
  - Container type specific constant (cube, dodecahedron)
  - Other constants are input from files

In summary, all of the above observations allow the interactions for two water molecules to be reduced to a single function where distance squared is the only variable and it can be used to retrieve pre-calculated intermediate or final results.

The following pseudo code illustrates how this may be applied to GROMACS water three atom to water three atom, Lennard-Jones and reaction field nonbonded interactions in GROMAS. Portions of the pseudo code is from the C source code of the GROMACS nb_kernel212 routine. It is not intended that the reader understand the pseudo code but only for the reader to see the magnitude of the difference.

```
bitpattern11.fval = rsq11;
iexp11            = EXP_ADDR(bitpattern11.bval);
addr11            = FRACT_ADDR(bitpattern11.bval);
result11.bval     = cinvsqrtexptab[iexp11] |
                     cinvsqrtfracttab[addr11];
lu11              = result11.fval;
rinv11            = (half*lu33*(three-((rsq1*lu11)*lu11)));
rinvsq11          = rinv11*rinv11;
rinvsix           = rinvsq11*rinvsq11*rinvsq11;
vnb6              = c6*rinvsix;
vnb12             = c12*rinvsix*rinvsix;
vnbtot            = vnbtot + vnb12-vnb6;
krsq              = krf*rsq11;
vcoul             = qqOO*(rinv11+krsq-crf);
fs11              = (twelve*vnb12-six*vnb6+qqOO*(rinv11-
                     two*krsq))*rinvsq11;
vctot             = vctot + vcoul;
```

The developed methodology avoids a Newton-Raphson 1/sqrt and retrieves three incremental data values based on distance squared (rsq11), each with a cache lookup using an integer index created with one/two instructions depending if the value is signed. Either single or double precision values may be retrieved. The tables are initialized the first time the nonbonded interaction is needed within the limits of the cache size. If the cache size is exceeded then the value can be calculated. The pseudo code is reduced to the following based on distance squared. This incrementally calculates the results based on three retrieved values and eliminates almost all floating point calculations in the inner force calculations and most of the remaining instructions are memory loads and register operations that execute in less than one CPU cycle.

```
vnbtot                  = vnbtot+ OO_VNBA(rsq11);
vctot                   = vctot+ OO_VCOUL(rsq11);
fs11                    = OO_FS11(rsq11);
```

1.5 The Algorithm

1.5.1 Implementation

The algorithm uses a processed form of the raw IEEE format representation as a table lookup index for each intermediate or final result. This is accomplished without compromising the 'floating' characteristic of the IEEE format. A table is created on the first pass through the nonbonded routine to initialize the table. The tables are based on the inter-particle distance squared and this eliminates the need to perform the 1/sqrt operation.

The index is created using the following pseudo code. The lower boundary of the distance squared values that are supported by the table is subtracted from the distance squared value (r^2) for the interaction, treating both native floating point values as 32 or 64 bit integers. The resulting value is shifted a number of bits to the right to reduce the precision and therefore reduces the spatial 'granularity' for the interaction. The resulting integer value is then used as an index to lookup the result or intermediate results in one or more tables or sections of tables and the value returned is either a single precision or double precision value depending on the requirements of the application. The same index may be used to retrieve multiple results. Multiple intermediate results may be used to incrementally calculate a final result. Applying the variable precision methodology to nonbonded interactions, *r* represents distance, r2 represents the floating point distance squared value used to create the index, F_{tab} is the lookup table for the reduced precision results, LOWER_BOUND is the lowest value for r^2 that is possible in floating point format (but used as an integer) and NUM_BITS is the number of bits to shift to the right to reduce the precision. There is no sign bit since distance squared (r^2) is always positive otherwise it would be necessary to mask it out or to use two tables for positive and negative results depending on the intended use. The range of r^2 is continuous and over a single range so no additional processing is required for boundary conditions or multiple table lookups for a single function.

$$F(r) \approx F_{tab}\,[(\text{r2.binary} - \text{LOWER\_BOUND}) >> \text{NUM\_BITS}].\text{float}$$

Multiple intermediate results can be stored as offsets in the same table eliminating the instructions that reload of the base address of the table. Using SSE2, SSE 4.1 and AVX2 this can be reduced from 11 to 7 to 3 instructions respectively that can be highly pipelined.

A number of these instructions operate in less than one clock cycle. SSE2 and SSE 4.1 can process four single precision values at one and AVX2 can process 8 values at once. The algorithm uses SIMD instructions for integer and logical operations that do not exist in either the SSE or AVX instruction set extensions. The algorithm can be coded to avoid the need for these instructions but with a performance penalty because of the increased number of instructions and memory accesses. This project did not evaluate the performance impact of using either SSE or AVX instruction sets. It was noted that the performance of the SSE2 and SSE 4.1 instruction set implementations on Intel Sandy Bridge and Haswell architecture CPUs was much higher than would have been predicted from instruction timings. This was possibly due to improvements in cache architecture and micro-operation (μops) processing.

The tables of intermediate results are generated by sequencing through all of the possible values of the variable precision representation so that the results 'float' in the same way as the IEEE 32 and 64 bit formats. As the table generation is performed, a simulated additional 'guard bit' to the right of the reduced precision mantissa is used before the calculation to initialize the table alternating on odd and even values of the mantissa.

This prevents a small divergent error from the binary truncation that occurs at runtime otherwise it would be necessary to treat the mantissa as a scaled base 2 value at runtime and this would defeat much of the performance gain of the algorithm.

All calculations for table entries are performed in double precision and the results stored in the tables as either single or double precision. No interpolation is required. The following diagram illustrates the 'floating' nature of the implementation. The mantissa portion of the IEEE 764 standard single precision format supports approximately 8M values for each power of two. By reducing the mantissa bits it is possible to reduce the precision without destroying the 'floating' property of the format. This makes the variable format suitable for use with functions/equations independent of their slope or continuity. When multiple results are used as part of incremental computation at runtime a small number of computer instructions are used to assemble or provide minimal calculation to solve equations or functions.

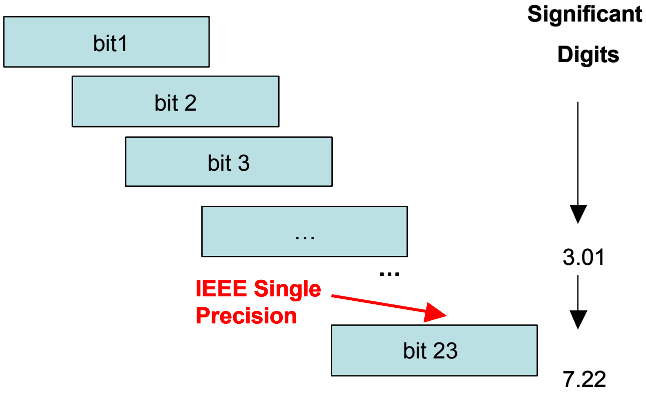


Figure 14 Variable Precision Mantissa 'Floats' Like IEEE

An alternative implementation would be to perform an integer conversion of the floating point value applying a scaling factor. In the following example the *int* function converts the results of the scaling factor times distance squared and looks the result up in a force table.

$$F(r) \approx F_{tab}\,[\mathrm{int}\,(\mathrm{scale} * r^2)]$$

Converting a floating point value to an integer causes a divergence from the IEEE standard and destroys the ‘float’ properties of the values and generates a diversion from the results obtained from those that would be generated by a floating point calculation. This diversion is highly function specific. Using a floating point to integer conversion results in fewer and fewer values to represent much larger forces as the distances become closer. When the algorithm is applied to other functions/equations the variations of the slope in portions of the functions/equations will show large differences in the offsetting error correction based on the slope on the axis.

In his work, Nilsson[40] applied an interpolation algorithm that uses an additional table and additional code at runtime to attempt to overcome this problem. His interpolation model was coded as follows.

Direct lookup

$$F(r) \approx F_{tab}[\text{index}]$$

Linear interpolation

$$F(r) \approx F_{tab}[\text{index}] + \alpha * (F_{tab}[\text{index}+1] - F_{tab}[\text{index})]) \equiv F_{tab}[\text{index}] + \alpha * \Delta F_{tab}[\text{index}]$$

Where

$$\text{index} = \text{int}(\text{scale} * r^2)$$

$$\alpha = \text{scale} * r^2 - \text{index}$$

If the integer conversion approach, even with linear interpolation, is applied to general computation the index will not appropriately sample the distribution of the function or equation's results based on the IEEE floating point properties.

It is highly undesirable to use an integer conversion to produce an index using a specific value when there are substantial differences in the results contained in the table that are being looked up. In this example there are larger and larger differences in the forces as the distance becomes smaller and fewer base 10 digits to represent the increasingly large forces.

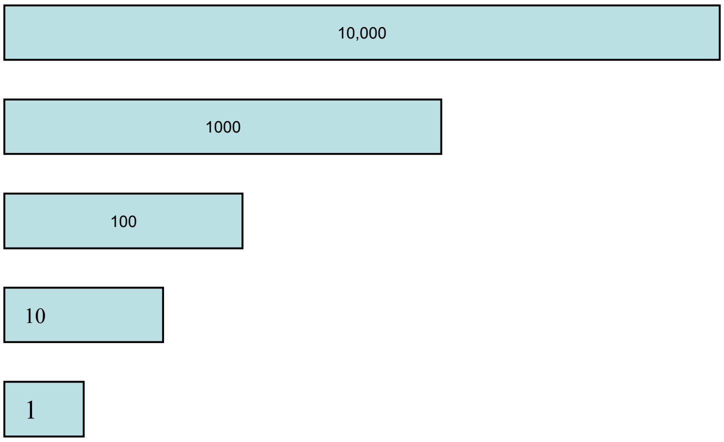


Figure 15 Effect of Integer Conversion on a Real Number

There is no ‘floating’ property in an integer conversion that would preserve the same precision independent of the distance. Figure 16 shows the increasing forward backward error correction needed with the 1/sqrt function that is at the core of many object interaction problems.

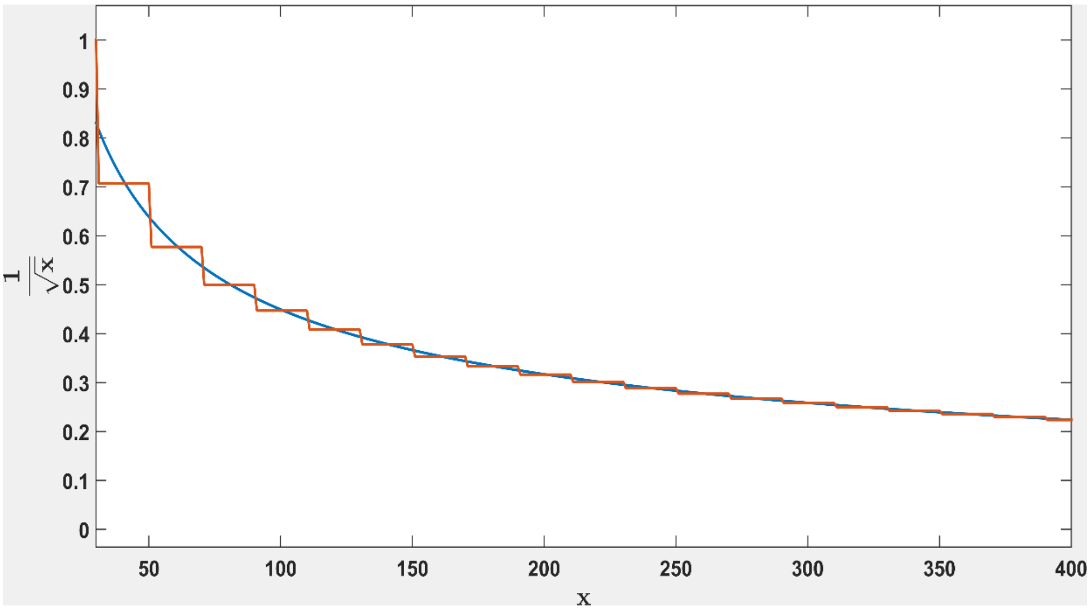
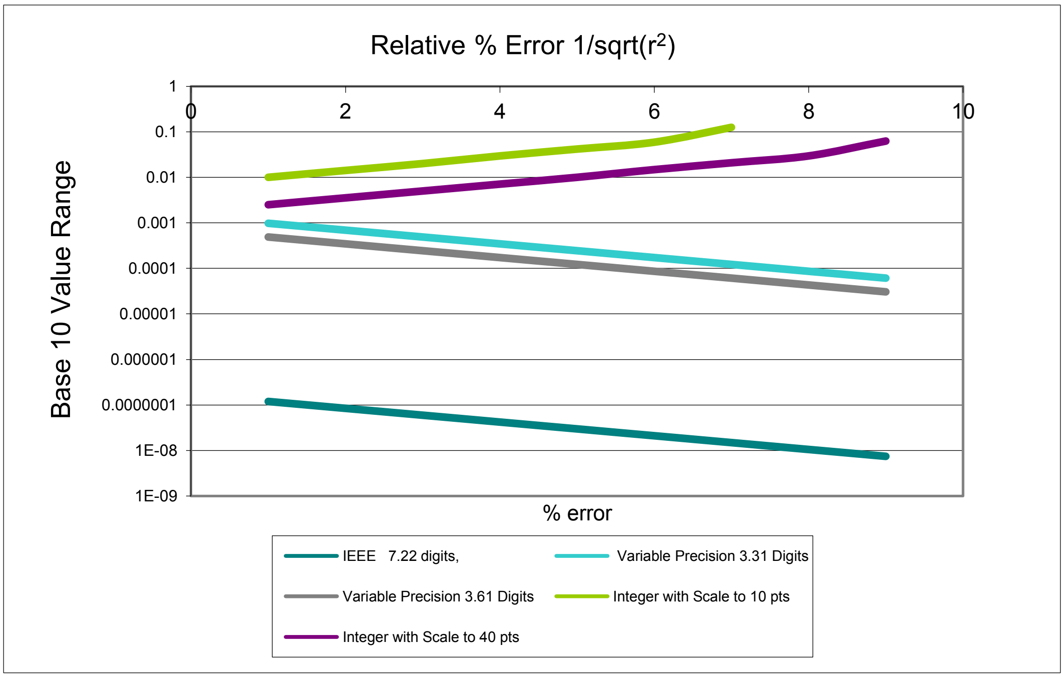


Figure 16 Increasing Deviations in 1/sqrt function with Integer Conversion Method

Figure 17 shows the variable precision implementation paralleling the floating point standard and an integer conversion based method diverging until it reaches a lower limit cutoff that is dependent on the scaling. This cutoff does not correspond to any



specified simulation parameters but is a result of the scaling before the integer conversion occurs.

Another of the side effects of an integer conversion method is the elimination of cases where a base 10 value may have more than one base 2 representation or it may have no base 2 representation at all. This property of the floating point representation is essential to preserving the same number of significant digits as values increase or decrease. The following table shows the nature of this property. These cohorts are the reason that programmers can not readily compare floating point variables with the same base 10 representation because they may have different internal representations.

Base 10	Base 10 Decimal	Base 2 Hexadecimal
2.500003e-01	0.2500000	3e80000b
2.500004e-01	0.2500000	3e80000c
2.500004e-01	0.2500000	3e80000d
2.500004e-01	0.2500000	3e80000e
2.500004e-01	0.2500000	3e80000f
2.500005e-01	0.2500000	3e800010
2.500005e-01	0.2500010	3e800011
2.500005e-01	0.2500010	3e800012
2.500006e-01	0.2500010	3e800013

Table 3 Example IEEE Floating Point Cohort

Figure 17 Relative % Error vs. IEEE 754 Single Precision

1.5.2 Performance Evaluation

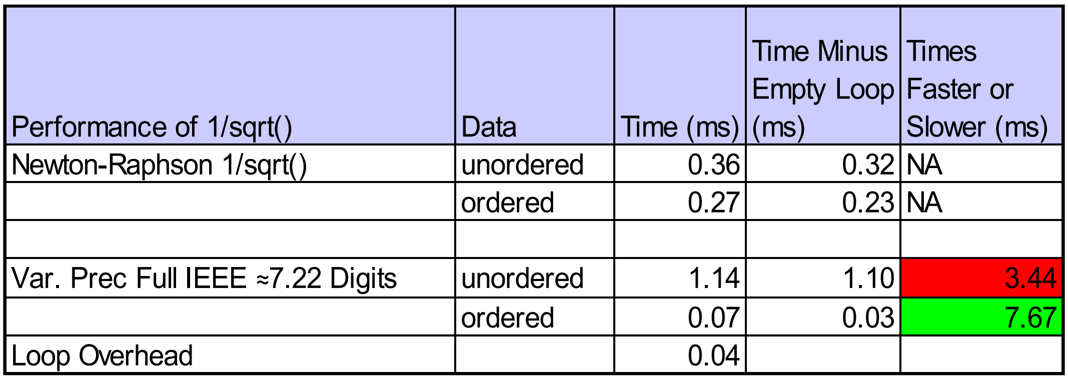
1.5.2.1 Performance Impact of Exceeding the CPU Cache

This algorithm exploits the available high speed CPU L2 or L3 cache memory to store each series of results used at the time the simulation is run.

Care must be taken not to use an excessive amount of cache memory or performance may become worse than computing the results every time.

An initial test was performed on an *ordered* sequence of real number values representing every possible binary representation of the single precision floating point numbers between the lower and upper bounds for the distance squared in our test water simulation. This was compared to an equal number of *unordered* distance squared values from an *actual* simulation. Table 4 shows the performance impact of having the distance squared values unordered.

The following figure shows that performing an index table lookup on an unordered set of values is 3.44 times slower and an ordered set of values is 7.67 times faster even when the full IEEE 754 ≈7.22 significant range is used. The data required for the ≈7.22 range is approximately 56MB. The time to sort them even with a fast single pass binary sort algorithm would make the access prohibitively long to use an indexed lookup at full IEEE 754 precision. The excellent performance even with a very large variable precision cache comes from the ordered nature of the data which benefits

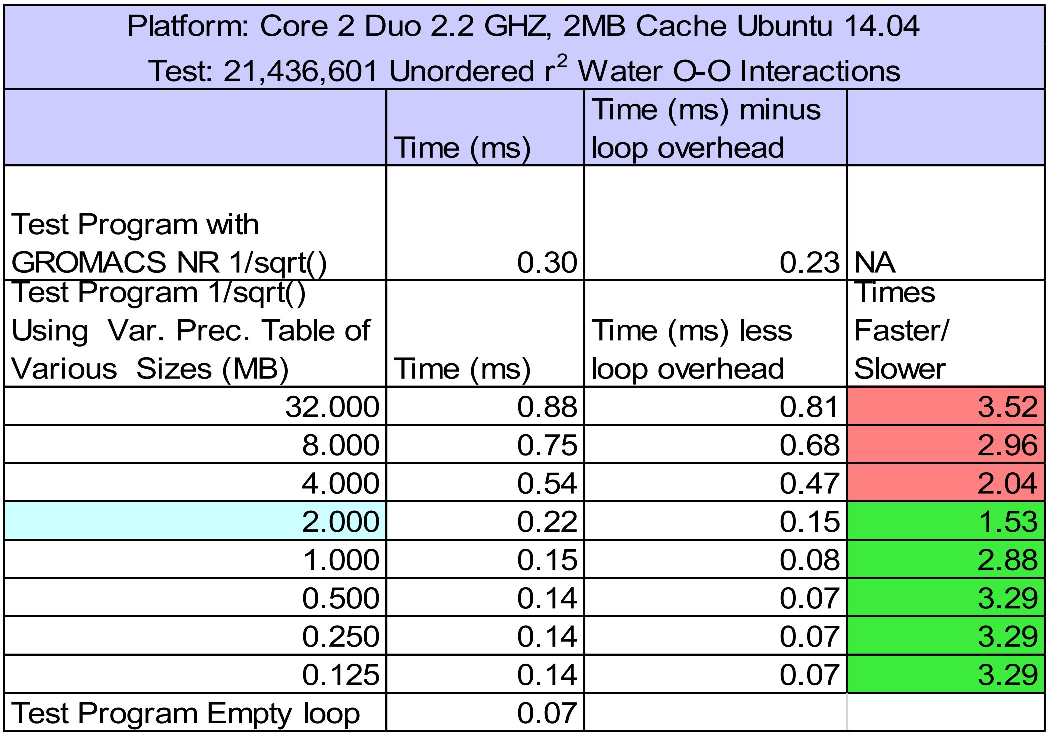


Performance of 1/sqrt()	Data	Time (ms)	Time Minus Empty Loop (ms)	Times Faster or Slower (ms)
Newton-Raphson 1/sqrt()	unordered	0.36	0.32	NA
	ordered	0.27	0.23	NA
Var. Prec Full IEEE ≈7.22 Digits	unordered	1.14	1.10	3.44
	ordered	0.07	0.03	7.67
Loop Overhead		0.04		

Table 4 Impact of Ordered vs. Unordered Values

from the memory pre-fetch and design of CPU caches and memory systems that are optimized for sequential access.

The only solution to the problem of unordered data using full IEEE single precision caches was to reduce the ‘granularity’ of the distance squared. Table 5 shows the impact of exceeding the 2MB CPU cache of an Intel Core 2 Duo 2.2GHZ processor even using a variable precision algorithm. The test was conducted on 21M unordered distance squared values for oxygen to oxygen interactions.



Platform: Core 2 Duo 2.2 GHZ, 2MB Cache Ubuntu 14.04 Test: 21,436,601 Unordered r^2 Water O-O Interactions			
	Time (ms)	Time (ms) minus loop overhead	
Test Program with GROMACS NR 1/sqrt()	0.30	0.23	NA
Test Program 1/sqrt() Using Var. Prec. Table of Various Sizes (MB)	Time (ms)	Time (ms) less loop overhead	Times Faster/ Slower
32.000	0.88	0.81	3.52
8.000	0.75	0.68	2.96
4.000	0.54	0.47	2.04
2.000	0.22	0.15	1.53
1.000	0.15	0.08	2.88
0.500	0.14	0.07	3.29
0.250	0.14	0.07	3.29
0.125	0.14	0.07	3.29
Test Program Empty loop	0.07		

Table 5 Impact of Exceeding the L2/L3 CPU Cache on Performance

This methodology works well when the pre-computed results can be contained within the CPU's L2/L3 cache. Figure 18 illustrates what happens to performance if the limits of the L2/L3 cache are exceeded. The difference between the Sandy Bridge and Haswell architecture is possibly due to improvements in the cache memory design and performance. Some architectures such as the IBM z196 have a very large L4 cache. This study evaluated Intel and AMD CPUs only but it could be easily be implemented on other architectures. This methodology is limited by the amount of L2/L3/L4 cache memory available. In the following example the CPU L2/L3 cache becomes fully utilized at a variable precision of about ≈5.12 significant digits. Other processors have greater L2/L3 cache memories that allow more extensive use of pre-calculated and incrementally calculated results. Even on older architecture CPUs with only 512KB of cache memory the algorithm can be used for water to water interactions (W3A-W3A Lennard-Jones reaction field) and also some intrinsic math functions over limited ranges. The slightly lower performance in Figure 18 in the range of ≈3.01 to ≈4.82 significant digits was repeatable on the Intel Core i7 "Sandy Bridge" and attributable to the CPU architecture. Other Intel and AMD processors also showed similar behavior but at different ranges of significant digits. This is probably due to differences in the CPU cache memory system design.

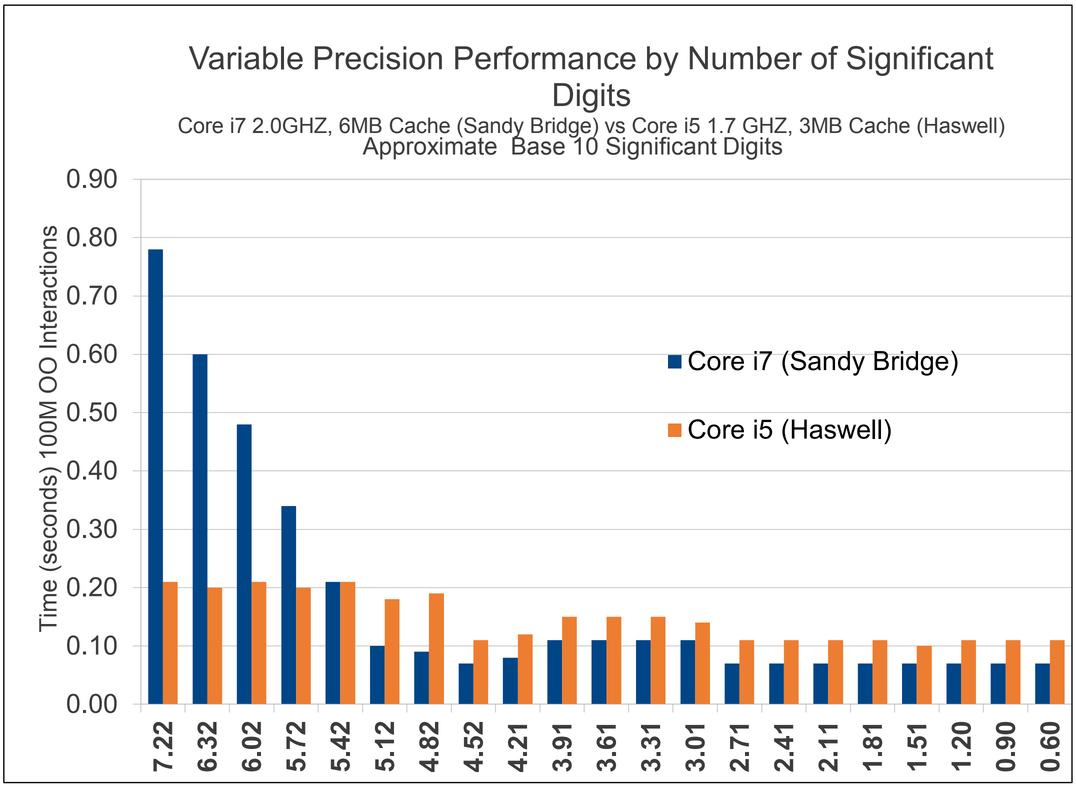


Figure 18 Impact of Exceeding CPU Cache Size Using Variable Precision

1.5.2.2 Improving Reciprocal Throughput

Using the algorithm can greatly reduce the number of instructions required to generate the object interaction results. The reciprocal throughput analysis is based on the work of Agner Fog.[49] Reciprocal throughput is one measure of performance.

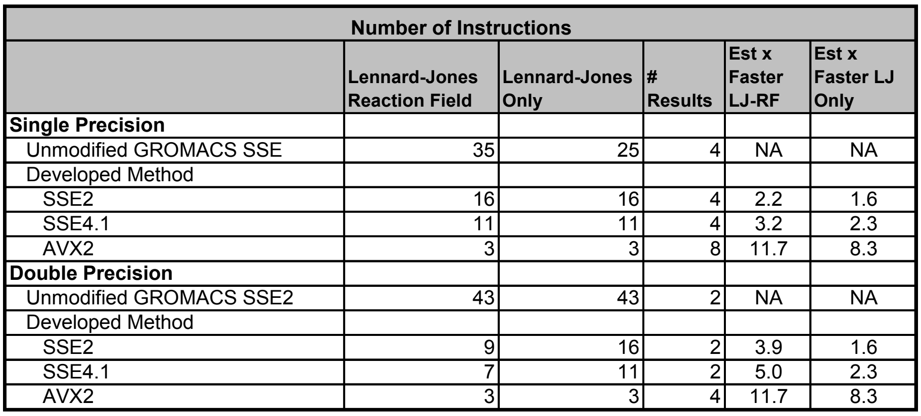
Reciprocal throughput has been defined as the average number of clock cycles per instruction for a series of independent instructions of the same kind in the same thread on a single core assuming that the operands of each instruction are independent of preceding instructions. The values used in this study are from Agner Fog's independent evaluation of the performance of a wide range of Intel and AMD processors. The values used are the reciprocals of the throughputs when the instructions are not part of a limiting chain of dependent instructions. For example, assuming that the operands are independent a reciprocal throughput of 2 cycles for an FMUL instruction means that another FMUL instruction can start 2 clock cycles after the previous FMUL and a value of 0.33 for ADD means that 3 integer additions can be performed per clock cycle.

Thus, the sum of the instruction cycles that a given algorithm uses may be used for a relative comparison but with limitations. One major limitation is memory architecture and whether or not the data is available in one of the levels of cache memory. Memory access takes 2-3 cycles if cached but several hundred if not. [50]

In principle if the number of computer instructions and the number of 'cycles' are reduced then the software should run faster. This however may be a deceiving measure because the number of cycles per instruction even for the same instruction varies greatly on what instructions are around it and where it is retrieving data. For example, modern CPUs will attempt to optimize on chip performance by reordering instructions, performing operations in parallel or attempting to predict branching. CPU instructions execute using micro operations that may be scheduled in parallel with neighboring instructions to avoid 'blocking' of program execution. Part of an instruction may execute in parallel with part of another instruction based on micro operations.[51]

Generally non-arithmetic instructions take fewer cycles than numeric instructions and are more easily optimized in the CPU pipeline frequently executing in less than one cycle. The number of cycles an instruction takes is also highly dependent on where the data resides. If the data is in the L2/L3 cache execution is very fast, but if it resides in main system memory the memory access could be 100 times slower.

Using the algorithm can greatly reduce the number of instructions required to generate the nonbonded interaction results. It may be noted in Table 6 that the SSE and AVX instruction sets are not included from testing. This is because they lack instructions for performing certain SIMD bit manipulation instructions that Intel later added in SSE2 and AVX2 that are essential to the developed algorithm. The following figure shows the reduction in the number of instructions required to obtain the equation results as compared with the GROMACS 4.5.3 assembly language code for atomic level interactions. Most of the instructions have a cycle time of 1 but the developed algorithm uses numerous memory and register instructions that have cycle times of .33 on Intel Sandy Bridge and .22 on the Intel Haswell architecture further improving the performance.



Number of Instructions					
	Lennard-Jones Reaction Field	**Lennard-Jones Only**	**# Results**	**Est x Faster LJ-RF**	**Est x Faster LJ Only**
Single Precision					
Unmodified GROMACS SSE	35	25	4	NA	NA
Developed Method					
SSE2	16	16	4	2.2	1.6
SSE4.1	11	11	4	3.2	2.3
AVX2	3	3	8	11.7	8.3
Double Precision					
Unmodified GROMACS SSE2	43	43	2	NA	NA
Developed Method					
SSE2	9	16	2	3.9	1.6
SSE4.1	7	11	2	5.0	2.3
AVX2	3	3	4	11.7	8.3

Table 6 Estimated Instruction Times

Table 6 shows a comparison of the number of instructions per result required to calculate the nonbonded interactions for two water molecules including the 1/sqrt and the Lennard-Jones and reaction field equations. Argon to argon interactions are also shown that only need to solve the Lennard-Jones equation. Figure 19 shows the estimated differences in performance based on instruction times. These are *only estimates* based on instruction times and *do not reflect* actual algorithmic performance. It is also important to note the number of results that are returned. For example, the AVX2 implementation can return 8 single precision values of 32 bits each and can return 4 double precision values of 64 bits each.

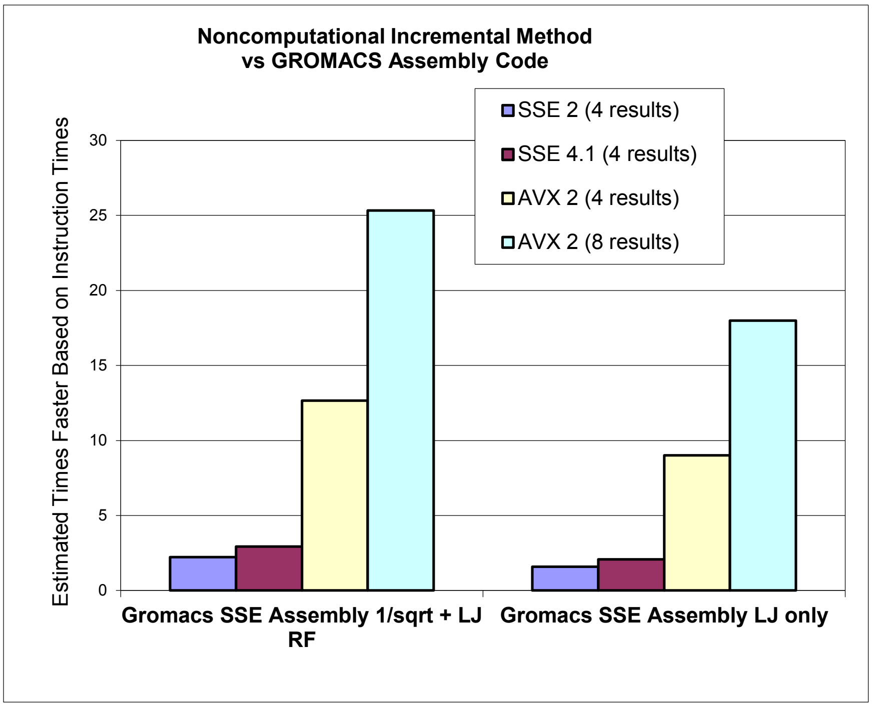


Figure 19 Performance Estimates Per Result Based on Instruction Count

1.5.3 Runtime Evaluation

In order to test application level performance a program was written that reads 10^8 oxygen to oxygen interactions and then processes them inside a timing loop. This was done in assembly code using the GROMACS version 4.5.3 SSE2 assembly code copied from the nonbonded kernel routines for Lennard-Jones reaction field, and Lennard-Jones only routine as well as for the noncomputational incremental method using SSE2 and SSE 4.1 with a special granularity of ≈3.01 significant digits to provide equivalent results.

The same test application was used to evaluate the performance of the developed algorithm when it was applied to basic C library math functions. The results are shown in Figure 20 comparing the developed algorithm versus conventional calculation. The Newton-Raphson 1/sqrt and Nilsson's floating point to integer conversion index lookup method are indicated in the chart below with labels 'NR 1/sqrt' and 'LN JCC' respectively. The chart also shows one of the limits of the developed method. The formula for the volume of a sphere 4/3 π r^3 is substantially slower than the developed method because there are so few instructions involved in

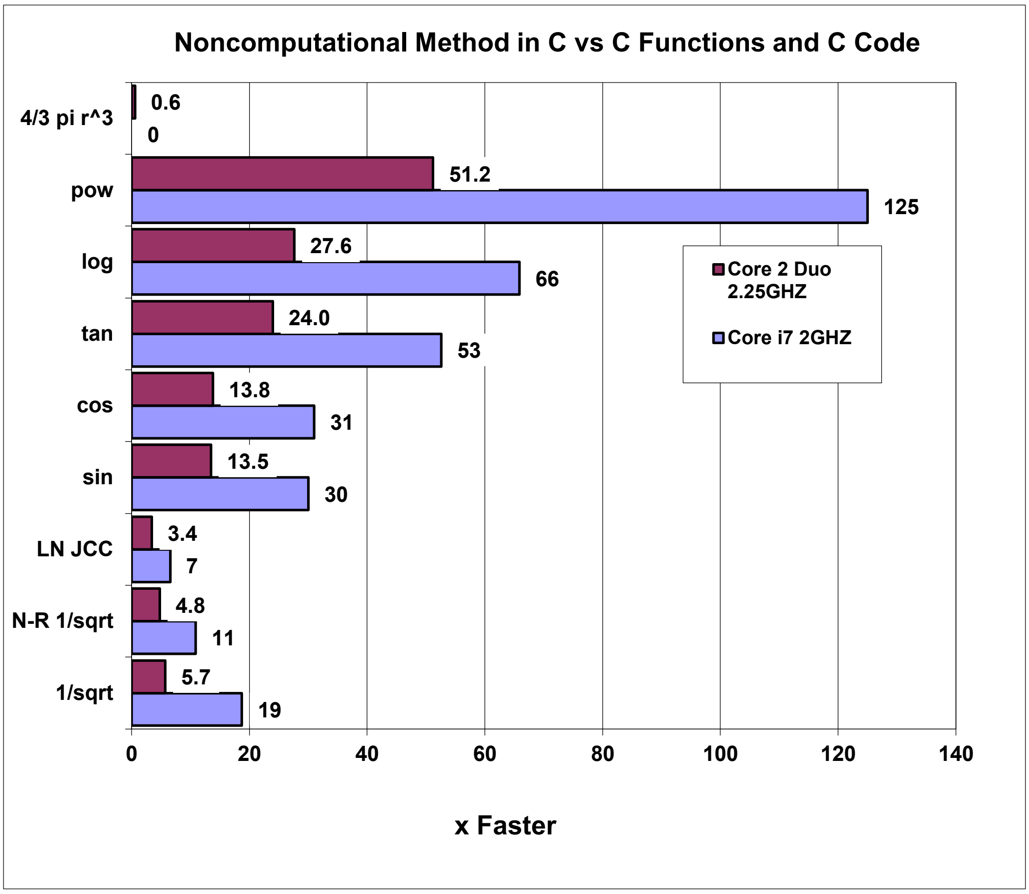


Figure 20 Noncomputational vs Computational Math Functions

the calculation and because the memory access time is slower than the calculation time.

1.5.3.1 Comparison of Force Only Tests

The force only tests for the nonbonded routines (excluding the distance calculations and the application of the forces after calculation) are presented in Table 7. The unmodified application SSE code was copied and pasted into the test application and changed only to reference local variables. The Lennard-Jones and reaction field force equations included the 1/sqrt calculation typical of force based object interaction applications. The noncomputational incremental method was also written in hand coded assembly code and inserted into the timing loop of the program.

The very large improvement in force calculation performance is due to the fact that the method does not require the calculation of the 1/sqrt, the reduced number of instructions and the benefit that the binary and integer operations receive in the CPU pipeline. This should be typical of other force based object interaction applications

The force only calculations represent only portion of the calculations within the inner processing routines. Distance calculations cannot benefit from the use of the developed method but can benefit in some architectures from fused multiply and add (FMA) instructions.

Time to Process 100M O-O Interactions		I7 server 2.67 GHZ	
Forces Only	**Time (sec)**	**Time less empty loop**	**x Faster**
Unmodified Applicaton SSE LJ Only(cut and pasted into test app)	7.9	7.76	
Incremental Method Equivalent (SSE 4.1) LJ Only Forces	0.38	0.24	32.33
Incremental Method (SSE2) Equivalent LJ Only Forces	0.39	0.25	31.04
Unmodified Applications SSE LJ + Reaction Field + 1/sqrt cut pasted into test app	16.78	16.64	
Incremental Method (SSE 4.1) Equivalent LJ + reaction Field + 1/sqrt	0.38	0.24	69.33
Incremental Method (SSE2) Equivalent LJ + reaction Field + 1/sqrt	0.39	0.25	66.56
Empty Loop	0.14		

Table 7 Performance of O-O Interactions

Intel Xeon 5650 2.67 GHZ 12MB Cache 6 cores (Westmere).

1.5.3.2 Comparison with the Full Inner Object Interaction Routines in GROMACS

The same performance test method was used for the assembly code to process *all* of the code within the inner most object interaction routine from GROMACS for the nonbonded routines. This code was also copied and pasted from the unmodified GROMACS inner processing routines into the timing loop of the test program. In order to test the noncomputational incremental method the same GROMACS assembly code was copied but the portions that calculate the 1/sqrt function and perform force calculations were replaced with the developed algorithm. The performance improvements of 2.15 and 3.18 times faster is in line with the reciprocal throughput estimate on the Intel Xeon 5650 2.67 GHZ 12MB Cache 6 cores (Westmere).

Time to Process 100M Interactions Intel Xeon 5650 server 2.67 GHZ			
Full Equivalent of the Inner Object Interaction Processes (Nonbonded forces in Assembly Code)	**Time (sec)**	**Time less empty loop**	**x Faster**
Unmodified Application SSE LJ Forces Only Cut and Pasted into Test App	21.00	20.86	
Incremental Method (SSE4.1) Equivalent LJ Forces Only	2.70	2.56	8.15
Incremental Method (SSE2) Equivalent LJ Forces Only	2.66	2.52	8.28
Unmodified Application SSE LJ + Reaction Field + 1/sqrt cut pasted into test app	23.02	22.88	
Incremental Method (SSE4.1) Equivalent LJ Reaction Field + 1/sqrt	6.70	6.56	3.18
Incremental Method (SSE2) Equivalent LJ Reaction Field + 1/sqrt	9.84	9.70	2.15
Empty Loop	0.14		

Table 8 Assembly Code Algorithm vs GROMACS SSE
Intel Xeon 5650 2.67 GHZ 12MB Cache 6 cores (Westmere)

1.5.3.3 Intel Core i7 'Sandy Bridge' and AMD Performance

The same tests were performed on the Intel Core i7 'Sandy Bridge' architecture that supports the AVX instruction set and on two AMD 64 core systems. The results showed an extreme improvement in performance between the generations of the Intel Core i7 architecture. The 'Sandy Bridge' architecture processed the code for the full interaction calculating the 1/sqrt, Lennard-Jones and reaction field forces 14.3 times faster and for Lennard-Jones force interactions 15.5 times faster than the unmodified GROMACS 4.5.3 SSE assembly code. It was also observed that the two AMD CPUs tested only performed about 1.7 times faster. The 64 core AMD server show the same performance improvement ratio as older AMD CPUs. There are fundamental differences between the AMD and Intel cache architectures that account for this. There were no values reported for the AMD CPUs for SSE 4.1 instructions because SSE 4.1 only exists on Intel CPUs. It is interesting to note that the SSE 4.1 implementation on Sandy Bridge was slightly *slower* than the SSE2 implementation even though fewer instructions were used.

The Intel 'Sandy Bridge' architecture has a number of improvements that may account for this difference. The most important is probably the improvements in the cache design, larger CPU cache and wider data paths. It functions almost as if it is processing 8 single precision values simultaneously rather than 4 even though no AVX instructions were coded into the test program and AVX2 instructions are not supported on Intel Core i7 Sandy Bridge. Figure 21 does not include testing with AVX or AVX2 instructions.

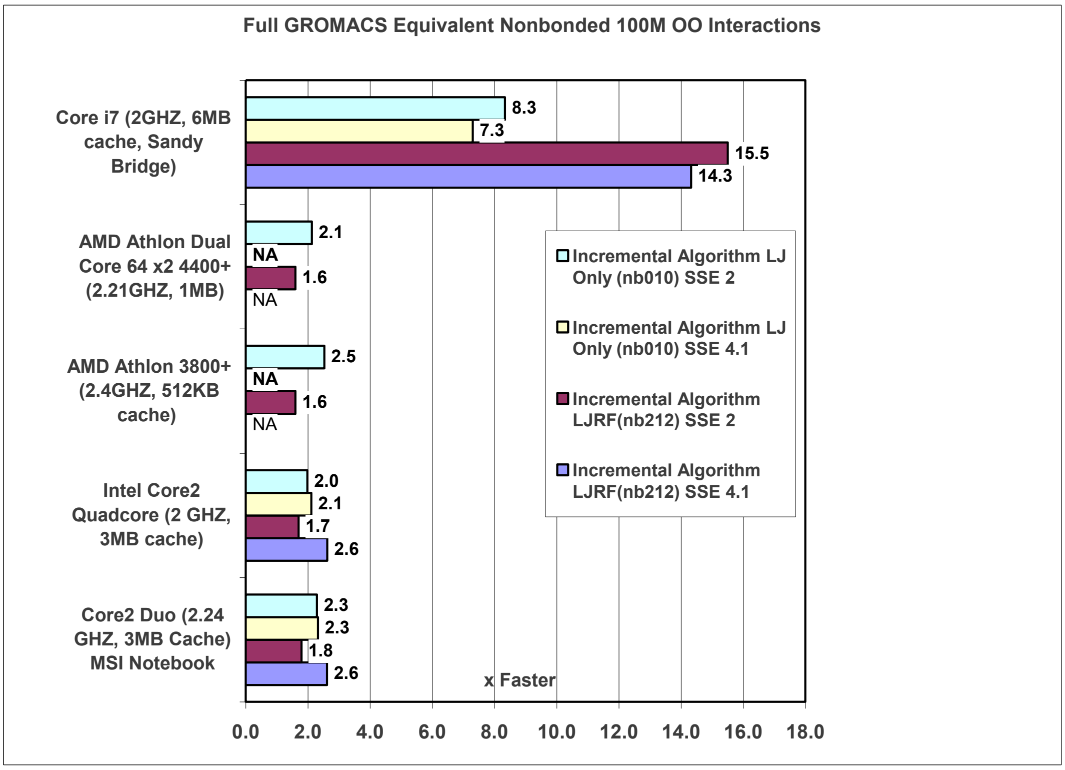


Figure 21 Developed Algorithm in Assembly vs GROMACS SSE Assembly

1.6 Limitations of the Methodology

Limitations Based On Application and CPU Architecture

The developed algorithm has a number of limitations. The most significant is the amount of CPU L2, L3, or L4 cache available for use with the lookup tables for the intermediate results. This study showed that there was a substantial reduction in performance as compared to actually computing the results when the tables used for the incremental results exceeded the CPU L2/L3 cache. It should also be considered that other applications may be using the core or processor that may cause cache misses for an application using the incremental lookup tables. Attention should be given to associating threads/processes using the algorithm with a physical core, CPU block and blade/node. If processes are allowed to move from core to core cache misses will result and this will have an adverse impact on performance. Therefore, associating a thread/process with a core and cache is essential. Differences between CPU cache designs and vendor 'families' of CPUs will also have significant impact on algorithmic performance as noted in the comparison of AMD and Intel CPUs.

Use of the algorithm in virtual machine environments with multiple virtual cores mapped to a single physical core was not evaluated. In a virtual environment it may be difficult or impossible to associate a thread/process with a physical core. If this cannot be done there will likely be a performance penalty.

The methodology requires application specific knowledge to allow the creation of reduced precision lookup tables for intermediate results. If this is not available it must be obtained by instrumenting the application to use data for the specific execution being performed. This is similar to the optimization that is done in some FFT libraries[52] and to the query execution plans that are supported in SQL databases[53].

The algorithm can currently be used only if the number of significant digits is less than or equal to ≈5.12 significant digits, or only one or few results/intermediate results are needed due to currently available L2/L3 CPU cache sizes. The exponent range must also be known so that each level representing a power of two can be initialized properly. The exponents do not need to be contiguous.

Only one floating point/integer parameter may generally be used as input to the developed algorithm for each result/intermediate result at runtime unless the granularity of the parameter is *extremely* coarse or the range of values is extremely limited. Use of more than one parameter with the developed algorithm results in a two or more dimensional array for use as a lookup table and the amount of memory required rapidly becomes large. All other values required for each function used to initialize the lookup tables must also be known and static at runtime.

The implementation of the table initialization and retrieval routines can be implemented to use fewer exponent levels of the IEEE 754 standard. If 80% of a function/equation's execution occurs within a range of values then only the exponent levels required to support the 80% need to be implemented as lookup tables and in all other cases the results would be returned by computation.

Functions that are cyclic can be implemented using smaller lookup tables and a small amount of additional code to manipulate the index value. Typically this code would consist of integer, and logical operations (and, or, shift) that execute in less than one cycle.

The AVX2 instruction set using the vsgather instruction and AVX2 integer and bit manipulations gives the optimal level of performance by significantly reducing the number of computer instructions.

1.7 Addressing the Limitations of the Methodology in Hardware

There are numerous processor based hardware architectures and platforms that could benefit from implementing the developed functionality in hardware. They include CPU (e.g. Intel, AMD, and IBM), GPU (e.g. NVIDIA, ATI, and Intel), smartphones, specialized ASICs and other processing hardware. There are two principal implementation requirements: 1) dedicated L2/L3/L4 cache or a high speed GPU/other device shared memory system (on-chip or off-chip) for application use so that cache misses would be eliminated and 2) additional processor instructions to manage the application caches and also to reduce the number of instructions required to obtain the incremental results.

Generalized software interfaces can be provided that are hardware independent vendor specific implementations. These software interfaces would represent an abstraction layer hiding an efficient hardware specific implementation and in the case that there was no hardware implementation they would use a software only implementation.

These improvements could be a major advance in general purpose computing giving developers the ability to implement their own problem domain specific noncomputational incremental solutions to equations and functions. This approach is similar to the microcoding capabilities that were available on early CPUs that allowed users to build their own instructions.

1.7.1 LxA Application Cache

There are several ways in which a Level x Application (LxA) cache could be supported. (x indicates that the cache is part of or parallels an existing processor cache such as L2, L3, L4, or vendor defined shared memory region). The general requirements for LxA caches are that they be large enough to be useful to store incremental results based on variable precision data formats. When implemented in hardware it should ideally also be possible to exploit the read only properties of the LxA caches after the cache is allocated and initialized so that many cores could share higher speed read only access after the cache is allocated and initialized.

This study showed that multiple tables with incremental results in the range of a few megabytes are sufficient to optimize HPC object interaction based applications. This technique could also be applied to other HPC applications where there is a high degree of interdependency between data processed between cores, processors or nodes. An LxA application cache should be at least 1-2MB.

1.7.2 Hardware Implementation Alternatives

The simplest and most straight forward approach to implementation would be to allocate a portion of the currently implemented processor or shared memory caches for application use and to add the additional instructions specified in sections 1.7.4 to 1.7.6.

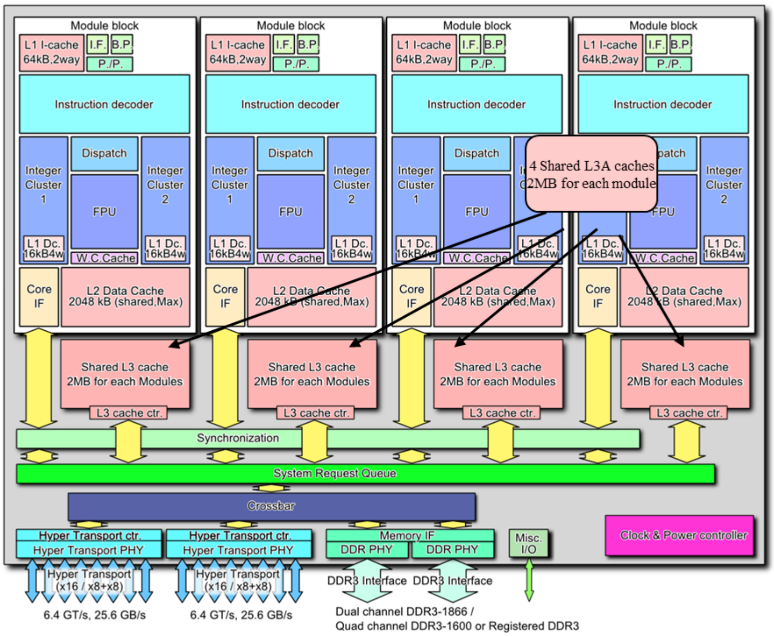
Alternative implementations would be to layer or tile another cache alongside the existing processor caches. Layering and tiling has been successfully used in processor architectures for other purposes and could be useful in implementing LxA caches.

1) Allocation of a portion of the current L2, L3, L4 or other processor/GPU caches or high speed shared memory can only be done in architectures where there are large CPU caches or large amounts of high speed shared memory. L1 instruction and data caches are too small to be useful for dedicated general purpose application use. This LxA cache is referenced as cache_id in the pseudo instruction definitions

2) An additional application level cache paralleling the existing processor, GPU or vendor specified caches or shared memory.

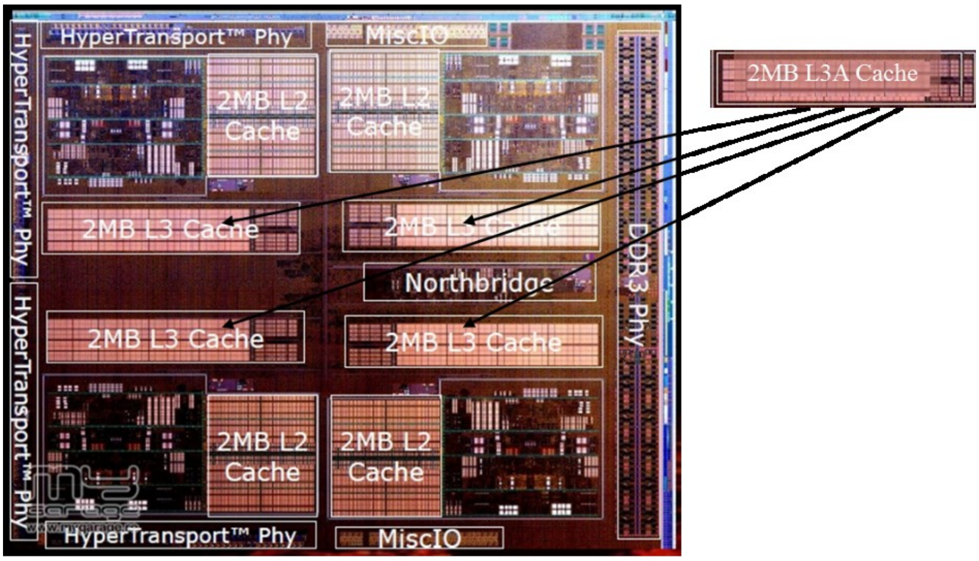
1.7.3 Design Approaches

There are several approaches for design and fabrication: on-chip as an allocated portion of the existing caches, layered above an existing cache/memory section on-chip, or off-chip. The caches could also be ‘tiled’, a technique used in system on a chip (SoC). Figures 22[54], 23[55], 24[56], 25[57], 26,[58] and 27[59] illustrate some possible design layouts and/or fabrication alternatives.



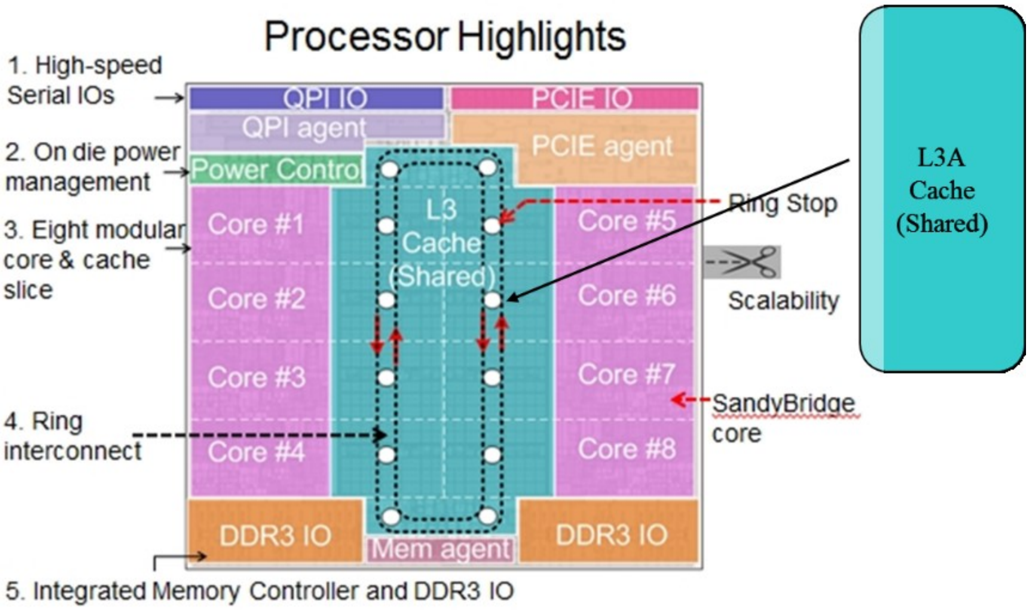
Each AMD Bulldozer with a 2MB Shared L3A Cache Paralleling the General Purpose Shared L3 Caches. Caches Overlaid with the same latency and distance from controller

Figure 22 Example AMD Bulldozer with LxA (L3A) Application Caches



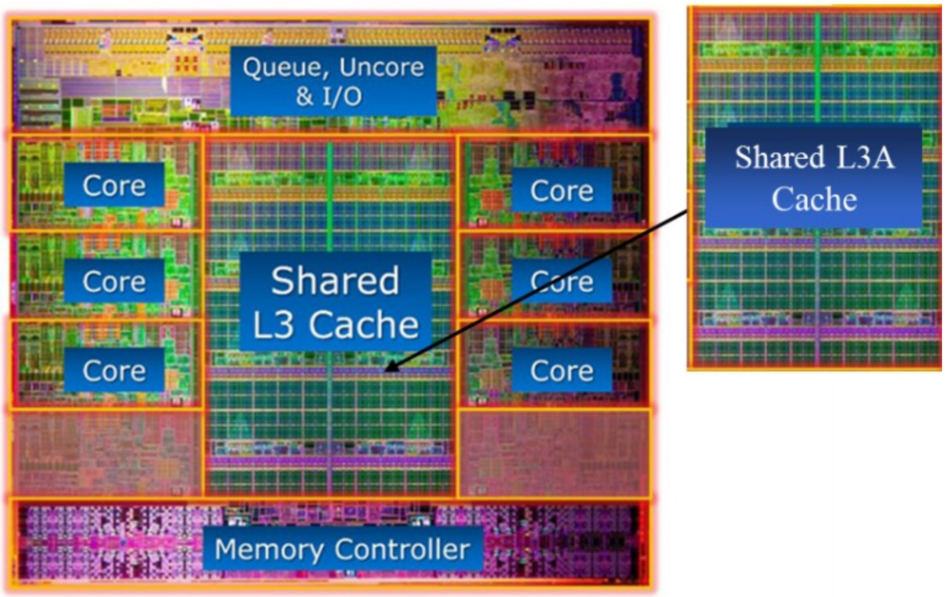
Each AMD Bulldozer Module Block with a 2MB Shared L3A Cache Paralleling each of the General Purpose Shared L3 Caches with the same latency and distance from controller

Figure 23 Example AMD Bulldozer with L3A Cache Die Detail



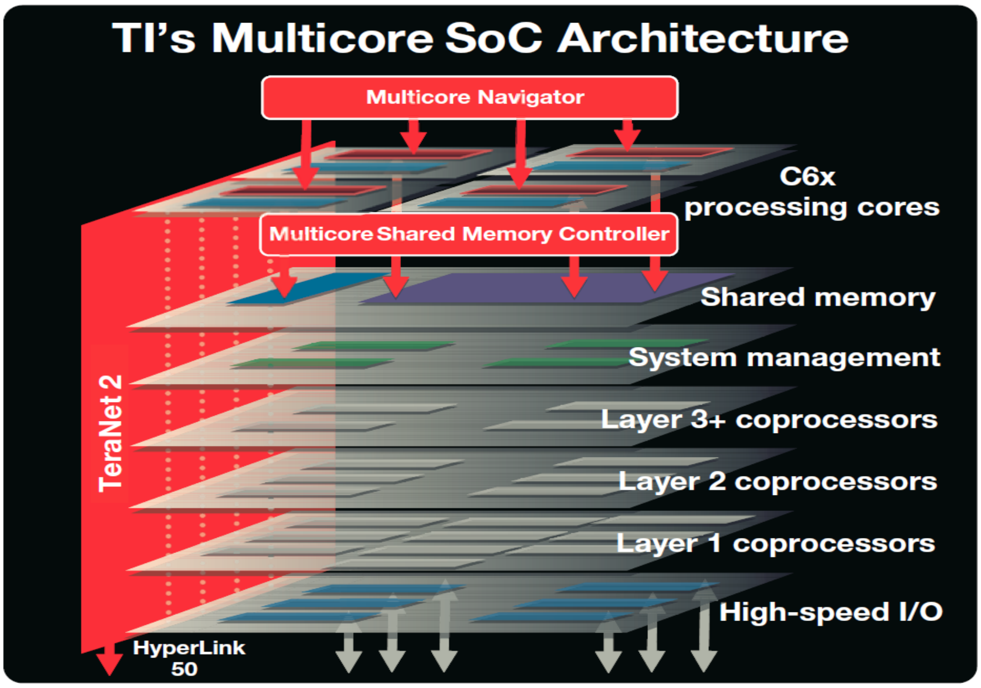
Shared L3A Cache Paralleling the General Purpose Intel Shared L3 Cache Across Cores. L3A Overlaid with the same latency and distance from controller

Figure 24 Example Intel Sandy Bridge Xeon Block Diagram with L3A Cache



Shared L3A Cache Paralleling the General Purpose Intel Shared L3 Cache **Across Cores. L3A Overlaid with the same latency and distance from controller**

Figure 25 Example Intel Core i7-390X Processor with L3A Die Detail



***1.** Multicore, multilayer architecture*

Figure 26 Texas Instruments Multicore, Multi-layer Chip LxA Could Be Added as a Second Shared Memory Layer

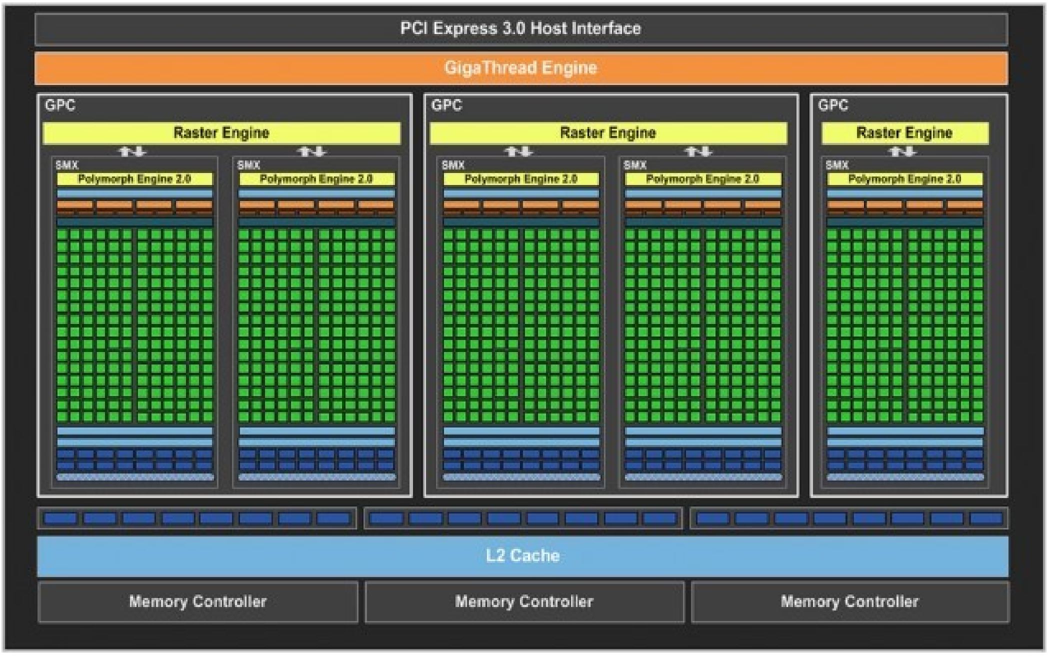


Figure 27 NVIDIA Kepler GPU Architecture L2A Cache Could be Overlaid or Parallel L2

1.7.4 LxA Cache Management Instructions

Pseudo Instructions

Lock/Unlock

LxA lock and unlock instructions are used for the management of the physical cache resource. These instructions manage the allocation of LxA memory resources. A software layer should be used as a centralized tool for dynamically allocating, managing and reorganizing the LxA cache resource.

LCKLXA32 cache_id, address, size
LCKLXA64 cache_id, address, size

> Sets the LxA cache address and memory allocation size for write and sets the LxA flag to locked for the specific LxA or vendor specified shared memory level. The size may be specified as a 32 or 64 bit integer. Results in a (NOP) if the LxA cache is already locked.

ULKLXA cache_id, address, size

> Unlock the LxA cache_id - clears the address and the LxA lock flag. If it is not locked to the specified, address, results the instruction results in a non-operation (NOP).

LXA Cache Status

LDLXAF register/flag

Returns the LxA Lock status. locked = 1, unlocked 0 or flag set or cleared.

LDLXAA register/address

Returns the current LxA address. If not locked the register or memory is unchanged.

LDLXAS32 register/memory
LDLXAS64 register/memory

Returns the 32bit/64bit integer size of the LxA cache.

Branch Instructions

JLXAL address

Jump if LxA cache is locked

JLXANL address

Jump if LxA cache is not locked

LxA Cache could be implemented using a write from only one core using existing processor memory instructions. By default core 0 could used to manage locks. LxA caches should be designed to have multi-way read only access for all cores on the chip/module. Applications using LxA cache need to be associated with a core and block that has the LxA cache. The association of an LxA cache in a physical processor cache or shared memory such as on a GPU would vendor specified.

1.7.5 Gather and Scatter Instructions for Use with LxA Caches

In order for the noncomputational incremental algorithm to execute with the fewest possible instructions the processor needs to support enhanced SIMD gather and scatter instructions. Gather and scatter instructions are already included in the Intel AVX2 instruction set extensions and exist in other processor architectures. These proposed instructions would access the LxA cache as if the LxA cache is any other memory in the application's address space.

Processor vendors such as Intel usually implement one instruction for each of various data types (e.g. single or double precision floating point values, integers). Variations of these instructions should be implemented for the various IEEE 754 data format specifications and for various integer formats supported by the processor.

1.7.6 Instructions to Support Noncomputational Incremental Methods with LxA Caches

Pseudo Instructions

SIMD ‘Noncomputational’ Instructions. The number of instructions required to manipulate data retrieved from noncomputational incremental tables can be reduced by 50% by adding the following instructions. The following instructions are based on SIMD gather instructions with additional capabilities to support the developed methodology.

These instructions provide support for index creation for the tables used by the developed method by providing the following functionality prior to performing the gather operation to retrieve the incremental or final results of a function or equation. These functions are performed in a single instruction.

The base format of instruction may operate on SIMD registers of various sizes and data types and these may be implemented on the processor as multiple instructions based on the number of single/double precision values that may be stored in an SIMD register. For example there may be single and double precision variations of this instruction on Intel platforms targeting xmm, ymm or larger SIMD registers on other Intel processors like Intel Phi.

1. integer subtraction of a base address from the value to be used as an index
2. mask of the value used to build the index
3. shift right to support the method’s variable precision format lookup

vadjust_gather	SIMD_R0, SIMD_R1, SIMD_R2 register, SIMD_R3, mask, immediate

SIMD_R0	Data from gather operation
SIMD_R1	Base address of lookup table
SIMD_R2	Floating Point/Integer Value to be used to construct the index
SIMD_R3/Memory	Mask

Intermediate	Value 1-64 for the right shift operation
(length)	Optional parameter (length of R0 data element)

The vadjust_gather instruction provides a fused integer subtract, right shift, and mask of an SIMD register/memory followed by a gather to retrieve indexed results at a variable precision. The optional length parameter is not required if multiple instructions based on data types are implemented.

The following examples are for variations of this instruction but there could be other variations based on data type:

vadjust_gatherh	half precision results
vadjust_gathers	single precision results
vadjust_gatherd	double precision results
vadjust_gatherq	quadruple precision results
vadjust_gatheri32	32 bit integer results
vadjust_gatheri64	64 bit integer results
vadjust_gatherd32	decimal 32 bit results
vadjust_gatherd64	decimal 64 bit results
vadjust_gatherd128	decimal 128 bit results

vadjust_scatter SIMD_R0, SIMD_R1, SIMD_R2 register, SIMD_R3, mask, immediate

SIMD_R0	Data for scatter operation
SIMD_R1	Base address of lookup table
SIMD_R2	Floating Point/Integer Value to be used to construct the index
SIMD_R3/Memory	Mask
Intermediate	Value 1-64 for the right shift operation
(length)	Optional parameter (length of R0 data element)

This instruction provides a fused integer subtract, right shift, and mask of an SIMD register/memory followed by a scatter of the register values based on a table index using variable precision. The optional length parameter is not required if multiple instructions based on data types are implemented.

The following examples are for variations of this instruction but there could be other variations based on data type:

vadjust_scatterh	half precision values
vadjust_scatters	single precision values
vadjust_scatterd	double precision values
vadjust_scatterq	quadruple precision values
vadjust_scatteri32	32 bit integer values
vadjust_scatteri64	64 bit integer values
vadjust_scatterd32	decimal 32 bit values

vadjust_scatterd64	decimal 64 bit values
vadjust_scatterd128	decimal 128 bit values

1.8 Conclusions

The major bottleneck in HPC applications performing object interactions is frequently the calculation of forces at a time step. HPC applications generally have been highly optimized to perform these functions using the latest instruction sets from the CPU vendors such as SSE4.1, AVX, AVX2, FMA4, etc. The calculation of the forces and intermediate results may be improved substantially by use of a noncomputational and incremental computation model that exploits a variable precision format based on the IEEE 754 standard for single precision values. This variable precision format effectively permits the simulation to run in a reduced granularity of 'space'. Object interaction applications typically do not use the entire range of the IEEE 754 general purpose floating point definition. Applications do not perform calculations at the level of quarks and galaxies at the same time. Object interaction applications use a 'coarse grain' spatial granularity appropriate to the problem being solved. Using a coarse grain approach allows the creation of indices for accessing pre-computed results without the artifacts associated with a simple conversion to integer lookup method.

It has been shown than a series of tables paralleling the IEEE 754 standard supporting variable precision coarse grain space using 3 significant digits precision can be generated to support atomic level object interactions in molecular dynamics software such as GROMACS. These assembly routines perform 15 times faster on a 2.0 GHZ Intel Core i7 'Sandy Bridge' and 2.6 times faster on an Intel Core2 Quad Core 2GHZ and 3.2 times faster on a first generation Intel Xeon Core i7 12 core server. The algorithm was also tested against an integer based lookup table method and was found to be 7 times faster on the Intel 'Sandy Bridge' Core i7 for the force only component of a molecular dynamics simulation. Improvements on various AMD CPUs showed an improvement of 1.6 times faster reflecting substantial differences in the CPU cache architectures of Intel and AMD.

Testing was also performed on a number of C programming library functions such as sqrt, log, tan, cos, etc. over a limited range of values with reduced precision with results in performance improvements that were in the range of 11 to 125 times faster over a specified range of values at a reduced precision.

Simulation specific performance improvements are dependent on the percent of CPU and clock time used for performing nonbonded calculations, the amount of cache memory used for the incremental result caches, the overall memory requirements of the simulation and CPU and system architecture especially CPU cache size and design, chip technology and HPC system architecture.

This study also explored modifications to processor architectures and the addition of support for application level caches using an allocated portion of existing L2, L3 or L4 caches or a proposed LxA application cache paralleling existing processor caches as a means to avoid cache misses. These enhancements in hardware would give developers powerful hardware to support the developed methodology through the proposed general purpose instructions for managing these caches and making optimal use of them. Additional instructions for retrieving and processing data using the developed algorithm could reduce the total number of instructions used by the algorithm and cycle times by 50%.

1.9 Acknowledgments

The author would like to express his thanks to Dr. Paulo Martel, Centre for Biomedical Research, the University of the Algarve, Faro, Portugal for his confidence that this project was feasible and for his support during the project.

The author is also grateful to members of the Computational Systems Biology Group, Center for Neuroscience and Cell Biology, UC-Biotech, University of Coimbra led by Dr. Armindo Salvador; the Biological Chemistry Group, Department of Chemistry, University of Coimbra led by Dr. Maria Joao Moreno; and to Dr. Claudio Soares, New University of Lisbon, Lisbon, Portugal for stimulating discussions about the project and suggestions on approaches to validation of the methodology.

The Biological Chemistry Group, Department of Chemistry, University of Coimbra also provided substantial computational resources that made the free energy studies possible. The author would like to thank Virtual Strategy, Inc., Boston, MA for funding this research and supporting the patent application.

Keywords: performance, incremental computation, non-computation, optimization, HPC, AVX2, variable precision

1.10 References

[14] Bowman, D., Provisional U.S. Patent Application # 62213053, 2015, “Service to Improve the Performance of Applications and Systems Using Noncomputational and Incremental Techniques Implemented in Hardware and Software”

[15] Verlet, L. Phys Rev 1969, 159, 98.

[16] Darden, T.; York, D.; Pedersen, L. J Chem Phys 1993, 98, 10089.

[17] NVIDIA Corporation, 2015, CUDA Runtime API, http://docs.nvidia.com/cuda/pdf/CUDA_Runtime_API.pdf

[18] Abraham, MJ ; van der Spoel, D ; Lindahl, E, Hess, B 2014, & the GROMACS development team, *GROMACS User Manual version 5.0.2*, www.gromacs.org

[19] Stone JE, Hardy DJ, Ufimtsev IS, Schulten K. 2010 “GPU-Accelerated Molecular Modeling Coming Of Age.” *Journal of molecular graphics & modelling*. 2010;29(2):116-125. doi:10.1016/j.jmgm.2010.06.010.

[20] Howes, L; Munshi, A, 2015. *"The OpenCL Specification Version: 2.1 Document Revision: 8"*, Khronos Group. Retrieved 16 April 2015.

[21] Intel Advanced Vector Extensions Programming Reference, website, Intel 2000-04-05.

[22] AMD64 Technology, AMD64 Architecture Programmer’s Manual Volume 6: 128-Bit and 256-bit XOP and FMA4 Instructions, Publication no. 43479 revision 3.01, 2009, Advanced Micro Devices

[23] Intrinsics for Intel® Advanced Vector Extensions 2, https://software.intel.com/en-us/node/513925, Intel Retrieved 06-09-2015

[24] Moore, Gordon E. 1965. "Cramming more components onto integrated circuits", *Electronics Magazine. p. 4. Retrieved 2006-11-11*

[25] ‘Moore’s Law is Dead says Gordon Moore’, http://www.techworld.com/news/operating-systems/moores-law-is-dead-says-gordon-moore-3576581, Apr 13, 2010

[26] 2011. "Cycles, cells and platters: an empirical analysis of hardware failures on a million consumer PCs. Proceedings of the sixth conference on Computer systems (EuroSys '11). pp 343-356.

[27] Páll, Szilárd, Abraham, Mark, James Kutzner, Carsten Hess, Berk Lindahl, Erik. 2015. “Tackling Exascale Software Challenges in Molecular Dynamics Simulations with GROMACS”, Editors, Markidis, Stefano Laure, Erwin. Book: Solving Software Challenges for Exascale, Publisher: Springer International Publishing, volume: 8759, Series Title: Lecture Notes in Computer Science, ISBN 978-3-319-15975-1, DOI 10.1007/978-3-319-15976-8_1

[28] Gruber CC, Pleiss J 2010 Sep 1, ‘Systematic benchmarking of large molecular dynamics simulations employing GROMACS on massive multiprocessing facilities’, *Journal of Computational Chemistry* 2011 Mar; 32(4):600-6. doi: 10.1002/jcc.21645. Epub.

[29] Amdahl, Gene M. (1967). "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities". AFIPS Conference Proceedings (30): 483–485. doi:10.1145/1465482.1465560.

[30] Rodgers, David P. (June 1985). "Improvements in multiprocessor system design". ACM SIGARCH Computer Architecture News archive (New York, NY, USA: ACM) 13 (3): 225–231. doi:10.1145/327070.327215. ISBN 0-8186-0634-7. ISSN 0163-5964.

[31] Daniels220, 13 April 2008, SVG Graph Illustrating Amdahl's Law, https://commons.wikimedia.org/wiki/File:AmdahlsLaw.svg

[32] Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, Compilers: Principles, Techniques, and Tools, Addison-Wesley, 1986. ISBN 0-201-10088-6

[33] Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman, Compilers: Principles, Techniques, and Tools (Second Edition), Addison-Wesley, 2006. ISBN 0-321-48681-1

[34] Flemming Nielson, Hanne Riis Nielson & Chris Hankin (1999). Principles of Program Analysis. Springer.

[35] https://msdn.microsoft.com/en-us/library/e7k32f4k.aspx (retrieved July, 23, 2015)

[36] https://wiki.scinet.utoronto.ca/wiki/images/2/2d/Compiler_qrg12.pdf (retrieved July 23, 2015)

[37] https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html (Retrieved July 23, 2015)

[38] IBM VisualAge C++ Professional for AIX Version 4.0, http://ps-2.kev009.com/basil.holloway/ALL%20PDF/10070600.pdf

[39] Visual Studio 2015 RTM, July 20, 2015, https://www.visualstudio.com/en-us/news/vs2015-vs.aspx#ManLang

[40] Neil McAllister, Microsoft's Roslyn: *Reinventing the compiler as we know it*, InfoWorld, 2011-10-20

[41]Higham, Nicholas J. Accuracy and Stability of Numerical Algorithms, Washington D.C.: Society for Industrial & Applied Mathematics, 2002.

[42]Forman S. Acton. Numerical Methods that Work,The Mathematical Association of America (August 1997).

[43] D. van der Spoel, E. Lindahl, B. Hess, A. R. van Buuren, E. Apol, P. J. Meulenhoff, D. P. Tieleman, A. L. T. M. Sijbers, K. A. Feenstra, R. van Drunen and H. J. C. Berendsen, Gromacs User Manual version 4.5.6, www.gromacs.org (2010)

[44] Intel® Xeon Phi™ Coprocessor System Software Developers Guide SKU# 328207-003EN March, 2014, Intel, xeon-phi-coprocessor-system-software-developers-guide.pdf

[45] Nilsson, L, 2009, *"Efficient table lookup without inverse square roots for calculation of pair wise atomic interactions in classical simulations"*, Journal of Computational Chemistry, Volume 30, Issue 9, pages 1490–1498, 15 July 2009, DOI: 10.1002/jcc.21169

[46] Fresheneesz, 14 April, 2007, Example of a floating point number, http://en.wikipedia.org/wiki/Single-precision_floating-point_format#/media/File:Float_example.svg

[47] Codekaizen, 21 February 2008, The memory format of an IEEE 754 double floating point value., https://commons.wikimedia.org/wiki/File:IEEE_754_Double_Floating_Point_Format.svg

[48] GROMACS 4.5.3, source code nb_kernel212_x86_64_sse.s, nb_kernel21.c

[49] Fog A, 2014, *Instruction Tables*, Technical University of Denmark. Copyright © 1996 - 2014

[50] Fog, A, 2014, *Optimizing software in C++ An optimization guide for Windows, Linux and Mac platforms*, University of Denmark. ©1996 – 2014.

[51] Fog, A. 2014. *The microarchitecture of Intel, AMD and VIA CPUs an optimization guide for assembly programmers and compiler makers*, University of Denmark. ©1996 - 2014

[52] http://www.fftw.org/fftw-wisdom.1.html, Written by Steven G. Johnson and Matteo Frigo. Copyright ©2003 Matteo Frigo. Copyright ©2003 Massachusetts Institute of Technology.

[53] Conference Paper, 275492, Surajit Chaudhuri, An overview of query optimization in relational systems, Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, 0-89791-996-3, Seattle, Washington, USA, 34-43, 1998, 10.1145/275487.275492, ACM

[54] *AMD Bulldozer block diagram (8 core CPU),* uploader, 26 October 2011, https://commons.wikimedia.org/wiki/File:AMD_Bulldozer_block_diagram_(8_core_CPU).PNG (retrieved July, 23, 2015)

[55]http://images.ht4u.net/reviews/2011/amd_bulldozer_fx_prozessoren//amd_bulldozer_die_8_core.png (retrieved July, 23, 2015)

[56] *Intel Sandy Bridge Xeon Block Diagram*, http://regmedia.co.uk/2011/02/24/intel_sandy_bridge_xeon_block.jpg (retrieved July, 23, 2015)

[57] *Intel Core i7 3960x Block Diagram*, http://techreport.com/r.x/core-i7-3960x/dieshot.jpg (retrieved July, 23, 2015)

[58] *Multicore SoC Architecture for Communications Infrastructure*, http://www.ti.com/ww/en/dsp/ci_platform/images/TIs-Multicore-SoC-Architecture_200.jpg (retrieved July, 23, 2015)

[59] *NVIDIA Kepler*, http://wccftech.com/review/asus-geforce-gtx-660-directcu-ii/ (retrieved July, 23, 2015)

Chapter 2 - Accelerating Molecular Dynamics Simulations Using Incremental and Noncomputational Algorithms

Accelerating Molecular Dynamics Simulations Using Incremental and Noncomputational Algorithms

2.1 Abstract

Molecular dynamics software maximizes performance by using methods to reduce the number of interactions, the latest computer instructions, multi-core and distributed computing architectures. Calculation of nonbonded interactions is the major performance problem. The speed of a simulation is limited by the processor architecture and how finely the simulation can be distributed across multiple processors/cores. Simulations in the range of 10s of thousands of atoms will not run significantly faster on a supercomputer than on a 64 core server. The developed algorithm provides a 'noncomputational' approach based on the definition of the simulation followed by incremental computation. It exploits a developed 'variable precision' numeric format. Improvements in the 'calculation' of nonbonded forces for water interactions for Lennard-Jones with Reaction Field on the Intel Core i7 'Sandy Bridge' of 14-15 times that of the GROMACS assembly language versions were achieved on a single thread/single physical core. Performance on a single thread/physical core is the fundamental unit of performance in single processor, GPU and distributed computer systems. The method can also be applied to core PME and other computationally intensive functions with MD software.

2.2 Introduction

Molecular dynamics simulations are important to the understanding of bio-molecular systems and are used for research in the areas of: membrane dynamics, protein folding and unfolding, protein binding, conformational transitions, protein dynamics, transport and macromolecular assembly, and small molecule behavior. These simulations can run for days or even weeks. Typically simulations are in the range of tens to hundreds of thousands of atoms. The primary performance problem is the calculation of the nonbonded force interactions (Coulomb[1] and van der Waals using the Lennard-Jones potential[2]) between the solvent molecules, typically water. Interactions between water and solute and solute to solute represent a much smaller part of the computational cost of simulations. Thus if solvent to solvent interactions cannot be made substantially faster, the simulation itself cannot be made significantly faster. The fundamental unit of performance in molecular dynamics software is the speed with which functions perform on one thread/core in a single physical processor, GPU or distributed environment. If this unit can be made substantially faster, so can program execution.

Since the late 1970s the ability to perform MD simulations has increased greatly from a 10 ps simulation of a 450 atom protein in a vacuum[3] to simulations in multiprocessor environments containing over one million atoms that run for hundreds of nanoseconds.[4,5] Using special hardware David Shaw has run simulations in the millisecond range on the Anton supercomputer.[6] There is also the need to perform free energy[7] simulation studies that involve a series of simulations with changes to simulation parameters to support multiple lambda values or replicated simulations to increase sampling[8]. Developers of MD software such as GROMACS[9,10] started in the 1990s by addressing the performance of the inverse square root function[11] as this is the largest computational contributor to the calculation of nonbonded interactions. Over the years as processor architectures, compilers and parallel processing technology developed, new features were added to MD software exploiting processor architectures and new instruction sets as they were developed in the industry. GROMACS currently supports x86, AMD64/x86-64, PPC, ARMx7 and SPARC VIII. It can also be compiled and run on many other operating systems and architectures. GROMACS supports NVIDIA GPUs, MPI, OpenMP for parallel processing for multiple machines across a network and for hybrid parallel environments.[12] For the last 25 years improvements in compiler optimization techniques have also greatly improved performance by avoiding computation within the scope of the compiler. Implicit solvent[13,14] models and coarse grain techniques have been developed to reduce the number of particles interactions,[15] but these methods are not widely used for most simulations due to limitations in the algorithms.

GPU support has been added to GROMACS in the last five years initially with OpenMM and in version 4.6 built-in support to offload computation to GPUs using NVDIA's CUDA[16] to Tesla, Fermi and Kepler cards.[17] There are however underlying limitations for GPU based computing including the limited bandwidth between the CPU and GPU and limitations in GPU hardware as a generalized coprocessor.[18] Recent advances in GPU memory architectures that include a larger number of cores and GBs of memory as well as the addition of GPU cache memory has greatly enhanced the ability to GPUs as general purpose computational engines. NVIDIA CUDA and OpenCL now make it possible to have TFLOPs available on the desktop or in servers. Software developed for GPUs must also be designed in a different way that CPU based software to make transfers between the CPU-GPU efficient and code designed for GPUs must make minimum use of shared memory so as not to block threads running on the GPU. GROMACS supports OpenCL as an interface to NVIDIA cards and it has been reported that there is OpenCL support for GROMACS using ATI graphics cards.[19]

Support for simple integer based lookup tables to eliminate calculations using the distance squared as the base for the lookup of forces was developed for CHARMM[26]. This approach has its own problems due to the conversion of floating point to integer indices to provide a simple index to obtain force results. As the distance becomes smaller the conversion to integer results in an increasingly large difference in the results obtained versus the results of the calculated value using floating point instructions. This results in large offsetting positive and negative error correction for forces as these increase and distance becomes smaller until eventually a cutoff occurs with a maximum force solely based on the conversion of floating point to integer and the size of the table.

The performance of MD simulations has probably benefited the most from specialized SIMD (single instruction multiple data) instructions and fused multiple and add instructions. For many years GROMACS has exploited the single instruction multiple data (SIMD) instruction sets SSE, SSE2 and recently added support for AVX[20] and Intel and AMD's FMA[21] instructions for fused multiply and add. Prior to GROMACS version 4.6 GROMACS achieved very high performance on a single processor/core by using hand coded assembly language to process four single precision values or two double precision values simultaneously. The GROMACS 4.5 assembly code for the nonbonded interactions was closely examined and *no* opportunities for further optimizations in the SSE version and only one instruction could be removed from the SSE2 (double precision) version. In version 4.6 the assembly code was replaced with high level language code using intrinsic functions to support newer instructions sets without the need to develop code in assembly language. This also has the benefit of exposing the code to the compiler global optimizer. It is rarely possible to improve upon the code generated by compilers with advanced global optimizers. With the support of the AVX, AVX2[22] and FMA instructions 8 single precision values may be processed simultaneously. The Intel Xeon Phi processor supports the AVX512 instruction set and can process 16 single precision values simultaneously.

2.2.1 Performance Limiting Factors for MD Simulations

Processing speed has been limited primarily by CPU clock speed (cycle/second GHZ) and the amount of data than can be processed in one cycle. In 1965 Gordon Moore, Intel co-founder, predicted that processing power would double approximately every 2 years[23] but in 2005 Moore declared that his law was 'dead'.[24] This was largely due to the limitations on CPU clock speed, heat dissipation on the chip and fabrication costs due to the on chip density. With current technology this limits the performance on a single core/processor primarily based on CPU clock speed (GHZ). There is a direct relationship between clock speed and power consumption and temperature though advances have been made in this area with the low power Intel 'Haswell' architecture.

Figure 28 shows that CPU clock speed has flattened since about 2003. See the blue line in the following figure. There are more cores/chip and more transistors/chip to support

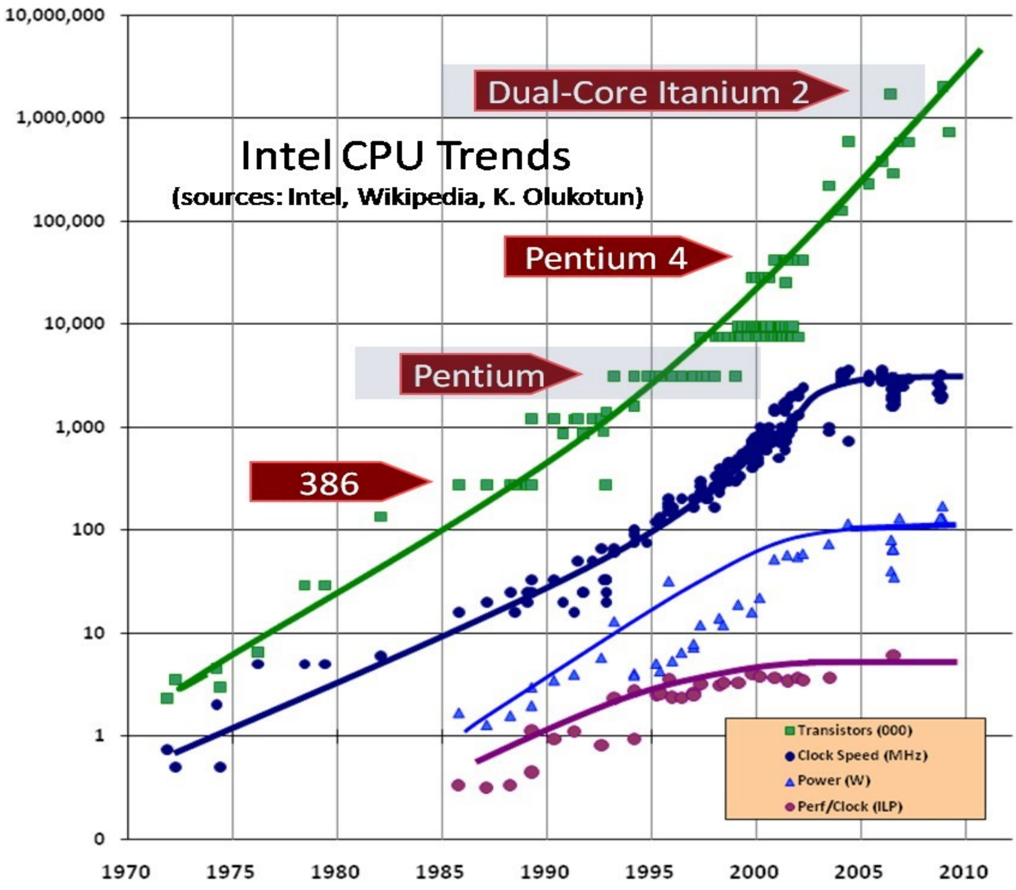


Figure 28 Intel CPU Trends – Limitations on Performance

© Herb Sutter, Used with permission

them. Techniques such as multiple cores on chip can reduce the communications costs between threads but CPU clock speed is not substantially increasing and is a fundamental barrier to the performance of MD simulations. Over-clocking of CPUs can be done but is limited by the amount of power consumed and heat generated. Over-clocking can also result in computational errors or damage to the chips.[25]

Studies have also shown that there are limits to the scalability of a simulation based on the number of atoms in the simulation[26]. There exists a minimum number of atoms that can be processed per processor/core before the communications costs and real time delays are greater than the real time performance gains. For simulations in the range of a few 10s of thousands of atoms this is less than 64 cores. This implies that a simulation in this size range will not run significantly faster on a system with 1000 cores. In fact if a simulation were to be split into too few atoms per core the communication cost could consume most of the real time and slow the real time to process the simulation. Molecular dynamics simulation software uses methods (e.g. lattice summation or spherical cutoffs) to reduce the number of interactions from $O(N^2)$ to $O(N\log N)$ or $O(N)$.[27,28] GROMACS exploits the latest in computer instructions, multi-core, and multiprocessor capabilities and tools such as MPI to decompose and distribute these atoms and interactions across cores and nodes.

Figure 29 shows the results of the study of Gruber and Pleiss[29] in 2010 that demonstrated the peak performance that can be achieved with different MD system sizes.

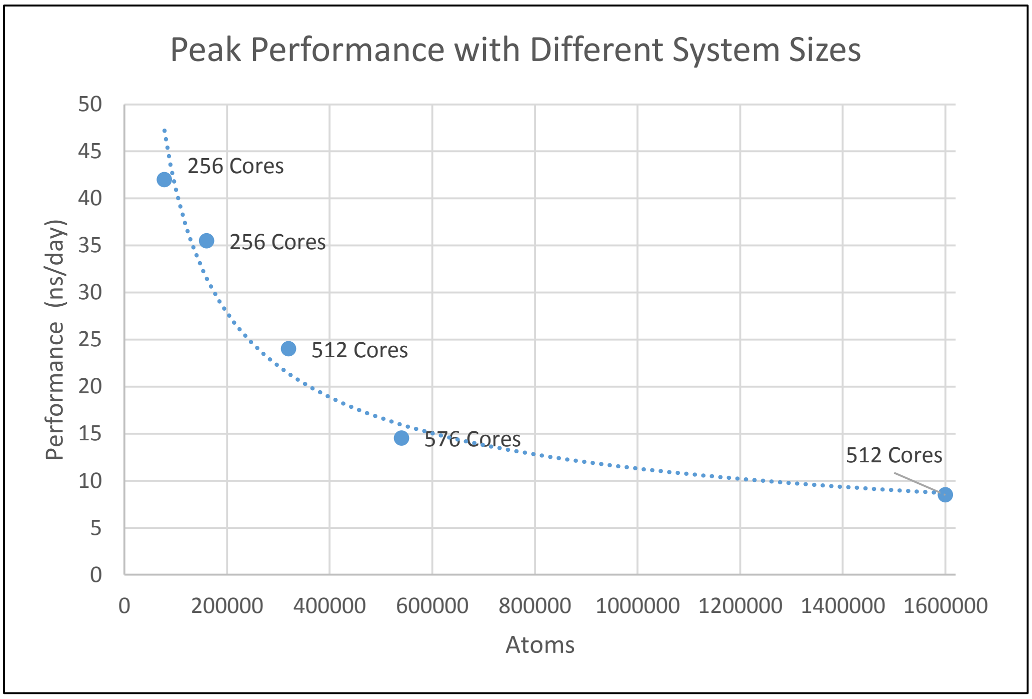


Figure 29 Peak Performance by system size. Adapted from Gruber and Pleiss

There are therefore fundamental limitations in simulation performance that cannot be overcome by improvements to programming: CPU clock speed, size and speed of L1, L2, L3, L4 cache memory and speed of main memory, transistor chip density, heat, and materials properties. Distances off chip to CPU blocks, main memory, blades or server nodes have significant delays in terms of clock cycles. Distance is a fundamental problem because of materials properties and the speed of light. When processing is performed in CPU registers frequently multiple instructions may be done in a single clock cycle. The further the data is from the registers the more cycles are required to obtain the data or a fraction of a cycle. Data from main memory may require over 100 cycles with data from other CPU blocks, blades or nodes taking significantly longer. Substantial improvements have been made in InfiniBand technology that is widely used in supercomputers but as of 2014 the theoretical effective transfer rate is 24 Gbs.[30] Due to advances in the Intel ‘Sandy Bridge’ and “Ivy Bridge’ architectures the minimum number of atoms/core/processor has been reduce from a range of ≈500-1000 to ≈150 allowing simulations to be distributed across more cores and processors. The fundamental performance problem with all of these improvements remains how fast the atoms on a single core/processor can be performed. Regardless of the number of atoms/core/processor, substantial improvements in performance must come from improving single thread/core/processor performance. This is the object of this project.

Other fundamental limitations exist to performance improvements such as those imposed by Amdahl's law[31]. The performance of a program that can be improved is limited by the percentage of the code that can be improved. The speedup of a program using multiple processors in parallel computing or using an improved algorithms is limited by this sequential fraction of the program. For example, if 95% of the performance of program can be parallelized or improved by code changes, the theoretical maximum speed up would be 20 × as shown in Figure 30.[32] It will not matter how many processors or how good the programming changes are assuming that the portions of the program that can be improved may be parallelized completely or that the performance of the code being changed may be reduced to zero percent of the total time for the program.

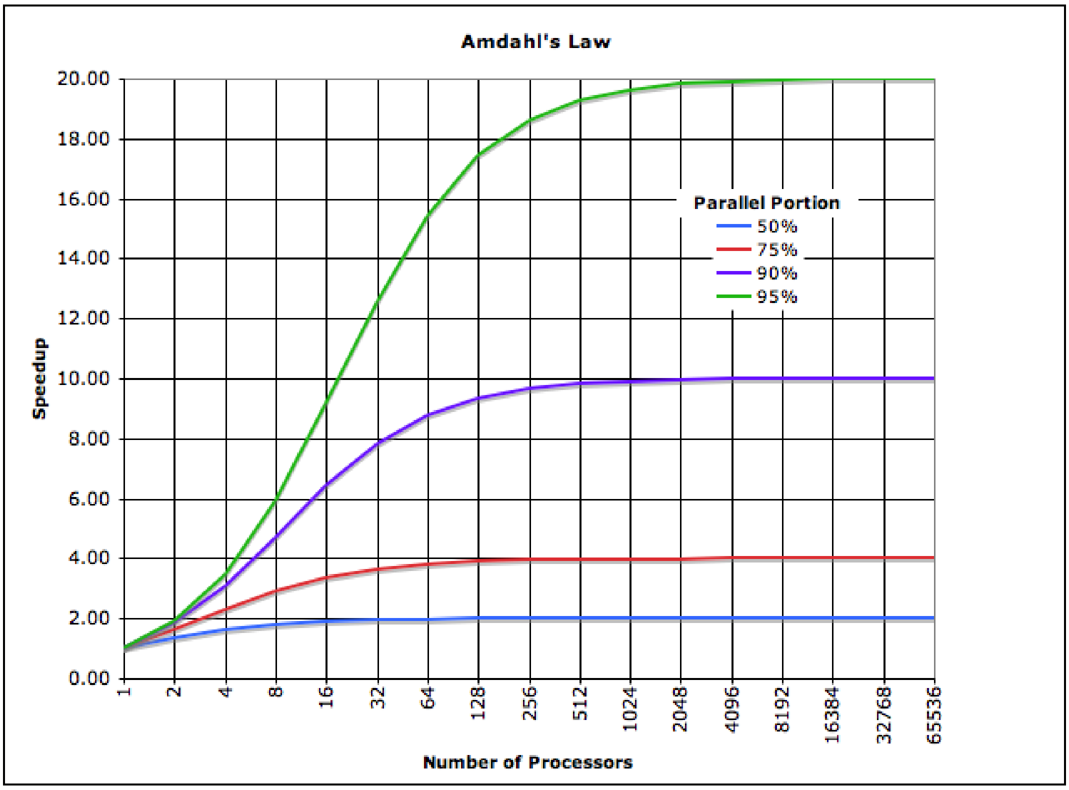


Figure 30 Performance Limitations - Amdahl's Law

In the case of molecular dynamics simulations this is not the case because it is not possible to distribute the numbers of molecules/atoms for typical simulations across large numbers of processors without causing large communication delays. The core processing of nonbonded interactions has a fundamental limit based on the number of atoms that can be processed per core.

Fortunately the percentage the CPU time and clock time for a simulation is concentrated in the nonbonded routines as shown from the summary statistics from a GROMACS simulation of a water box. (See Figure 31.) The performance improvement that may be obtained for processing the nonbonded interaction has some additional limitations based on the percentage of water that is part of the simulation. GROMACS has specialized routines to optimize the performance of water to water interactions. Proteins in water typically benefit more from the developed algorithm more than membrane simulations that may only contain 60-70 percent water.

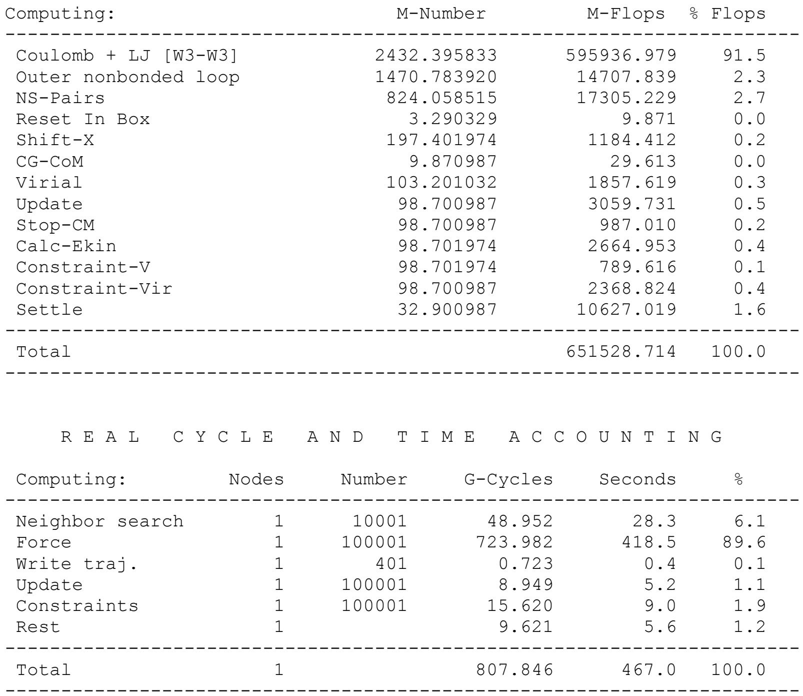
2.3 Methodology

The fundamental question is if single thread/core/processor performance cannot be substantially increased by faster CPU clock times, offloading to GPUs or other processors in a network where will the breakthroughs in performance occur. This study has developed an algorithm that avoids as much of the computation for the nonbonded interactions as possible and the algorithm can be applied to other computationally intensive functions in MD and other applications. It can effectively raise the scalability bar for all single core/processor and multiprocessor environments. This algorithm can increase the number of nanoseconds/day that can be run for a given simulation on a single core/processor.

The developed algorithm optimizes the performance of the nonbonded code by using runtime knowledge of the simulation being performed just as computer language compiler uses global optimizers to determine what part of a software program does not change at compilation time. In the case of the nonbonded routines for a given simulation typically values for charge constants, Lennard-Jones parameters, reaction field parameter and even box type are used as constants at *runtime* and are not known to be constants when the GROMACS software is built and thus cannot be optimized by computer language optimizers.

The developed methodology and algorithms in both the C language and hand coded assembler routines provide a ‘noncomputational’ approach based on the definition of the simulation and the incremental ‘assembly’ of pre-calculated results to obtain the final result. This approach may be applied to any GROMACS version but the test environment was based on version 4.5.3. The developed method uses Lennard-Jones and reaction field as the test model. With the advent of Verlet cutoff schemes reaction field has once again become a viable high performance alternative to PME in GROMACS version 5.

These incremental results are each stored in a table and accessed with memory, integer and bit manipulation instructions only. These instructions can be efficiently executed in the CPU pipeline and many of these instructions execute in less than one clock cycle and operate only with CPU registers. No floating point instructions are required except where pre-calculated results cannot be used for intermediate or final results that are used external to the function being optimized. For MD simulations processed using GROMACS this approach effectively produces a ‘coarse grain’ *space* for the simulation to run in using the distance squared (r^2). The algorithm does not however reduce the cost of processing the neighborhood list or communication overhead.



```
Computing:                         M-Number         M-Flops  % Flops
-----------------------------------------------------------------------
 Coulomb + LJ [W3-W3]            2432.395833      595936.979    91.5
 Outer nonbonded loop            1470.783920       14707.839     2.3
 NS-Pairs                         824.058515       17305.229     2.7
 Reset In Box                       3.290329           9.871     0.0
 Shift-X                          197.401974        1184.412     0.2
 CG-CoM                             9.870987          29.613     0.0
 Virial                           103.201032        1857.619     0.3
 Update                            98.700987        3059.731     0.5
 Stop-CM                           98.700987         987.010     0.2
 Calc-Ekin                         98.701974        2664.953     0.4
 Constraint-V                      98.701974         789.616     0.1
 Constraint-Vir                    98.700987        2368.824     0.4
 Settle                            32.900987       10627.019     1.6
-----------------------------------------------------------------------
 Total                                            651528.714   100.0
-----------------------------------------------------------------------


    R E A L   C Y C L E   A N D   T I M E   A C C O U N T I N G

 Computing:         Nodes     Number     G-Cycles    Seconds     %
-----------------------------------------------------------------------
 Neighbor search        1      10001       48.952       28.3     6.1
 Force                  1     100001      723.982      418.5    89.6
 Write traj.            1        401        0.723        0.4     0.1
 Update                 1     100001        8.949        5.2     1.1
 Constraints            1     100001       15.620        9.0     1.9
 Rest                   1                   9.621        5.6     1.2
-----------------------------------------------------------------------
 Total                  1                 807.846      467.0   100.0
-----------------------------------------------------------------------
```

Figure 31 Performance Summary - GROMACS Water Box Simulation

The objective of this research is to address the approximately 70-90% of the CPU cost of most molecular dynamics simulations.

The Coulomb and Lennard-Jones equations required to determine the results of the nonbonded interactions are easily solved but usually represent 70-90%+ of the computation time for a simulation. These simple equations require few parameters and at runtime are dependent on atomic level charges, C6 and C12 Lennard-Jones parameters as can be seen from the following equation.

$$V(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right],$$

Equation 8 Lennard-Jones Potential

The inner loops of the GROMACS nonbonded routines use the Lennard-Jones potential to calculate the Pauli repulsion and attractive dispersive forces for atoms or molecules. In Eq. 8 ε is the depth of the potential well, σ is the distance where the inter-particle potential is zero, and r is inter-particular distance. [33,34]

Electrostatic interactions may be calculated as follows where the value of the electrostatic force $\boldsymbol{F}$ acting on two point charges q_1 and q_2 is defined as follows. k_e is Coulomb's constant and *r* is the distance.

$$|\boldsymbol{F}| = k_e \frac{|q_1 q_2|}{r^2}$$

Equation 9 Coulomb's Equation

In order to improve the performance of these equation solutions it is necessary to eliminate or reduce substantially the computer instructions and CPU cycles required to solve them. Fortunately at runtime many of the values required to do this are loaded into the program based on force field and water model definitions and used as constants. This allows the exchange of computational code for data assembled with a few non-floating point instructions at runtime that may be easily pipelined by the CPU and most of the instructions execute in less than one cycle.

In order to achieve the objective of this project a new mathematical and floating point format model has also been developed providing a variable precision floating point calculation model based on the IEEE 754 standard to reduce the size of the pre-calculated tables. This variable precision format parallels the IEEE standard without the need for additional tables or runtime operations. This algorithm may be used in any application that is computationally intensive. There are no approximations other than those reflected in the reduced precision using a subset of the IEEE floating point definitions. All calculations that were used in the evaluation with GROMACS for the incremental calculations are performed in double precision and stored as single precision when the tables are initialized. The tables representing computational increments could also be stored as double precision if needed. GROMACS builds are by default single precision so the pre-calculated results were stored as single precision. When the tables containing the incremental pre-calculated results are generated the variable precision algorithm adjusts the floating point representation to parallel the IEEE 754 standard but at reduced precision. It uses a simulated 'guard' bit to avoid creating a very small cumulative divergent error. When the tables are created the exponent portion of the floating point representation of the value of interest (e.g. distance squared in the case of the nonbonded GROMACS routines) *may* be masked with a binary value to alter or create one or more ranges of exponents and/or eliminate the sign bit of the floating point representation. The precision of the resulting value is reduced shifting the floating point value in binary form to the right reducing the mantissa. The right shift effectively divides the representation by a power of two and reduces the size of the tables required. After this is performed, and a simulated 'guard' bit is added enabling the calculation of the values stored in the table to be treated as a scaled base 2 value. This provides a mixed stability model with a very small offsetting forward and backward error correction paralleling the IEEE standard. This is necessary because dividing an integer by 2 causes a truncation on odd values creating a cumulative error. A floating point to integer conversion is not used because this creates a divergent error relative to the IEEE standard by eliminating the 'floating' property of IEEE 754 compliant CPUs and data formats. In the case of the nonbonded GROMACS routines[35] the floating point to integer conversion has an increasingly large offsetting corrective error as the distance squared decreases. This has an undesirable effect on the results for the force values using such an approach as the distance squared becomes smaller.

The algorithm is designed to scale using single instruction multiple data (SIMD) extensions to instruction sets such as SSE, SSE2, SSE4.1, AVX, AVX2 and AVX512 Intel Xeon Phi)[36]. In these instructions sets it is possible to process 128, 256 or 512 bits of data in parallel. This may be in the form of 4, 8, or 16 single precision values or 2, 4, or 8 double precision values. The Intel Xeon Phi coprocessor processes 512 bits but requires special programming when used as a coprocessor. It has been reported that a native version of GROMACS has been compiled for Intel Xeon Phi. The default GROMACS builds are for single precision (SSE). Single and double precision C language code, and double precision (SSE2) are also supported. With the release of version 4.6 additional support for FMA and AVX instructions has been added as configuration options at build time. The variations of the developed algorithm were evaluated against the standard GROMACS 4.5.3 builds for single precision SSE, SSE2 double precision and single precision C. Additionally an AVX2 version was developed for performance comparison on the Intel 'Haswell' architecture. AVX support was not provided because like SSE it lacks support for certain integer and binary instructions that make the developed algorithm efficient. At the time the algorithm was developed an AVX enabled version of GROMACS was not available.

The methodology also could benefit from new computer instructions that s could reduce the number of instructions from three on AVX2 or Intel Xeon Phi to two and support the allocation and management of a portion of L2, L3 and L4 caches for use by applications. A test program also evaluated performance of the algorithms versus the performance of gcc 4.7 C library functions such as sin, cos, tan, sqrt, log, pow and other functions over a limited range of values. This suggests that PME and other computationally intensive portions of GROMACS may substantially benefit from replacing these functions with the developed method. Included in the test was the C version of the GROMACS Newton-Raphson 1/sqrt and Lennart Nilsson's floating point integer conversion based table lookup method are indicated in the chart below with labels 'NR 1/sqrt' and 'LN JCC' respectively. Figure 32 also shows some of the limits of the developed method. The calculation of the 4/3 π r^3 sphere volume formula is substantially slower when compared with the other cases because there are so few instructions involved in the calculation and the access to memory is slower than the computation. The developed method should not be used for functions that are implemented in only a few instructions.

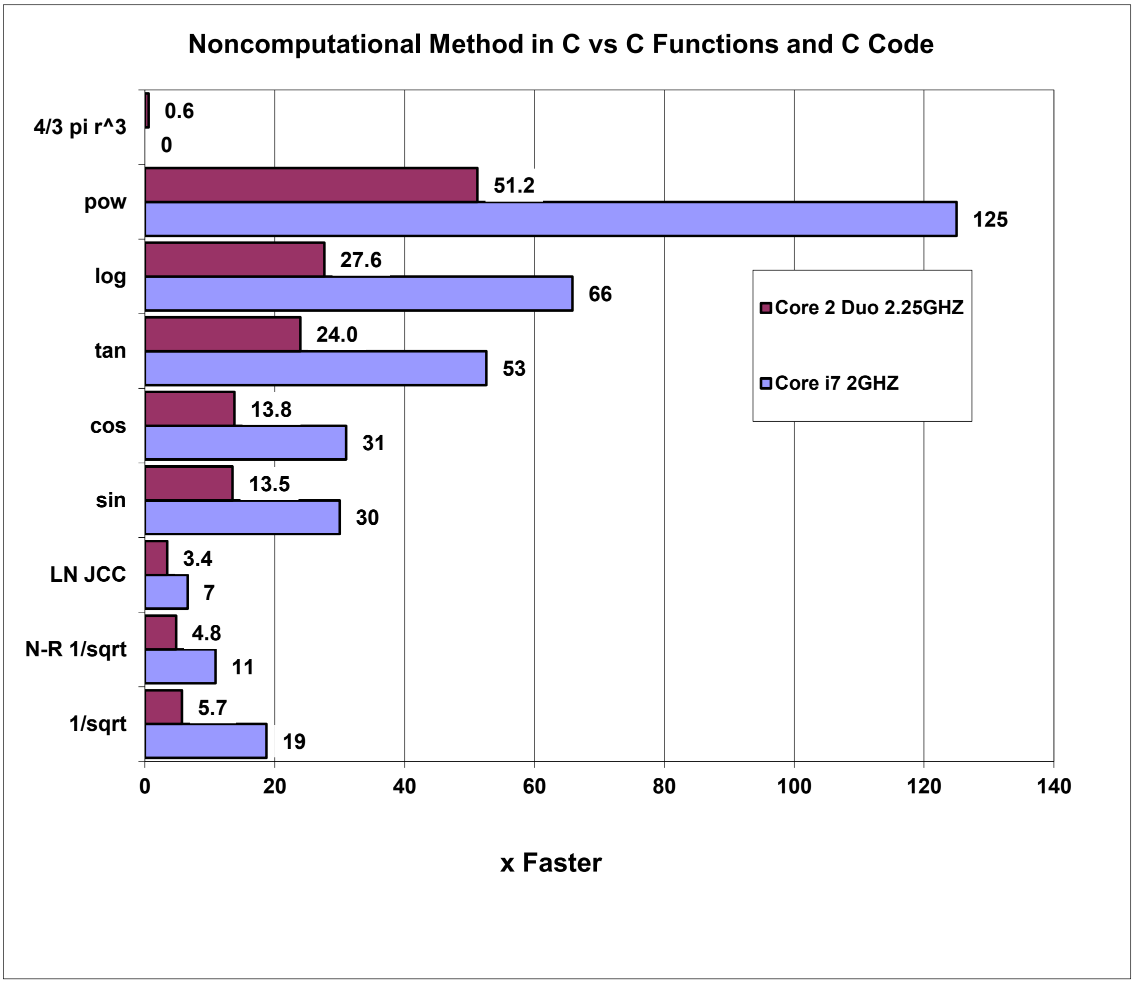


Figure 32 Comparison of C Library Functions vs Developed Algorithm Core 2 Duo 2.25 GHZ vs Core i7 2 GHZ 'Sandy Bridge'

The IEEE 754 representation of a single precision floating point value has a 23 bit explicit mantissa that defines the precision of the single precision format. GROMACS builds representing the 23 variations in precision possible using the developed algorithm for single precision were developed for use in the comparison with the standard build versions for GROMACS single and double precision.

Each variation of the algorithm was produced by reducing the precision of the distance squared value used in calculating the nonbonded interactions. These versions represent the spatial and computation granularities possible with the algorithm using a variable precision floating point representation.[37] Each variation is created by reducing the single precision mantissa by one bit.

It was observed that the standard GROMACS build versions (C, SSE and SSE2) have a natural deviation between each other and this deviation was used to validate the algorithms to determine if they were ‘good enough’. If the results of the new methodology and algorithms were within the deviation between the GROMACS algorithms this study considers the results ‘good enough’ and ‘equivalent’ for use with molecular dynamics simulations.

Validation was performed at two levels: 1) computational - examining forward/backward error, stability and possible divergence from the IEEE 754 standard [38] 2) the developed algorithm as compared with the GROMACS 4.5.3 build alternatives. This study presents computational and performance results based on water boxes, amino acid and protein simulations.

Water Box Validation

The following analysis represents the results of a 1ns SPC water box simulation of 329 water molecules using the GROMACS 4.5.3 Lennard-Jones reaction field nonbonded routines with the Berendsen temperature coupling. The results were generated from a series of simulations using the GROMACS SSE build, C build and variations of the developed algorithm from ≈7.22 significant digits to ≈2.11 significant digits. The

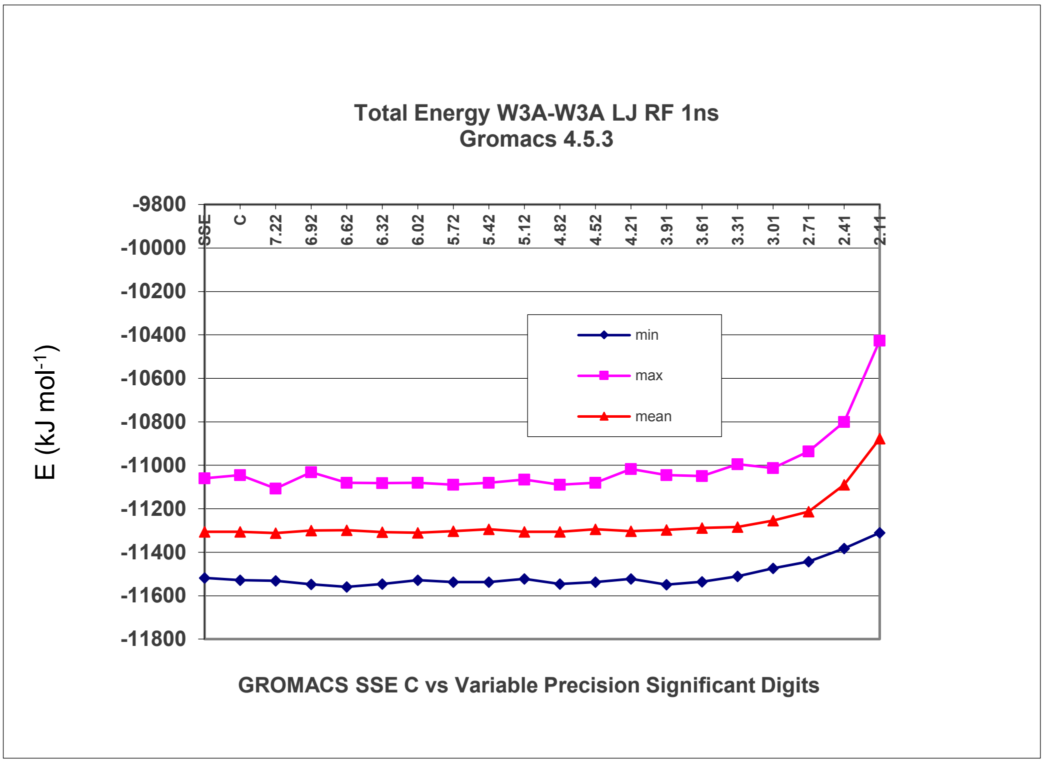
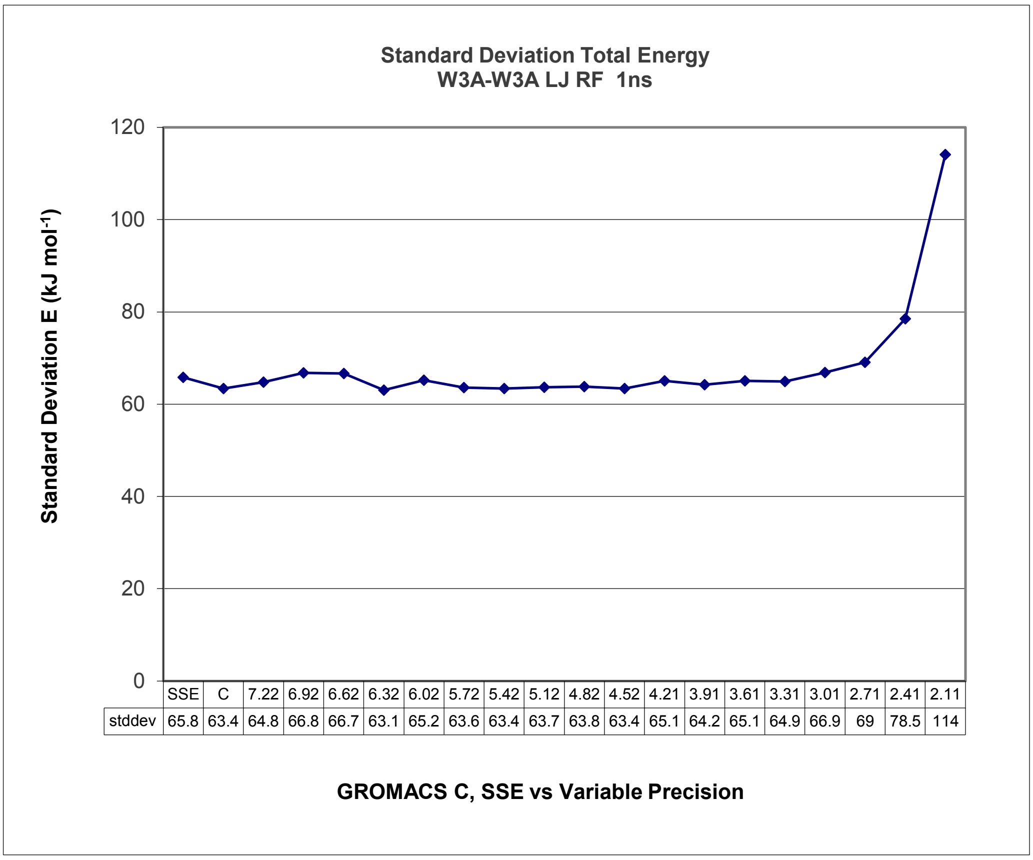


Figure 33 Total Energy – Water Box Simulation

GROMACS g_energy utility was used for analysis. Tests were attempted using variable precision of less than ≈2.11 significant digits but the system blew up. Figure 34 shows the variation in mean, max and minimum values for the total energy.

Figure 36 show the standard deviation increasing after ≈2.71 significant digits to 114 kJ mol^{-1}.



	SSE	C	7.22	6.92	6.62	6.32	6.02	5.72	5.42	5.12	4.82	4.52	4.21	3.91	3.61	3.31	3.01	2.71	2.41	2.11
stddev	65.8	63.4	64.8	66.8	66.7	63.1	65.2	63.6	63.4	63.7	63.8	63.4	65.1	64.2	65.1	64.9	66.9	69	78.5	114

Figure 34 Increase in Standard Deviation with Reduced Precision

Figure 35 shows the degree to which the standard deviation of the variable precision algorithm lies between the standard deviations of the C and SSE GROMACS builds. The blue line shows the difference between the standard deviation of variable precision and the standard deviation of the C language version and the purple line shows the difference between the standard deviation of the variable precision and the standard deviation of the GROMACS SSE version. Note the difference between the standard deviations of the GROMACS C build and the GROMACS SSE build. Once again after ≈3.31 or ≈3.01 significant digits there are increases in the standard deviations as the 'spatial' granularity is reduced. Variable precision formats within or close to the

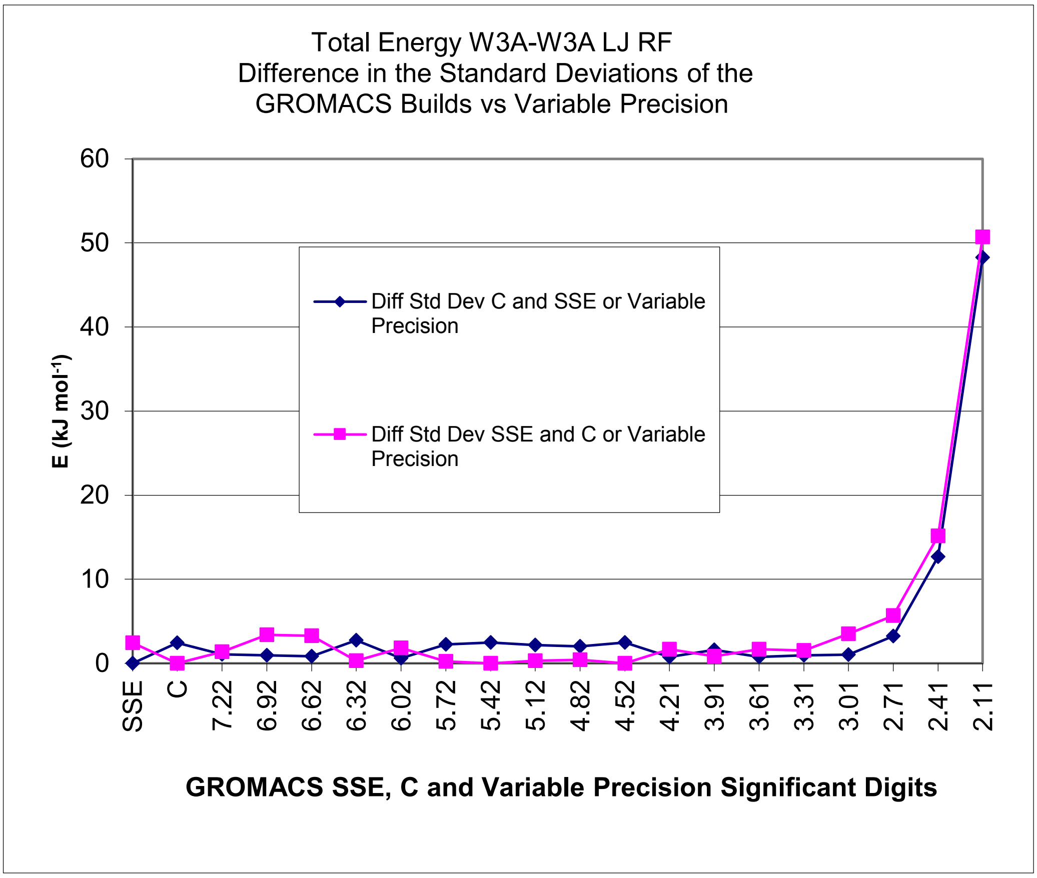


Figure 35 Standard Deviation in Total Energy

deviation between the GROMACS builds are considered to likely be 'good enough' for MD simulations.

The system temperature is well regulated. The Berendsen temperature coupling was used.

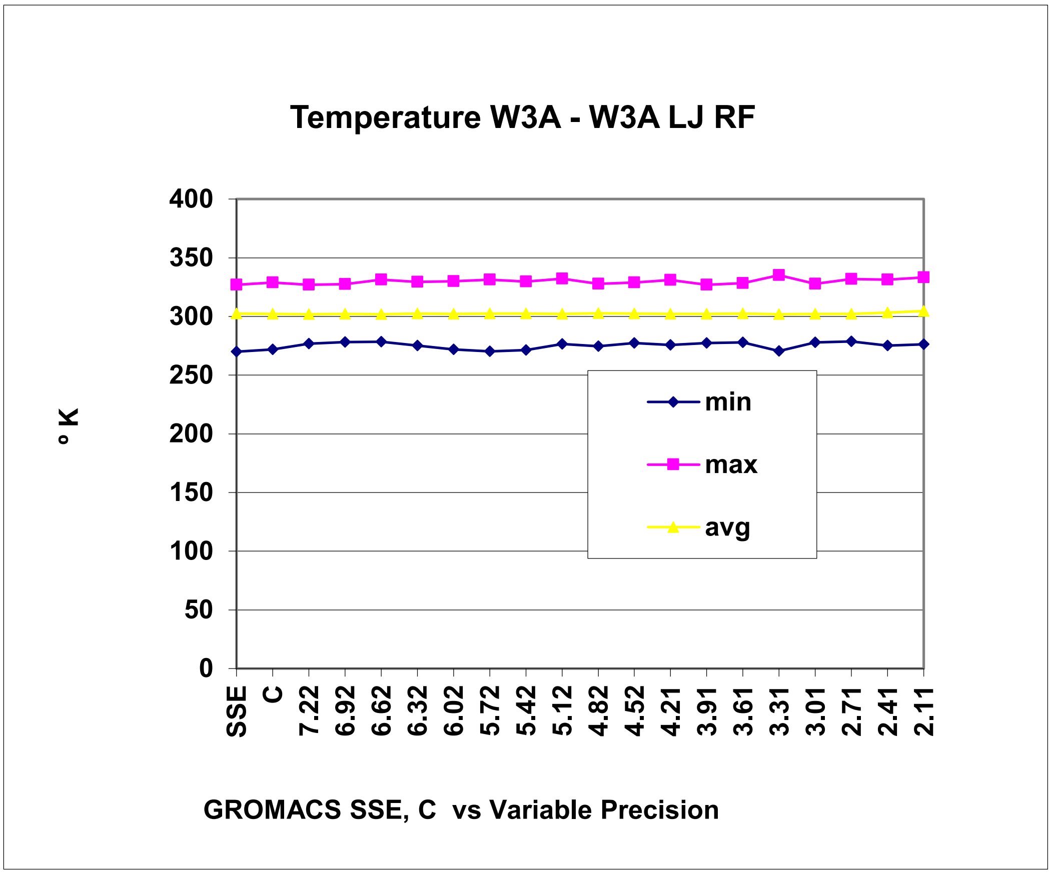


Figure 36 Temperature Stability with Reduced Precision

2.3.1 Validation Using Protein Simulations

Protein simulations were run using the standard GROMACS builds and the developed algorithm. The following protein studies were conducted: 1VII (Chicken Villin Headpiece), 1LYD (T4-Lysozyme), 2INT (Human Interleukin0-4), and BPTI (Proteinase Inhibitor (Tyrpsin)). The studies were analyzed using examining root mean square deviation (RMSD), energy and hydrogen bonds using the GROMACS utilities g_rms, g_energy and g_hbond.

The variations between multiple replicate runs of the same simulation and using different standard build options for GROMACS (single precision SSE, double precision SSE2, and single precision C language versions) show results that vary based on how GROMACS is built and is not repeatable from one run of the same simulation to another.

The results show a similar variation between the standard GROMACS builds and a corresponding similar variation between the developed algorithms. It is however impossible to compare these algorithms to demonstrate that the developed algorithm is 'good enough' because of the complexity and size of these simulations. A number of experienced GROMACS users including one of the GROMACS developers were of the opinion that the results 'looked ok'. The options of users or even GROMACS developers do not constitute evidence that the methodology is 'good enough'. Figure 37 illustrates the natural variation between the GROMACS builds as compared with the developed algorithms supporting the approximately 3.01 significant digits.

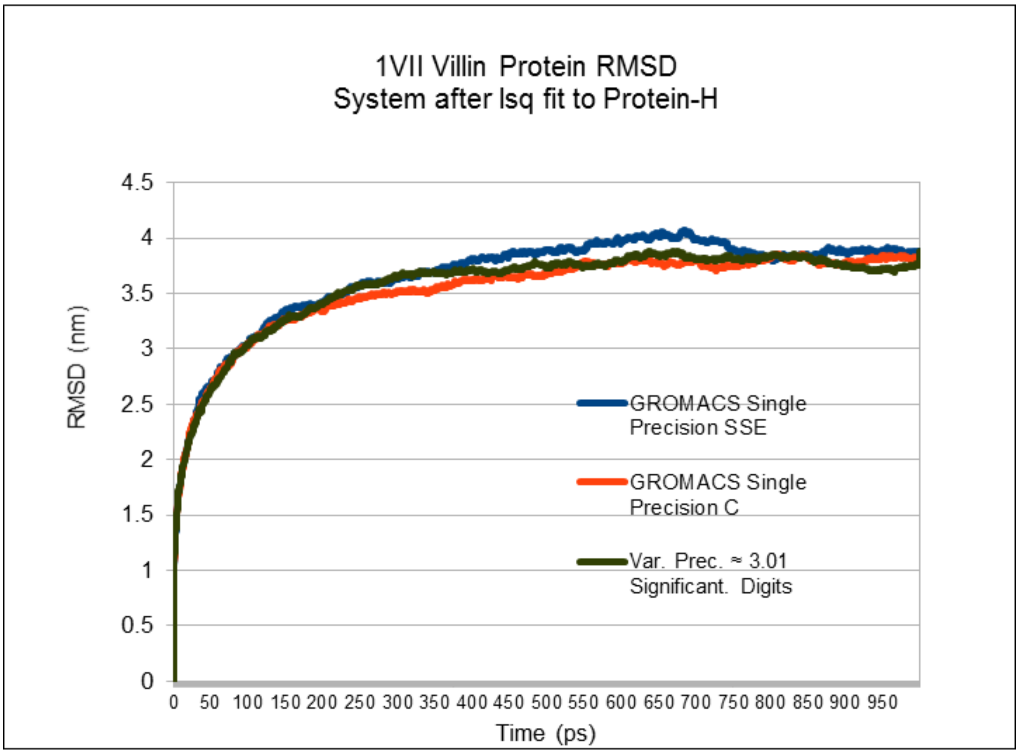


Figure 37 RMSD 1VII Chicken Villin Headpiece

Multiple runs of protein simulations always show a level of variation and therefore these complex systems cannot be used to 'prove' equivalence.

In the study of Nilsson using a less computationally robust integer based conversion approach tested on very short simulations, it was reported that the RMS relative force error for a DHFR system "was sufficient for simulation of biomolecules." Further the study analyzed the drift in total energy using CHARMM showing that a direct lookup table approach had a 35 K/ns drift as compared with the 0.008 K/ns drift for the standard version of CHARMM. Using a linear interpolation method this could be reduced to 0.16- 0.19 K/ns depending on points/table. The drift in total energy was expressed as $\Delta E_{tot} = \Delta E_{pot} + \Delta E_{kin}$ assuming that the entire drift in energy was converted to heat.[39]

2.3.2 Amino Acid Studies (1µs)

The algorithm was also evaluated using long 1µs simulations of amino acids in water to look for computational artifacts that might not be visible in shorter simulations. Arginine and cysteine were used for these studies and variable precision ‘spatial’ granularities of the developed algorithm were tested to ≈2.11 significant digits. Implementations of the developed algorithm with precisions less than ≈2.11 crashed. These mean total energy of the reduced precision variations behaved similar to the variable precision water box simulations.

The results of the developed algorithm were evaluated against the standard single precision SSE and single precision C versions of GROMACS. No computational divergence was encountered other than an increase in the standard deviation in total energy similar to the increase in the standard deviation for the water box simulations.

All tests were performed using Lennard-Jones with reaction field with the SPC water model and the GROMOS 43a1 force field using GROMACS 4.5.3 using a single core.

2.3.3 GROMACS Regression Test Suites

The developed algorithms were tested in all versions of GROMACS from 3.3 to 4.5.3 using the GROMACS regression test suites available on the GROMACS site for versions 3.3.3 to 4.5.3. This was useful to validate where the ‘spatial granularity’ thresholds were. The GROMACS test suites regressiontest.git, gmxtest-3.3.2, gmxtesst-3.3.3, and gmxtest-4.0.2 were used. Tests using the test suite for GROMACS 3.3.3 showed that all non-bonded tests passed for the first 17 versions of the developed algorithm with reduce computational and spatial granularity to a distance squared of ≈2.11 significant digits. Some of these should not have passed due to the large spatial ‘jumps’ that this reduction in precision implies.

The regression test suite for GROMACS 4.5.3 showed that all non-bonded tests pass for the first 14 versions with a reduced granularity to ≈3.01 significant digits. This is in line with the expectations based on an assumption that a spatial granularity in the range of 1pm should be viable and is consistent with the water box testing that showed total energy values reported by g_energy started to increase dramatically after spatial granularity fell below ≈3.01 significant digits.

2.3.4 Test Environments

Performance testing was performed on the following systems: AMD Opteron 6272 2.1 GHZ 2MB CPU cache (Bulldozer) 64 core server, Intel Xeon 5650 2.67 GHZ 12MB Cache 6 cores (Westmere-EP 32nm), Intel Core i7 (2630QM) 2.0 GHZ 6 MB Cache (Sandy Bridge), Intel Core i7 (930) 2.8 GHZ, 8 MB cache (Nehalem), Intel Core 2 Duo 2.24 GHZ 3 MB cache, Intel Core 2 Quad Core 2.2 GHZ 3 MB cache, AMD Athlon 3800+ 2.4 GHZ 512KB cache, AMD Athlon X2 4400+ 2.3 GHZ 512KB cache and Intel Core i5 1.7 GHZ 3MB cache (4201U (Haswell) ULT).

All testing was performed on Ubuntu 14.04. The Intel Software Development Emulator (SDE) was used for the initial development and testing of the Intel AVX2 instruction set version (Intel 'Haswell' architecture). The gcc 4.7 was used for all non AVX2 development and the gcc 4.8 C compiler was used for building the AVX2 executables.

2.3.5 Testing with Water Models

Simulations of a water box containing 987 atoms, using the SPC water model, GROMOS 43a1 force field and using the NVT ensemble were run. The water box simulation used the GROMACS 4.5.3 nonbonded kernels routines for Lennard-Jones and Reaction Field for electrostatics. These simulations were executed using 22 different variations of the algorithm and the three standard GROMACS single and double precision builds. The 22 different versions of the algorithm reduce the size of the mantissa for the IEEE 754 representation of the real number value. By reducing the size of the mantissa the approximate number of decimal digits supported can be reduced from ≈7.22 to ≈2.11. This reduced computational and spatial 'granularity' was used to minimize the memory required for pre-computed tables.

Large fluctuations in the simulation results occurred with water box simulations as the 'granularity' approached 1 significant digit. Similar results were seen in small molecule and protein simulations. This is a result of the 'spatial granularity' being reduced too much and the atoms 'jumping' too far.

2.3.6 A 'Noncomputational' Incremental Algorithm

The algorithm exploits the internal structure of the IEEE 754 floating point standard's representation of single and double precision numerical values. Once these values are processed they are used to index tables of pre-calculated results and then to incrementally compute additional intermediate or final results. The objective of the methodology is to avoid computation at the core of the nonbonded routines and to move this computation to the generation of pre-computed tables are generated only on first use. Compiler global optimizers have been using techniques similar to this since the 1980s (generating partial results once and saving them in a temporary variable for use local use later).

Figure 38 illustrates the differences between the methodologies. A computer program may be considered to be sequences of instructions and functions that have a start and an end with branches representing calls to the functions of the program.

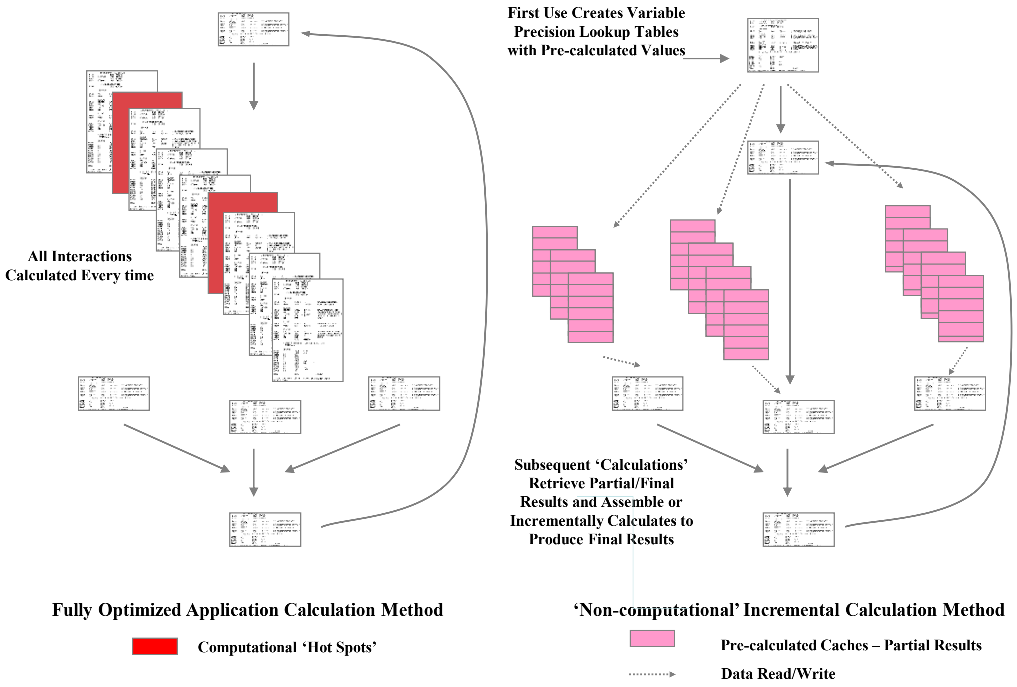

First Use Creates Variable Precision Lookup Tables with Pre-calculated Values
All Interactions Calculated Every time
Subsequent 'Calculations' Retrieve Partial/Final Results and Assemble or Incrementally Calculates to Produce Final Results
Fully Optimized Application Calculation Method
🟥 Computational 'Hot Spots'
'Non-computational' Incremental Calculation Method
Pre-calculated Caches – Partial Results
Data Read/Write

Figure 38 Differences between Computational and Noncomputation Models

In the case of molecular dynamics simulation software the routines for computing the nonbonded interactions use the greatest amount of CPU time and real clock time. In the computational model every function calculates everything every time the function is executed. If the algorithms in these portions of the program have already been coded optimally the only alternative is to not perform the calculations. In the noncomputational incremental method when the program is run the simulation is analyzed for a number of key parameters: type of force field and water model, cutoffs and type of simulation box (cubic, dodecahedron), definition of what is being simulated, and other simulation specific parameters. This information is used to construct a series of small memory caches of pre-computed results the first time the results are used that can be assembled or incrementally calculated at runtime for all subsequent processing eliminating the need to recomputed the entire equation or function. The number of precomputed cache tables created is dependent on CPU cache memory available.

2.4 Background

2.4.1 IEEE 754 Floating Point Standard

The IEEE 754 floating point standard defines digital representations for ranges of real numbers. The single precision real number format is represented using 32 bits and the double precision format is represented in 64 bits. Other formats such as half precision (16 bits) and quad-precision (128 bits) but are not implemented in most MD software such as GROMACS. All binary representations have three components: a fraction (mantissa), an exponent and a sign. The differences between the formats are in the number of bits used to represent the exponents and mantissas.

IEEE 754 Representations			
	Width	Range	Precision[a]
Single Precision	32 bits	$\pm1.18\times10^{-38}$ to $\pm3.4\times10^{38}$	≈ 7.2 digits
Double Precision	64 bits	$\pm2.23\times10^{-308}$ to $\pm1.80\times10^{308}$	≈ 15.9 digits
[a] Decimal digits precision is mantissa bits * Log10(2)			

Table 9 IEEE 754 Ranges Supported

A single precision value is represented in a 32 bit binary format as shown in figure 39.[40] For single precision values there are is a sign bit, 23 bits plus an implicit 24th bit for the mantissa, and an 8 bit exponent that is biased by 127.

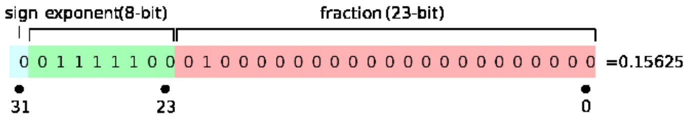


Figure 39 IEEE 754 Single Precision Format

A single precision value is represented using the following formula

The following equation is used to convert the binary 32 bit representation of a single precision value to a base 10 format where *I* is the first bit of the mantissa to the maximum supported by the format. Thus the value represented in Figure 39 is 0.15625.

$$x = (-1)^{sign}(1 + \sum_{i=1}^{23} \lceil b_{23-i} \rceil 2^{-i}) \times 2^{(e-127)}$$

Formula 1 IEEE 754 Single Precision Value

The 64 bit double precision representation has additional bits for both the exponent and mantissa as illustrated in figure 40.[41]

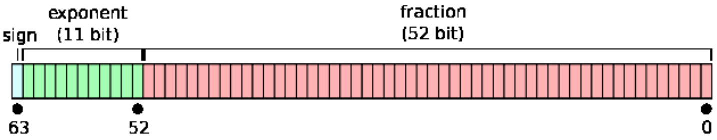


Figure 40 IEEE 754 Double Precision Format

The binary representation of a double precision value is converted to the base 10 format using the following formula. This formula varies from the single precision formula only in the number of bits that are used for the exponent and mantissa.

$$x = (-1)^{sign}(1 + \sum_{i=1}^{52} \lceil b_{52-i} \rceil 2^{-i}) \times 2^{(e-127)}$$

Formula 2 IEEE 754 Double Precision Value

By manipulating the binary format of the floating point representation of data it is possible to vary the precision of a floating point value. These reduced precision representations and knowledge of the equation or function can be used to create indices. These indices can then be used for accessing pre-computed results that are part of function, equation or application solution eliminating runtime computation by one or more table lookups. The multiple results returned can be ‘assembled’ or incrementally calculated to product the final results. Due to current CPU cache limitations variable precision can operate only over one or more small ranges of the IEEE format.

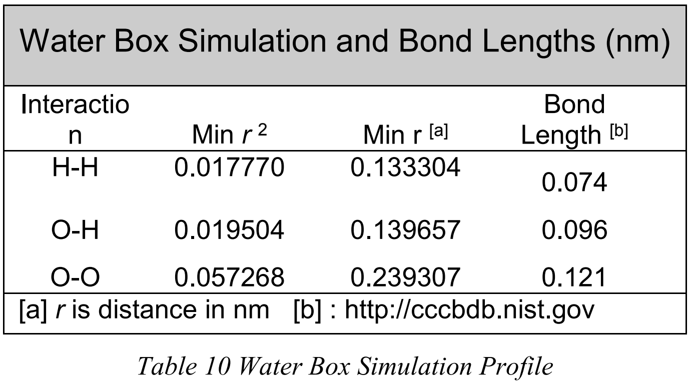
If variable precision floating point representations are used computationally intensive portions of functions, equations, or solutions can be completely avoided without violating the ‘floating’ characteristics of the IEEE 754 standard. The computationally intensive portions of the solution become ‘noncomputational’ at runtime. Use of variable precision requires application and solution specific knowledge that must be provided either by the developer or obtainable at runtime. This information is available to GROMACS at runtime based on the simulation parameter file (mdp), the force field and water model definitions and the simulation (.gro, .pdb) file itself. It also requires knowledge of the properties of the CPU and especially the cache size available to process the simulation.

2.4.2 Understanding What Is Being Modeled

The objective of MD software should be to make the *simulation being performed* run as fast as possible not to just be a general purpose MD tool. MD simulations are models that use limited precision constants and constraints that are approximations for the atoms and/or groups of atoms being modeled. Each force field and water model has its own constants and parameters with limited precision.

MD simulations are performed at the atomic level in ‘boxes’ or other containers with sizes in a range of nanometers not meters and the atomic level bonds and forces operate over picometers. The full IEEE 754 is therefore unnecessary. GROMACS is not modeling quarks or galaxies.

Simulations are run in a virtual real space that maps to a range of computational values. This real space and computational space can be represented by fewer bits for both the exponent and mantissa portions of the IEEE 754 representation used for general computation because of the nature of MD simulations. In simple water models such as SPC, SPCE and TIP3P water has fixed physical dimensions and constant properties such as charges at runtime in GROMACS.



Water Box Simulation and Bond Lengths (nm)

Interactio n	Min r^2	Min r [a]	Bond Length [b]
H-H	0.017770	0.133304	0.074
O-H	0.019504	0.139657	0.096
O-O	0.057268	0.239307	0.121

[a] *r* is distance in nm [b] : http://cccbdb.nist.gov

Table 10 Water Box Simulation Profile

The bond lengths for simulation purposes are usually considered fixed based on experimental data if we are using a bond constraining algorithm like LINCS or SHAKE.

Solute components of a simulation also have experimentally measured bonds in the range of picometers. It is important to note that bond lengths vary between actual force fields models, as they are also adjustable parameters. Also there are slightly different values coming from different experimental (and theoretical) methods. These differences give credibility to the hypothesis that MD simulations may run using a spatial and computational granularity of 1 pm.

Water Box Simulation and Bond Lengths (nm)			
Bond	Length	Bond	Length
H—H	74	H--C	109
C—C	154	H--N	101
N—N	145	H--O	96
O—O	148	H--F	92
F—F	142	H--Cl	127
Cl-Cl	199	H--Br	141
Br-Br	228	H--I	161
I—I	267	C--C	154
C—C	154	C=C	134
C—N	147	C≡C	120
C—O	143	O—O	148
C—S	182	O=O	121
C—F	135	N—N	145
C—Cl	177	N≡N	110
C—Br	194	C—I	214
Source: http://cccbdb.nist.gov			

Table 11 Experimental Bond Lengths

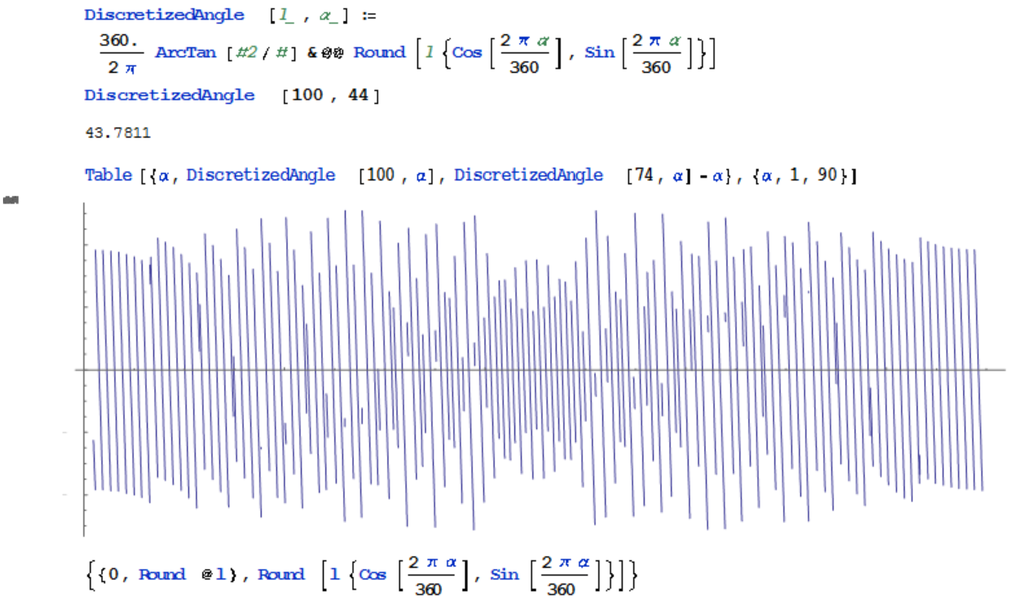
With few exceptions single precision is used for simulations. There are a few exceptions the most common is energy minimization (e.g. Steepest Descents, Conjugate Gradient or L-BFGS). It is unnecessary for GROMACS to perform MD simulations in a spatial granularity the size of a subatomic particle. GROMACS should be able to operate on a reduced precision form of the IEEE format. During the water box simulation that was used to provide the base for reproducibility for this study the following was observed during a 1 ns simulation where *r* represents the distance between interactions. The maximum distance in the water box simulation was the cutoff of 1.4 nm. Experimental bond angles have also been determined ± *n* degrees for H_2O.

Experimental Bond Angles (Degrees)				
	Min	Max	Average	
H2O	104.48	111.3	107.89	±4.82
H2N	103.25	121.6	109.85	±6.27
HCN	101.91	131	114.53	±7.25
CCH	109.9	129.2	120.1	±10.49
Source: http://cccbdb.nist.gov				

Table 12 Experimental Bond Angles

Using this data and the bond lengths for water, a mathematical model was created to determine the angular error for a water molecule rotating in 1 pm granular space. This calculation assumes that the water molecules are not flexible (the GROMACS default) and *only* shows the computational impact of rotating an inflexible water molecule through a coarser grain space. This difference is close to the range of experimentally determined values

An analysis using Mathematica[42] of a water molecule rotated through a discrete computation space of 1 pm shows an angular error of ± 0.47 degrees with no divergence.The angular error for rotating a water molecule through 1 pm discrete space is calculated as follows and shown in Figure 41.



```
DiscretizedAngle [l_ , α_] :=
 360./(2 π) ArcTan [#2 / #] & @@ Round [l {Cos [2 π α/360], Sin [2 π α/360]}]
DiscretizedAngle [100 , 44]

43.7811

Table [{α, DiscretizedAngle [100 , α], DiscretizedAngle [74, α] - α}, {α, 1, 90}]
```

```
{{0, Round @1}, Round [1 {Cos [2 π α/360], Sin [2 π α/360]}]}
```

Figure 41 Angular Error for a 1 pm Discrete Space

It was theorized that MD simulations could calculate nonbonded interactions in granular space of 1 pm based on the distance squared value in the nonbonded routines and give ‘equivalent’ results to the standard GROMACS builds. This implies that a variable form of the IEEE 754 format could be used to generate the results for nonbonded interactions using relatively small amounts of CPU cache memory for pre-computed tables. Values representing the 1 pm granular space may be represented as a reduced precision IEEE 754 format of three significant digits.

This is accomplished by reducing the number of bits *i* in the mantissa. The following defines the variable precision format for ≈3.01 to ≈3.91 significant digits over a limited range of exponents. The value of *i* represents the number of bits for the mantissa and the *e* must be in the set of biased values representing the range of powers of 2 that the application requires.

$$x = (-1)^{sign}(1 + \sum_{i=1}^{9,10,11,12} \lceil b_{23-i} \rceil 2^{-i}) \times 2^{(e \in \{120,121,...128\})-127}$$

Formula 3 Variable Precision 3 Significant Digits, 7 IEEE 754 Biased Format Exponents

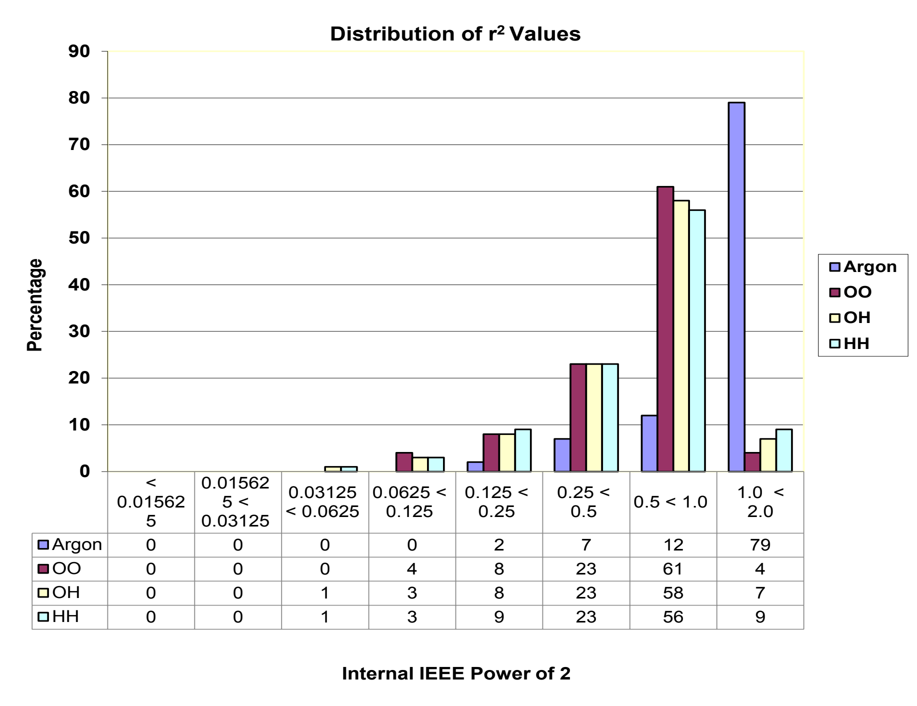
The test GROMACS water box simulation with a cutoff of 1.4 nm uses seven base 2 exponent ranges and only *three* of the ranges account for over 80% of the values in the simulation.

The most memory efficient alternative supports ≈3.01 significant digits and uses 28,672 bytes of storage for each single precision pre-calculated value (7 base 2 exponent ranges * 1024 entries/power of two * 4 bytes/single precision entry). The distance squared (r^2) and the type of nonbonded interaction is used to determine what intermediate or final results are created and are available for use during the execution of the simulation. The other alternatives of interest are ≈3.31, ≈3.61, and ≈3.91 significant digits requiring 57344, 114,688 and 229,376 bytes of storage respectively. Any of these fits easily within the CPU L2/L3 caches. The following shows the memory requirements for variable precision where 1.0 equals 1 nm^2 for all the GROMACS results and intermediates for water three atom to water three atom (W3A-W3A) for Lennard-Jones and Reaction Field. The use of a precision of ≈2.71 significant digits was also evaluated.

Mantissa Bits	Base 10 Significant Digits	Number of Mantissa Values	Distance Squared (r^2)	Number of Floating Point Values	Memory Required Per Result	W3A-W3A Interactions LJ + RF (bytes)
9	2.71	512	> 1 pm	3,584	14,336	100,352
10	3.01	1,024	1 pm	7,168	28,672	200,704
11	3.31	2,048	1 pm	14,336	57,344	401,408
12	3.61	4,096	1 pm	28,672	114,688	802,816
13	3.91	8,192	1 pm	57,344	229,376	1,605,632
14	4.21	16,384	0.1 pm	114,688	458,752	3,211,264
15	4.52	32,768	0.1 pm	229,376	917,504	6,422,528
16	4.82	65,536	0.1 pm	458,752	1,835,008	12,845,056
17	5.12	131,072	0.01 pm	917,504	3,670,016	25,690,112

Table 13 Memory Requirements for Variable Precision Format

This permits the use of the algorithm on solvent-solvent, solvent-solute and some solute-solute interactions depending on what is being simulated and the amount of CPU cache available. This study evaluated the variations of the algorithm for use in MD simulations using water boxes, small molecule, proteins and free energy studies. The number bits of the IEEE format biased exponent (base 2 powers) representation that is required is application and simulation specific. For the GROMACS nonbonded routines the uses the value 1.0 to represent 1 nm^2. Figure 42 shows the distribution of distance squared (r^2) values across the base 2 exponents for a water box and liquid argon simulation. The base 2 exponents are represented as decimal ranges to correspond to how the bits of the IEEE 754 format are used to form decimal values.



	< 0.015625	0.015625 < 0.03125	0.03125 < 0.0625	0.0625 < 0.125	0.125 < 0.25	0.25 < 0.5	0.5 < 1.0	1.0 < 2.0
Argon	0	0	0	0	2	7	12	79
OO	0	0	0	4	8	23	61	4
OH	0	0	1	3	8	23	58	7
HH	0	0	1	3	9	23	56	9

Figure 42 Distribution of Distance Squared for Ar-Ar and Water - Water

Simulation specific runtime ‘constants’ not known at compile time also may be useful in reducing the number and size of tables used for the developed method. These runtime constants can be used to reduce or eliminate calculations that are performed at runtime. In our water box example supporting Lennard-Jones reaction field for water three atom to water three atom the following may be observed from executions of the GROMACS 4.5.3 nonbonded routine nb_kernel212a.c. The GROMACS inner nonbonded calculations may therefore be eliminated/reduced based on simulation specific knowledge that is available when the simulation is run. Much of this data is available from the .mdp, water model, and force field chosen.

The noncomputational incremental algorithm can exploit the following:

- Distance squared is always within a limited range and is always a positive value.
- The Newton-Raphson 1/sqrt() is not needed because pre-calculated results are used
- Calculations of powers of the inverse square of the distance squared is not needed
- Vnb6, vnb12 and other intermediate results are not needed
- Variables twelve and six are constants
- qOO, qOH, qHH are simulation specific charge constants known based on the type of water model used
- facel which is container specific constant
- c6 and c12 Lennard-Jones constants
- The krf and crf variables are used as constants at runtime

The variables fs11 and vnba need to be returned from a lookup table for intermediate results based on distance squared due to their requirement for final results.

In summary, all of the above observations allow nonbonded interactions for water three atom to water three atom, Lennard-Jones and reaction field, values to be reduced to a single function where distance squared is the only variable that is needed to retrieve pre-calculated intermediate or final results.

The following pseudo code illustrates how this may be applied to GROMACS water three atom to water three atom, Lennard-Jones and reaction field nonbonded interactions in GROMACS. Portions of the pseudo code are from the C source code of the GROMACS nb_kernel212 routine. It is not intended that the reader understand the pseudo code but only understand the magnitude of the difference.

```
bitpattern11.fval = rsq11;
iexp11            = EXP_ADDR(bitpattern11.bval);
addr11            = FRACT_ADDR(bitpattern11.bval);
result11.bval     = cinvsqrtexptab[iexp11] |
                      cinvsqrtfracttab[addr11];
lu11              = result11.fval;
rinv11            = (half*lu33*(three-((rsq1*lu11)*lu11)));
rinvsq11          = rinv11*rinv11;
rinvsix           = rinvsq11*rinvsq11*rinvsq11;
vnb6              = c6*rinvsix;
vnb12             = c12*rinvsix*rinvsix;
vnbtot            = vnbtot + vnb12-vnb6;
krsq              = krf*rsq11;
vcoul             = qqOO*(rinv11+krsq-crf);
fs11              = (twelve*vnb12-six*vnb6+qqOO*(rinv11-
                      two*krsq))*rinvsq11;
vctot             = vctot + vcoul;
```

Pseudo-Code 1 GROMACS Code Required to Process Nonbonded LJ and Reaction Field

The developed methodology avoids the GROMACS NR 1/sqrt and retrieves three incremental data values based on distance squared (rsq11 in the sample code), each with a cache lookup using an integer index created with one/two instructions depending if the value is signed. Either single or double precision values may be retrieved. The tables are initialized the first time the nonbonded interaction is needed within the limits of the cache size. If the cache size is exceeded then the value can be calculated. The pseudo code is reduced to the following based on distance squared. This incrementally calculates the results based on three retrieved values and eliminates almost all floating point calculations in the inner nonbonded force calculations.

```
vnbtot              = vnbtot+ OO_VNBA(rsq11);
vctot               = vctot+ OO_VCOUL(rsq11);
fs11                = OO_FS11(rsq11);
```

Pseudo-Code 2 Equivalent Code for the Developed Method

2.5 Design of the 'Noncomputational' Incremental Model

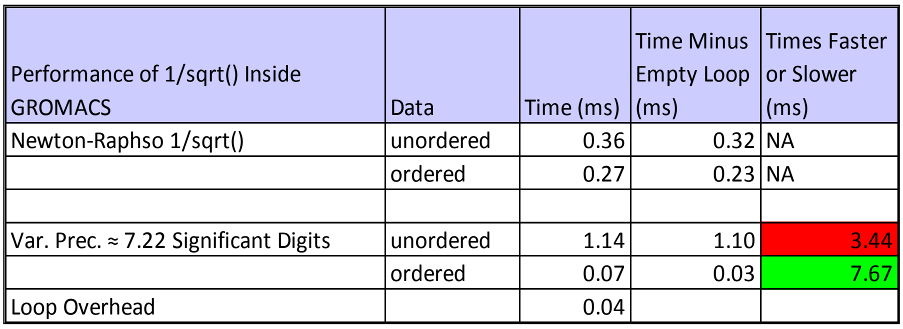
This algorithm exploits the available high speed CPU L2 or L3 cache memory to store each series of results used at the time the simulation is run.

Care must be taken not to use an excessive amount of cache memory or performance may become worse than computing the results every time.

Initially a test was performed on an *ordered* sequence of real number values representing every possible binary representation of the real numbers between the lower and upper bounds for the distance squared for a liquid argon simulation based on the definition of the simulation. This was compared to an equal number of *unordered* distance squared values from an actual liquid argon simulation. Table 12 shows the performance impact of having the distance squared values unordered.

The following table shows that an unordered set of values is 3.44 times slower and an ordered set of values is 7.67 times faster even when the full IEEE 754 ≈7.22 significant range is used. The data required for the ≈7.22 range is approximately 56MB. The time to sort them even with a fast binary sort algorithm making only a single pass through the data would be take prohibitively long. The excellent performance even with a very large variable precision cache comes from the ordered nature of the data which benefits from the memory pre-fetch and design of CPU caches and memory systems.

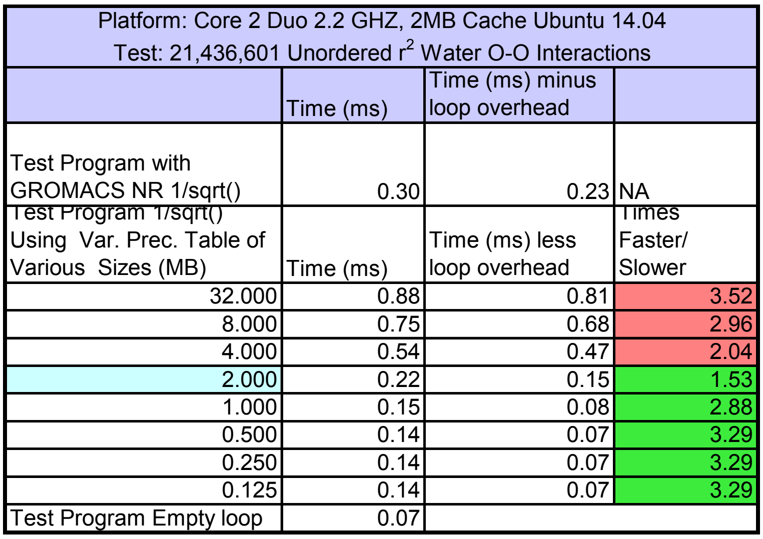
The only solution to the problem of unordered data with full IEEE single precision caches was to reduce the 'granularity' of the distance squared and to develop a variable precision algorithm. It was observed that there was a variation in the GROMACS 4.5.3 C single precision build in the lower 2 bits of the result from the Newton-Raphson 1/sqrt function and the result obtained from the C library function based calculation. This meant that in theory the floating point precision could be reduced and still provide a single precision version of GROMACS that would be 'good enough' for use in MD simulations. The following table shows the impact of exceeding the 2MB CPU cache of an Intel Core 2 Duo 2.2 GHZ processor even using



Performance of 1/sqrt() Inside GROMACS	Data	Time (ms)	Time Minus Empty Loop (ms)	Times Faster or Slower (ms)
Newton-Raphso 1/sqrt()	unordered	0.36	0.32	NA
	ordered	0.27	0.23	NA
Var. Prec. ≈ 7.22 Significant Digits	unordered	1.14	1.10	3.44
	ordered	0.07	0.03	7.67
Loop Overhead		0.04		

Table 14 Impact of Ordered vs Unordered Data

a variable precision algorithm. The test was conducted on 21M unordered distance squared O-O interactions.



Platform: Core 2 Duo 2.2 GHZ, 2MB Cache Ubuntu 14.04 Test: 21,436,601 Unordered r^2 Water O-O Interactions			
	Time (ms)	Time (ms) minus loop overhead	
Test Program with GROMACS NR 1/sqrt()	0.30	0.23	NA
Test Program 1/sqrt() Using Var. Prec. Table of Various Sizes (MB)	Time (ms)	Time (ms) less loop overhead	Times Faster/ Slower
32.000	0.88	0.81	3.52
8.000	0.75	0.68	2.96
4.000	0.54	0.47	2.04
2.000	0.22	0.15	1.53
1.000	0.15	0.08	2.88
0.500	0.14	0.07	3.29
0.250	0.14	0.07	3.29
0.125	0.14	0.07	3.29
Test Program Empty loop	0.07		

Figure 43 Performance Impact of Using Too Much CPU Cache

Figure 39 illustrates the differences between the current calculated every time method in GROMACS for determining the nonbonded interactions versus the ‘noncomputational’ incremental method. If too high a percentage of the CPU cache is used because the tables are too large performance gains decrease and can result in a loss of performance. In the testing performed it was found that if over approximately fifty percent (50%) of the CPU cache was used the performance could vary substantially between multiple executions of the test program.

This methodology works well when the pre-computed results can be contained within the CPU’s L2/L3 cache. Figure 44 illustrates what happens to performance if the limits of the L2/L3 cache are exceeded. Some architectures such as the IBM z196 have a very large L4 cache. This study evaluated Intel and AMD CPUs only but it could be easily be implemented on other architectures. This methodology is limited by the amount of L2/L3/L4 cache memory available. In the following example the CPU L2/L3 cache becomes fully utilized at a variable precision of about ≈5.12 significant digits. Other processors have greater L2/L3 cache memories that allow more extensive use of pre-calculated and incrementally calculated results. Even on older architecture CPUs with only 512KB of cache memory the algorithm can be used for water to water interactions (W3A-W3A Lennard-Jones reaction field) and also some intrinsic math functions over limited ranges. The slightly lower performance in

Figure 44 in the range of ≈3.01 to ≈4.82 significant digits was repeatable on the Intel Core i7 "Sandy Bridge" and attributable to the CPU cache architecture. Other Intel and AMD processors also showed similar behavior but at different ranges of significant digits. The Intel Core i5 'Haswell' platform gave unexpectedly little degradation in performance as the size of the application cache exceeded the physical CPU cache size.

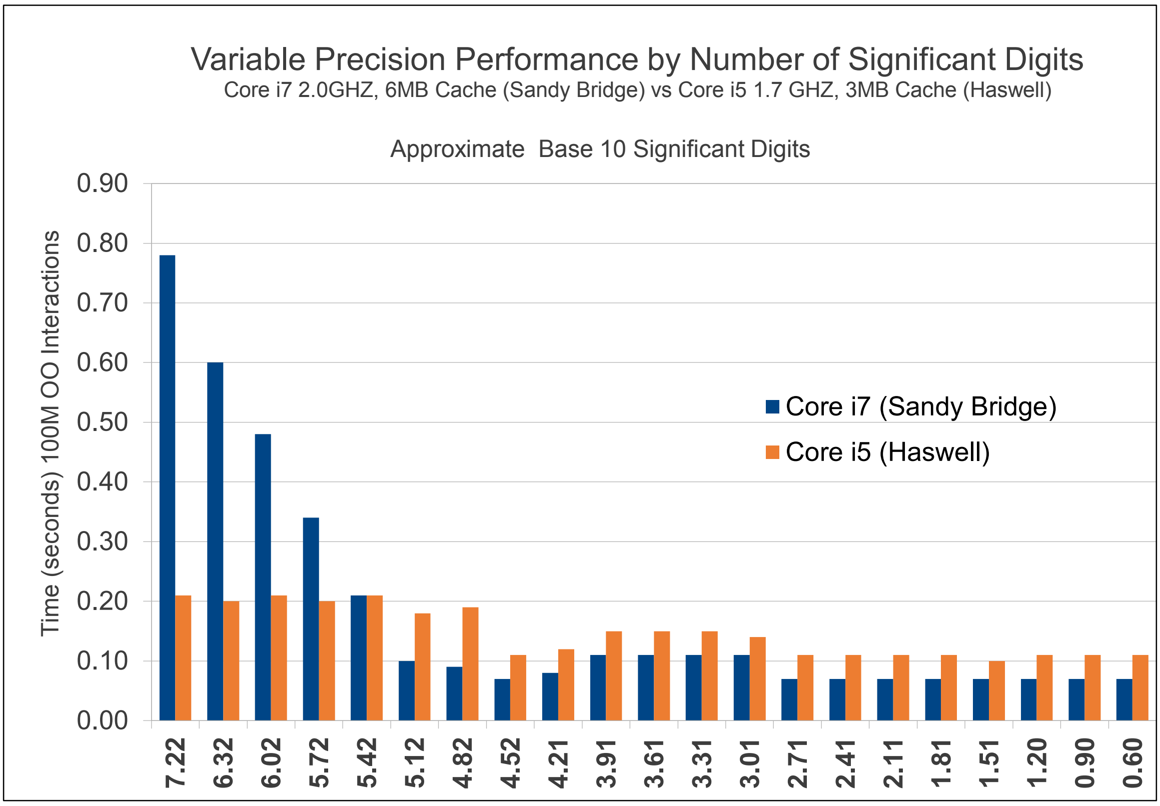


Figure 44 Variable Precision Performance vs. Significant Digits

2.6 The Algorithm

The algorithm uses a processed form of the raw IEEE format representation as a table lookup index for each intermediate or final result. This is accomplished without compromising the 'floating' characteristic of the IEEE format. A table is created and initialized on the first pass through the nonbonded routine and the values later used as 'constants'. The tables are based on the inter-particle distance squared and this eliminates the need to perform the 1/sqrt operation. The values of the various intermediate results may be looked up at runtime and can assembled or incrementally calculated to produce a final results such as forces.

The index is created using the following pseudo code. The lower boundary of the distance squared values that are supported by the table is subtracted from the distance squared value (r^2) for the interaction, treating both native floating point values as 32 or 64 bit integers. The resulting value is shifted a number of bits to the right to reduce the precision and therefore reduces the spatial 'granularity' for the interaction. The resulting integer value is then used as an index to lookup the result or intermediate results in one or more tables or sections of tables and the value returned is either a single precision or double precision value depending on the requirements of the application. The same index may be used to retrieve multiple results. Multiple intermediate results may be used to incrementally calculate a final result. Applying the variable precision methodology to nonbonded interactions, *r* represents distance, r^2 represents the floating point distance squared value used to create the index, F_{tab} is the lookup table for the reduced precision results, LOWER_BOUND is the lowest value for r^2 that is possible in floating point format (but used as an integer) and NUM_BITS is the number of bits to shift to the right to reduce the precision. There is no sign bit since distance squared (r^2) is always positive otherwise it would be necessary to mask it out or to use two tables for positive and negative results depending on the intended use. The range of r^2 is continuous and over a single range so no additional processing is required for boundary conditions or multiple table lookups for a single function.

$$F(r) \approx F_{tab}[(\text{r2.binary} - \text{LOWER\_BOUND}) >> \text{NUM\_BITS}].\text{float}$$

Multiple intermediate results can be stored as offsets in the same table eliminating the instructions that reload of the base address of the table. Using SSE2, SSE 4.1 and AVX2 this can be reduced from 11 to 7 to 3 instructions respectively that can be highly pipelined. The reduction in the number of instructions used from the SSE2 and the SSE 4.1 version did not yield a significant increase in performance and on the Intel Core i7 'Sandy Bridge' architecture it showed a slight reduction in performance. SSE 4.1 was not supported on the AMD systems available.

A number of these instructions use less than one clock cycle. SSE2 and SSE 4.1 can process four single precision values at one and AVX2 can process 8 values at once. The algorithm requires SIMD instructions for integer and logical operations that do not exist in either SSE or AVX and does not support these instruction set extensions.

The tables of intermediate results are generated by sequencing through all of the possible values of the variable precision representation so that the results ‘float’ in the same way as the IEEE 32 and 64 bit formats. As the table generation is performed a simulated additional ‘guard bit’ is added to the right of the reduced precision mantissa before the latter is used to initialize the table alternating on odd and even values of the mantissa.

This prevents a small divergent error from the binary truncation that occurs at runtime otherwise it would be necessary to treat the mantissa as a scaled base 2 value at runtime and this would defeat much of the performance gain of the algorithm.

All calculations for table entries are performed in double precision and the results stored in the tables as either single or double precision. No interpolation is required. The following figure illustrates the ‘floating’ nature of the implementation. The mantissa portion of the IEEE 764 standard single precision format supports approximately 8M values for each power of two. By reducing the mantissa bits it is possible to reduce the precision without destroying the ‘floating’ property of the format. This makes the variable format suitable for use with functions/equations independent of their slope or continuity.

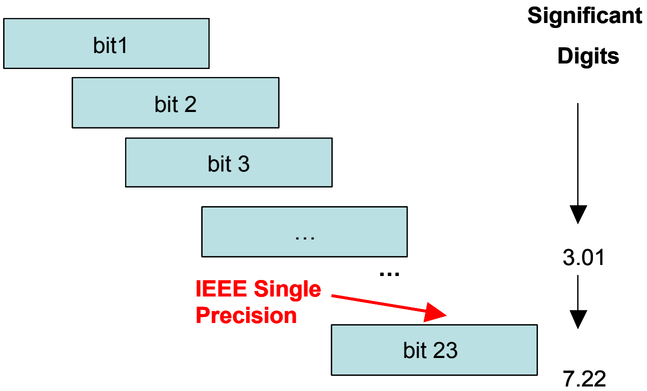


Figure 45 Variable Precision Mantissa 'Floats' Like IEEE 754

An alternative implementation would be to perform an integer conversion of the floating point value applying a scaling factor. In the following example the *int* function converts the results of the scaling factor times distance squared and looks the result up in a force table.

$$F(r) \approx F_{\text{tab}}[\text{int}(\text{scale}*r^2)]$$

Converting a floating point value to an integer causes a divergence from the IEEE standard and destroys the ‘float’ properties of the values and generates a diversion of the results obtained from those that would be generated by a floating point calculation. This diversion is highly function specific. Using a floating point to integer conversion results in fewer and fewer values to represent much larger forces as the distances become closer. When the algorithm is applied to other functions/equations the variations of the slope in portions of the functions/equations will show large differences in the offsetting adjustment based on the slope on the axis.

If the integer conversion approach is applied to general computation the index will not appropriately sample the distribution of function results. Figure 46 shows that the floating point to integer method does not sample the function/equation uniformly based on the IEEE floating point property.

It is highly undesirable to use an integer conversion to produce an index using a specific value when there are substantial differences in the results contained in the table that are being looked up. In this case there are larger and larger differences in the forces as the distance becomes smaller and fewer base 10 digits to represent the increasingly large forces. [26]

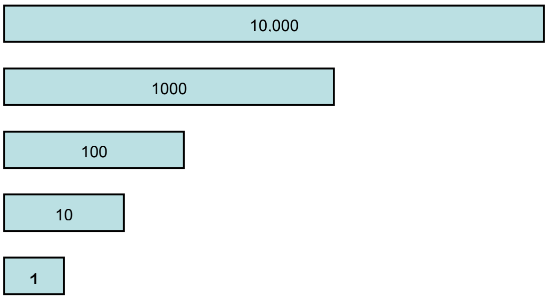


Figure 46 Effect of Integer Conversion on a Real Number

There is no ‘floating’ property in an integer conversion that would preserve the same precision independent of the distance. Figure 48 shows the increasingly large error forward backward error correction required with the integer conversion method for the 1/sqrt function. The same problem was shown in the study of Nilson in 2009 when the above methodology was applied to force equations. [7]

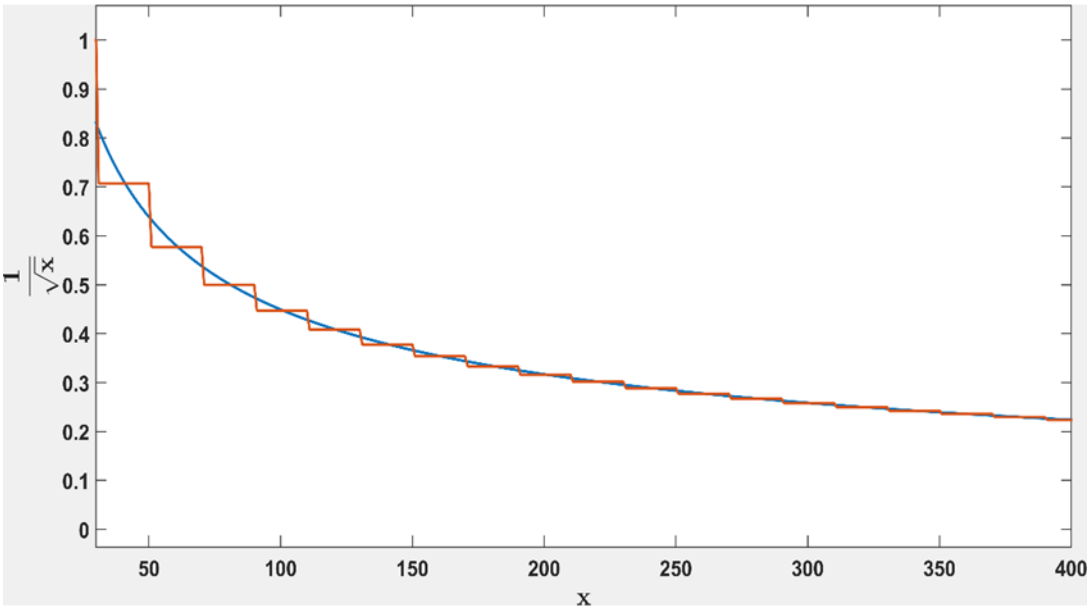


Figure 48 Increasing Deviations in 1/sqrt function with Integer Conversion

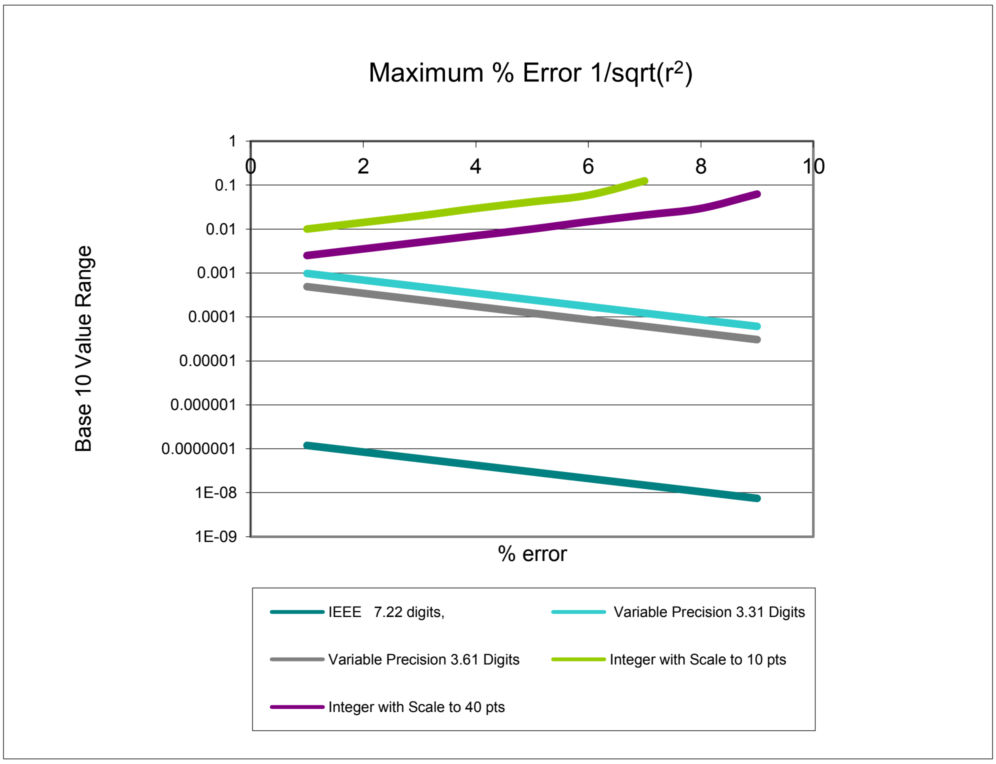


Figure 47 Method Relative Error vs. IEEE 754 Standard in Percentage

Figure 47 shows the variable precision implementation paralleling the floating point standard and an integer conversion based method diverging until it reaches a lower limit cutoff that is dependent on the scaling. This cutoff does not correspond to any specified simulation parameters but is a result of the scaling before the integer conversion occurs.

Another of the side effects of an integer conversion method is the elimination of cases where a base 10 value may have more than one base 2 representation or it may have no base 2 representation at all. This property of the floating point representation is essential to preserving the same number of significant digits as values increase or decrease. The following table shows the nature of this property. These cohorts are the reason that programmers can not readily compare floating point values with the same base 10 representation because they may have different internal representations.

	Base 10	Base 2
Base 10	Decimal	Hexadecimal
2.500003e-01	0.2500000	3e80000b
2.500004e-01	0.2500000	3e80000c
2.500004e-01	0.2500000	3e80000d
2.500004e-01	0.2500000	3e80000e
2.500004e-01	0.2500000	3e80000f
2.500005e-01	0.2500000	3e800010
2.500005e-01	0.2500010	3e800011
2.500005e-01	0.2500010	3e800012
2.500006e-01	0.2500010	3e800013

Table 15 Example IEEE Floating Point Cohort

2.7 Performance Testing of the Algorithm

2.7.1 Improving Reciprocal Throughput

Using the algorithm can greatly reduce the number of instructions required to generate the object interaction results. The reciprocal throughput analysis is based on the work of Agner Fog.[43] Reciprocal throughput is one measure of performance.

Reciprocal throughput has been defined as the average number of clock cycles per instruction for a series of independent instructions of the same kind in the same thread on a single core assuming that the operands of each instruction are independent of preceding instructions. The values used in this study are from Agner Fog’s independent evaluation of the performance of a wide range of Intel and AMD processors. The values used are the reciprocals of the throughputs when the instructions are not part of a limiting chain of dependent instructions. For example, assuming that the operands are independent a reciprocal throughput of 2 cycles for an FMUL instruction means that another FMUL instruction can start 2 clock cycles after the previous FMUL and a value of 0.33 for ADD means that 3 integer additions can be performed per clock cycle.

Thus, the sum of the instruction cycles that a given algorithm uses may be used for a relative comparison but with limitations. One major limitation is memory architecture and whether or not the data is available in one of the levels of cache memory. Memory access takes 2-3 cycles if cached but several hundred if not.[44]

In principal if the number of computer instructions and the number of 'cycles' are reduced then the software should run faster. This however may be a deceiving measure because the number of cycles per instruction even for the same instruction varies greatly on what instructions are around it and where it is retrieving data. For example, modern CPUs will attempt to optimize on chip performance by reordering instructions, performing operations in parallel or attempting to predict branching. CPU instructions execute using micro operations that may be scheduled in parallel with neighboring instructions to avoid 'blocking' of program execution. Part of an instruction may execute in parallel with part of another instruction based on micro operations.[45]

Generally non-arithmetic instructions take fewer cycles than numeric instructions and are more easily optimized in the CPU pipeline frequently executing in less than one cycle. The number of cycles an instruction takes is also highly dependent on where the data resides. If the data is in the L2/L3 cache execution is very fast, but if it resides in main system memory the memory access could be 100 times slower.

Using the present algorithm can greatly reduce the number of instructions required to generate the nonbonded interaction results. It may be noted in Figure 45 that the SSE and AVX instruction sets are not included from testing. This is because they lack instructions for performing certain SIMD bit manipulation instructions that Intel later added in SSE2 and AVX2 that are essential to the developed algorithm. The following figure shows the reduction in the number of instructions required to obtain the equation results as compared with the GROMACS 4.5.3 assembly language code. Most of the instructions have a cycle time of 1 but the developed algorithm uses numerous memory and register instructions that have cycle times of .33 cycles/instruction on Intel Sandy Bridge and .22 cycles/instruction on Intel Haswell architecture further improving the performance.

Number of Instructions					
	Lennard-Jones Reaction Field	Lennard-Jones Only	# Results	Est x Faster LJ-RF	Est x Faster LJ Only
Single Precision					
GROMACS 4.5.3 SSE	35	25	4	NA	NA
Developed Method					
SSE2	16	16	4	2.2	1.6
SSE4.1	11	11	4	3.2	2.3
AVX2	3	3	8	11.7	8.3
Double Precision					
GROMACS 4.5.3 SSE2	43	43	2	NA	NA
Developed Method					
SSE2	9	16	2	3.9	1.6
SSE4.1	7	11	2	5.0	2.3
AVX2	3	3	4	11.7	8.3

Table 16 Instruction Counts

Figure 50 shows a comparison of the number of instructions per result required to calculate the nonbonded interactions for SPC water to water (1/sqrt, Lennard-Jones, reaction field and argon (Lennard-Jones only). Argon to Argon interactions are also shown that only need to solve the Lennard-Jones equation. These are *only estimates* based on instruction times and *do not reflect* actual algorithmic performance

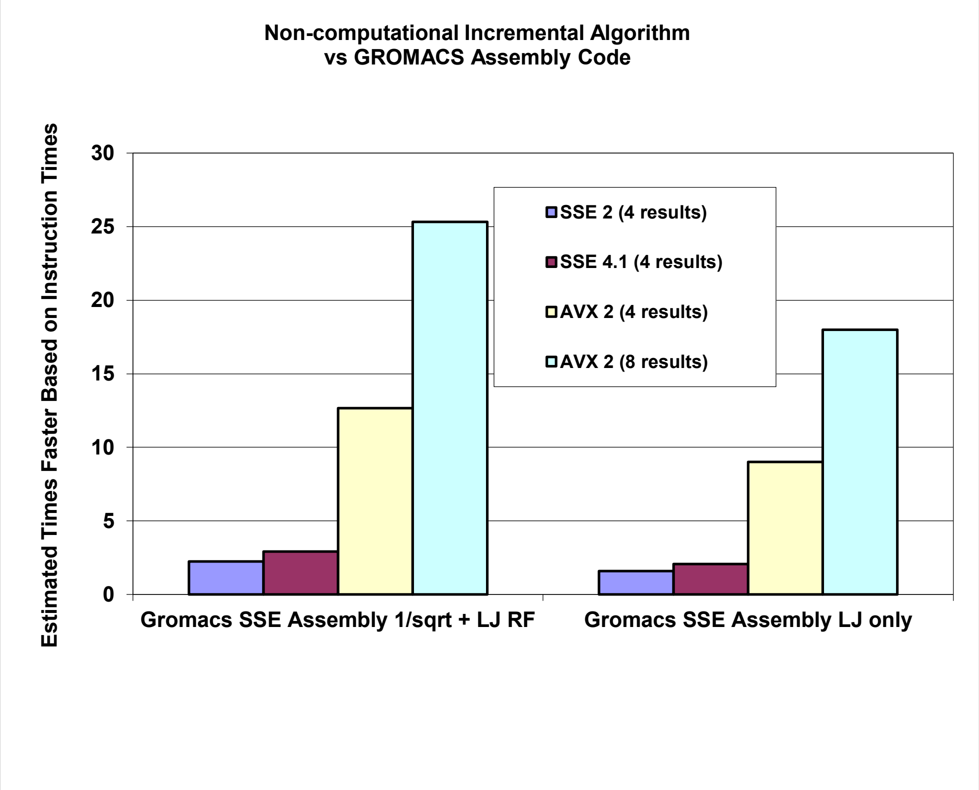


Figure 49 Performance Estimates Per Result Based on Instruction Count

2.7.2 Runtime Testing

In order to test the application level performance a C program was written that reads 100M oxygen to oxygen interactions and then processes them inside a timing loop. The gcc 4.7 compiler was used to output the assembly language version of the program and test code from GROMACS and the developed algorithm was inserted into the assembly language test loop. The GROMACS version 4.5.3 SSE and SSE2 assembly code from the nonbonded kernel routines for Lennard-Jones reaction field (nb_kernel212), and Lennard-Jones only (nb_kernel010) as well as for the noncomputational incremental method using SSE2, SSE 4.1 and AVX2 with a special granularity of ≈3.01 significant digits was used for comparison.

2.7.3 Comparison of Force Only Tests

The forces only tests for the nonbonded routines (excluding the distance calculations and the application of the forces after calculation) showed the following results. The GROMACS 4.5.3 SSE code was copied and pasted and changed only to reference local variables. The Lennard-Jones Reaction Field testing included the 1/sqrt calculation for the GROMACS versions. The noncomputational incremental method was written in hand coded assembly code and inserted into the timing loop of the program.

The very large improvement in force calculation performance is due to the fact that the method does not require the calculation of the 1/sqrt, the reduced number of instructions and the benefit that the binary and integer operations receive in the CPU pipeline.

The force only calculations represent only portion of the calculations within the nonbonded routines. Distance calculations cannot benefit from the use of the developed method.

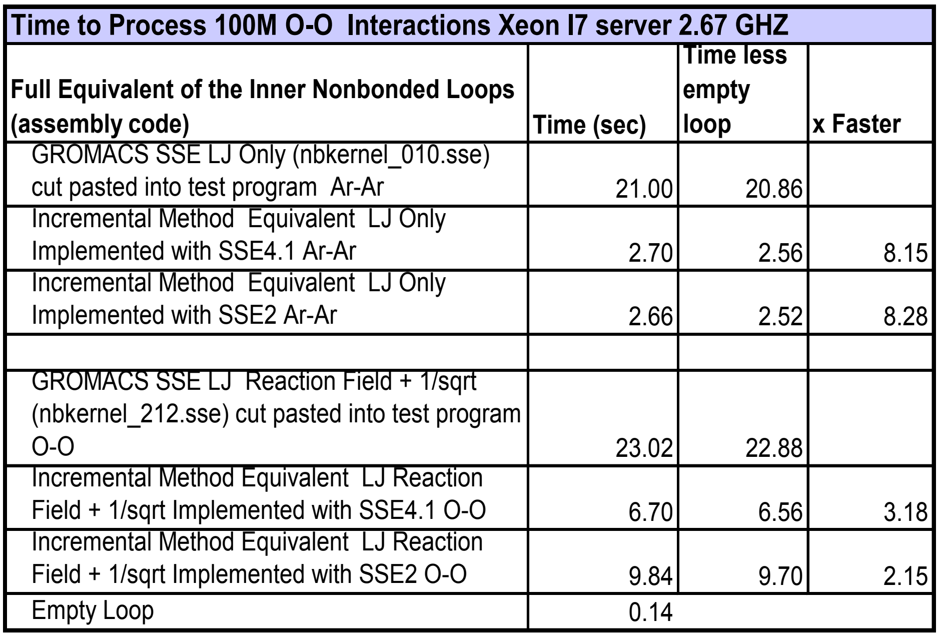
Time to Process 100M O-O Interactions		I7 server 2.67 GHZ	
Nonbonded Forces Only	Time (sec)	Time minus empty loop	x Faster
GROMACS SSE LJ Only (nbkernel_010.sse) cut pasted into test program Ar-Ar	7.9	7.76	
Incremental Method Equivalent LJ Only Implemented with SSE4.1 Ar-Ar	0.38	0.24	32.33
Incremental Method Equivalent LJ Only Implemented with SSE2 Ar-Ar	0.39	0.25	31.04
GROMACS SSE LJ Reaction Field + 1/sqrt (nbkernel_212.sse) cut pasted into test program O-O	16.78	16.64	
Incremental Method Equivalent LJ Reaction Field + 1/sqrt Implemented with SSE4.1 O-O	0.38	0.24	69.33
Incremental Method Equivalent LJ Reaction Field + 1/sqrt Implemented with SSE2 O-O	0.39	0.25	66.56
Empty Loop	0.14		

Table 17 Performance of O-O Interactions

Intel Xeon 5650 2.67 GHZ 12MB Cache 6 cores (Westmere)

2.7.4 Comparison with the Full Nonbonded Kernel Routines

The same performance test method was used as the force only testing except the full nonbonded assembly code from the GROMACS nonbonded routines was copied and pasted into the timing loop of the test program. In order to test the noncomputational incremental method the same GROMACS assembly code was copied but the portions that calculate the 1/sqrt function and perform force calculations was replaced with the new algorithm. The performance improvements of 2.15 and 3.18 times faster is in line with the reciprocal throughput estimate on the Intel Xeon 5650 2.67 GHZ 12MB Cache 6 cores (Westmere).



Time to Process 100M O-O Interactions Xeon I7 server 2.67 GHZ			
Full Equivalent of the Inner Nonbonded Loops (assembly code)	**Time (sec)**	**Time less empty loop**	**x Faster**
GROMACS SSE LJ Only (nbkernel_010.sse) cut pasted into test program Ar-Ar	21.00	20.86	
Incremental Method Equivalent LJ Only Implemented with SSE4.1 Ar-Ar	2.70	2.56	8.15
Incremental Method Equivalent LJ Only Implemented with SSE2 Ar-Ar	2.66	2.52	8.28
GROMACS SSE LJ Reaction Field + 1/sqrt (nbkernel_212.sse) cut pasted into test program O-O	23.02	22.88	
Incremental Method Equivalent LJ Reaction Field + 1/sqrt Implemented with SSE4.1 O-O	6.70	6.56	3.18
Incremental Method Equivalent LJ Reaction Field + 1/sqrt Implemented with SSE2 O-O	9.84	9.70	2.15
Empty Loop	0.14		

Table 18 Assembly Code Algorithm vs GROMACS SSE

Intel Xeon 5650 2.67 GHZ 12MB Cache 6 cores (Westmere)

2.7.5 Intel Core i7 ‘Sandy Bridge’ and AMD Performance

The same tests were performed on the Intel Core i7 ‘Sandy Bridge’ architecture that supports the AVX instruction set and on two AMD systems. The results showed an extreme improvement in performance between the generations of the Intel Core i7 architecture. The ‘Sandy Bridge’ architecture processed the full nonbonded interactions for 100M O-O interactions 14.3 times faster and for Ar-Ar interactions 15.5 times faster than the GROMACS 4.5.3 SSE assembly code. It was also observed that the two AMD CPUs tested only performed about 1.7 times faster. These were older AMD CPUs but the same performance improvement was observed on a 64 core AMD server. There are fundamental differences between the AMD and Intel cache architectures that may account for this. There were no values reported for the AMD CPUs for SSE 4.1 instructions because SSE 4.1 only exists on the Intel CPUs. It is interesting to note that the SSE 4.1 implementation on ‘Sandy Bridge’ was slightly *slower* than the SSE2 implementation even though fewer instructions were used.

The Intel ‘Sandy Bridge’ architecture has a number of improvements that may account for this difference. Most important are probably the new cache design, larger CPU cache and wider data paths. It functions almost as if it is processing 8 single precision values simultaneously rather than 4 even though no AVX instructions were coded into the test program and AVX2 instructions are not supported on Intel Core i7 Sandy Bridge.

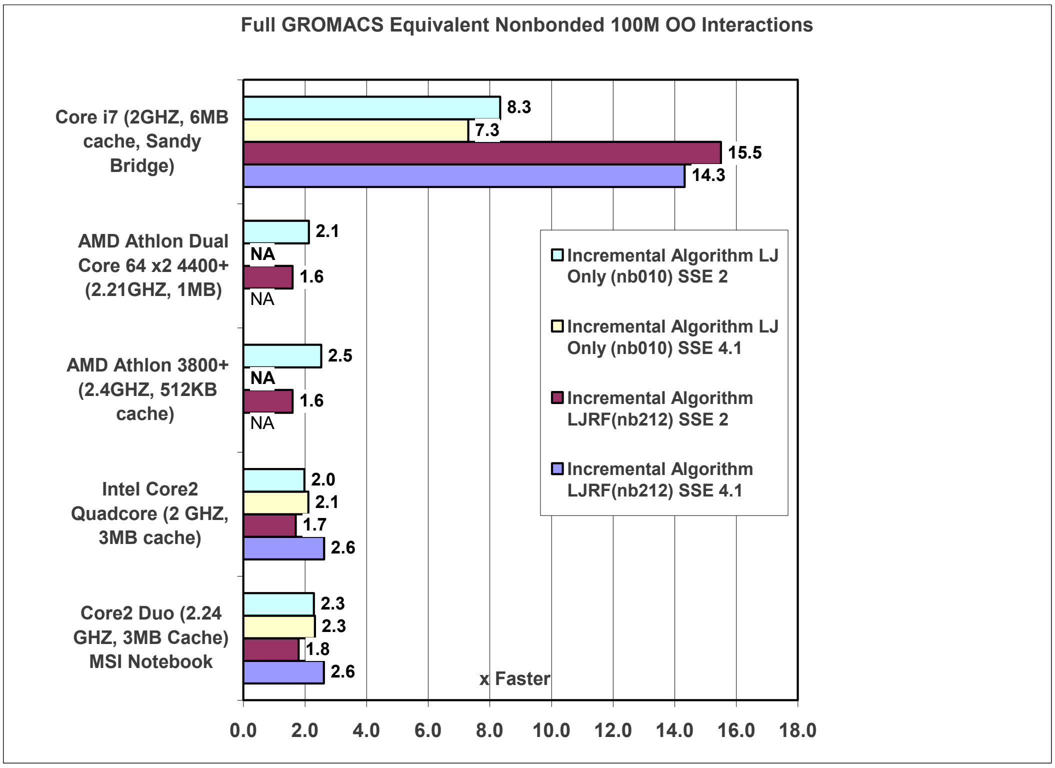


Figure 50 Variable Precision Assembly Code Algorithm vs GROMACS SSE

2.8 Conclusions

The major bottleneck in molecular dynamics simulations is the calculation of nonbonded interactions at each time step. GROMACS 4.5.3 has highly optimized hand coded SSE and SSE2 assembly code to perform these functions. The calculation of the forces and intermediate results may be improved substantially by use of a noncomputational and incremental computation model that exploits a variable precision format based on the IEEE 754 standard for single precision values. This variable precision format effectively permits the simulation to run in 1pm ‘space’. Using a coarse grain approach allows the creation of indices for accessing pre-computed results without the artifacts associated with a simple conversion to integer lookup method.

It has been shown than a series of tables paralleling the IEEE 754 standard supporting variable precision coarse grain space using 3 significant digits precision can be generated to support water to water and water to solute interactions using the GROMACS Lennard-Jones reaction field (nbkernel212) and the Lennard-Jones only (nbkernel010) assembly language routines. These assembly routines perform 15 times faster on an 2GHZ Intel Core i7 ‘Sandy Bridge’ and 2.6 times faster on an Intel Core2

Quad core 2.0 GHZ and 3.2 times faster on an Intel Xeon 5650 2.67 GHZ 12MB Cache 6 cores 12 threads (Westmere). Improvements on various AMD CPUs showed an improvement of 1.6 times faster.

Testing was also performed on a number of C programming library functions such as sqrt, log, tan, cos, etc. with results in performance improvements that were in the range of 11 to 125 times faster over a specified range of values at a reduced precision. The algorithm was also tested against an integer based lookup table method and was found to be 7 times faster on the Intel ‘Sandy Bridge’ Core i7.

This algorithm may be applied to the real space part of PME and other computationally intensive portions of GROMACS or other MD Software.

The performance of the algorithm is highly dependent on the percent of CPU and clock time used for performing nonbonded calculations, the amount of cache memory used for the incremental result caches, the overall memory requirements of the simulation, processor technology used, cache architecture, motherboard/blade design, node configuration and network bandwidth. In all cases tested Intel processors performed better than AMD with Intel ‘Sandy Bridge’ and ‘Ivy Bridge’ greatly exceeding the performance of earlier Intel architectures. Preliminary testing on the Intel ‘Haswell’ architecture shows a further increase in performance based on improvements in the cache architecture and the addition of new instructions such as vsgather that permit the developed algorithm to reduce the number of instructions by half. It is expected at in AVX512 environments and other environments with the ability to process more data per cycle the algorithm will continue to greatly exceed the performance of calculating the results where the developed algorithm can be used.

The developed algorithm has a number of limitations. The most significant is the amount of CPU L2, L3, L4 cache available for use with the lookup tables for the intermediate results. This study showed that there was a substantial reduction in performance as compared to actually computing the results when the tables used for the incremental results exceeded the CPU L2/L3 cache. The maximum variable precision format using CPUs with 2-8MB L2/L3 cache is ≈5.12 significant digits. It should also be considered that other applications may be using the core or processor that may cause cache misses for an application using the incremental lookup tables. Attention should be given to associating threads/processes using the algorithm with a physical core, CPU block and blade/node. If processes are allowed to move from core to core cache misses will result and this will have an adverse impact on performance. Therefore, associating a thread/process with a core and cache is essential.

2.9 Acknowledgments

The author would like to express his thanks to Dr. Paulo Martel, Centre for Biomedical Research, the University of the Algarve, Faro, Portugal for his confidence that this project was feasible and for his support during the project.

The author is also grateful to members of the Computational Systems Biology Group, Center for Neuroscience and Cell Biology, UC-Biotech, University of Coimbra led by Dr. Armindo Salvador; the Biological Chemistry Group, Department of Chemistry, University of Coimbra led by Dr. Maria Joao Moreno; and to Dr. Claudio Soares, New University of Lisbon, Lisbon, Portugal for stimulating discussions about the project and suggestions on approaches to validation of the methodology.

The Biological Chemistry Group, Department of Chemistry, University of Coimbra also provided substantial computational resources that made the free energy studies possible. The author would like to thank Virtual Strategy, Inc., Boston, MA for funding this research and supporting the patent application.

Keywords: performance, incremental computation, non-computation, nonbonded, AVX2

2.10 References

[1] van Gunsteren, Wilfred F.; Berendsen, Herman J. C.; Rullmann, Johan A. C. (1 January 1978). "Inclusion of reaction fields in molecular dynamics. Application to liquid water". Faraday Discussions of the Chemical Society 66: 58. doi:10.1039/DC9786600058

[2] Lennard-Jones, J. E. (1924), "On the Determination of Molecular Fields", Proc. R. Soc. Lond. A 106 (738): 463–477, Bibcode:1924RSPSA.106..463J, doi:10.1098/rspa.1924.0082

[3] McCammon, J. A.; Gelin, B. R.; Karplus, M. Nature 1977, 267, 585.

[4] Freddolino, P. L.; Arkhipov, A. S.; Larson, S. B.; McPherson, A.; Schulten, K. Structure 2006, 14, 437

[5] Sanbonmatsu, K. Y.; Tung, C. S. J Struct Biol 2007, 157, 470.

[6] David E. Shaw, Martin M. Deneroff, Ron O. Dror, Jeffrey S. Kuskin, Richard H. Larson, John K. Salmon, Cliff Young, Brannon Batson, Kevin J. Bowers, Jack C. Chao, Michael P. Eastwood, Joseph Gagliardo, J.P. Grossman, C. Richard Ho, Douglas J. Ierardi, István Kolossváry, John L. Klepeis, Timothy Layman, Christine McLeavey, Mark A. Moraes, Rolf Mueller, Edward C. Priest, Yibing Shan, Jochen Spengler, Michael Theobald, Brian Towles, and Stanley C. Wang (July 2008). "Anton, A Special-Purpose Machine for Molecular Dynamics Simulation" (PDF). Communications of the ACM (ACM) 51 (7): 91–97. doi:10.1145/1364782.1364802. ISBN 978-1-59593-706-3.

[7] Nilsson, L, 2009, "Efficient table lookup without inverse square roots for calculation of pair wise atomic interactions in classical simulations", Journal of Computational Chemistry, *Volume 30, Issue 9,* pages 1490–1498, 15 July 2009, DOI: 10.1002/jcc.21169

[8] Elofsson, A.; Nilsson, L. J Mol Biol 1993, 233, 766.

[9] Lindahl, E, Hess, B and van der Spoel, D 2001, 'GROMACS 3.0: a package for molecular simulation and trajectory analysis.' Journal of Molecular Modeling, 7(8): p. 306-317.

[10] D. van der Spoel, E. Lindahl, B. Hess, A. R. van Buuren, E. Apol, P. J. Meulenhoff, D. P. Tieleman, A. L. T. M. Sijbers, K. A. Feenstra, R. van Drunen and H. J. C. Berendsen, Gromacs User Manual version 4.5.6, www.gromacs.org (2010)

[11] Hess, B, et al., 2008, 'GROMACS 4: Algorithms for highly efficient, load-balanced, and scalable molecular simulation.' Journal of Chemical Theory and Computation, 4(3): p. 435-447.

[12] Abraham, MJ ; van der Spoel, D ; Lindahl, E, Hess, B 2014, & the GROMACS development team, *GROMACS User Manual version 5.0.2*, www.gromacs.org

[13] Roux, B.; Simonson, T. Biophys Chem 1999, 78(1/2), 1.

[14] Feig, M.; Brooks, I.; Charles L. Curr Opin Struct Biol 2004, 14, 217.

[15] Malevanets, A.; Kapral, R. J Chem Phys 2000, 112, 7260.

[16] NVIDIA Corporation, 2015, CUDA Runtime API, http://docs.nvidia.com/cuda/pdf/CUDA_Runtime_API.pdf

[17] Stone JE, Hardy DJ, Ufimtsev IS, Schulten K. 2010 "GPU-Accelerated Molecular Modeling Coming Of Age." Journal of molecular graphics & modelling. 2010;29(2):116-125. doi:10.1016/j.jmgm.2010.06.010.

[18] Stone JE, Hardy DJ, Ufimtsev IS, Schulten K. 2010 "GPU-Accelerated Molecular Modeling Coming Of Age." Journal of molecular graphics & modelling. 2010;29(2):116-125. doi:10.1016/j.jmgm.2010.06.010.

[19] Mirco Wahab, gromacs.org_gmx_developers-developers@maillist.sys.kth.se, June 7, 2015

[20] Intel Advanced Vector Extensions Programming Reference, website, Intel 2000-04-05.

[21] AMD64 Technology, AMD64 Architecture Programmer's Manual Volume 6: 128-Bit and 256-bit XOP and FMA4 Instructions, Publication no. 43479 revision 3.01, 2009, Advanced Micro Devices

[22] Intrinsics for Intel® Advanced Vector Extensions 2, https://software.intel.com/en-us/node/513925, Intel Retrieved 06-09-2015

[23] Moore, Gordon E. 1965. "Cramming more components onto integrated circuits", *Electronics Magazine. p. 4. Retrieved 2006-11-11*

[24] 'Moore's Law is Dead says Gordon Moore', http://www.techworld.com/news/operating-systems/moores-law-is-dead-says-gordon-moore-3576581, Apr 13, 2010

[25] 2011. "Cycles, cells and platters: an empirical analysis of hardware failures on a million consumer PCs. Proceedings of the sixth conference on Computer systems (EuroSys '11). pp 343-356.

[26] Páll, Szilárd, Abraham, Mark, James Kutzner, Carsten Hess, Berk Lindahl, Erik. 2015. "Tackling Exascale Software Challenges in Molecular Dynamics Simulations with GROMACS", Editors, Markidis, Stefano Laure, Erwin. Book: Solving Software Challenges for Exascale, Publisher: Springer International Publishing, volume: 8759, Series Title: Lecture Notes in Computer Science, ISBN 978-3-319-15975-1, DOI 10.1007/978-3-319-15976-8_1

[27] Verlet, L. Phys Rev 1969, 159, 98.

[28] Darden, T.; York, D.; Pedersen, L. J Chem Phys 1993, 98, 10089.

[29] Gruber CC, Pleiss J 2010 Sep 1, 'Systematic benchmarking of large molecular dynamics simulations employing GROMACS on massive multiprocessing facilities.' Journal of Computational Chemistry 2011 Mar; 32(4):600-6. doi: 10.1002/jcc.21645. Epub.

[30] http://www.infinibandta.org/content/pages.php?pg=technology_overview

[31] Amdahl, Gene M. (1967). "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities". AFIPS Conference Proceedings (30): 483–485. doi:10.1145/1465482.1465560.

[32] Daniels220, 13 April 2008, SVG Graph Illustrating Amdahl's Law, https://commons.wikimedia.org/wiki/File:AmdahlsLaw.svg

[33] Lennard-Jones, J. E. (1924), *On the Determination of Molecular Fields*, *Proc. R. Soc. Lond. A* 106 (738): 463–477, Bibcode:1924RSPSA.106..463J, doi:10.1098/rspa.1924.0082

[34] Frenkel, D. & Smit, B. (2002), *Understanding Molecular Simulation* (Second ed.), San Diego: Academic Press, ISBN 0-12-267351-4.

[35] GROMACS 4.5.3, source code nb_kernel212_x86_64_sse.s, nb_kernel212a.

[36] Intel® Xeon Phi™ Coprocessor System Software Developers Guide SKU# 328207-003EN March, 2014, Intel, xeon-phi-coprocessor-system-software-developers-guide.pdf

[37] Bowman, D, 2015. Unpublished. "Optimizing Applications in HPC Environments Using Incremental and Noncomputational Methods"

[38] IEEE Computer Society (August 29, 2008). "IEEE Standard for Floating-Point Arithmetic". IEEE. doi:10.1109/IEEE STD.2008.4610935. ISBN 978-0-7381-5753-5. IEEE Std 754-2008

[39] MacKerell Juniorperiod, A.D.; Bashford, D; Belott, M.; Dunbrack, R.L.; Evanseck, J.D.; Field, M.J.; Fischer, S.; Gao, J.; Guo, H.; Ha, S.; Joseph-McCarthy, D.; Kuchnir, L.; Kuczera, K.; Lau, P.T.K.\; Mattos, C.; Michnick, S.; Ngo, T.; Nguyen, D.T.; Prodhom, B.; Reiher, W.E.; Roux, B.; Schlenkrich, M.; Smith, J. C.; Stote, R.; Straub, J.; Watanabe, M.; Wiorkiewicz-Kuczera, J.; Yin, D.;Karplus, M. J Phys Chem B 1998, 102, 3586

[40] Fresheneesz, 14 April, 2007, Example of a floating point number, http://en.wikipedia.org/wiki/Single-precision_floating-point_format#/media/File:Float_example.svg

[41] Codekaizen, 21 February 2008, The memory format of an IEEE 754 double floating point value., https://commons.wikimedia.org/wiki/File:IEEE_754_Double_Floating_Point_Format.svg

[42] Wolfram Research, Inc., Mathematica, Version 10.1, Champaign, IL (2015).

[43] Fog A, 2014, *Instruction Tables*, Technical University of Denmark. Copyright © 1996 - 2014

[44] Fog, A, 2014, Optimizing software in C++ An optimization guide for Windows, Linux and Mac platforms, University of Denmark. ©1996 – 2014.

[45] Fog, A. 2014. The microarchitecture of Intel, AMD and VIA CPUs. An optimization guide for assembly programmers and compiler makers, University of Denmark. ©1996 - 2014

Chapter 3 - Free Energy Study Validation of Incremental and Noncomputational Performance Algorithms Using GROMACS

A Free Energy Study Validation of Incremental and Noncomputational Performance Algorithms Using GROMACS

3.1 Abstract

Noncomputational incremental algorithms applied to GROMACS 4.5.3 have been shown to be 14-15 times faster in determining nonbonded interactions than the GROMACS assembly code on an Intel Core i7 “Sandy Bridge”. These algorithms exploit a variable precision numeric format that eliminates the need for the inverse square root and greatly reduces the number of computer instructions inside the nonbonded routines by reducing the ‘spatial granularity’ of the spatial distance supporting incremental computation. This algorithm was previously validated for mathematical and computational stability and performance. It was also tested using water boxes and proteins. The results from water box and protein simulations appeared to show that the developed algorithm was suitable for MD simulations but this was somewhat subjective based on energy drift, RMSD and other factors. The present study used a methodology to determine statistical equivalence to evaluate if the results produced by the developed algorithms were ‘good enough’ for MD simulations. This study used a statistically significant number of free energy studies on 5 amino acid side chain analogues and compared the results with those produced with existing GROMACS 4.5.3 build versions and experimental data. It demonstrated that the developed algorithm produced statistically equivalent results as compared to the existing GROMACS 4.5.3 builds. This study was modeled after studies used to validate force fields and water models that can be tied to experimental data. Using these small systems it is possible to obtain a statistically significant number of samples allowing the use of statistical equivalence methodologies.

3.2 Introduction

There has been significant research in validating commonly used force fields and water models in the last ten years. These studies have frequently used amino acid side chain analogues in water as the basis of their investigation and sought to compare the results with experimental data. Perhaps the two most significant studies were performed by Hess, et. al.[105] and Shirts, et. al.[106] These studies used extremely large amount of computational resources to achieve these objectives. In 2003 Shirts et al. used Folding@Home[107] (http://folding.stanford.edu) at Stanford University to perform their study. This network of home based volunteered computers at the time had approximately 90,000 computers around the world running the Folding@Home client software. The study investigated 3 force field parameter sets, 15 amino acid side chain analogues, 5 trials each of 1.2 ns using 61 lambda values. This study used an estimated 140 CPU years, estimated on the processing capability of Celeron processors. The work was performed in less than 2 months and the complete study used approximately 200 CPU years. In 2006 Hess and Vegt performed a systematic comparison of force fields and water models also using a vast amount of computational resources. In their study it was reported that they used 1 μs for each residue with one force field and water model. The results of these studies could then be tied to experimental data.

The objective of the current study was to use a similar methodology to validate the viability of high performance incremental noncomputational algorithms for use in performing molecular dynamics simulations using GROMACS 4.5.3.[108,109,110] The computational resources to obtain statistical equivalence required a large number of iterations for each GROMACS build option (C single precision, SSE single precision and SSE2 double precision) and variations of the developed algorithm, for each amino acid analogue, force field and water model. The statistical methodology of this project was modeled after the methodology used in clinical trials for new drugs.

There are 23 possible variations in the incremental noncomputational algorithm representing variable precision 'spatial granularity' for the distance squared where 1.0 represents 1nm^2. In addition to these variations the GROMACS C single precision, SSE single precision and SSE2 double precision builds also need to be performed to determine statistical equivalence with the developed algorithms. The goal was not to validate the correctness of the simulations but to validate the results of the developed algorithm versus the widely used GROMACS builds. There are a total of 26 builds/variations possible and some number of iterations required for each combination of amino acid analogue and water model in order to obtain a statistically significant sample size. This study performed from 40-60 iterations of each free energy workflow for each algorithm/build, water model and amino acid analogue selected to produce a sample size sufficient for the equivalence testing. If the study were to include all combinations of amino acid analogues and water models the amount of CPU time required would have been 1040 times that of the previously referenced studies. The scope of this study was also not to revalidate or evaluate all combinations of force field and water models using the developed algorithms but to determine if the algorithms were statistically equivalent for use in performing MD simulations. This would not have been computationally possible with available resources and it was also unnecessary given it had been already established by existing studies. Using free energy workflows also enabled the results to be tied to experimental results. For this reason only five amino acid side chain analogues, one water model and one force field model were selected. The number of builds/variations was also reduced. All three GROMACS builds needed to be run in order to establish a zone of equivalence and a zone of superiority. It was previously established that exceeding the CPU cache memory to store incremental result tables would result in a significant performance loss.[111] Existing CPUs have limited L2/L3 cache memory in the range of 2MB - 8MB and each variable precision form of the developed algorithm requires CPU cache memory. If only water models using Lennard-Jones and reaction field[112] were supported variable precision alternatives in the range of ≈4.21 to ≈7.22 significant digits would require too much memory on 2MB CPU cache processors. It was also observed that variable precision versions with less than ≈3.01 significant digits would likely cause reductions in the granularity of the distance squared that would be too large to be useful for molecular dynamics. It was theorized that 1pm was adequate for MD simulations based on the atomic and molecular distances involved and that larger granularity would result in spatial 'jumps' that would be too large. This reduced the number of variations of the developed algorithm to four. (≈3.01, ≈3.31, ≈3.61, and ≈3.91) Using this approach allowed a quantifiable means of comparison against the existing builds for the GROMACS software that was feasible with the computational resources available.

3.1.1 Why Use Free Energy Studies for Validation

The calculation of free energies[113,114, 115] in molecular dynamics simulations has been an important area of research for many years. Free energy is an important quantity to determine because it quantifies the way a molecular process will operate and the probability that the system will achieve a specific state. Calculating free energies from MD simulations help in the understanding of atomic level processes. Absolute free energy of a system using the following can only be calculated in a limited number of cases. This effectively can only be done for small simple systems governed by a simple Hamiltonian. For larger simulations such as protein simulations this is normally not possible. In order to obtain a free energy estimate for a given system several things must be determined. Free energy *F* for a system using the canonical ensemble (or NVT ensemble, which has a constant number of particles, volume and temperature) is determined by the following equation.

$$F = -(1/\beta)(\ln Q)$$

The value *β* is the inverse of the temperature divided by Boltzmann's constant k_B and *Q* is the partition function. A classical description of the system in Cartesian coordinates is used, assuming the system is at equilibrium. [116]

The free energy workflows for this study are modeled after those developed by David Mobley[117] and updated by Justin Lemkul[118] in their protocol to calculate the change in free energy for the decoupling of van der Walls interactions between an amino acid analogues and a water box. The study did not investigate electrostatics in order to parallel the method used by Lemkul and due to issues with version 4.5.3 of GROMACS. This model was chosen because it uses very simple systems with a small water box and one amino acid analogue molecule where experimentally determined free energy values exist. This model is one of those included in the studies of Shirts et al. and Hess et al. in the analysis of force fields and the free energies of hydration of amino acid side chain analogues. The data analysis was performed using the Bennett Acceptance Ratio method[119] for calculating free energy differences.[120,121,122] This study chose to analyze the free energy by turning off only the van der Waals (vdW) interactions between the amino acid analogue and water and to attempt to reproduce the results obtained by the standard GROMACS builds and tie to both experimental and theoretical work of others.

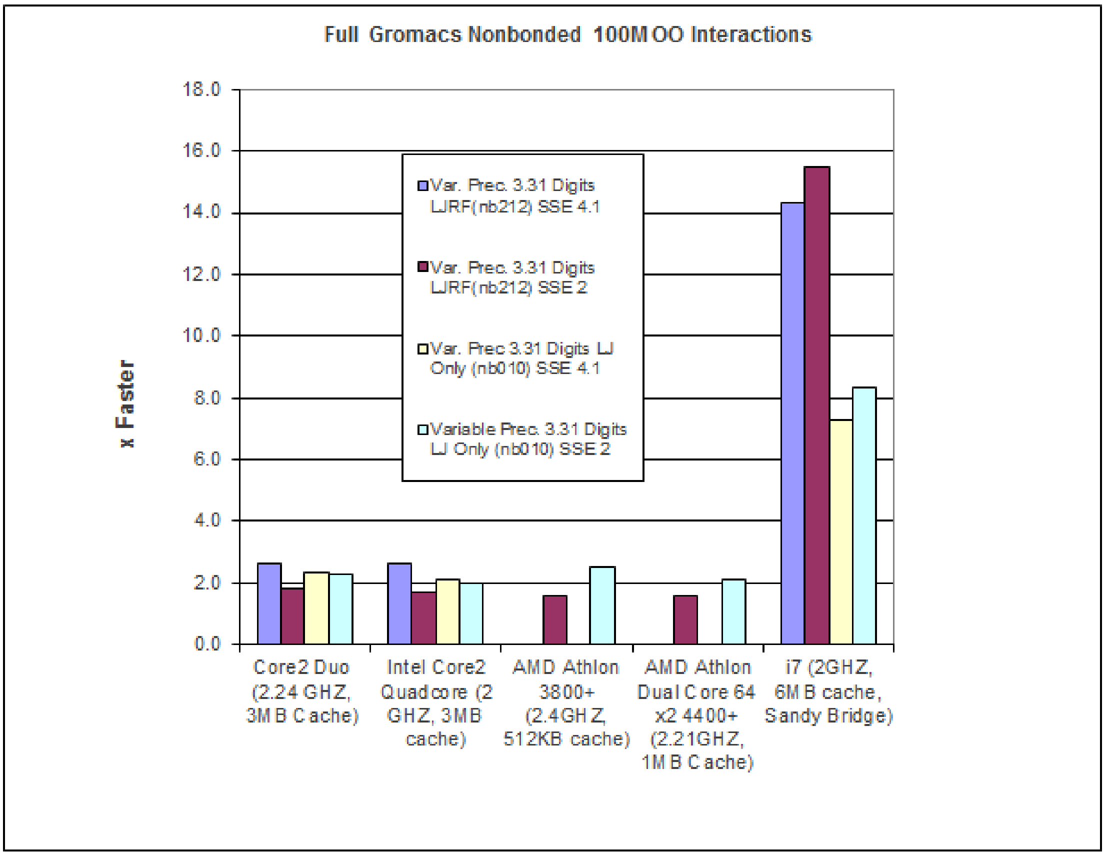


Figure 51 Comparison of Developed Algorithm with GROMACS Assembly Code

3.1.2 Why is this Validation Study Important

Performance enhancements to molecular dynamics software are extremely important because they expand the range of simulations that can be run and the chemical and biological problems that can be investigated. At the core of molecular dynamics software are inner computational loops that calculate forces, energies and distances for nonbonded interactions. These nonbonded interactions account for most of the computational costs of a simulation. The largest part of these interactions are the solvent to solvent interactions followed by solvent to solute and solute to solute interactions. Generally the largest single problem is the processing of interactions between water molecules. The SPC, SPC/E and TIP3P water models are most commonly used and they have nine interactions between the combinations of atoms pairs (OO, OH and HH). A noncomputational incremental algorithm was developed that increases the performance of the GROMACS 4.5.3 nonbonded kernel routines for nonbonded interactions for Lennard-Jones plus reaction field and Lennard-Jones only nonbonded kernels in GROMACS 4.5.3 by a factor of 14-15 times.[123]

This was achieved through a variable precision numeric format that effectively reduces the granularity of the computational and real space used in MD simulations to approximately 1 pm. This is accomplished by creating a variable version of the IEEE 754 floating point standard to support with precision ranges from ≈3.01 to ≈3.91 significant digits. Ordinary single precision calculations are performed with ≈7.22 significant digits. In the case of the GROMACS software, the inner nonbonded loops use distance squared units of 1.0 as 1 nm^2 and the seventh significant digit would represent the size of a subatomic particle. Computational and mathematical analysis demonstrated that the results paralleled those that would have been computed by using the IEEE 754 floating point instructions.

Simulations run in a real space that maps to a range of computational values. This real space and computational space can be represented by fewer bits for both the exponent and mantissa portions of the IEEE 754 representation used for general computation because of the nature of MD simulations.

Water Box Simulation and Bond Lengths (nm)

Interaction	Min r^2	Min r [a]	Bond Length [b]
H-H	0.017770	0.133304	0.074
O-H	0.019504	0.139657	0.096
O-O	0.057268	0.239307	0.121

[a] *r* is distance in nm   [b] : http://cccbdb.nist.gov

Table 19 Water Box Simulation Profile

The bond lengths for simulation purposes are usually considered fixed based on experimental data. Using this data and the bond lengths for water a mathematical model was created to determine the angular error for a water molecule rotating in 1 pm granular space. An analysis of a water molecule rotated through a discrete computation space of 1 pm shows an angular error of ± 0.47 degrees with no divergence.

The prior study of the noncomputational incremental algorithm theorized that MD simulations could calculate nonbonded interactions in granular space of ± 1 pm based on the distance squared in the nonbonded routines and give ‘equivalent’ results to the standard GROMACS builds.

In prior research the noncomputational incremental algorithm (See Chapter 2) was also shown to parallel the IEEE 754 floating point standard. There was no mathematical divergence because of the forward backward error correction that was generated in the tables containing the pre-calculated results. These tables are used to incrementally calculate energies and forces.

Why Test the Algorithm with Free Energy Studies

After the testing of the algorithm computationally and mathematically the question still remained was it ‘good enough’ for molecular dynamics simulations. The algorithm was shown to work mathematically and computationally but could it be successfully used in MD simulations. Is a spatial granularity of 1pm sufficient without causing adverse side effects? The algorithm has a forward backward error correction but does it function within the limits required for MD simulations. A number of investigators including one of the GROMACS developers said that the results of protein and water box simulations ‘looked good enough’ but this is only a small sample of subjective opinions.

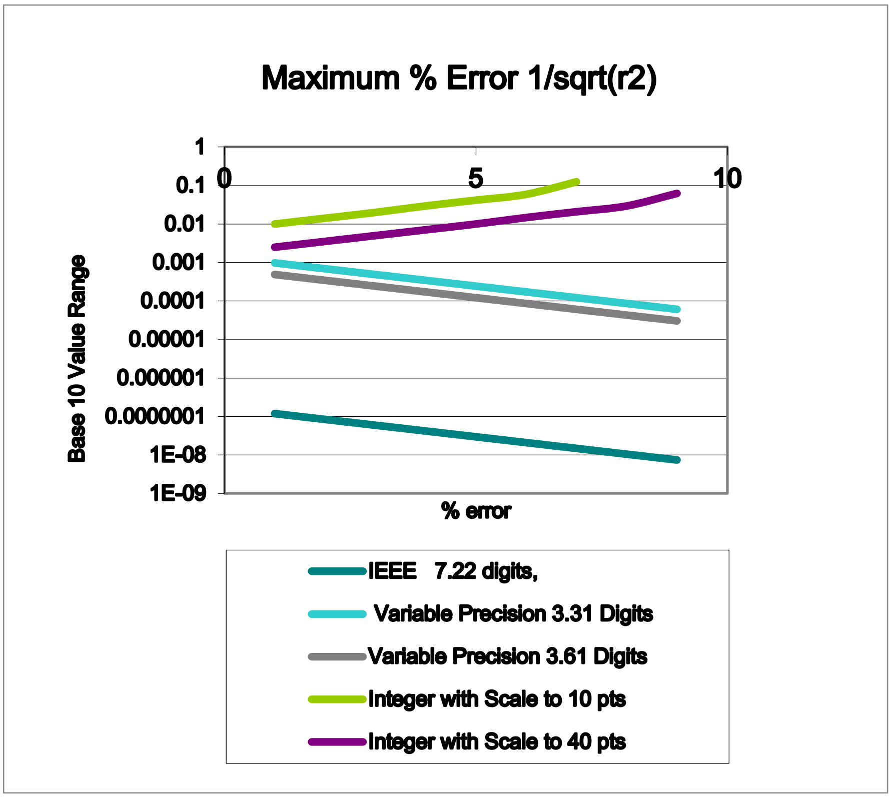


Figure 52 Relative Error vs. IEEE 754 Standard in Percentage

A large number of water box, amino acid, and protein simulations had already been run using the 4 different versions of the developed algorithm and the three build options for GROMACS 4.5.3 C language single precision, SSE single precision, and SSE2 double precision. The simulations were run multiple times but due to the inherent nature of molecular dynamics simulations three is significant variation from one simulation run to another even with the GROMACS builds. In complex systems such as proteins in water multiple runs of a simulation may show different unfolding paths and the deviation in the results of the simulation may be extremely different. For these reasons this study of free energies was performed.

3.3 Methodology

3.3.1 Test Environments

Software

The 64 bit version for Windows of R version 3.1.2 (R Foundation for Statistical Computing)[124] was used for normality testing and statistical analysis. Descriptive statistics were generated using StatView for Windows Version 5.0[125] and Microsoft Excel 2007 was used for general analysis.

All test systems used the Ubuntu 14.04 operating system, the gcc 4.7 compiler and GROMACS 4.5.3.

Hardware

All free energy simulations for this study were performed using GROMACS 4.5.3 on Ubuntu 12.04 on an AMD Opteron 6272 2.1 GHZ 2MB CPU cache (Bulldozer) 64 core server, and on an Intel Xeon 5650 2.67 GHZ 12MB Cache 6 cores (Westmere-EP 32nm). The gcc 4.7 compiler was used for development.

Earlier algorithmic, performance, small molecule and protein simulations were performed on the above platforms and other platforms including the following systems: Intel Core i7 (2630QM) 2.0 GHZ 6 MB Cache (Sandy Bridge), Intel Core i7 (930) 2.8 GHZ, 8 MB cache (Nehalem), Intel Core 2 Duo 2.24 GHZ 3 MB cache, Intel Core 2 Quad Core 2.2 GHZ 3 MB cache, AMD Opteron 6272 2.1 GHZ 2MB CPU cache (Bulldozer) 64 core server, AMD Athlon 3800+ 2.4 GHZ 512KB cache, AMD Athlon

X2 4400+ 2.3 GHZ 512KB cache and Intel Core i5 1.7 GHZ 3MB cache (4201U (Haswell) ULT).

Summary of Prior Testing with Water Boxes

Simulations of a water box containing 987 atoms, using the SPC water model, GROMOS 43a1 force field and using the NVT ensemble were run. The water box simulation used the GROMACS 4.5.3 nonbonded kernels routines for Lennard-Jones and reaction field for electrostatics. These simulations were executed using 22 different variations of the algorithm and the three standard GROMACS single precision C, single precision SSE and double precision SSE2 builds. The 22 different versions of the algorithm reduce the size of the mantissa for the IEEE representation of the real number value. By reducing the size of the mantissa the approximate number of decimal digits supported can be reduced from ≈7.22 to ≈2.11.

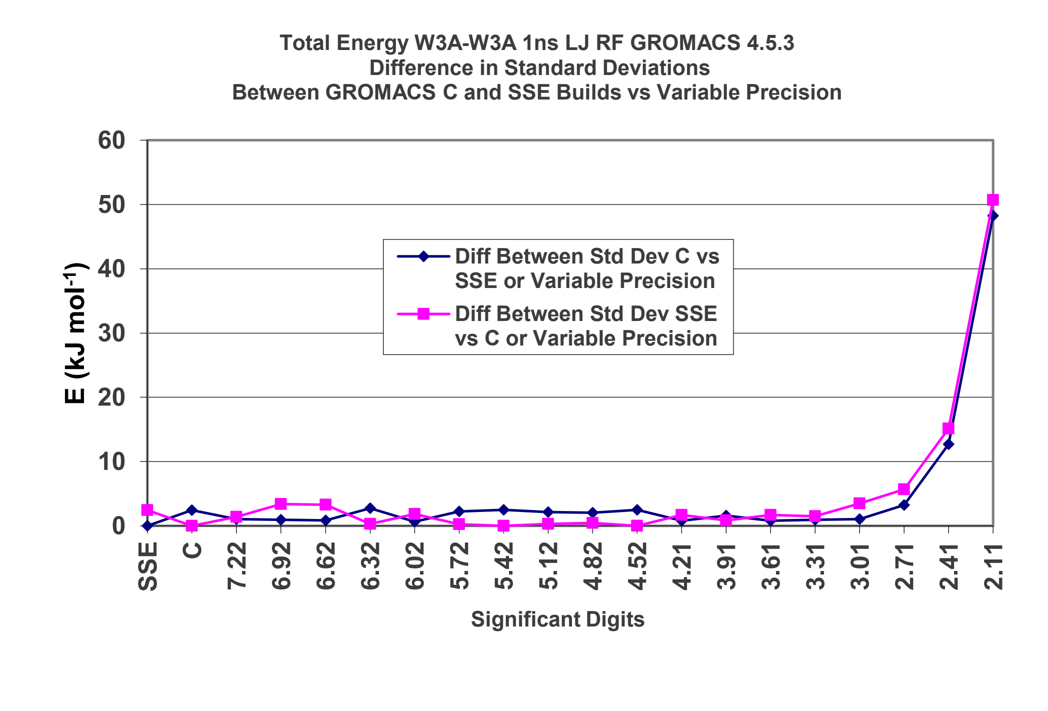


Figure 53 Changes in the Standard Deviation with Reduced Precision

This reduced computational and spatial ‘granularity’ was used to minimize the memory required for pre-computed tables. Figure 50 shows large standard deviations in total energy and other simulation results occurred as the ‘granularity’ approached ≈2.11 significant digits and simulations crashed at coarser granularity. Similar results were seen in small molecule and protein simulations. This is a result of the ‘spatial granularity’ being reduced too much and the atoms ‘jumping’ too far. The results shown in Figure 50 show the difference between the standard deviation in total energy of the build/algorithm and the GROMACS C or SSE versions used as a reference point. It can be seen that there is a discrepancy between the standard deviations of the energy calculated with the two GROMACS builds. Until ≈3.01 significant digits this discrepancy is similar to that between the standard deviation of the energy calculated with the build/algorithm and the standard deviation of either the C or SSE builds. This was extremely important because it shows that a granularity of less than ≈3.01 significant digits is going to start to cause large fluctuations in total energy. The GROMACS g_energy utility was used to obtain information such as total energy, kinetic energy, potential energy, etc. This permitted a reduction in the number of variable precision variations of the developed algorithm that needed to be tested and reduced dramatically the computational resources that were required.

3.3.2 Free Energy Study Methodology Using Amino Acid Side Chain Analogues

Free energy studies of amino acid side chain analogues have been used to demonstrate the viability of force field and water models and to show that these results conform to experimental data. These small short simulations consist of a small water box and one amino acid side chain analogue. The small size of these simulations and their validation with experimental data make them ideal for use in testing new performance algorithms and looking for undesirable side effects. The methodology for this study uses two major components: 1) use of free energy for amino acid analogue studies and 2) determining what is 'good enough' through statistical equivalence. The free energy data that is collected is used to support statistical equivalence test using the results from the iterations for each combination of force field, water model, amino acid analogue and developed algorithm variant and standard GROMACS build. The goal is *not* to

Amino acids and side chain analogs used in this study			
Amino acid	**Abbreviation**	**Analogue**	**Natural occurrence**
alanine	Ala	methane	7.80%
asparagine	Asn	acetamide	4.30%
leucine	Leu	isobutane	9.10%
serine	Ser	methanol	6.80%
threonine	Thr	ethanol	5.90%

Table 20 Amino acid side chain analogues used in the study

revalidate the GROMACS versions against all combinations of force fields, water models and amino acid side chain. Extensive studies have already been performed to validate water models and force fields using vast amounts of computer time.[126], [127] It is also not the goal of this study to attempt equivalence testing on all amino acid side chain analogues or more than one force field or water model. Five amino acid side analogues each with one force field and water model were chosen for evaluation. Table 20 shows the natural occurrences in proteins. Methane was used because it was used in the protocol of Justin Lemkul. Additionally the analogues isobutane, methane and methanol represent the analogues of the three highest natural occurrences. Ethanol and acetamide were chosen because they are analogues for polar amino acids (asparagine and threonine respectively) and also have a high natural occurrence.

This statistical methodology coupled with the methodology of previous studies used to validate water models and force fields was the only way to prove mathematically that the developed algorithms were ‘good enough’ for the intended purpose. This study examined five amino acid side chain analogues using the OPLSAA[128] force field and the TIP3P water model.

The workflow for the free energy studies was based on the free energy protocol of methane in water by Justin Lemkul.[119] The study included a water box of 241 TIP3P water molecules and a single molecule of methane. The OPLSAA force field was used. The .mdp, .gro and .top files are from the protocol except with the changes noted for methane in water. The .gro and .pdb files for the other amino acid analogues were readily available from other studies. Charge interactions between solute and water were turned off prior to the van der Waals (vdW) terms to avoid charges approaching too closely after the vdW repulsive terms are turned off, which would result in an unstable system. The methodology adapted from the Lemkul protocol assumes that charges have previously been turned off and that only the van der Waal terms remain, and will gradually be turned off between the solvent and solute.

The only other changes made were to replace PME by reaction field since no implementation of the developed method exists for PME, and to change the temperature from 300 K to 298 K to better reflect experimental data and the work of Shirts et al.[107] The time for the production simulation run associated with each lambda was increased to 5 ns to look for potentially hidden computational side effects that might not be visible in shorter simulations.

The free energy change of changing a system from state A to state B, ΔG_{AB}, is a function of the coupling parameter (lambda λ). Figure 55 shows that this parameter designates the level of change that occurs between states A and B. This is the degree to which the Hamiltonian has been altered and the system transformed. Simulations using different values of lambda permit the plotting of a $\partial H/\partial \lambda$ curve, from which ΔG_{AB} can be determined. A key issue in free energy calculations is determining how many lambda values (points) will be used to describe the change from state A ($\lambda = 0$) to state B ($\lambda = 1$). The goal is to collect an adequate sample of data to produce a viable $\partial H/\partial \lambda$ curve.

This study used a linear series of λ values with an equidistant λ spacing of 0.05 and ranging from 0 to 1 for decoupling the vdW interactions. Linear λ spacing values of 0.05-0.1 are commonly used but in many cases molecules will need many more lambda points, such as systems that have strong interactions through hydrogen bonding.

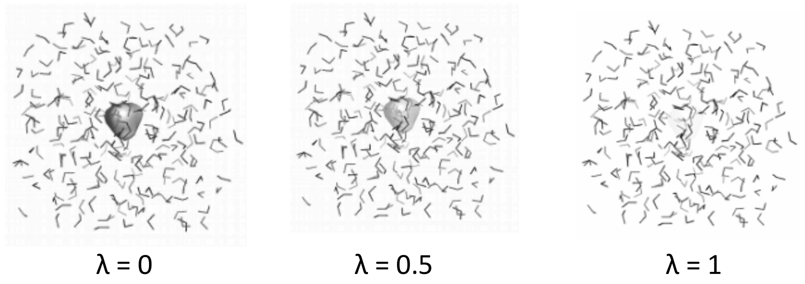


Figure 54 Free Energy - change from state A (λ = 0) to state B (λ = 1)

Adapted from Justin Lemkul [107]

Lambda spacing may need to be decreased so that more points are and distributed asymmetrically due to variations in the slope. Shirts et. al. used 61 λ values in their work but shorter simulation times of 1.2ns whereas this study used 5ns simulations because part of the object was to look for the longer term impact of the developed algorithm to the free energy results.

Due to the non-linear dependence of the energy on the λ values decoupling of the van der Waals interactions can sometimes be problematic. For reasons described by Shirts et al. and elsewhere, many more λ values may be necessary to adequately describe the transformation, particularly in regions where the slope of the ∂H/∂λ curve is steep. In the present study, the linear λ spacing of 0.05 was found to be sufficient.

For each value of λ, it is essential that a free energy workflow be performed (energy minimization, equilibration, and production data collection). These jobs were run as batches performing large numbers of iterations of the same workflow for a given combination of algorithm, granularity, and amino acid analogue. The following workflow was performed for each iteration. Each step in the workflow was run using the GROMACS 4.5.3 SSE version except the production step where the algorithm the various GROMACS builds and the variable precision algorithms were used. This minimized the differences in the test environment by keeping all steps constant except the production data step. With statistical equivalence or superiority testing the test environment must keep as many variables constant as possible to minimize side effects that would skew the results. The workflow used is as follows:

1. Steepest descents minimization
2. L-BFGS minimization
3. NVT equilibration
4. NPT equilibration
5. Production data based on the NPT ensemble

Both the steepest descents and L-BFGS minimization steps were used to provide a better minimization of the starting structure. Lemkul reported that the L-BFGS minimization converges prematurely and may result in unstable systems, however when it is used in conjunction with steepest descents it yields a better minimization. [107] The double precision SSE2 build of GROMACS 4.5.3 was used for the minimization steps of the workflow and the single precision SSE build of GROMACS 4.5.3 was used for NVT and NPT equilibration steps. The production NPT step was used for comparing the algorithms. It used the variations of the developed algorithm and the GROMACS C single precision, SSE single precision and SSE2 double precision builds.

3.3.3 Free Energy Study - Preliminary Results

Figure 56 shows the results from a methane in water free energy study using the 3 standard GROMACS 4.5.3 build options and 4 variable precision algorithms. It shows a comparison of the results for ≈3.01 and ≈3.91 digit versions of the developed algorithm compared to the GROMACS 4.5.3 double precision, C single precision and SSE single precision builds. It can also be seen in this simple system that the reaction field and GROMACS PME models give virtually identical results. It can be seen that

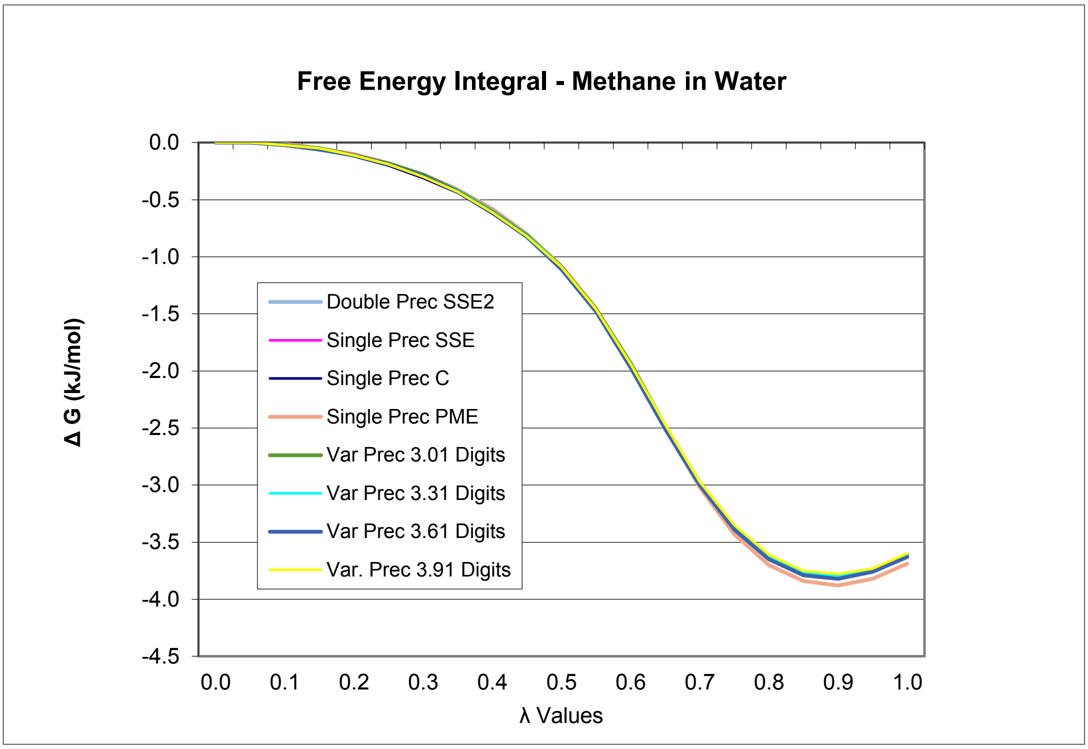


Figure 55 Free Energy Integral - Methane in Water

both the free energy integral and free energy differences for methane in water are virtually identical to the GROMACS builds. In figure 57 the free energy differences

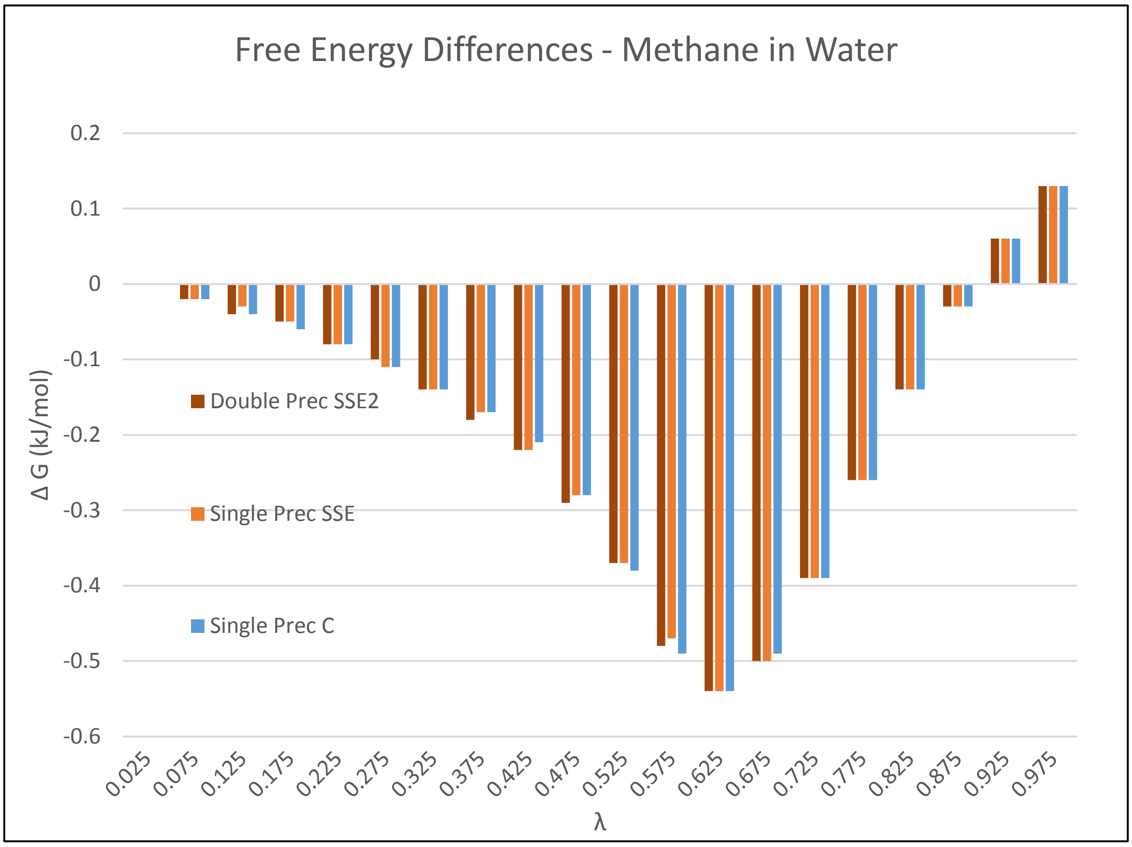


Figure 56 Variation between GROMACS Builds

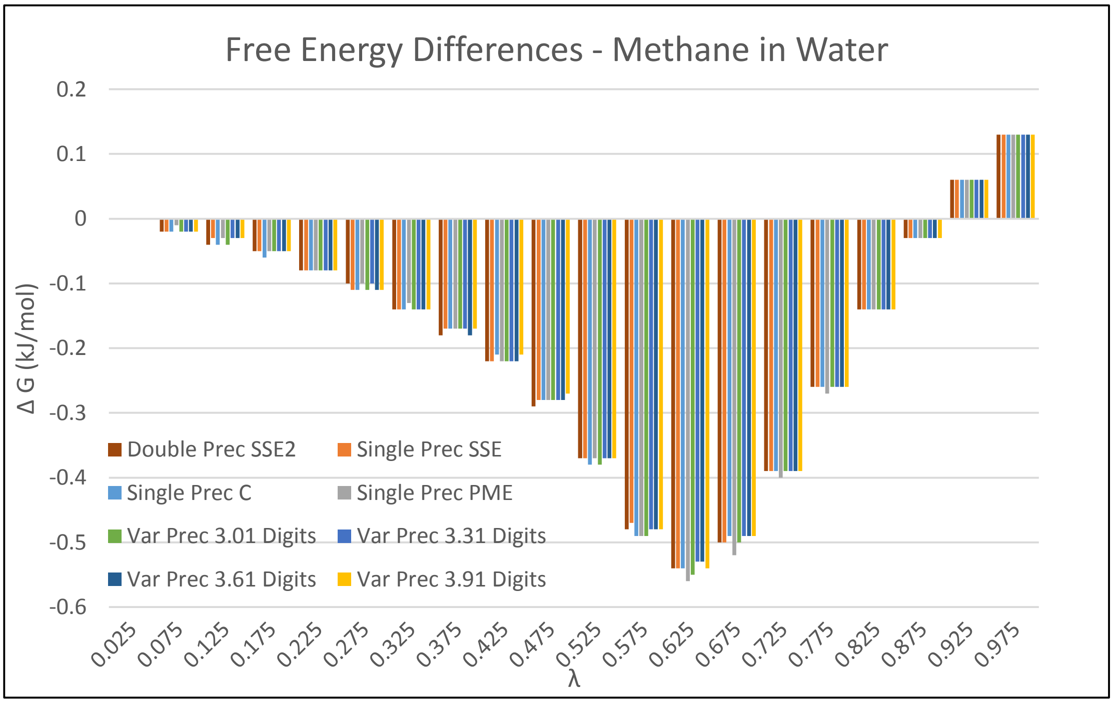


Figure 57 Variation between GROMACS Builds and Variable Precision

show a variation between the GROMACS double precision SSE2, single precision SSE, and single precision C builds that illustrates that the variation between the GROMACS builds can be used to establish a zone of equivalence. In Figure 58 the free energy differences are shown for all standard GROMACS builds, PME and the developed algorithm showing that they are virtually indistinguishable.

Statistical Methodology

It is useful to see graphically that the results of the free energy study for methane in water look 'good enough' when the noncomputational incremental algorithms are used with spatial and computational granularity of three significant digits. In order to determine if the results are "good enough" an analysis of statistical equivalence was needed[129]. This methodology is well defined and frequently used in clinical trials of new drugs.[130,131]

Standards for the design of statistical equivalence and non-inferiority studies have been developed such as The Consolidated Standards of Reporting Trials Statement (CONSORT)[132] and the 2010 extension and 2012 update to CONSORT for non-inferiority and equivalence trials. In this study if a molecular dynamics simulation of an amino acid analogue is a 'disease' then each build variation of GROMACS 4.5.3 and the developed algorithm may be considered and alternative 'treatment'. In 2006, a survey of the methodology used in 162 clinical trial studies from 2003 and 2004 was made by Le Henanff et. al.[98] of the use of non-inferiority and equivalence methods in clinical trials. They cited two common issues with these studies that relate to this current work: 1) definition and justification of the margin of equivalence or non-inferiority and 2) failure to define how the sample size was determined.[133]

3.4 Statistical Equivalence

The purpose of this study is to evaluate the statistical equivalence[134], superiority or inferiority of the noncomputational incremental versions of the developed algorithm using a variable precision floating point format versus the standard single C and SSE, and double precision builds of GROMACS 4.5.3 and thus determine if a variable precision format was 'good enough' to be used in molecular dynamics simulations using GROMACS.

Since no two algorithms or methods yield results that are exactly equal, we must define what effects are important and what difference in these effects would be important. The effect differs depending on what is being studied. In this study the delta in free energy (ΔG) is considered the principal effect of the algorithm.

Standard statistical tests can be used to show that there is no ‘statistically significant difference’ but this is not the same as showing that two sets of data representing two methods are equivalent or ‘good enough’ for the intended purpose. Statistically significant means that there is strong evidence that the difference is not zero, but it does not show if the difference is large enough to eliminate the conclusion that the two sets of data representing different methods are ‘functionally’ equivalent to data representing different methods that are ‘generally accepted’ as ‘good enough’. Statistical methods have been developed for testing equivalence using either confidence intervals or p-values.[136] These methods are frequently used in clinical trials and other applications. This study uses confidence intervals as a means of determining if the developed algorithms are statistically equivalent to the standard GROMACS builds.

3.4.1 Defining a Zone of Equivalence and a Zone of Superiority

In order to perform a study of statistical equivalence it is necessary to define an acceptable range beyond which the results would not be considered acceptable. Zones of equivalence, superiority or inferiority need to be defined and this definition is based on the problem being analyzed. What is ‘good enough’ is problem and methodology specific. For example in a clinical trial with a control group with systolic blood pressure of 163, if the alternative treatment group had a systolic blood pressure of 160, clinicians would not change their practice for just 3 points[135]. There is however the question of how many points would be considered to be sufficient to change general clinical treatment. This becomes subjective judgement call based on the test or trial being performed, if not 3 points perhaps 5 or 10 points in this example. This decision must be made by experts in the field using data from an acceptable sample size. A question of personal interest in the results arises. Obviously the developers of the drug, algorithm or other method being developed have a vested interest in the success of the results, other groups may have different professional opinions and perhaps government regulators have an extremely conservative perspective. The same applies to the suitability of an algorithm for use in performing molecular dynamics simulations. It can be said that a new drug/algorithm can be considered equivalent to the standard if it falls within the range of X, where X is the mean of the standard therapy/algorithm and I_2 is the upper equivalence interval and I_1 is the lower equivalence interval. I_1 and I_2 may not be equidistant from the center of the zone of equivalence and may include an additional subjective non-statistically based value to increase the size of the zone of equivalence, superiority or inferiority. This may be expressed as:

$$[X_{\text{Standard}} - I_1] \text{ to } [X_{\text{Standard}} + I_2]$$

From one perspective the smaller the interval the more equivalent the drug/algorithm will have to be compared to existing drugs/algorithms in order to be accepted. The greater the interval is the more easily it will be for the drug/algorithm to acceptable for approval or for the algorithm to be accepted for general use. There is however a side effect that when the interval I is small a larger sample size is required and it is more difficult to determine equivalence.[136] In this study each of the standard GROMACS 4.5.3 builds were considered as if they were a 'treatment' or algorithmic alternative and therefore collectively the zone of equivalence was defined based on the formula below. The mean of the aggregated datasets from the three builds of GROMACS samples was not used as the basis as the definition of a zone of equivalence because there are subtle differences in how the 1/sqrt function is implemented and what computer instructions are used for performing the 1/sqrt and force calculations. This means that the implementations for how forces are calculated yields slightly different spatial positions, force results and accumulated statistics. Equivalence testing using the zone of equivalence testing model in this study could also be useful to validate new GPU, multicore/multiprocessor implementations, new implicit water models or even major version changes to GROMACS itself as part of the test suite. It could also be useful as a generalized testing methodology for complex software.

Data for this study was shown to have a normal distribution so that $I_1 = I_2$. This study defined the zone of equivalence using the following formula without the addition/subtraction of subjective modifications to the zone of equivalence based on the opinions of 'experts' using the confidence interval of the replicate data. It could easily be argued that due to limitations in force field models, water models and how the calculations are performed that the size of these zones could be increased, but that could lead to debates about how much should be allowed and still be ‘good enough’.

$$\min([\ [X_{\text{Gromacs c}} - I_{\text{Gromacs c}}], [X_{\text{Gromacs sse}} - I_{\text{Gromacs sse}}], [X_{\text{Gromacs sse2}} - I_{\text{Gromacs sse2}}])$$

to

$$\max([\ [X_{\text{Gromacs c}} + I_{\text{Gromacs c}}], [X_{\text{Gromacs sse}} + I_{\text{Gromacs sse}}], [X_{\text{Gromacs sse2}} + I_{\text{Gromacs sse2}}]\ )$$

and a new algorithm *t* is equivalent if the following criteria are met:

$$[X_{\text{t}} - I_{\text{t}}] \ \geq\ \min([\ [X_{\text{Gromacs c}} - I_{\text{Gromacs c}}], [X_{\text{Gromacs sse}} - I_{\text{Gromacs sse}}], [X_{\text{Gromacs sse2}} - I_{\text{Gromacs sse2}}])$$

and

$$[X_{\text{t}} + I_{\text{t}}] \ \leq \max([\ [X_{\text{Gromacs c}} + I_{\text{Gromacs c}}], [X_{\text{Gromacs sse}} + I_{\text{Gromacs sse}}], [X_{\text{Gromacs sse2}} + I_{\text{Gromacs sse2}}]\ )$$

A zone of superiority was also defined using the GROMACS SSE2 double precision build as the definition of the zone of superiority because it is computationally more robust supporting over 15 significant digits. The following defines the zone of superiority:

$$\min([\ [X_{\text{Gromacs sse2}} - I_{\text{lowerbound sse2}}])$$

to

$$\max\ [X_{\text{Gromacs sse2}} + I_{\text{lowerbound sse2}}]\ )$$

An algorithm *t* would be considered within the zone of superiority if it meets the following criteria.

$$[X_{\text{t}} - I_{\text{t}}] \ \geq\ [X_{\text{Gromacs sse2}} - I_{\text{Gromacs sse2}}]$$

and

$$[X_{\text{t}} + I_{\text{t}}] \ \leq\ [X_{\text{Gromacs sse2}} + I_{\text{Gromacs sse2}}]$$

The zone of equivalence for this study is specified conservatively in that the developed algorithm/variant must fit *completely* between the minimum value and the maximum values supported by the three ‘accepted’ GROMACS build algorithms without any discussion as to whether or not it is possible to extend the lower or upper end of the zone of equivalence. In the above *X* is the mean of the Δ G values for the replicate series of each GROMACS build. *I* is the confidence interval for the Δ G values of each replicate series determined by the GROMACS 4.5.3 g_bar utility using the Bennett Acceptance Ratio method for calculating free energy differences. Figure 59 illustrates the methodology of the study. The zone of equivalences is defined by the three GROMACS builds as indicated by the two black lines for the C and SSE single precision builds and the purple line for the GROMACS SSE2 double precision build.

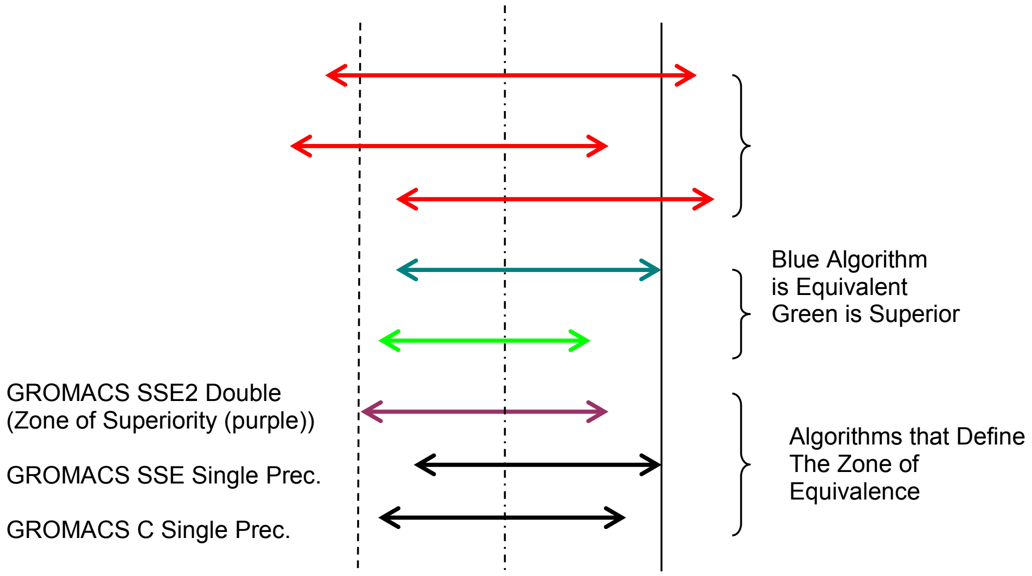


Figure 58 Illustration of Zone of Equivalence and Zone of Superiority

The purple line indicates the zone of superiority within the zone of equivalence. The algorithm represented in green is within the zone of equivalence but it is also within the zone of superiority. The algorithm indicated in blue is within the zone of equivalence. In this study all of the variants based on the data collected were within the zone of equivalence. The algorithms in red indicate possible algorithms that would fall outside of the zone of equivalence. Variants of the developed algorithm with reduced precision less than ≈3.01 significant digits would begin to fall outside of the zone of equivalence. Evaluation of variable precision versions of the developed algorithm with less than ≈3.01 significant digits were not included because it was already evident from the results of water box simulations from a prior study that at precisions less than ≈3.01 significant digits caused increases in total energy.

3.4.2 Sample Data Generation

For each amino acid analogue studied the following procedure was performed to generate the data required for statistical analysis. For example, this study uses multiple runs of a free energy simulation with 21 lambdas of methane in water, following a protocol developed by Justin Lemkul (with the exception that PME was changed to Reaction Field). Each simulation representing a single lambda of a free energy series is 5 ns and is processed through a full complement of minimization, equilibration, NVT, and NVP runs before the production MD run is performed. The mdp, gro and .top files are from the protocol by Justin Lemkul. Each data point represents the change in free energy (Δ G) associated with the 21 simulations that are part of a single free energy study. The workflow implemented use the GROMACS 4.5.3 double precision SSE2 build for the two minimization steps and the GROMACS 4.5.3 SSE single precision for the two equilibration steps. The production workflow was performed for each of the tested builds and algorithms. This is repeated for each algorithm GROMACS 4.5.3 SSE single precision, C single precision, and SSE2 double precision, and the developed algorithm using ≈3.91, ≈3.61, ≈3.31 and ≈3.01 significant digits.

3.4.3 Sample Size Estimation and Confidence Intervals

In order to demonstrate statistical significance only the three standard GROMACS builds and only four variable precision forms of the developed algorithm were used. The study began by running 20 iterations of each combination of build/algorithm and amino acid analogue (20 iterations x 21 lambdas x 5 ns). The standard deviation was used to provide an initial estimate of what the sample size needed to be for each analogue. Additional iterations for each amino acid analogue was executed until the desired level of statistical significance a 95% confidence interval was achieved. There was substantial variability between the builds and variants with the GROMACS C single precision build frequently indicating the largest sample requirement. The average sample size for each run was used except where the average sample size was less than forty and then forty was used (methane and methanol). Table 21 shows the results of the sample size estimates based on the data series for each build or algorithmic variant, and the sample size actually run for each amino acid analogue used in the study.

Amino Acid Analogue	GROMACS C Single Precision	GROMACS Single Precision SSE	GROMACS Double Precision SSE2	Variable Precision 3.91 Sign Digits	Variable Precision 3.61 Sign Digits	Variable Precision 3.31 Sign Digits	Variable Precision 3.01 Sign Digits	Avg Sample Size	# Used in Study
Acetamide	56	59	44	55	28	38	42	46	46
Ethanol	75	55	53	61	60	50	61	59	60
Isobutane	61	37	34	39	43	63	43	46	46
Methane	29	37	28	23	26	20	34	28	40
Methanol	20	23	33	28	21	24	31	26	40

Table 21 Sample Size Estimates - Amino Acid Analogues Based on Data Series

Each replicate was a complete run of the production workflow for all lambda values for each algorithm and amino acid analogue yielding a set of ΔG values. The sample size was determined using R-2.15.1 for Windows (The R Foundation for Statistical Computing) using the following formula:

$$n = ((SD * z(0.95)) \; / \, E \, (\pm 2\%))^{2}$$

In the above formula *n* is the sample size, *SD* is the standard deviation and *z(0.95)* is the zscore for 95% confidence interval. A margin of error *E* of 2% was chosen because it is less than or equal to the estimated error of the Δ G values for the tests and it is the maximum supported by the sample size. The SD is taken from the Bennet's Method output in GROMACS for a full free energy calculation of all lambdas for the entire replicate series. The sample size for each analogue studied was increased for each replicate until the average sample size for the analogue achieved a 95% confidence interval.

Confidence intervals were computed assuming a normal distribution using Microsoft Excel 2003. The population size was consider to be the number of data points in the data sets (40-60). A minimum of 40 data points were used for each combination of amino acid analogue and algorithm/build.

3.4.4 Resource and Data Requirements for the Study

The computational costs of this study were high due to the large numbers of simulations required for each algorithm multiplied by the number of lambda values times the number of amino acid analogues. Table 22 shows the number of iterations required for each amino acid analogue and the number of microseconds of simulation required for each amino acid analogue for this study. In order to obtain a single Δ G value for the statistical study for each analogue it was necessary to run a complete workflow totaling 105 ns for each study (5ns times 21 lambda points).

Amino Acid Analogue	Estimated Sample Size	# Builds or Algorithms	Samples in the Study	Simulation Time µs
Acetamide	46	7	46	33.8
Ethanol	59	7	60	44.1
Isobutane	46	7	46	33.8
Methane	28	7	40	29.4
Methanol	26	7	40	29.4
Total			232	170.5
Note: 5ns x 21 λ values = 105ns / analogue/ΔG				

Table 22 Total Microseconds of Simulation Time Needed for Study

3.5 Normality Testing

R-2.15.1 for Windows was used for all normality testing and for producing QQ plots and histograms of data samples. The data values associated with each build/algorithm were tested for normality using the following statistical tests from the *nortest* package for R-2.15.1. Each normality test has its own strength and weaknesses especially on relatively small sample sizes. The Anderson-Darling test generally has better results using small sample sizes. It was observed that even using the R-2.15.1 functions to create a ‘small’ random sample of ‘normal’ data there was significant variation between the results generated by these tests. It is extremely important to compare the results of these normality tests with QQ plots and histograms. Even when the R-2.15.1 function to create a random set of normal distribution data the histograms and QQ plots may not appear ‘normal’.

The following normality tests used were:

1. Anderson-Darling
2. Shapiro-Francia
3. Shapiro-Wilk

4. Pearson chi-square
5. Lilliefors (Kolmogorov-Smirnov)
6. Cramer-von Mises.

3.5.1 Interpreting the Normality Tests

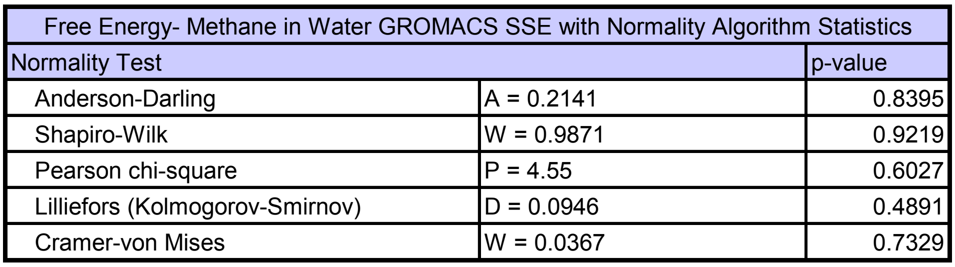
The Anderson-Darling test is a modified form of the Kolmogorov-Smirnov (K-S) test and but gives more weight to the tails than the K-S test. A p-value ≥ 0.05 indicates normality. The Anderson-Darling test may be used with small sample sizes (≤ 25).[137]

The Shapiro-Wilk[138] test has the best power for a given significance followed by the Anderson-Darling test. Data is considered to be normal if p-value is ≥ 0.05.[139]

The Pearson chi-square test was also performed. It however is unreliable if the expected frequencies are too low. It is normally acceptable if no more than 20% of the cases have expected frequencies below 5. It is of limited value based on the nature of the data set used in this study but also frequently indicated normality.[140]

The Lilliefors (Kolmogorov-Smirnov) was also used to test normality. It is used when the mean and standard deviation of the theorized normal distribution are not known. In this case they are estimated from the sample data. This test also indicated a normal data distribution.[141,142]

The Cramer-von Mises test for normality was also used. This test was developed in 1928 and uses one or two samples. It is an alternative test to Kolmogorov-Smirnov. A p-value ≥ 0.05 is considered to show normality.[143,144] The Shapiro-Francia test supports a number of data points between 5 and 5000 and was also used. Table 23 shows the results for normality testing for methane in water using 40 data points.



Free Energy- Methane in Water GROMACS SSE with Normality Algorithm Statistics		
Normality Test		p-value
Anderson-Darling	A = 0.2141	0.8395
Shapiro-Wilk	W = 0.9871	0.9219
Pearson chi-square	P = 4.55	0.6027
Lilliefors (Kolmogorov-Smirnov)	D = 0.0946	0.4891
Cramer-von Mises	W = 0.0367	0.7329

Table 23 Normality Test Results Methane in Water GROMACS SSE

Whenever normality tests are performed it is important to examine the quantile to quantile plots (Q-Q plots) and histograms. Figures 61 shows the Q-Q plot and histogram respectively for the data samples for methane in water for 40 data points using GROMACS 4.5.3 SSE single precision for the production simulations. This histogram and quantile to quantile plot give a clearer and more visual assessment of normality.

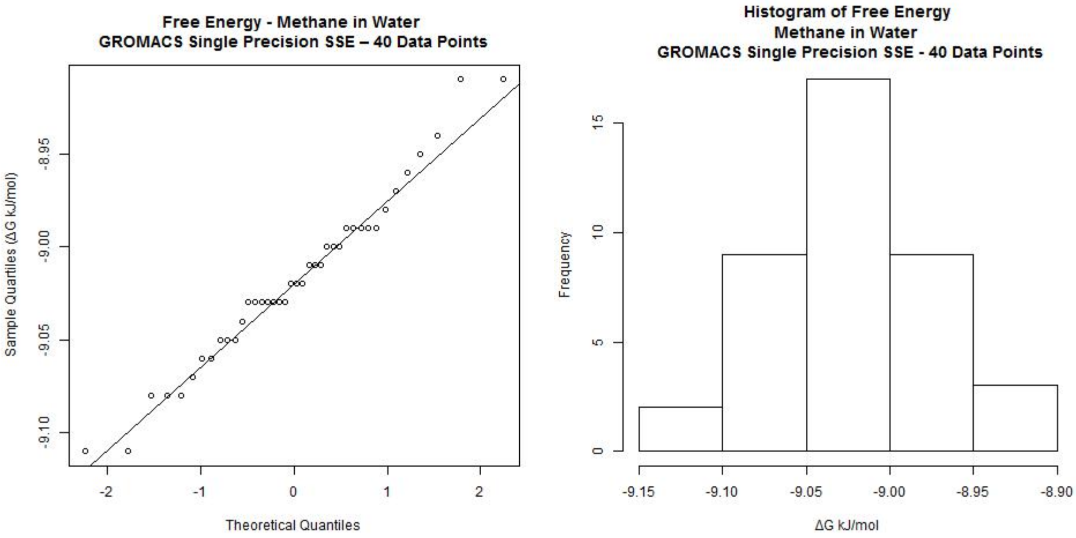


Figure 59 Q-Q Plot and Histogram - Methane in Water GROMACS SSE

3.5.2 Normality Test Results

The following results are for methane in water. Tables 24 and 25 show the p-value results for each normality test used for each of the GROMACS 4.5.3 builds and the each variant of the developed algorithm supporting a variable number of significant digits. The test results for all algorithms and variants showed normality except for the GROMACS 4.5.3 C single precision build. A data set fails the above normality test if the p-value did not meet the ≥ 0.05 criteria. With small sample sizes normality tests may fail even for tests such as Anderson-Darling that were designed for small sample sizes. This was the only build/algorithm that failed to pass at least one normality test for any of the amino acid analogues studied.

	Normality Tests							
	Anderson-Darling		Shapiro-Francia		Shapiro-Wilk		Pearson Chi-Square	
Amino Acid Analogue	A	p-value	W		W	p-value	P	p-value
Methane								
GROMACS SSE	0.3226	0.5160	0.9776	0.5138	0.9757	0.5347	4.55	0.6027
GROMACS C Sing Prec	1.4219	0.0010	0.9107	0.0055	0.9029	0.0023	13.10	0.0415
GROMACS SSE2 Double	0.7646	0.0429	0.9530	0.0887	0.9495	0.0727	19.85	0.0029
Variable Prec. 3.01	0.6847	0.0682	0.9561	0.1114	0.9565	0.1273	15.80	0.0149
Variable Prec. 3.31	0.6178	0.1006	0.9646	0.2070	0.9634	0.2186	10.40	0.1088
Variable Prec. 3.61	0.5067	0.1896	0.9650	0.2124	0.9686	0.3239	15.35	0.0177
Variable Prec. 3.91	0.5543	0.1432	0.9528	0.0877	0.9584	0.1471	5.45	0.4875
Num Passed Normality Test		5		6		6		3

Table 24 Methane in Water - Normality Test Results by Algorithm

	Normality Test				Wilcoxon Signed Rank Test Nonparametric		Num Norm Tests Passed / Algorithm
	Lilliefors (Kolmogorov-Smirnov)		Cramer-von Mises				
Amino Acid Analogue	D	p-value	W	p-value	V	p-value	
Methane							
GROMACS SSE	0.1017	0.3731	0.0510	0.4873	0	3.58E-08	6
GROMACS C Sing Prec	0.2000	0.0003	0.2179	0.0030	0	3.52E-08	0
GROMACS SSE2 Double	0.1285	0.0942	0.1289	0.0433	0	3.59E-08	3
Variable Prec. 3.01	0.1467	0.0299	0.1143	0.0687	0	3.54E-08	4
Variable Prec. 3.31	0.1432	0.0379	0.1055	0.0906	0	3.54E-08	5
Variable Prec. 3.61	0.1601	0.0113	0.0922	0.1387	0	3.50E-08	4
Variable Prec. 3.91	0.1249	0.1200	0.0936	0.1329	0	3.52E-08	6
Num Passed Normality Test		3		5			

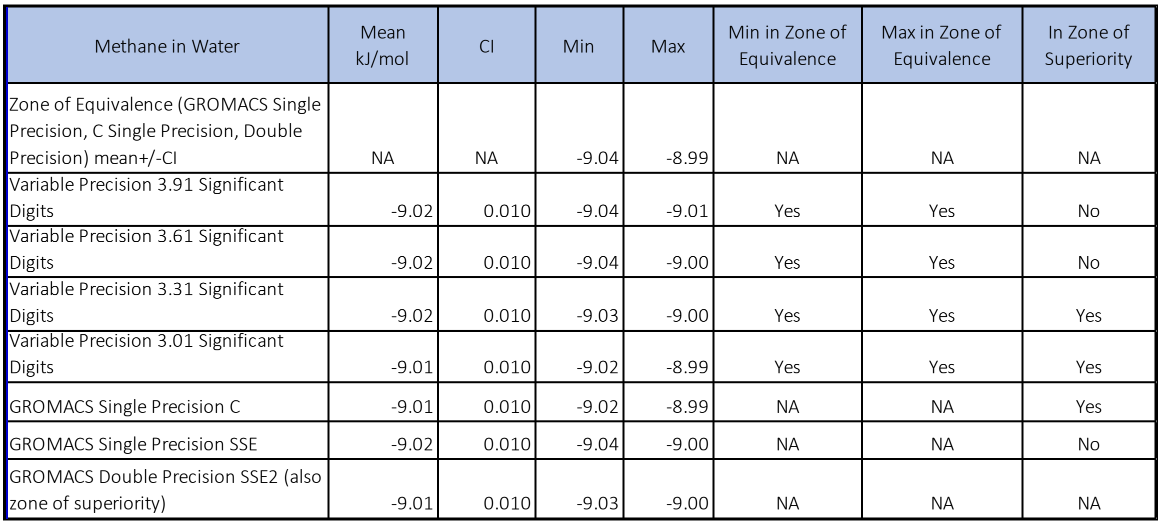
Table 25 Methane in Water - Normality Test Results by Algorithm

3.6 Equivalence Testing

3.6.1 Results for the Zone of Equivalence Testing

Zone of equivalence testing was performed in Microsoft Excel 2003 using the mean ± the confidence interval. The results in Table 26 were obtained for methane in water with a confidence level of 95% with 2% margin of error. Details on the other amino acid analogues that are part of this study may be found in the supplemental materials. The zone of equivalence was defined treating each GROMACS build as if it were a separate 'treatment' method for performing molecular dynamics simulations. This was done because the code that is generated especially for spatial calculations differs in each version. For example, the C version uses a Newton-Raphson method with a limited size table for the seed values for the 1/sqrt, the GROMACS SSE version uses the single precision assembly language instruction approximating 1/sqrt as a seed value and the SSE2 double precision uses double precision instructions. This means that there are three distinct methods of determining the 1/sqrt value that is key to force and distance calculations. Additionally the force calculations are coded to use single precision C, single precision SSE instructions and double precision SSE2 instructions that causes some variation in computational results

NA indicates not applicable since the standard GROMACS builds are by definition part of the zone of equivalence and the GROMACS SSE2 double precision is by definition the zone of superiority because of the number of significant digits supported. The statistical equivalence method that is being used is confidence interval (CI) based where the upper and lower equivalence interval are the confidence interval without any subjective knowledge based alteration of the intervals.



Methane in Water	Mean kJ/mol	CI	Min	Max	Min in Zone of Equivalence	Max in Zone of Equivalence	In Zone of Superiority
Zone of Equivalence (GROMACS Single Precision, C Single Precision, Double Precision) mean+/-CI	NA	NA	-9.04	-8.99	NA	NA	NA
Variable Precision 3.91 Significant Digits	-9.02	0.010	-9.04	-9.01	Yes	Yes	No
Variable Precision 3.61 Significant Digits	-9.02	0.010	-9.04	-9.00	Yes	Yes	No
Variable Precision 3.31 Significant Digits	-9.02	0.010	-9.03	-9.00	Yes	Yes	Yes
Variable Precision 3.01 Significant Digits	-9.01	0.010	-9.02	-8.99	Yes	Yes	Yes
GROMACS Single Precision C	-9.01	0.010	-9.02	-8.99	NA	NA	Yes
GROMACS Single Precision SSE	-9.02	0.010	-9.04	-9.00	NA	NA	No
GROMACS Double Precision SSE2 (also zone of superiority)	-9.01	0.010	-9.03	-9.00	NA	NA	NA

Table 26 Equivalence Test Results Methane in Water

3.6.2 Results for the Zone of Superiority Testing

The zone of superiority was assumed in the study to be the GROMACS SSE2 double precision build because of its greater precision. The differences between the confidence intervals for the GROMACS C single precision, SSE and SSE2 double decision builds were considered to be insufficient to establish a viable range of superiority because of the small difference between the ΔG ± values of the three GROMACS builds. In Figure 61 the GROMACS C single precision results are also within the zone of superiority.

The GROMACS SSE single precision results are only 0.01 kJ/mol different. The figure also shows the two variants of the developed algorithm with the *least* significant digits as being within the zone of superiority. These results along with those from testing with the other amino acid analogues are evidence that there is no meaningful difference between the zone of equivalence and the zone of superiority. In many cases the GROMACS single precision SSE and C builds were determined to be within the double precision SSE2 defined zone of equivalence.

3.7 Summary of Statistical Study

The above testing was performed for all five amino acid analogues and it was found that the developed algorithm with variable precisions ≈3.01, ≈3.31, ≈3.61, and ≈3.91 significant digits were all shown to be normal for most of the normality tests used. They were also within the zone of equivalence defined by the three GROMACS 4.5.3 builds single precision SSE, single precision C and double precision SSE2. There was only one exception acetamide in water with a variable precision of ≈3.01 it was found to be .01 kJ/mol outside the lower bound of the zone of equivalence. It was however interesting to note that for some amino acid analogues such as methane the GROMACS single precision C build did not pass the normality tests. The statistical results for the other amino acid analogues, histograms, quantile to quantile plots (Q-Q plots), normality test results, sample size estimates, R language scripts and raw data are contained in the supplemental materials.

3.7.1 Summary by Algorithm and Amino Acid Analogue

The following tables give credibility that the data for each combination of an amino acid analogue and algorithm is normal. It should be remembered that the data sets are relatively small (40-60). It should also be noted that even with data generated by the R language's function to generate a sample of 'normal' data of arbitrary size the normality tests may fail and the histograms and Q-Q plots may not visually appear normally distributed. The table below shows the results of the six normality tests that were used for all amino acid analogues for all 7 algorithms/builds. It shows that the Shapiro-Wilk and Cramer-von Misses normality tests obtained the best results on the sample data and the Pearson Chi-Square showed the poorest results on the data sets.

Number of Algorithms/Builds Passed Per Amino Acid Analogue Per Normality Test							
	Normality Test						
Amino Acid Analogue	Anderson-Darling	Shapiro-Francia	Shapiro-Wilk	Pearson Chi-Square	Lilliefors (Kolmogorov-Smirnov)	Cramer-von Mises	Average Passed
Acetamide	7	6	7	6	6	7	6.5
Ethanol	5	5	5	5	5	6	5.2
Isobutane	6	6	6	6	6	6	6.0
Methane	5	6	6	3	3	5	4.7
Methanol	7	7	7	5	7	7	6.7
Average	6.0	6.0	6.2	5.0	5.4	6.2	

Table 27 Normality Tests Passed by Each Amino Acid Analogue

Table 28 shows the results for each of the algorithms/builds with a count of the number of normality tests for all amino acid analogues. This reveals that for each algorithm/build approximately the same number of normality tests passed further lending credence that the data distribution for the study is normal. Only the GROMACS C single precision methane data failed all normality tests. There are a few possible explanations. 1) Normality tests do not always indicate normality even when the histogram visually appears to have a normal distribution, especially with a small sample. Normality tests that claim to be viable on data sets as small as seven are frequently not reliable even when using the R language function to generate a sample of 'normal' data. 2) There is a small cumulative difference in the results obtained using the Newton-Raphson 1/sqrt in the GROMACS C version as compared to the C library 1/sqrt function in the lower order 2 bits of the floating point representation of the result. This difference probably is too small to be seen except in very long simulations, if at all, since there are other limitations in MD simulations such as limited precision in constants. 3) The methane histogram for the GROMACS C build 'appears' similar to the GROMACS SSE2 double precision that tests normal.

Total Number of Normality Tests Passed by Build/Algorithm for All Amino Acid Analogues	
GROMACS SSE Single Prec.	27
GROMACS C Sing Prec.	24
GROMACS SSE2 Double Prec.	24
Variable Prec. 3.01	24
Variable Prec. 3.31	23
Variable Prec. 3.61	26
Variable Prec. 3.91	26

Table 28 Count of Normality Tests Passed by Algorithm/Build

The data sets representing the zone of equivalence for each of the analogues (aggregate replicates for GROMACS C, SSE, SSE2) show that fewer normality tests pass than the individual builds. This is also observed when all replicates for all builds for an analogue were combined (three GROMACS builds and the four variable precisions builds). An examination of the histograms show that there usually is a tighter clustering around the mean but in some cases there is a slight skewing that is similar to the skewing in the individual results from the GROMACS builds.

Results for Other Amino Acid Analogues

The following results were reported for the other amino acid analogues studied. Statistical equivalence was shown using the developed algorithm with acetamide, ethanol, isobutene, and methanol in addition to methane in water.

Amino Acid Analogue in Water		Mean ΔG kJ/mol	CI	Min	Max	Min in Zone Equivalence	Max in Zone Equivalence	In Zone ofSuperiority
Acetamide	Zone of Equivalence mean ± CI	NA	NA	-3.95	-3.89	NA	NA	NA
	3.91 Significant Digits	-3.93	0.02	-3.95	-3.91	Yes	Yes	Yes
	3.61 Significant Digits	-3.93	0.02	-3.95	-3.91	Yes	Yes	Yes
	3.31 Significant Digits	-3.91	0.02	-3.93	-3.89	Yes	Yes	Yes
	3.01 Significant Digits	-3.94	0.02	-3.96	-3.92	No	Yes	No
	GROMACS Single Precision C	-3.92	0.02	-3.95	-3.89	NA	NA	Yes
	GROMACS Single Precision SSE	-3.92	0.02	-3.95	-3.89	NA	NA	Yes
	GROMACS Double Precision SSE2 (also zone of superiority)	-3.92	0.03	-3.95	-3.89	NA	NA	NA
Ethanol	Zone of Equivalence mean ± CI	NA	NA	-9.48	-9.4	NA	NA	NA
	3.91 Significant Digits	-9.44	0.02	-9.47	-9.41	Yes	Yes	No
	3.61 Significant Digits	-9.44	0.02	-9.47	-9.41	Yes	Yes	No
	3.31 Significant Digits	-9.44	0.02	-9.46	-9.41	Yes	Yes	Yes
	3.01 Significant Digits	-9.43	0.02	-9.46	-9.4	Yes	Yes	Yes
	GROMACS Single Precision C	-9.44	0.02	-9.47	-9.41	NA	NA	Yes
	GROMACS Single Precision SSE	-9.43	0.02	-9.46	-9.4	NA	NA	Yes
	GROMACS Double Precision SSE2 (also zone of superiority)	-9.46	0.02	-9.48	-9.44	NA	NA	NA
Isobutane	Zone of Equivalence mean ± CI	NA	NA	-9.91	-9.85	NA	NA	NA
	3.91 Significant Digits	-9.88	0.02	-9.9	-9.86	Yes	Yes	No
	3.61 Significant Digits	-9.88	0.02	-9.9	-9.86	Yes	Yes	No
	3.31 Significant Digits	-9.87	0.02	-9.89	-9.85	Yes	Yes	No
	3.01 Significant Digits	-9.88	0.02	-9.9	-9.86	Yes	Yes	No
	GROMACS Single Precision C	-9.87	0.02	-9.89	-9.85	Yes	Yes	No
	GROMACS Single Precision SSE	-9.89	0.02	-9.91	-9.87	Yes	Yes	Yes
	GROMACS Double Precision SSE2 (also zone of superiority)	-9.89	0.02	-9.91	-9.87	NA	NA	NA
Methanol	Zone of Equivalence mean ± CI	NA	NA	-4.81	-4.76	NA	NA	NA
	3.91 Significant Digits	-4.78	0.02	-4.8	-4.76	Yes	Yes	Yes
	3.61 Significant Digits	-4.78	0.01	-4.8	-4.76	Yes	Yes	Yes
	3.31 Significant Digits	-4.78	0.02	-4.8	-4.76	Yes	Yes	Yes
	3.01 Significant Digits	-4.79	0.02	-4.81	-4.77	Yes	Yes	Yes
	GROMACS Single Precision C	-4.79	0.01	-4.81	-4.77	NA	NA	Yes
	GROMACS Single Precision SSE	-4.79	0.02	-4.81	-4.77	NA	NA	Yes
	GROMACS Double Precision SSE2 (also zone of superiority)	-4.78	0.02	-4.8	-4.76	NA	NA	NA

Table 29 Zone of Equivalence Results - All Amino Acid Analogues in Study

3.8 Conclusions

This study performed over 170 μs of simulations on five amino acid analogues to provide the basis for the analysis of statistical equivalence and superiority of the developed algorithm for improving the performance of MD simulations versus the standard GROMACS builds. A very strict definition of the zone of equivalence was used based on the widely used single precision C, SSE and double precision GROMACS 4.5.3 builds. If the developed variable precision algorithm did not fall *completel*y within this zone then it was not considered equivalent. This is more stringent that the methods used in clinical trials where there is usually a 'range' outside of the upper and lower limit of statistical that is included in the zone of equivalence considered 'good enough' by experts in the field. This study showed that when the noncomputational incremental algorithm is used in free energy studies of five amino acid analogues with a reduced precision between ≈3.01 and ≈3.91 significant digits, the results of the simulations were statistically equivalent and indistinguishable to the GROMACS builds. Prior studies of these amino acid analogues by others showed that the results of free energy studies of these simple systems could be used to validate GROMACS simulation results with experimentally determined values. This study provides strong evidence that the developed algorithm may be used in general purpose molecular dynamics simulations without causing side effects. It also suggests that testing of new performance algorithms including GPU and alternative distributed computational models can use equivalence testing as a means of validation with prior software versions and experimental data. Equivalence testing can also be applied to the evaluation of new/modified force field and water models.

The supplemental materials contain detailed of normality tests results, sample size estimation, quantile-quantile plots, histograms, and descriptive statistics. Also included are the R language scripts and data tables.

3.9 Acknowledgments

The author would like to express his thanks to Dr. Paulo Martel, Centre for Biomedical Research, the University of the Algarve, Faro, Portugal for his confidence that this project was feasible and for his support during the project.

The author is also grateful to members of the Computational Systems Biology Group, Center for Neuroscience and Cell Biology, UC-Biotech, University of Coimbra led by Dr. Armindo Salvador; the Biological Chemistry Group, Department of Chemistry, University of Coimbra led by Dr. Maria Joao Moreno; and to Dr. Claudio Soares, New University of Lisbon, Lisbon, Portugal for stimulating discussions about the project and suggestions on approaches to validation of the methodology.

The Biological Chemistry Group, Department of Chemistry, University of Coimbra also provided substantial computational resources that made the free energy studies possible. The author would like to thank Virtual Strategy, Inc., Boston, MA for funding this research and supporting the patent application.

Keywords: performance, statistical equivalence, non-computation, nonbonded, free energy

3.10 References

[105] Hess, B. and N. F. A. van der Vegt (2006). "Hydration Thermodynamic Properties of Amino Acid Analogues: A Systematic Comparison of Biomolecular Force Fields and Water Models." *The Journal of Physical Chemistry B* 110(35): 17616-17626.

[106] Shirts, M. R., J. W. Pitera, et al. (2003). "Extremely precise free energy calculations of amino acid side chain analogs: Comparison of common molecular mechanics force fields for proteins." *The Journal of Chemical Physics* 119(11): 5740-5761.

[107] Pande, V. S., Baker, I., Chapman, J., Elmer, S. P., Khaliq, S., Larson, S. M., Rhee, Y. M., Shirts, M. R., Snow, C. D., Sorin, E. J. and Zagrovic, B. (2003), Atomistic protein folding simulations on the submillisecond time scale using worldwide distributed computing. Biopolymers, 68: 91–109. doi: 10.1002/bip.10219

[108] Hess, B., Kutzner, C., van der Spoel, D., Lindahl, E. GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation. J. Chem. Theory Comp. 4(3):435–447, 2008.

[109] Bekker, H., Berendsen, H. J. C., Dijkstra, E. J., Achterop, S., van Drunen, R., van der Spoel, D., Sijbers, A., Keegstra, H., Reitsma, B., Renardus, M. K. R. Gromacs: A parallel computer for molecular dynamics simulations. In Physics Computing 92 (Singapore, 1993). de Groot, R. A., Nadrchal, J., eds. . World Scientific.

[110] D. van der Spoel, E. Lindahl, B. Hess, A. R. van Buuren, E. Apol, P. J. Meulenhoff, D. P. Tieleman, A. L. T. M. Sijbers, K. A. Feenstra, R. van Drunen and H. J. C. Berendsen, Gromacs User Manual version 4.5.6, www.gromacs.org (2010)

[111] Bowman, D, 2015, Unpublished, "Optimizing Applications in HPC Environments Using Incremental and Noncomputational Methods"

[112] Tironi, I. G., Sperb, R., Smith, P. E., van Gunsteren, W. F. A generalized reaction field method for molecular dynamics simulations. J. Chem. Phys. 102:5451–5459, 1995.

[113] C. D. Christ, A. E. Mark, and W. F. van Gunsteren (2010) J. Comput. Chem. 31: 1569-1582. DOI

[114] A. Pohorille, C. Jarzynski, and C. Chipot (2010) J. Phys. Chem. B 114: 10235-10253. DOI

[115] A. Villa and A. E. Mark (2002) J. Comput. Chem. 23: 548-553. DOI

[116] Clara D Christ, Alan E. Mark, Wilfred F. van Gunsteren, Basic Ingredients of Free Energy calculations: A Review, 2009, *Journal of Computational Chemistry* DOI 10.1002

[117] Mobley, D. L., É. Dumont, et al. (2010). "Comparison of Charge Models for Fixed-Charge Force Fields: Small Molecule Hydration Free Energies in Explicit Solvent." *The Journal of Physical Chemistry B* 115(5): 1329-1332.

[118] Justin A. Lemkul, "GROMACS Tutorial Free Energy Calculations: Methane in Water", Bevan Laboratory, Virginia Tech, Blacksburg, VA 24061, USA, http://www.bevanlab.biochem.vt.edu/Pages/Personal/justin/gmx-tutorials/free_energy_old/index.html

[119] Bennett, Charles H. (1976). "Efficient estimation of free energy differences from Monte Carlo data", Journal of Computational Physics Volume 22, Issue 2, October 1976, Pages 245-266

[120] C.D. Christ, A.E. Mark, W.F. van Gunsteren (2010) J. Comput. Chem. 31: 1569-1582. DOI

[121] A. Pohorille, C. Jarzynski, and C. Chipot (2010) *J. Phys. Chem.* B 114: 10235-10253. DOI

[122] A. Villa and A.E. Mark (2002) *J. Comput. Chem*. 23: 5488-553. DOI

[123] Bowman, D, 2015, Unpublished, "Accelerating Molecular Dynamics Simulations with Incremental and Non-omputational Algorithms"

[124] R version 3.1.2 (2014-10-31) Copyright (C) 2014 The R Foundation for Statistical Computing Platform: x86_64-w64-mingw32/x64 (64-bit)

[125] StatView for Windows, Version 5.0, SAS Institute Inc. Copyright © 1992-1998.

[126] Berk Hess and Nico F. A. van der Vegt, 2006,, "Hydration Thermodynamic Properties of Amino Acid Analogues: A Systematic Comparison of Biomolecular Force Fields and Water Models", Journal of Physical Chemistry B, 110 (35), pp 17616–17626
DOI: 10.1021/jp0641029

[127] Michael R Shirts, Vijay S Pande, 2005, "Solvation free energies of amino acid side chain analogs for common molecular mechanics water models", Journal of Chemical Physics. 122, 134508 (2005); http://dx.doi.org/10.1063/1.1877132

[128] William L. Jorgensen, David S. Maxwell, and, and Julian Tirado-Rives, (1996) "Development and Testing of the OPLS All-Atom Force Field on Conformational Energetics and Properties of Organic Liquids", Journal of the American Chemical Society, 118 (45), 11225-11236, DOI

[129] Jacques S. Lee, MD, "Understanding equivalence trials (and why we should care)", CJEM 2001;2(3):194-196

[130] Blackwelder, W.C. `Equivalence trials.' Encyclopedia of Biostatistics, 1997.

[131] Pamela J. Atherton Skaff, Jeff A. Sloan, Mayo Clinic, Rochester, MN 55905. "Design and Analysis of Equivalence Clinical Trials Via the SAS® System", www.sas.com

[132] Piaggio G, Elbourne DR, Pocock SJ, Evans SJW, Altman DG, for the CONSORT Group. "Reporting of non-inferiority and equivalence randomized trials. Extension of the CONSORT 2010 statement". *JAMA*. 2012; 308(24): 2594-2604. doi:10.1001/jama.2012.87802. PMID: 23268518

[133] Le Henanff A, Giraudeau B, Baron G, Ravaud P. "Quality of Reporting of Noninferiority and Equivalence Randomized Trials". *JAMA*.2006;295 (10):1147-1151. doi:10.1001/jama.295.10.1147.

[134] Wellek, S. (2010) *Testing Statistical Hypotheses of Equivalence*, Chapman and Hall/CRCm, ISBN: 978-1439808184.

[135] Norman, G. and Streiner, D. (2008) *Biostatistics the Bare Essentials*, People's Medical Publishing House, ISBN: 978-1-55009-347-6

[136] Kendall, Philip C.; Marrs-Garcia, Abbe; Nath, Sanjay R.; Sheldrick, Radley C. "Normative comparisons for the evaluation of clinical significance", Journal of Consulting and Clinical Psychology, Vol 67(3), Jun 1999, 285-299. http://dx.doi.org/10.1037/0022-006X.67.3.285

[137] Anderson, T.W.; Darling, D.A. (1954). "A Test of Goodness of Fit". J*ournal of the American Statistical Association*, Vol 49, Iss. 268

[138] Shapiro, S. S.; Wilk, M.B. (1965). "An analysis of variance test for normality (complete samples)". *Biometrika* 52 (3-4): 591-611. doi:10.1093/biomet/52.3-4.591. JSTOR 2333709. MR 205384. p. 593

[139] Razali, Nornadiah; Wah, Yap Bee (2011)/ "Power comparisions of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests". *Journal of Statistical Modeling and* Analytics 2 (1): 21-33.

[140] Plackett, R. L. (1983). "Karl Pearson and the Chi-Squared Test". International Statistical Review (International Statistical Institute (ISI)) 51 (1): 59-72. doi:10.2307/1402731. JSTOR 1402731

[141] Lilliefors, H. (June 1967), "On the Kolmogorov-Smirnov test for normality with mean and variance unknown", *Journal of the American Statistical Association,* Vol. 62. pp. 399-402.

[142] Lilliefors, H. (1969), "On the Kolmogorov-Smirnov test for the exponential distribution with mean unknown", Journal of the American Statistical Association., Vol. 64. pp. 387-389.

[143] Cramer, H. On the composition of elementary errors", *Scandinavian Actuarial Journal 1928*

[144] Anderson, T. W. (1962). "On the distribution of the Two-Sample Cramer-von Mises Criterion", *The Annals of Mathematical Statistics* (Institute of Mathematical Statistics) 33 (3): 1148-1159. doi:10.1214/aoms/1177704477. ISSN 0003-481.

General Discussion

Biomolecular simulation software such as GROMACS have proven to be important tools to study the physico-chemical properties of biomolecules and their mechanisms and to describe extremely short-lived molecular phenomena otherwise difficult to describe.[1, 2] The state of the art has come a long way since the first protein simulations in the early 70's, both in terms of theoretical methods and the computer hardware used. Extensive work has been performed to develop and validate force fields, water models and algorithms with experimental data especially using free energy studies. Simulation of the molecular dynamics (MD) of solvated proteins with atomic detail requires the use of systems with the order of tens to hundreds of thousands of atoms. Particle simulations with detailed molecular potentials can be extremely heavy in systems of this size, and until recently only relatively short simulation times (10-100 ns) were accessible with most computer systems available.

Molecular dynamics software maximizes performance by using methods to reduce the number of interactions, the latest computer instructions, multi-core and distributed computing architectures. Calculation of nonbonded interactions is the major performance problem. The speed of a simulation is limited by the processor architecture and how finely the simulation can be distributed across multiple processors/cores. Simulations in the range of 10s of thousands of atoms will not run significantly faster on a supercomputer than on a 64 core server. Improvements in the calculation of nonbonded forces for water interactions for Lennard-Jones with Reaction Field on the Intel Core i7 'Sandy Bridge' of 14-15 times that of the GROMACS 4.5.3 hand coded assembly language versions were achieved on a single thread on a single core. Part of this is due to the elimination of the 1/sqrt calculation. This is the fundamental unit of performance in any computational environment whether it is on a single processor, GPU and distributed computer systems. The

[1] Karplus, M. and J.A. McCammon, *Molecular dynamics simulations of biomolecules.* Nature Structural Biology, 2002. 9(9): p. 646-652.

[2] Karplus, M. and J. Kuriyan, *Molecular dynamics and protein function.* Proceedings of the National Academy of Sciences of the United States of America, 2005. 102(19): p. 6679-6685.

method could also be applied to core PME and other computationally intensive functions with MD software.

This algorithm was validated for mathematical and computational stability and performance. It was also tested using water boxes and proteins. The results from water box and protein simulations appeared to show that the developed algorithm was suitable for MD simulations but this was based on a relatively small number of simulations examining energy drift, RMSD and other factors. Protein and small molecule studies can validate a new performance algorithm only at a high level.

The developed algorithm is also generally applicable to most HPC applications and hardware that perform object to object interactions and that have *already* been fully optimized using optimizing compilers, off-loaded to GPUs, run in distributed environments and uses the latest and most efficient algorithms. After developers have fully optimized their code and it cannot be made faster by design or computation a way must be found to completely avoid computations or the application will not run any faster.

The algorithm may be used in a broad class of HPC applications that run on supercomputers. These applications are generally used to address two major classes of problems: 1) problems with large amounts of data that has relatively few/no interdependencies and can be processed as many streams and 2) problems that are computationally intensive because they have large numbers of interactions between many objects. These problems may be in many diverse areas such as: weather forecasting, oceanography, climate change, the evolution of galaxies, development of stars and clusters, black holes, particle physics, molecular dynamics, protein folding, fluid dynamics, economics or other applications. The first class of applications can be easily distributed over an unlimited number of processors or cores. The second class of applications generally have a core internal computationally intensive code section (frequently to calculate forces) and a large number of object interactions. Software simulating large numbers of interactions uses algorithms (e.g. lattice summation or spherical cutoffs) to reduce the number of interactions from $O(N^2)$ to O(NlogN) or O(N). They exploit the latest processor architectures, OpenMP, MPI, vector instruction sets such as AVX, AVX512 and FMA and offload work to GPU coprocessors using NVIDIA CUDA or OpenCL. Even with these techniques there are limits to both the number of object interactions that can be processed and the amount of time that these interactions may be simulated independent of the problem being solved.

The method was also used to develop C library math functions that operate within a limited precision and range of exponents. The performance of the developed functions was compared to the standard C library functions with performance speedups in the range of 11 to 125 times faster than using gcc 4.7. The developed method may be applied to simple functions, equations or the simultaneous solution of multiple equations.

The developed algorithm has a number of limitations. The most significant is the amount of CPU L2, L3, L4 cache available for use with the lookup tables for the intermediate results. This study showed that there was a substantial reduction in performance as compared to actually computing the results when the tables used for the incremental results exceeded the CPU L2/L3 cache. It should also be considered that other applications may be using the core or processor that may cause cache misses for an application using the incremental lookup tables. Attention should be given to associating threads/processes using the algorithm with a physical core, CPU block and blade/node. If processes are allowed to move from core to core cache misses will result and this will have an adverse impact on performance. Use of the algorithm in virtual machine environments with multiple virtual cores mapped to a single physical core was not evaluated. In a virtual environment it may be difficult or impossible to associate a thread/process with a physical core. If this cannot be done there will likely be a performance penalty.

The methodology requires application specific knowledge to allow the creation of reduced precision lookup tables for intermediate results. If this is not available it must be obtained by instrumenting the application to collect the data for the specific execution being performed. This is similar to the auto optimization that is done in some FFT libraries.

The algorithm can be used if the number of significant digits is less than or equal to ≈4.21 significant digits. This limitation is due to the current size of processor cache memory. Use of greater than ≈4.21 significant digits can/will result in table sizes that result in cache misses and a degradation in performance. The exponent range must also be known so that each level representing a power of two can be initialized properly. The exponents do not need to be contiguous.

Only one floating point/integer value may be used as a function at runtime. All other values required to initialize the lookup tables must be known and static at runtime.

The implementation of the table initialization and retrieval routines can be implemented to use fewer exponent levels of the IEEE 754 standard. If 80% of a function/equation's execution occurs within a range of values then only the exponent levels required to support the 80% need to be implemented as lookup tables and in all other cases the results would be returned by computation.

Functions that are cyclic can implemented using smaller lookup tables and a small amount of additional code to manipulate the index value. Typically this code would consist of integer, and, or instructions or shift operations that execute in less than one cycle.

The AVX2 instruction set using the vsgather instruction and AVX2 integer and bit manipulations gives the optimal level of performance by significantly reducing the number of computer instructions.

In order to provide validation at the force field and water model levels free energy studies of amino acid analogues were necessary.

This study used a methodology to determine statistical equivalence similar to what is performed in clinical trials to evaluate if the results produced by the developed algorithms were 'good enough' for MD simulations.

This study used a statistically significant number of free energy studies on 5 amino acid side chain analogues and compared the results with existing GROMACS 4.5.3 build versions and experimental data. It demonstrated that the developed algorithm produced statistically equivalent results as compared to the existing GROMACS 4.5.3 builds. This study was modeled after studies used to validate force fields and water models that can be tied to experimental data. Using these small systems it was possible to obtain a statistically significant number of samples and use statistical equivalence methodologies and to demonstrate that the developed algorithm produced statistically equivalent results to the three GROMACS 4.5.3 builds (C single precision, SSE single precision and SSE2 double precision)

Conclusions

The ‘calculation’ of the forces and intermediate results may be improved substantially by use of a noncomputational and incremental computation model that exploits a variable precision numeric format based on the IEEE 754 standard for single precision values on a single thread and single physical core using the hand coded assembly versions of GROMACS 4.5.3. It was shown that variable precision formats in the range of ≈3.01 to ≈3.91 significant digits for the spatial calculations at the core of the non-bonded routines effectively permits the simulation to run in 1pm ‘space’. Using a coarse grain approach to obtaining the force results allows the creation of indices for accessing pre-computed results without the artifacts associated with a simple conversion to integer lookup method.

It has been shown than a series of tables paralleling the IEEE 754 standard supporting variable precision coarse grain space using 3 significant digits precision can be generated to support water to water and water to solute interactions using the GROMACS Lennard-Jones reaction field and the Lennard-Jones only assembly language routines. These assembly routines perform 15 times faster on an 2GHZ Intel Core i7 ‘Sandy Bridge’ and 2.6 times faster on an Intel Core2 Quad core 2 GHZ and 3.2 times faster on an Intel Xeon 5650 2.67 GHZ 12MB Cache 6 cores (Westmere-EP 32nm). Improvements on various AMD CPUs showed an improvement of 1.6 times faster.

Testing was also performed on a number of C programming library functions such as sqrt, log, tan, cos, etc. with results in performance improvements that were in the range of 11 to 125 times faster over a specified range of values at a reduced precision. The algorithm was also tested against an integer based lookup table method and was found to be 7 times faster on the Intel ‘Sandy Bridge’ Core i7.

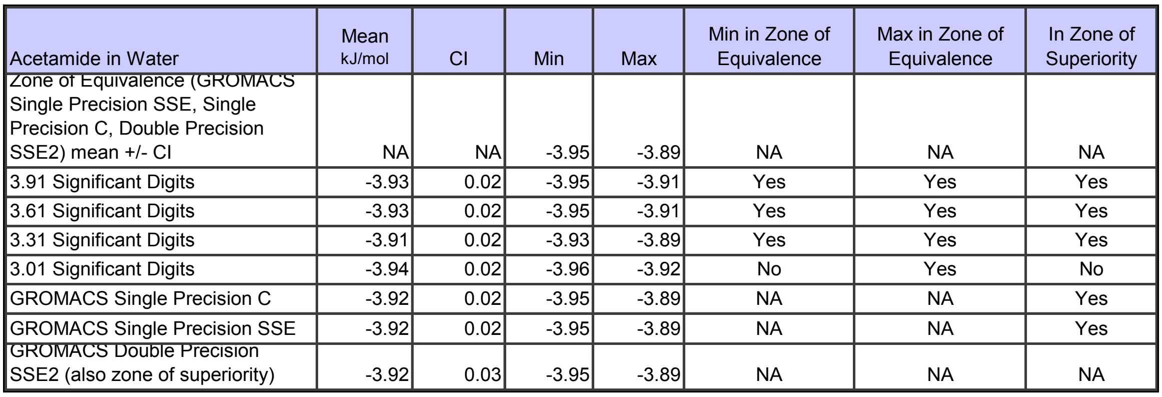
The performance of the algorithm is highly dependent on the percent of CPU and clock time used for performing nonbonded calculations, the amount of cache memory used for the incremental result caches, the overall memory requirements of the simulation, processor technology used, cache architecture, motherboard/blade design, node configuration and network bandwidth. In all cases tested Intel processors performed better than AMD with Intel ‘Sandy Bridge’ and ‘Ivy Bridge’ greatly exceeding the performance of earlier Intel architectures. Preliminary testing on the Intel ‘Haswell’ architecture shows a further increase in performance based on improvements in the cache architecture and the addition of new instructions such as *vsgather* that permit the developed algorithm to reduce the number of instructions by half. In AVX512 and other environments with higher data per cycle processing power, it is expected that the algorithm will continue to greatly exceed the performance of calculation-based conventional approaches.

The developed algorithm must be validated on a case by case basis to ensure that an application continues to meet user requirements after implementing the developed method. In order to accomplish this for the molecular dynamics software GROMACS a free energy study performed over 170 μs of simulations on five amino acid analogues was carried out providing a basis for the analysis of statistical equivalence and superiority of the developed algorithm when compared to the standard GROMACS builds. A very strict definition of the zone of equivalence was used based on the widely used single precision C, SSE and double precision GROMACS 4.5.3 builds. If the developed variable precision algorithm did not fall *completel*y within this zone then it was not considered equivalent. This is more stringent that the methods used in clinical trials where there is usually a ‘range’ outside of the upper and lower limit of statistical that is included in the zone of equivalence considered ‘good enough’ by experts in the field. This study showed that when the noncomputational incremental algorithm is used in free energy studies of five amino acid analogues with a reduced precision between ≈3.01 and ≈3.91 significant digits the results of the simulations were statistically equivalent and indistinguishable to the GROMACS builds. Prior studies of these amino acid analogues by others showed that the results of free energy studies of these simple systems could be used to validate GROMACS simulation results with experimentally determined values. This study provides strong evidence that the developed algorithm may be used in general purpose molecular dynamics simulations without causing side effects.

Appendix 1 – Supplemental Materials Chapter 3

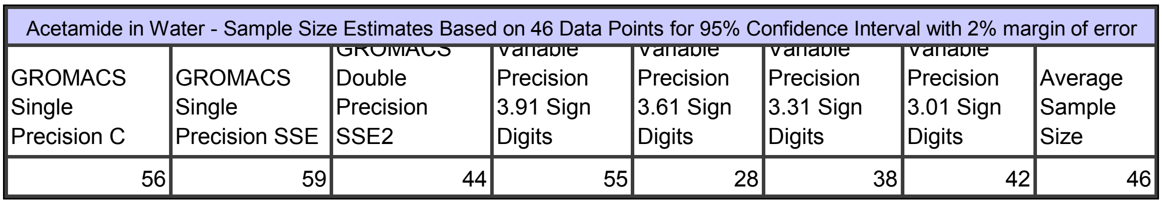
1.1 Equivalence Test Results and Sample Sizes

1.1.1 Acetamide in Water



Acetamide in Water	Mean kJ/mol	CI	Min	Max	Min in Zone of Equivalence	Max in Zone of Equivalence	In Zone of Superiority
Zone of Equivalence (GROMACS Single Precision SSE, Single Precision C, Double Precision SSE2) mean +/- CI	NA	NA	-3.95	-3.89	NA	NA	NA
3.91 Significant Digits	-3.93	0.02	-3.95	-3.91	Yes	Yes	Yes
3.61 Significant Digits	-3.93	0.02	-3.95	-3.91	Yes	Yes	Yes
3.31 Significant Digits	-3.91	0.02	-3.93	-3.89	Yes	Yes	Yes
3.01 Significant Digits	-3.94	0.02	-3.96	-3.92	No	Yes	No
GROMACS Single Precision C	-3.92	0.02	-3.95	-3.89	NA	NA	Yes
GROMACS Single Precision SSE	-3.92	0.02	-3.95	-3.89	NA	NA	Yes
GROMACS Double Precision SSE2 (also zone of superiority)	-3.92	0.03	-3.95	-3.89	NA	NA	NA

Figure 60 Equivalence Test Results Acetamide in Water



Acetamide in Water - Sample Size Estimates Based on 46 Data Points for 95% Confidence Interval with 2% margin of error

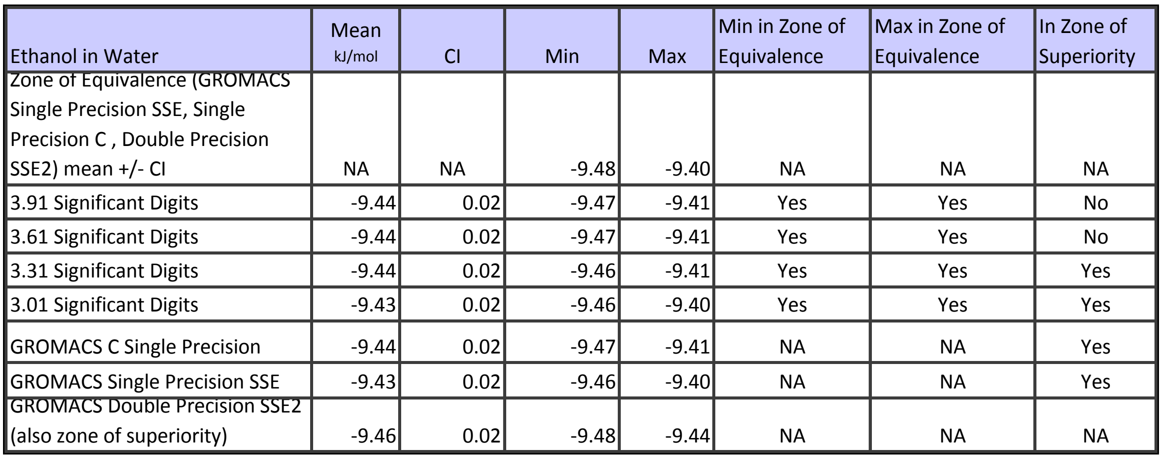
GROMACS Single Precision C	GROMACS Single Precision SSE	GROMACS Double Precision SSE2	Variable Precision 3.91 Sign Digits	Variable Precision 3.61 Sign Digits	Variable Precision 3.31 Sign Digits	Variable Precision 3.01 Sign Digits	Average Sample Size
56	59	44	55	28	38	42	46

Figure 61 Acetamide in Water - Sample Size Estimates Base on 46 data points

The sample sizes for the two GROMACS 4.5.3 single precision builds are greater than that for any of the variants of the developed algorithm. The average sample size estimate of 46 was used for analysis.

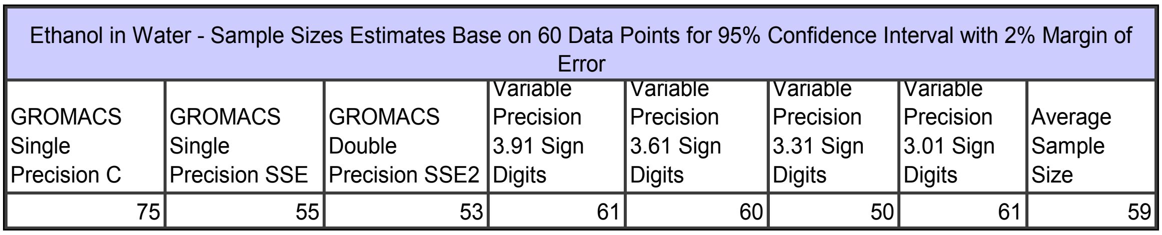
The results of the variable precision algorithm for ≈3.01 significant digits for acetamide is slightly lower than the lower bound of the zone of equivalence (.01 kJ/mol). This is probably of no consequence to molecular dynamics simulations. It is also the only amino acid analogue studied that fell outside of the zone of equivalence. When zones of equivalence are defined there is usually a range a little below or above the range that would be determined by the means and confidence intervals of the various methods or treatments that are already considered 'good enough.' In clinical trials for example there are small variations above and below the treatment results of available drugs that are considered to be part of the zone of equivalence or zone of superiority. This study used a more conservative methodology where the zone of equivalence was defined as *completely* within the equivalence zone defined by the three GROMACS 4.5.3 builds.

1.1.2 Ethanol in Water



Ethanol in Water	Mean kJ/mol	CI	Min	Max	Min in Zone of Equivalence	Max in Zone of Equivalence	In Zone of Superiority
Zone of Equivalence (GROMACS Single Precision SSE, Single Precision C , Double Precision SSE2) mean +/- CI	NA	NA	-9.48	-9.40	NA	NA	NA
3.91 Significant Digits	-9.44	0.02	-9.47	-9.41	Yes	Yes	No
3.61 Significant Digits	-9.44	0.02	-9.47	-9.41	Yes	Yes	No
3.31 Significant Digits	-9.44	0.02	-9.46	-9.41	Yes	Yes	Yes
3.01 Significant Digits	-9.43	0.02	-9.46	-9.40	Yes	Yes	Yes
GROMACS C Single Precision	-9.44	0.02	-9.47	-9.41	NA	NA	Yes
GROMACS Single Precision SSE	-9.43	0.02	-9.46	-9.40	NA	NA	Yes
GROMACS Double Precision SSE2 (also zone of superiority)	-9.46	0.02	-9.48	-9.44	NA	NA	NA

Figure 62 Equivalence Test Results Ethanol in Water

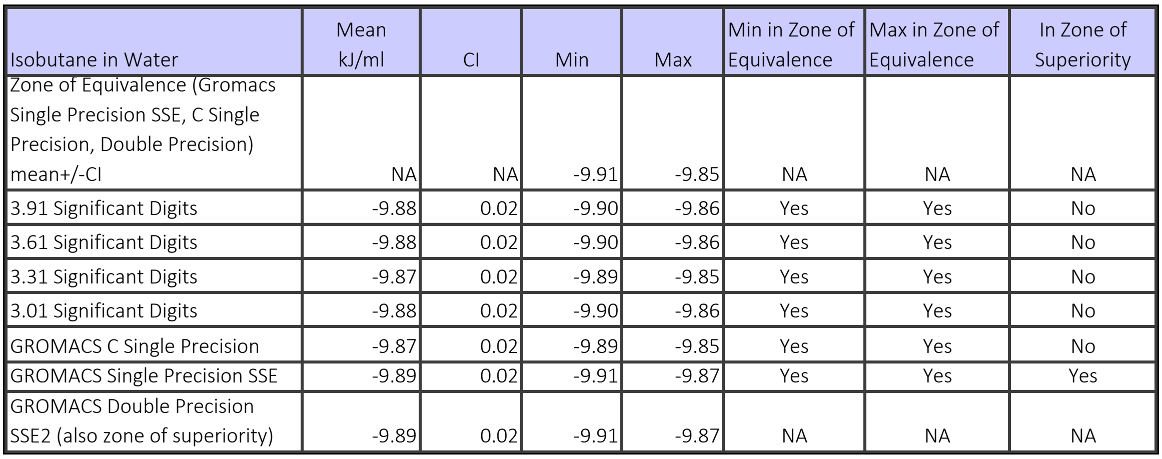


Ethanol in Water - Sample Sizes Estimates Base on 60 Data Points for 95% Confidence Interval with 2% Margin of Error							
GROMACS Single Precision C	GROMACS Single Precision SSE	GROMACS Double Precision SSE2	Variable Precision 3.91 Sign Digits	Variable Precision 3.61 Sign Digits	Variable Precision 3.31 Sign Digits	Variable Precision 3.01 Sign Digits	Average Sample Size
75	55	53	61	60	50	61	59

Figure 63 Ethanol in Water - Sample Size Estimates Base on 60 data points

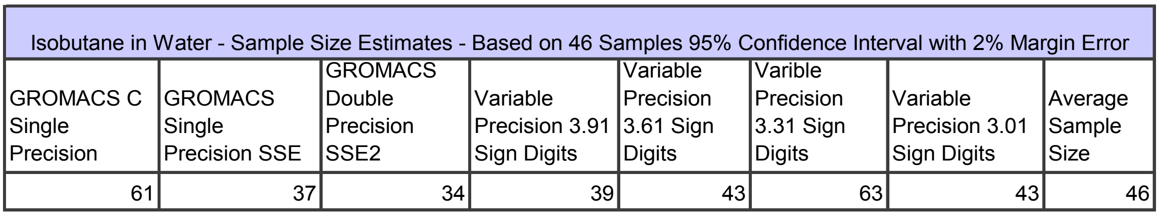
Ethanol in water shows higher sample requirement for the GROMACS C single precision than the GROMACS SSE or SSE2 double precision. A sample size of 60 was used for analysis.

1.1.3 Isobutane in Water



Isobutane in Water	Mean kJ/ml	CI	Min	Max	Min in Zone of Equivalence	Max in Zone of Equivalence	In Zone of Superiority
Zone of Equivalence (Gromacs Single Precision SSE, C Single Precision, Double Precision) mean+/-CI	NA	NA	-9.91	-9.85	NA	NA	NA
3.91 Significant Digits	-9.88	0.02	-9.90	-9.86	Yes	Yes	No
3.61 Significant Digits	-9.88	0.02	-9.90	-9.86	Yes	Yes	No
3.31 Significant Digits	-9.87	0.02	-9.89	-9.85	Yes	Yes	No
3.01 Significant Digits	-9.88	0.02	-9.90	-9.86	Yes	Yes	No
GROMACS C Single Precision	-9.87	0.02	-9.89	-9.85	Yes	Yes	No
GROMACS Single Precision SSE	-9.89	0.02	-9.91	-9.87	Yes	Yes	Yes
GROMACS Double Precision SSE2 (also zone of superiority)	-9.89	0.02	-9.91	-9.87	NA	NA	NA

Figure 64 Equivalence Test Results Isobutane in Water

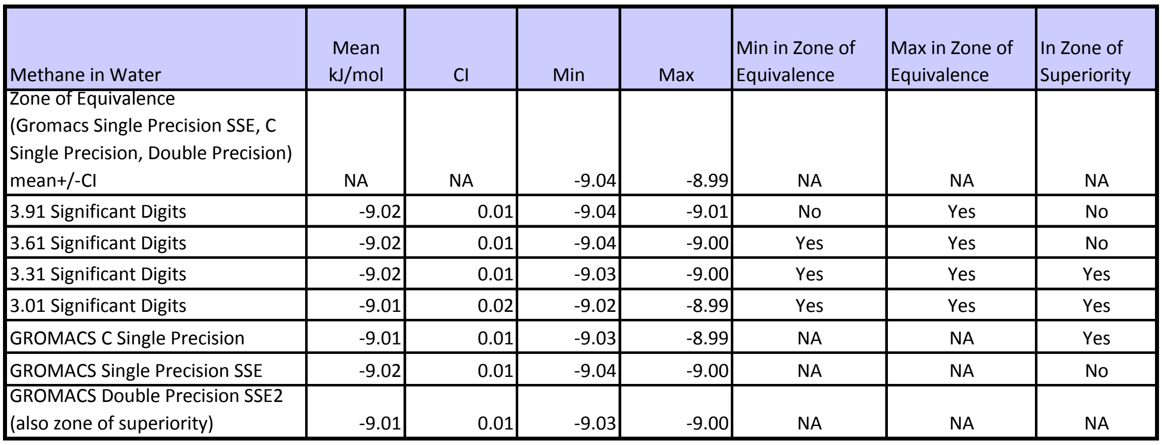


Isobutane in Water - Sample Size Estimates - Based on 46 Samples 95% Confidence Interval with 2% Margin Error							
GROMACS C Single Precision	GROMACS Single Precision SSE	GROMACS Double Precision SSE2	Variable Precision 3.91 Sign Digits	Variable Precision 3.61 Sign Digits	Varible Precision 3.31 Sign Digits	Variable Precision 3.01 Sign Digits	Average Sample Size
61	37	34	39	43	63	43	46

Figure 65 Isobutane in Water - Sample Size Estimates Base on 46 data points

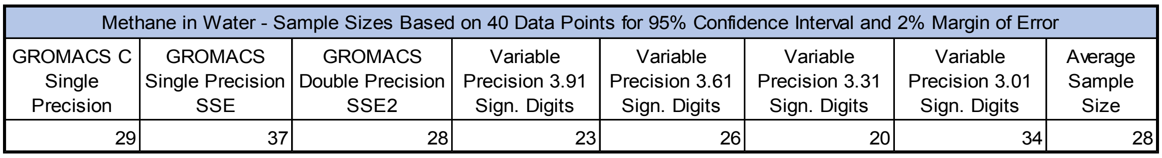
Isobutane in water also shows that GROMACS 4.5.3 C single precision build has a larger sample size than that for the other builds/variants of the developed algorithm except for the developed algorithm using ≈3.31 significant digits. The average sample size estimate of 46 was used for analysis.

1.1.4 Methane in Water



Methane in Water	Mean kJ/mol	CI	Min	Max	Min in Zone of Equivalence	Max in Zone of Equivalence	In Zone of Superiority
Zone of Equivalence (Gromacs Single Precision SSE, C Single Precision, Double Precision) mean+/-CI	NA	NA	-9.04	-8.99	NA	NA	NA
3.91 Significant Digits	-9.02	0.01	-9.04	-9.01	No	Yes	No
3.61 Significant Digits	-9.02	0.01	-9.04	-9.00	Yes	Yes	No
3.31 Significant Digits	-9.02	0.01	-9.03	-9.00	Yes	Yes	Yes
3.01 Significant Digits	-9.01	0.02	-9.02	-8.99	Yes	Yes	Yes
GROMACS C Single Precision	-9.01	0.01	-9.03	-8.99	NA	NA	Yes
GROMACS Single Precision SSE	-9.02	0.01	-9.04	-9.00	NA	NA	No
GROMACS Double Precision SSE2 (also zone of superiority)	-9.01	0.01	-9.03	-9.00	NA	NA	NA

Figure 66 Equivalence Test Results Methane in Water

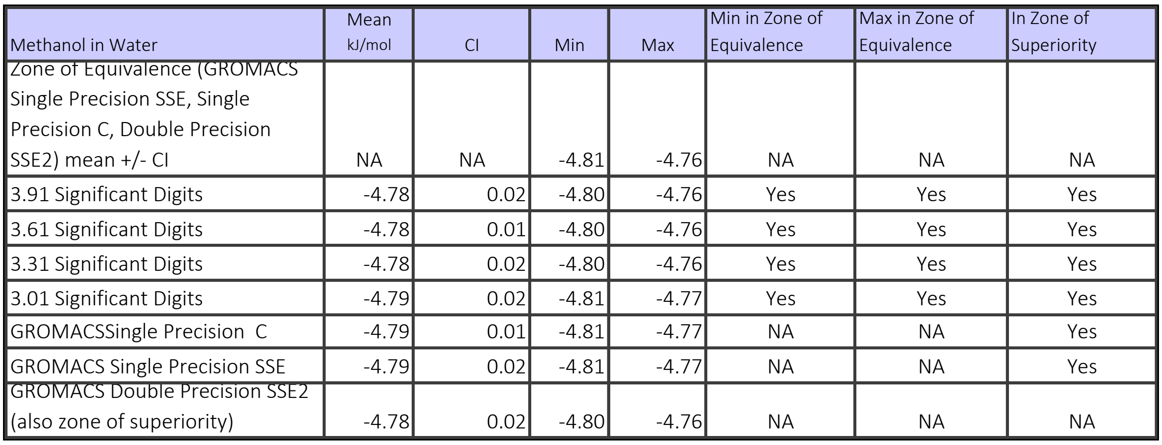


Methane in Water - Sample Sizes Based on 40 Data Points for 95% Confidence Interval and 2% Margin of Error							
GROMACS C Single Precision	GROMACS Single Precision SSE	GROMACS Double Precision SSE2	Variable Precision 3.91 Sign. Digits	Variable Precision 3.61 Sign. Digits	Variable Precision 3.31 Sign. Digits	Variable Precision 3.01 Sign. Digits	Average Sample Size
29	37	28	23	26	20	34	28

Figure 67 Methane in Water - Sample Size Estimates Base on 40 data points

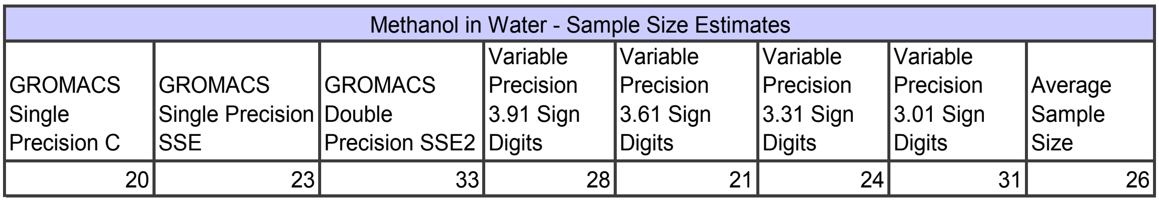
The ≈3.31 and ≈3.01 significant digit variants of the developed algorithm have values that indicate that they are in the zone of superiority. However, this is not meaningful because the variants with greater precision and the GROMACS 4.5.3 single precision SSE are not in the zone of superiority. This lends credibility to the theory that the differences between the zone of equivalence and zone of superiority are so small that they are not meaningful. A sample size of 40 was used for the analysis.

1.1.5 Methanol in Water



Methanol in Water	Mean kJ/mol	CI	Min	Max	Min in Zone of Equivalence	Max in Zone of Equivalence	In Zone of Superiority
Zone of Equivalence (GROMACS Single Precision SSE, Single Precision C, Double Precision SSE2) mean +/- CI	NA	NA	-4.81	-4.76	NA	NA	NA
3.91 Significant Digits	-4.78	0.02	-4.80	-4.76	Yes	Yes	Yes
3.61 Significant Digits	-4.78	0.01	-4.80	-4.76	Yes	Yes	Yes
3.31 Significant Digits	-4.78	0.02	-4.80	-4.76	Yes	Yes	Yes
3.01 Significant Digits	-4.79	0.02	-4.81	-4.77	Yes	Yes	Yes
GROMACSSingle Precision C	-4.79	0.01	-4.81	-4.77	NA	NA	Yes
GROMACS Single Precision SSE	-4.79	0.02	-4.81	-4.77	NA	NA	Yes
GROMACS Double Precision SSE2 (also zone of superiority)	-4.78	0.02	-4.80	-4.76	NA	NA	NA

Figure 68 Equivalence Testing Results Methanol in Water



Methanol in Water - Sample Size Estimates							
GROMACS Single Precision C	GROMACS Single Precision SSE	GROMACS Double Precision SSE2	Variable Precision 3.91 Sign Digits	Variable Precision 3.61 Sign Digits	Variable Precision 3.31 Sign Digits	Variable Precision 3.01 Sign Digits	Average Sample Size
20	23	33	28	21	24	31	26

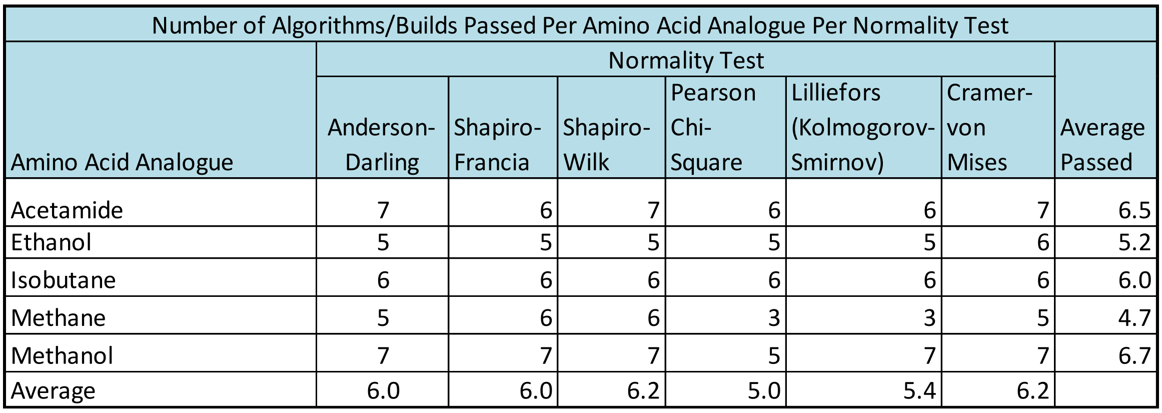
Figure 69 Methanol in Water Sample Size Estimates Base on 40 data points

A sample size of 40 was used even though the maximum estimated sample size estimate was 33 and the average sample estimate was 26.

2.1 Normality Test Details

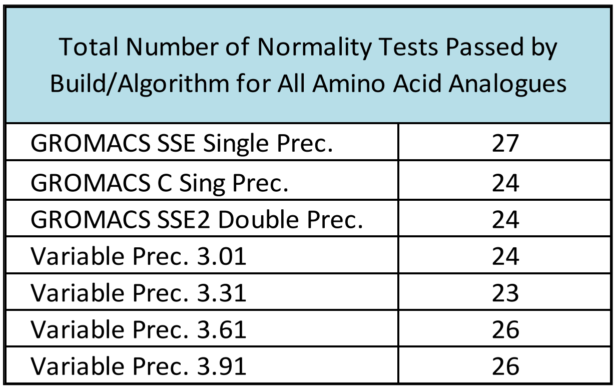
2.1.1 Summary by Algorithm and Amino Acid Analogue

The following tables give credibility that the data for each combination of an amino acid analogue and algorithm is normal. It should be remembered that the data sets are relatively small (40-60). It should also be noted that even with data generated by the R language's function to generate a sample of 'normal' data of arbitrary size the normality tests may fail and the histograms and Q-Q plots may not visually appear normally distributed. The table below shows the results of the six normality tests that were used for all amino acid analogues for all 7 algorithms/builds. It shows that the Shapiro-Wilk and Cramer-von Misses normality tests most frequently indicated that the study data sets are normal. The Pearson Chi-Square indicated normality on the fewest number of data sets.



Number of Algorithms/Builds Passed Per Amino Acid Analogue Per Normality Test							
	Normality Test						
Amino Acid Analogue	Anderson-Darling	Shapiro-Francia	Shapiro-Wilk	Pearson Chi-Square	Lilliefors (Kolmogorov-Smirnov)	Cramer-von Mises	Average Passed
Acetamide	7	6	7	6	6	7	6.5
Ethanol	5	5	5	5	5	6	5.2
Isobutane	6	6	6	6	6	6	6.0
Methane	5	6	6	3	3	5	4.7
Methanol	7	7	7	5	7	7	6.7
Average	6.0	6.0	6.2	5.0	5.4	6.2	

The chart to the right shows the results for each of the algorithms/builds with a count of the number of normality tests for all amino acid analogues. This reveals that for each algorithm/build approximately the same number of normality tests passed further lending credence that the data distribution for the study is normal. Only the GROMACS C single precision methane data failed all normality tests.



Total Number of Normality Tests Passed by Build/Algorithm for All Amino Acid Analogues	
GROMACS SSE Single Prec.	27
GROMACS C Sing Prec.	24
GROMACS SSE2 Double Prec.	24
Variable Prec. 3.01	24
Variable Prec. 3.31	23
Variable Prec. 3.61	26
Variable Prec. 3.91	26

The data sets representing the zone of equivalence builds and that for all data points including all builds and variable precision algorithms show fewer normality tests pass. An examination of the histograms show that there usually is a tighter clustering around the mean but in some cases there is a slight skewing that is similar to the skewing in the individual results from the GROMACS builds.

2.1.2 Acetamide

	Normality Tests							
	Anderson-Darling		Shapiro-Francia		Shapiro-Wilk		Pearson Chi-Square	
Amino Acid Analogue	A	p-value	W		W	p-value	P	p-value
Acetamide								
GROMACS SSE	0.4352	0.2872	0.9503	0.0479	0.9573	0.0901	7.04	0.4244
GROMACS C Sing Prec	0.4749	0.2294	0.9707	0.2499	0.9763	0.4637	7.48	0.3808
GROMACS SSE2 Double	0.6202	0.1001	0.9690	0.2172	0.9660	0.1958	20.52	0.0045
Variable Prec. 3.01	0.2580	0.7033	0.9886	0.8664	0.9831	0.7325	10.52	0.1609
Variable Prec. 3.31	0.3741	0.4022	0.9819	0.5935	0.9835	0.7520	8.78	0.2686
Variable Prec. 3.61	0.3588	0.4369	0.9805	0.5364	0.9823	0.7010	10.96	0.1405
Variable Prec. 3.91	0.4648	0.2430	0.9733	0.3088	0.9662	0.1986	6.61	0.4707
Num Passed Normality Test		7		6		7		6
Zone of Equivalence	0.8125	0.0348	0.9807	0.0467	0.9842	0.1132	23.30	0.0253
All Data	0.5541	0.1519	0.9929	0.1202	0.9943	0.2660	39.04	0.0028

	Normality Test				Wilcoxon Signed Rank Test Nonparametric		Num Norm Tests Passed / Algorithm
	Lilliefors (Kolmogorov-Smirnov)		Cramer-von Mises				
Amino Acid Analogue	D	p-value	W	p-value	V	p-value	
Acetamide							
GROMACS SSE	0.1377	0.0287	0.0653	0.3158	0	3.59E-09	4
GROMACS C Sing Prec	0.1218	0.0849	0.0841	0.1787	0	3.58E-09	6
GROMACS SSE2 Double	0.1165	0.1217	0.1121	0.0738	0	3.55E-09	5
Variable Prec. 3.01	0.0822	0.6091	0.0436	0.6071	0	3.55E-09	6
Variable Prec. 3.31	0.0986	0.3174	0.0672	0.2980	0	3.56E-09	6
Variable Prec. 3.61	0.0868	0.5207	0.0534	0.4537	0	5.73E-10	6
Variable Prec. 3.91	0.1067	0.2109	0.0716	0.2607	0	3.58E-09	6
Num Passed Normality Test		6		7			
Zone of Equivalence	0.0934	0.0049	0.1642	0.0152	0	2.20E-16	1
All Data	0.0519	0.0365	0.0950	0.1312	0	2.20E-16	4

2.1.3 Ethanol

	Normality Tests							
	Anderson-Darling		Shapiro-Francia		Shapiro-Wilk		Pearson Chi-Square	
Amino Acid Analogue	A	p-value	W		W	p-value	P	p-value
Ethanol								
GROMACS SSE	0.2579	0.7072	0.9897	0.8208	0.9872	0.7822	5.63	0.6882
GROMACS C Sing Prec	0.3036	0.5619	0.9898	0.8267	0.9858	0.7096	13.70	0.0899
GROMACS SSE2 Double	0.6713	0.0757	0.9764	0.2522	0.9723	0.1891	11.50	0.1749
Variable Prec. 3.01	0.7648	0.0442	0.9544	0.0269	0.9540	0.0240	20.67	0.0081
Variable Prec. 3.31	0.2134	0.8455	0.9917	0.9093	0.9888	0.8562	4.90	0.7682
Variable Prec. 3.61	0.5563	0.1448	0.9563	0.0321	0.9577	0.0365	5.63	0.6882
Variable Prec. 3.91	0.7627	0.0447	0.9612	0.0524	0.9644	0.0770	29.10	0.0003
Num Passed Normality Test		5		5		5		5
Zone of Equivalence	0.5876	0.1239	0.9922	0.3774	0.9899	0.2341	46.67	0.0000
All Data	0.5586	0.1483	0.9957	0.2683	0.9961	0.3824	51.72	0.0001

	Normality Test				Wilcoxon Signed Rank Test Nonparametric		Num Norm Tests Passed / Algorithm
	Lilliefors (Kolmogorov-Smirnov)		Cramer-von Mises				
Amino Acid Analogue	D	p-value	W	p-value	V	p-value	
Ethanol							
GROMACS SSE	0.0710	0.6372	0.0394	0.6862	0	1.65E-11	6
GROMACS C Sing Prec	0.0813	0.4164	0.0529	0.4625	0	1.65E-11	6
GROMACS SSE2 Double	0.1295	0.0139	0.1177	0.0625	0	1.64E-11	5
Variable Prec. 3.01	0.0951	0.1946	0.1139	0.0705	0	1.65E-11	2
Variable Prec. 3.31	0.0748	0.5528	0.0339	0.7848	0	1.63E-11	6
Variable Prec. 3.61	0.0981	0.1615	0.0870	0.1648	0	1.64E-11	4
Variable Prec. 3.91	0.1351	0.0083	0.1485	0.0240	0	1.60E-11	2
Num Passed Normality Test		5		6			
Zone of Equivalence	0.0773	0.0106	0.1021	0.1043	0	2.20E-16	4
All Data	0.0444	0.0462	0.0986	0.1172	0	2.20E-16	4

2.1.4 Isobutane

	Normality Tests							
	Anderson-Darling		Shapiro-Francia		Shapiro-Wilk		Pearson Chi-Square	
Amino Acid Analogue	A	p-value	W		W	p-value	P	p-value
Isobutane								
GROMACS SSE	0.4145	0.3224	0.9681	0.2027	0.9685	0.2436	14.43	0.0440
GROMACS C Sing Prec	0.4356	0.2866	0.9740	0.3268	0.9654	0.1849	7.04	0.4244
GROMACS SSE2 Double	0.3147	0.5325	0.9814	0.5732	0.9741	0.3890	6.61	0.4707
Variable Prec. 3.01	0.2961	0.5789	0.9840	0.6777	0.9796	0.5908	6.17	0.5196
Variable Prec. 3.31	0.8848	0.0217	0.9490	0.0432	0.9475	0.0376	10.96	0.1405
Variable Prec. 3.61	0.6975	0.0641	0.9580	0.0887	0.9511	0.0518	10.09	0.1837
Variable Prec. 3.91	0.6182	0.1013	0.9538	0.0632	0.9528	0.0602	8.78	0.2686
Num Passed Normality Test		6		6		6		6
Zone of Equivalence	0.8250	0.0324	0.9788	0.0309	0.9755	0.0137	30.26	0.0026
All Data	0.9708	0.0144	0.9835	0.0014	0.9824	0.0005	53.39	2.28E-05

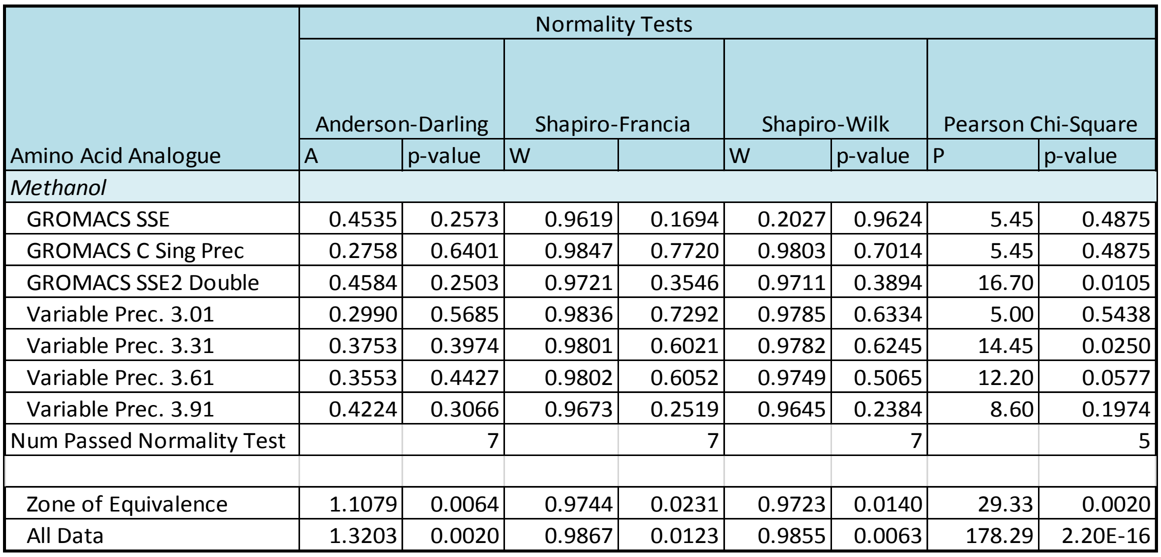
	Normality Test				Wilcoxon Signed Rank Test Nonparametric		Num Norm Tests Passed / Algorithm
	Lilliefors (Kolmogorov-Smirnov)		Cramer-von Mises				
Amino Acid Analogue	D	p-value	W	p-value	V	p-value	
Isobutane							
GROMACS SSE	0.0997	0.3015	0.0608	0.3621	0	3.56E-09	5
GROMACS C Sing Prec	0.0773	0.7027	0.0569	0.4085	0	3.60E-09	6
GROMACS SSE2 Double	0.0870	0.5165	0.0487	0.5208	0	3.51E-09	6
Variable Prec. 3.01	0.1049	0.2314	0.0482	0.5286	0	3.58E-09	6
Variable Prec. 3.31	0.1727	0.0015	0.1740	0.0109	0	3.59E-09	1
Variable Prec. 3.61	0.0904	0.4545	0.0867	0.1651	0	3.55E-09	6
Variable Prec. 3.91	0.1194	0.0980	0.0944	0.1300	0	3.52E-09	6
Num Passed Normality Test		6		6			
Zone of Equivalence	0.0780	0.0388	0.1170	0.0650	0	2.20E-16	1
All Data	0.0662	0.0017	0.1299	0.0438	0	2.20E-16	0

2.1.5 Methane

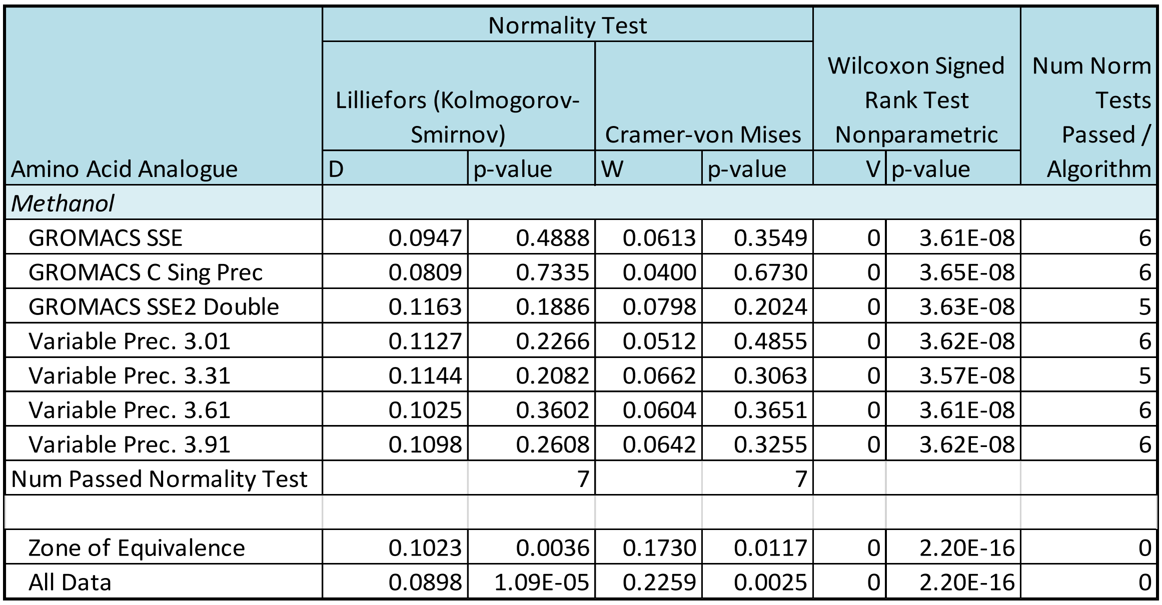
	Normality Tests							
	Anderson-Darling		Shapiro-Francia		Shapiro-Wilk		Pearson Chi-Square	
Amino Acid Analogue	A	p-value	W		W	p-value	P	p-value
Methane								
GROMACS SSE	0.3226	0.5160	0.9776	0.5138	0.9757	0.5347	4.55	0.6027
GROMACS C Sing Prec	1.4219	0.0010	0.9107	0.0055	0.9029	0.0023	13.10	0.0415
GROMACS SSE2 Double	0.7646	0.0429	0.9530	0.0887	0.9495	0.0727	19.85	0.0029
Variable Prec. 3.01	0.6847	0.0682	0.9561	0.1114	0.9565	0.1273	15.80	0.0149
Variable Prec. 3.31	0.6178	0.1006	0.9646	0.2070	0.9634	0.2186	10.40	0.1088
Variable Prec. 3.61	0.5067	0.1896	0.9650	0.2124	0.9686	0.3239	15.35	0.0177
Variable Prec. 3.91	0.5543	0.1432	0.9528	0.0877	0.9584	0.1471	5.45	0.4875
Num Passed Normality Test		5		6		6		3
Zone of Equivalence	0.8250	0.0324	0.9788	0.0309	0.9755	0.0137	30.26	0.0026
All Data	0.9708	0.0144	0.9835	0.0014	0.9824	0.0005	53.39	2.28E-05

	Normality Test				Wilcoxon Signed Rank Test Nonparametric		Num Norm Tests Passed / Algorithm
	Lilliefors (Kolmogorov-Smirnov)		Cramer-von Mises				
Amino Acid Analogue	D	p-value	W	p-value	V	p-value	
Methane							
GROMACS SSE	0.1017	0.3731	0.0510	0.4873	0	3.58E-08	6
GROMACS C Sing Prec	0.2000	0.0003	0.2179	0.0030	0	3.52E-08	0
GROMACS SSE2 Double	0.1285	0.0942	0.1289	0.0433	0	3.59E-08	3
Variable Prec. 3.01	0.1467	0.0299	0.1143	0.0687	0	3.54E-08	4
Variable Prec. 3.31	0.1432	0.0379	0.1055	0.0906	0	3.54E-08	5
Variable Prec. 3.61	0.1601	0.0113	0.0922	0.1387	0	3.50E-08	4
Variable Prec. 3.91	0.1249	0.1200	0.0936	0.1329	0	3.52E-08	6
Num Passed Normality Test		3		5			
Zone of Equivalence	0.0780	0.0388	0.1170	0.0650	0	2.20E-16	1
All Data	0.0662	0.0017	0.1299	0.0438	0	2.20E-16	0

2.1.6 Methanol



	Normality Tests							
	Anderson-Darling		Shapiro-Francia		Shapiro-Wilk		Pearson Chi-Square	
Amino Acid Analogue	A	p-value	W		W	p-value	P	p-value
Methanol								
GROMACS SSE	0.4535	0.2573	0.9619	0.1694	0.2027	0.9624	5.45	0.4875
GROMACS C Sing Prec	0.2758	0.6401	0.9847	0.7720	0.9803	0.7014	5.45	0.4875
GROMACS SSE2 Double	0.4584	0.2503	0.9721	0.3546	0.9711	0.3894	16.70	0.0105
Variable Prec. 3.01	0.2990	0.5685	0.9836	0.7292	0.9785	0.6334	5.00	0.5438
Variable Prec. 3.31	0.3753	0.3974	0.9801	0.6021	0.9782	0.6245	14.45	0.0250
Variable Prec. 3.61	0.3553	0.4427	0.9802	0.6052	0.9749	0.5065	12.20	0.0577
Variable Prec. 3.91	0.4224	0.3066	0.9673	0.2519	0.9645	0.2384	8.60	0.1974
Num Passed Normality Test		7		7		7		5
Zone of Equivalence	1.1079	0.0064	0.9744	0.0231	0.9723	0.0140	29.33	0.0020
All Data	1.3203	0.0020	0.9867	0.0123	0.9855	0.0063	178.29	2.20E-16



	Normality Test				Wilcoxon Signed Rank Test Nonparametric		Num Norm Tests Passed / Algorithm
	Lilliefors (Kolmogorov-Smirnov)		Cramer-von Mises				
Amino Acid Analogue	D	p-value	W	p-value	V	p-value	
Methanol							
GROMACS SSE	0.0947	0.4888	0.0613	0.3549	0	3.61E-08	6
GROMACS C Sing Prec	0.0809	0.7335	0.0400	0.6730	0	3.65E-08	6
GROMACS SSE2 Double	0.1163	0.1886	0.0798	0.2024	0	3.63E-08	5
Variable Prec. 3.01	0.1127	0.2266	0.0512	0.4855	0	3.62E-08	6
Variable Prec. 3.31	0.1144	0.2082	0.0662	0.3063	0	3.57E-08	5
Variable Prec. 3.61	0.1025	0.3602	0.0604	0.3651	0	3.61E-08	6
Variable Prec. 3.91	0.1098	0.2608	0.0642	0.3255	0	3.62E-08	6
Num Passed Normality Test		7		7			
Zone of Equivalence	0.1023	0.0036	0.1730	0.0117	0	2.20E-16	0
All Data	0.0898	1.09E-05	0.2259	0.0025	0	2.20E-16	0

3.1 Descriptive Statistics

3.1.1 Acetamide in Water

Descriptive Statistics

	Double Prec SSE2	Single Prec SSE	Single Prec C	Var Prec 3.01	Var Prec 3.31	Var Prec 3.61	Var Prec 3.91
Mean	-3.924	-3.921	-3.917	-3.936	-3.912	-3.934	-3.925
Std. Dev.	.068	.079	.076	.066	.063	.054	.075
Std. Error	.010	.012	.011	.010	.009	.008	.011
Count	46	46	46	46	46	46	46
Minimum	-4.050	-4.190	-4.100	-4.070	-4.050	-4.060	-4.060
Maximum	-3.760	-3.770	-3.710	-3.800	-3.750	-3.800	-3.770
Variance	.005	.006	.006	.004	.004	.003	.006
Coef. Var.	-.017	-.020	-.019	-.017	-.016	-.014	-.019
Range	.290	.420	.390	.270	.300	.260	.290
Sum	-180.500	-180.380	-180.200	-181.050	-179.960	-180.950	-180.570
Sum Squares	708.475	707.604	706.172	712.786	704.212	711.934	709.070
Skew ness	.097	-.750	-.033	.065	.049	.180	.302
Kurtosis	-.089	1.499	.593	-.670	-.131	-.222	-.386
Median	-3.920	-3.920	-3.915	-3.940	-3.900	-3.940	-3.935
IQR	.080	.090	.080	.090	.090	.090	.090
Mode	-3.920	-3.960	-3.920	-3.890	-3.870	-3.980	•
10% Tr. Mean	-3.925	-3.917	-3.917	-3.937	-3.912	-3.934	-3.927
MAD	.030	.040	.040	.050	.040	.040	.045

3.1.2 Ethanol in Water

Descriptive Statistics

	Double SSE2	Single SSE	Single Std C	Var Prec 3.01	Var Prec 3.31	Var Prec 3.61	Var Prec 3.91
Mean	-9.462	-9.425	-9.442	-9.428	-9.443	-9.440	-9.436
Std. Dev.	.074	.075	.088	.080	.072	.079	.080
Std. Error	.010	.010	.011	.010	.009	.010	.010
Count	60	60	60	60	60	60	60
Minimum	-9.620	-9.590	-9.630	-9.570	-9.610	-9.700	-9.600
Maximum	-9.300	-9.250	-9.250	-9.200	-9.290	-9.310	-9.190
Variance	.005	.006	.008	.006	.005	.006	.006
Coef. Var.	-.008	-.008	-.009	-.008	-.008	-.008	-.008
Range	.320	.340	.380	.370	.320	.390	.410
Sum	-567.720	-565.530	-566.530	-565.660	-566.580	-566.380	-566.180
Sum Squares	5372.090	5330.739	5349.730	5333.231	5350.521	5346.804	5343.040
Skew ness	-.045	.221	.005	.782	-.134	-.767	.499
Kurtosis	-.779	-.350	-.620	.527	-.329	.966	.950
Median	-9.460	-9.430	-9.440	-9.435	-9.440	-9.435	-9.440
IQR	.120	.105	.135	.110	.085	.090	.085
Mode	•	-9.440	-9.520	-9.490	-9.400	-9.420	-9.440
10% Tr. Mean	-9.461	-9.428	-9.443	-9.434	-9.442	-9.434	-9.439
MAD	.060	.050	.075	.055	.040	.045	.040

3.1.3 Isobutane in Water

Descriptive Statistics

	Double SSE2	Single SSE	Single Std C	Var Prec 3.01	Var Prec 3.31	Var Prec 3.61	Var Prec 3.91
Mean	-9.887	-9.890	-9.866	-9.881	-9.872	-9.876	-9.879
Std. Dev.	.060	.062	.080	.067	.081	.067	.064
Std. Error	.009	.009	.012	.010	.012	.010	.009
Count	46	46	46	46	46	46	46
Minimum	-10.000	-10.000	-10.000	-10.000	-10.000	-10.000	-9.990
Maximum	-9.770	-9.710	-9.700	-9.720	-9.640	-9.740	-9.710
Variance	.004	.004	.006	.004	.007	.004	.004
Coef. Var.	-.006	-.006	-.008	-.007	-.008	-.007	-.006
Range	.230	.290	.300	.280	.360	.260	.280
Sum	-454.780	-454.960	-453.840	-454.510	-454.130	-454.290	-454.450
Sum Squares	4496.352	4499.926	4477.912	4491.055	4483.646	4486.711	4489.851
Skewness	.089	.310	.095	.156	.705	-.379	.670
Kurtosis	-.459	.118	-.770	-.492	-.008	-.547	.532
Median	-9.880	-9.880	-9.865	-9.885	-9.890	-9.875	-9.880
IQR	.080	.090	.120	.100	.110	.090	.080
Mode	-9.880	-9.840	•	-9.910	•	-9.890	-9.880
10% Tr. Mean	-9.888	-9.893	-9.868	-9.881	-9.877	-9.873	-9.883
MAD	.040	.040	.060	.045	.055	.045	.040

3.1.4 Methane in Water

Descriptive Statistics

	Double SSE2	Single SSE	Single Std C	Var Prec 3.01	Var Prec 3.31	Var Prec 3.61	Var Prec 3.91
Mean	-9.010	-9.017	-9.008	-9.005	-9.019	-9.019	-9.023
Std. Dev.	.040	.047	.041	.045	.036	.039	.035
Std. Error	.006	.007	.006	.007	.006	.006	.006
Count	40	40	40	40	40	40	40
Minimum	-9.080	-9.110	-9.070	-9.120	-9.090	-9.100	-9.090
Maximum	-8.910	-8.910	-8.920	-8.920	-8.940	-8.910	-8.910
Variance	.002	.002	.002	.002	.001	.002	.001
Coef. Var.	-.004	-.005	-.005	-.005	-.004	-.004	-.004
Range	.170	.200	.150	.200	.150	.190	.180
Sum	-360.410	-360.680	-360.330	-360.190	-360.780	-360.770	-360.930
Sum Squares	3247.447	3252.337	3246.008	3243.500	3254.106	3253.934	3256.811
Skewness	.373	.219	.768	-.568	-.329	.548	.497
Kurtosis	-.658	.018	-.295	.187	-.289	.315	1.310
Median	-9.020	-9.020	-9.020	-9.000	-9.015	-9.030	-9.020
IQR	.065	.060	.040	.060	.045	.050	.040
Mode	-8.980	-9.030	-9.020	-9.000	-9.020	-9.030	-9.010
10% Tr. Mean	-9.012	-9.018	-9.012	-9.002	-9.018	-9.021	-9.024
MAD	.030	.030	.020	.030	.025	.030	.020

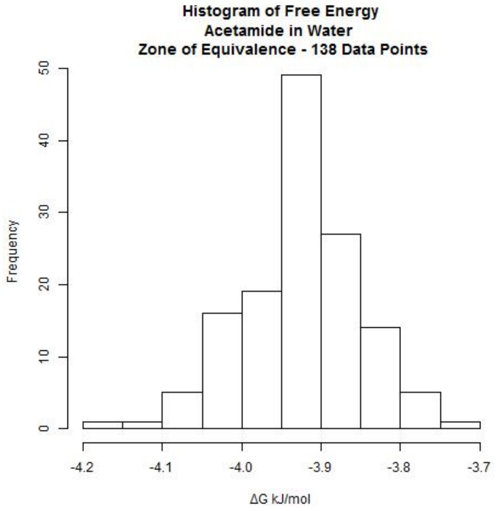
3.1.5 Methanol in Water

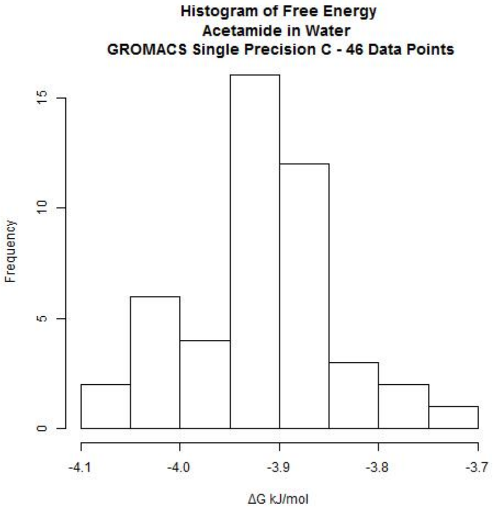
Descriptive Statistics

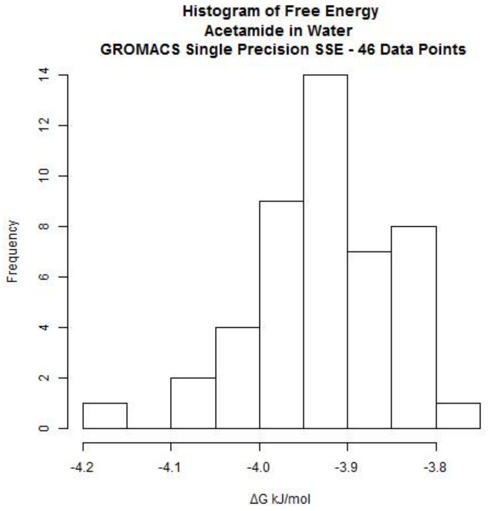
	Double SSE2	Single SSE	Single Std C	Var Prec 3.01	Var Prec 3.31	Var Prec 3.61	Var Prec 3.91
Mean	-4.779	-4.790	-4.791	-4.788	-4.776	-4.781	-4.775
Std. Dev.	.059	.049	.045	.057	.050	.047	.054
Std. Error	.009	.008	.007	.009	.008	.007	.009
Count	40	40	40	40	40	40	40
Minimum	-4.930	-4.920	-4.880	-4.910	-4.890	-4.870	-4.870
Maximum	-4.670	-4.700	-4.690	-4.680	-4.670	-4.680	-4.640
Variance	.003	.002	.002	.003	.003	.002	.003
Coef. Var.	-.012	-.010	-.009	-.012	-.011	-.010	-.011
Range	.260	.220	.190	.230	.220	.190	.230
Sum	-191.170	-191.620	-191.630	-191.530	-191.020	-191.260	-191.020
Sum Squares	913.783	918.048	918.132	917.218	912.315	914.597	912.332
Skewness	-.369	-.595	.168	-.084	-.217	.058	.479
Kurtosis	-.343	.325	-.743	-.386	-.286	-.374	-.168
Median	-4.790	-4.790	-4.795	-4.790	-4.780	-4.780	-4.780
IQR	.080	.060	.065	.075	.070	.055	.085
Mode	-4.800	•	-4.800	-4.790	-4.780	•	•
10% Tr. Mean	-4.776	-4.787	-4.792	-4.788	-4.774	-4.782	-4.778
MAD	.050	.030	.035	.040	.030	.025	.040

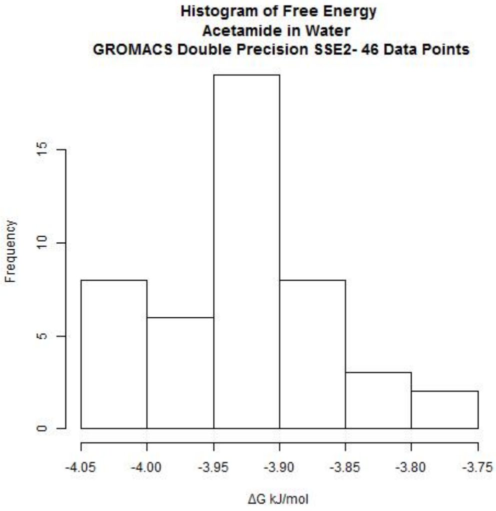
4.1 Histograms

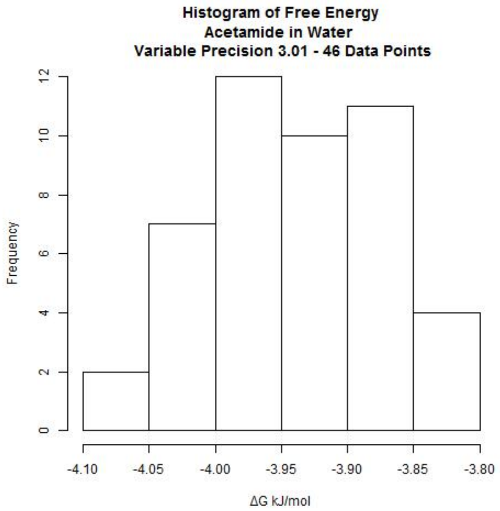
4.1.1 Acetamide in Water

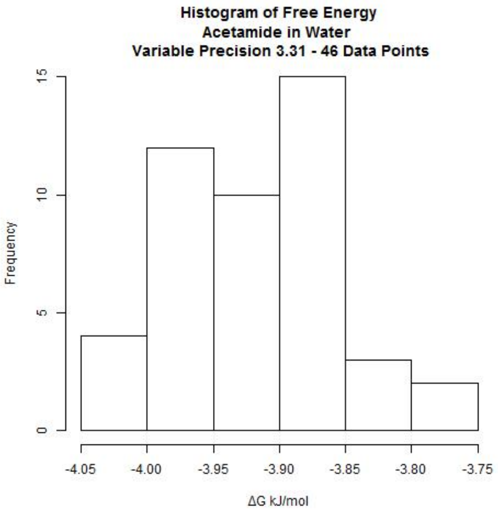


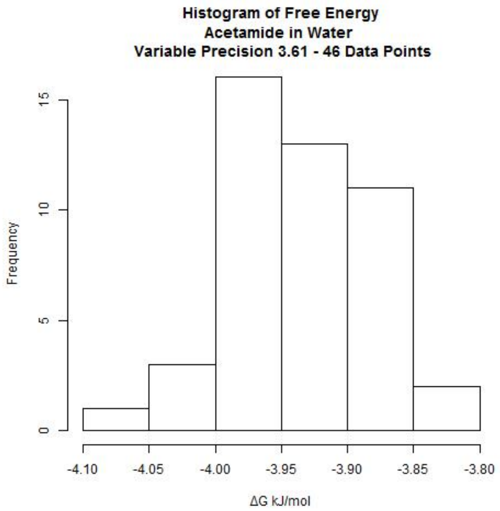


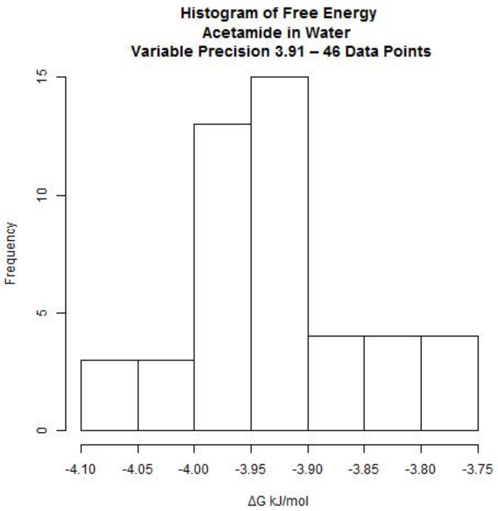




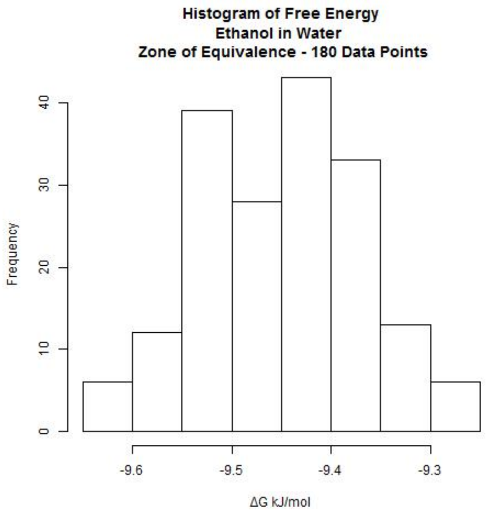

Histogram of Free Energy
Acetamide in Water
Variable Precision 3.01 - 46 Data Points
Frequency
0
2
4
6
8
10
12
-4.10
-4.05
-4.00
-3.95
-3.90
-3.85
-3.80
ΔG kJ/mol

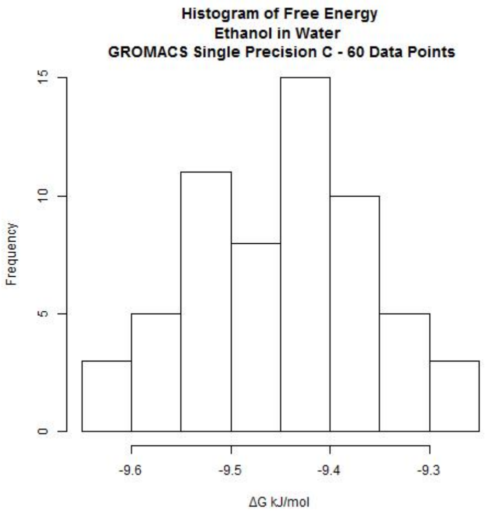

Histogram of Free Energy
Acetamide in Water
Variable Precision 3.31 - 46 Data Points
Frequency
0
5
10
15
-4.05
-4.00
-3.95
-3.90
-3.85
-3.80
-3.75
ΔG kJ/mol

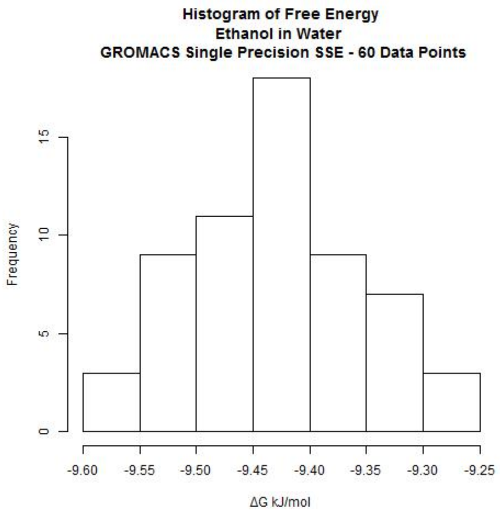

Histogram of Free Energy
Acetamide in Water
Variable Precision 3.61 - 46 Data Points
Frequency
0
5
10
15
-4.10
-4.05
-4.00
-3.95
-3.90
-3.85
-3.80
ΔG kJ/mol

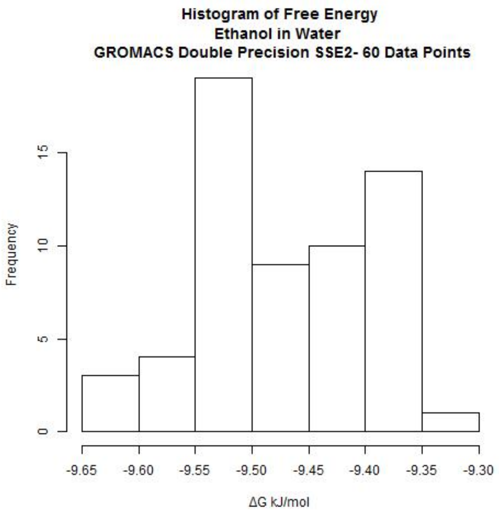

Histogram of Free Energy
Acetamide in Water
Variable Precision 3.91 – 46 Data Points
Frequency
0
5
10
15
-4.10
-4.05
-4.00
-3.95
-3.90
-3.85
-3.80
-3.75
ΔG kJ/mol

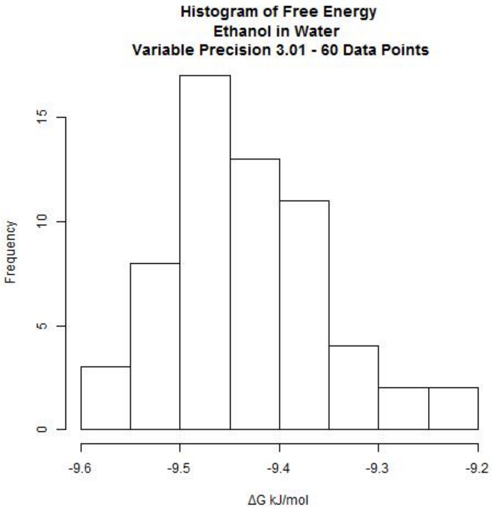
4.1.2 Ethanol in Water

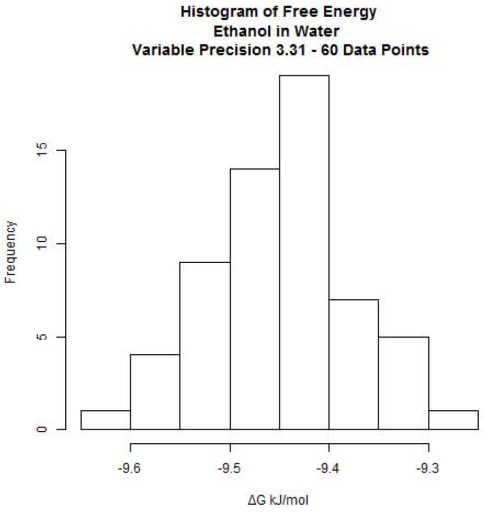


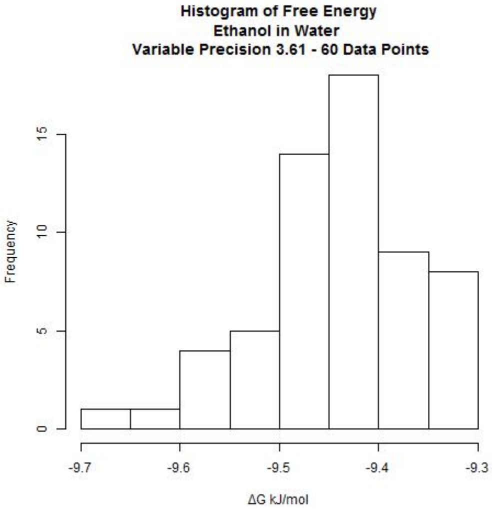


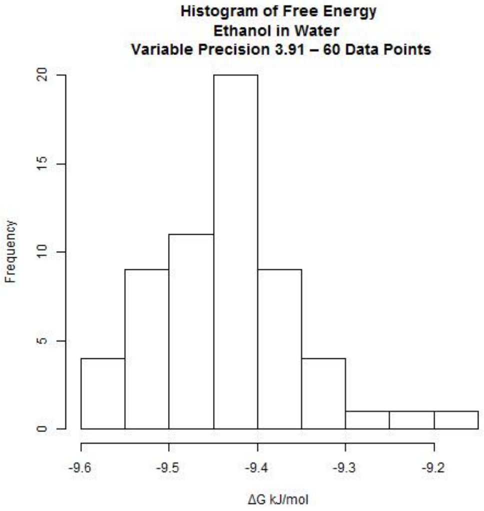




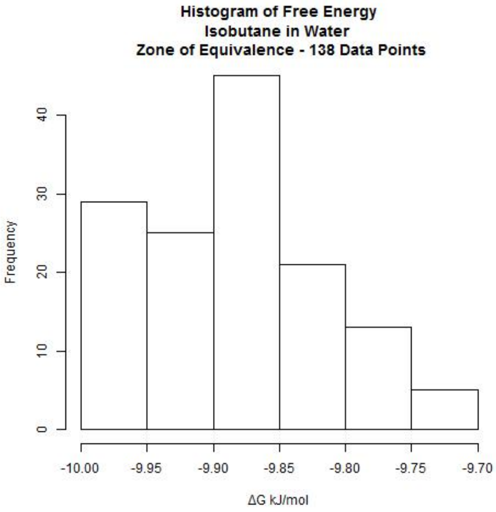

Histogram of Free Energy
Ethanol in Water
Variable Precision 3.01 - 60 Data Points
Frequency
0
5
10
15
-9.6
-9.5
-9.4
-9.3
-9.2
ΔG kJ/mol

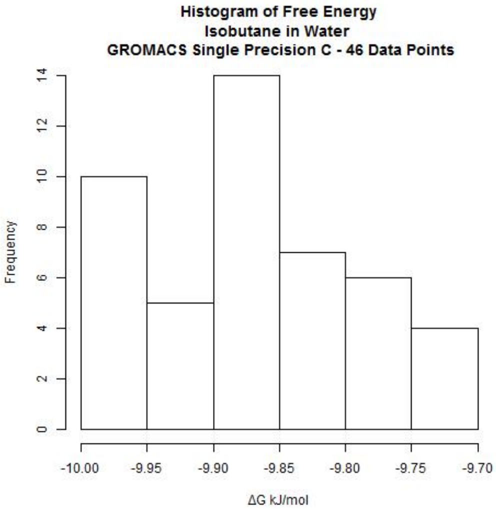

Histogram of Free Energy
Ethanol in Water
Variable Precision 3.31 - 60 Data Points
Frequency
0
5
10
15
-9.6
-9.5
-9.4
-9.3
ΔG kJ/mol

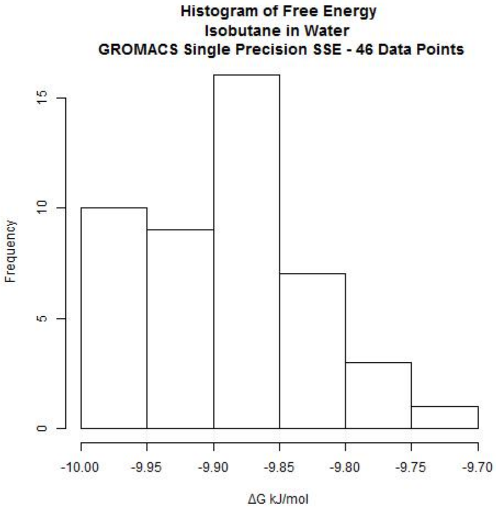

Histogram of Free Energy
Ethanol in Water
Variable Precision 3.61 - 60 Data Points
Frequency
0
5
10
15
-9.7
-9.6
-9.5
-9.4
-9.3
ΔG kJ/mol

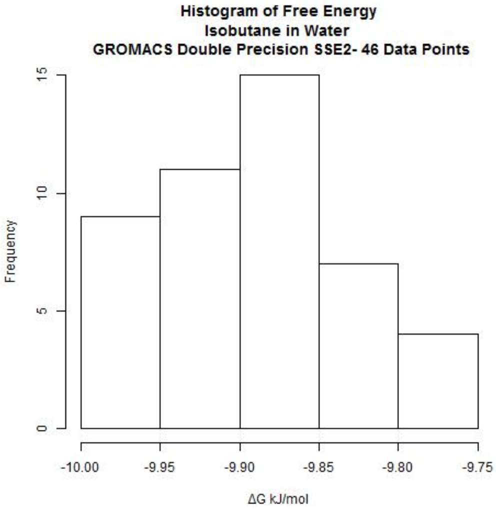

Histogram of Free Energy
Ethanol in Water
Variable Precision 3.91 – 60 Data Points
Frequency
0
5
10
15
20
-9.6
-9.5
-9.4
-9.3
-9.2
ΔG kJ/mol

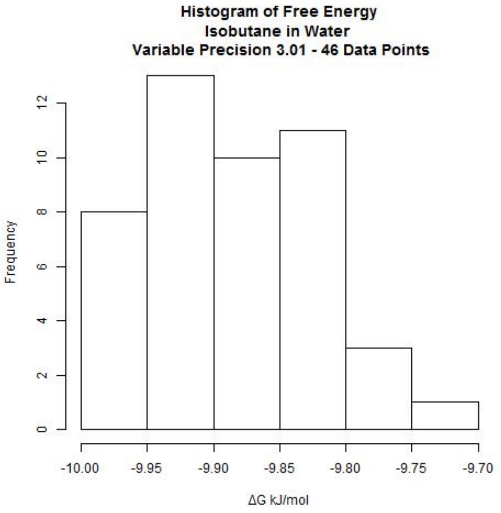
4.1.3 Isobutane in Water

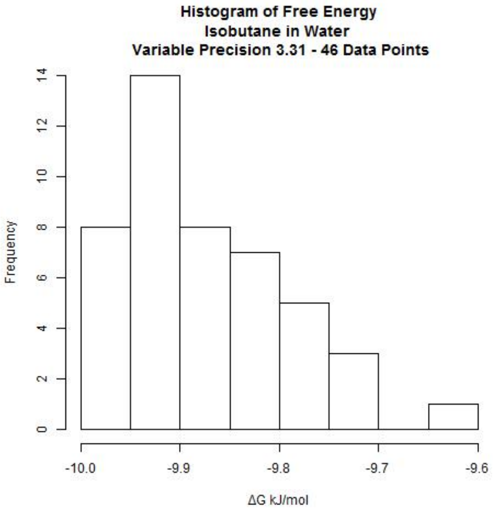


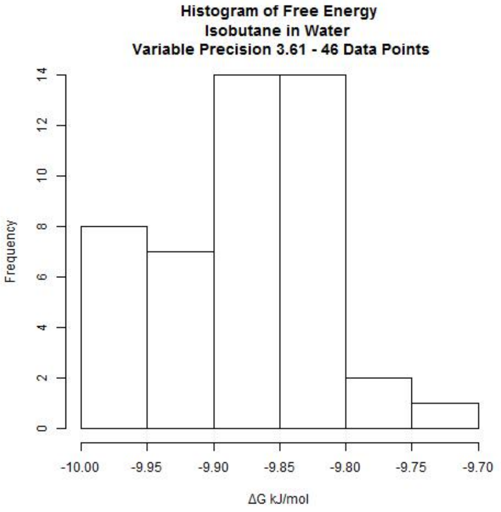


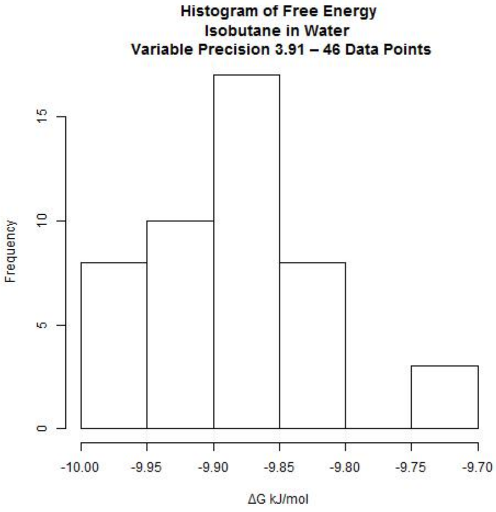




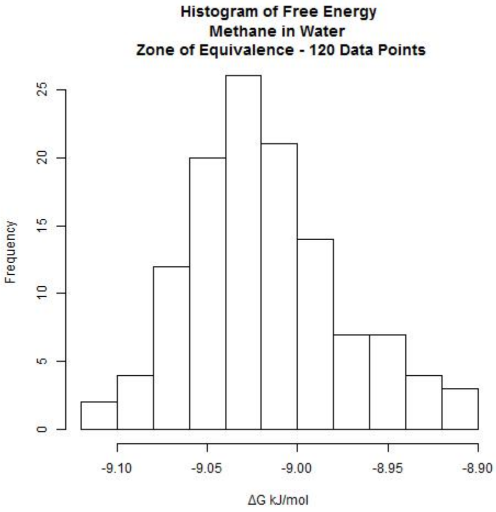

Histogram of Free Energy
Isobutane in Water
Variable Precision 3.01 - 46 Data Points
Frequency
0
2
4
6
8
10
12
-10.00
-9.95
-9.90
-9.85
-9.80
-9.75
-9.70
ΔG kJ/mol

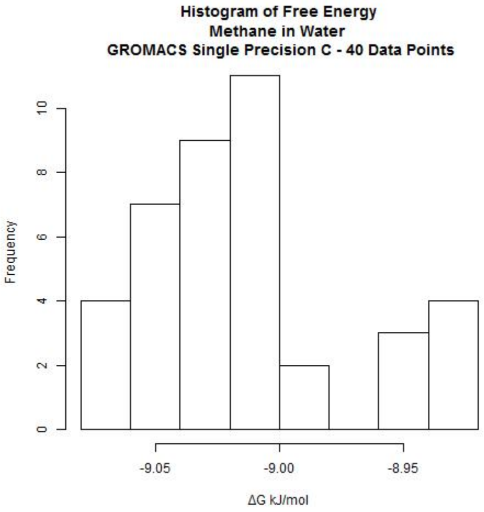

Histogram of Free Energy
Isobutane in Water
Variable Precision 3.31 – 46 Data Points
Frequency
0
2
4
6
8
10
12
14
-10.0
-9.9
-9.8
-9.7
-9.6
ΔG kJ/mol

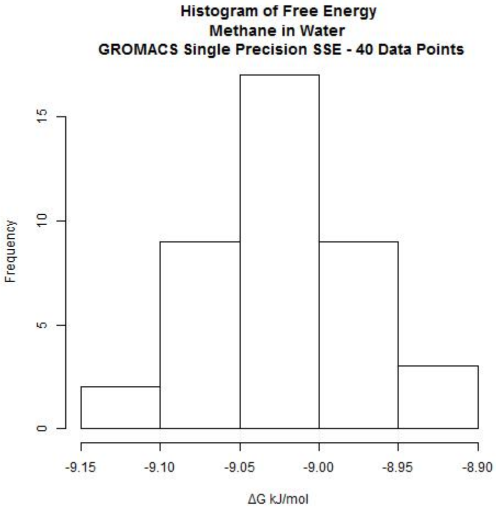

Histogram of Free Energy
Isobutane in Water
Variable Precision 3.61 - 46 Data Points
Frequency
0
2
4
6
8
10
12
14
-10.00
-9.95
-9.90
-9.85
-9.80
-9.75
-9.70
ΔG kJ/mol

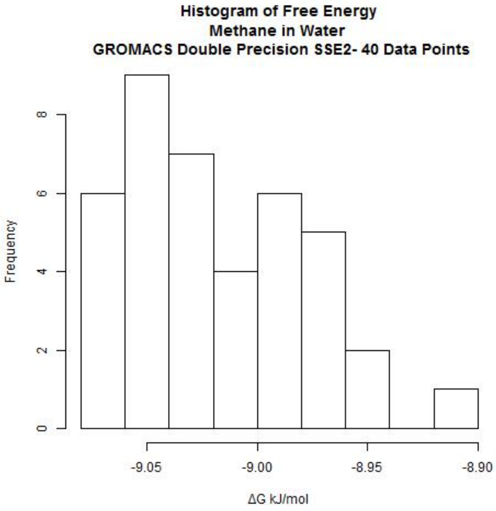

Histogram of Free Energy
Isobutane in Water
Variable Precision 3.91 – 46 Data Points
Frequency
0
5
10
15
-10.00
-9.95
-9.90
-9.85
-9.80
-9.75
-9.70
ΔG kJ/mol

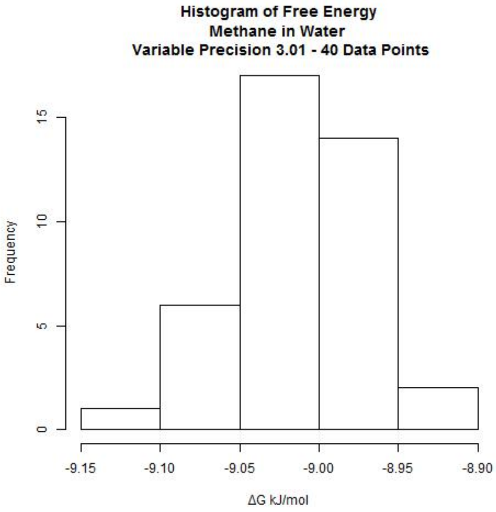
4.1.4 Methane in Water

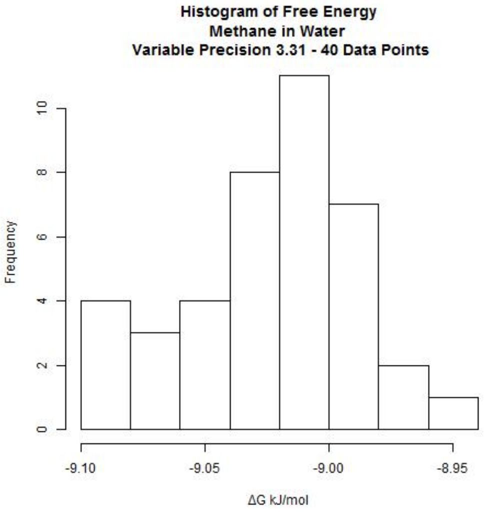


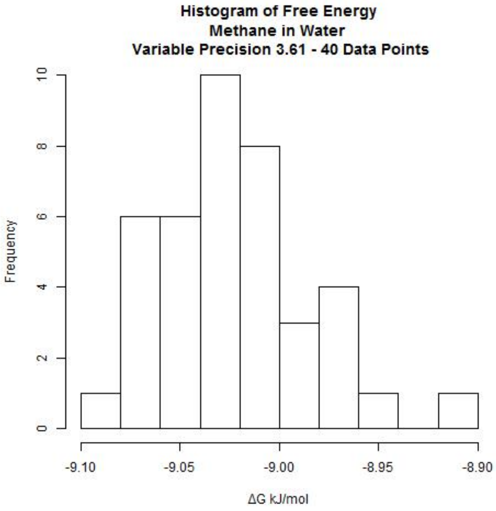


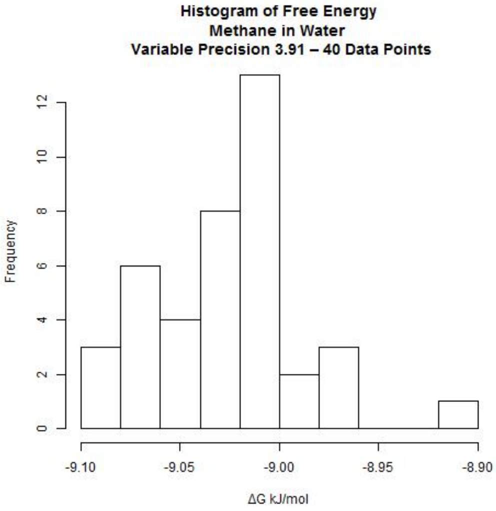




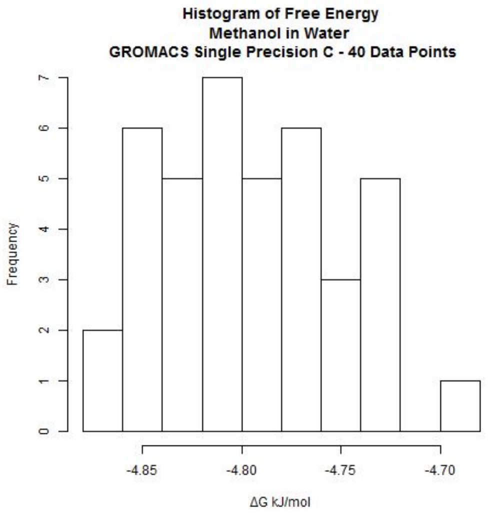

Histogram of Free Energy
Methane in Water
Variable Precision 3.01 - 40 Data Points
Frequency
0
5
10
15
-9.15
-9.10
-9.05
-9.00
-8.95
-8.90
ΔG kJ/mol

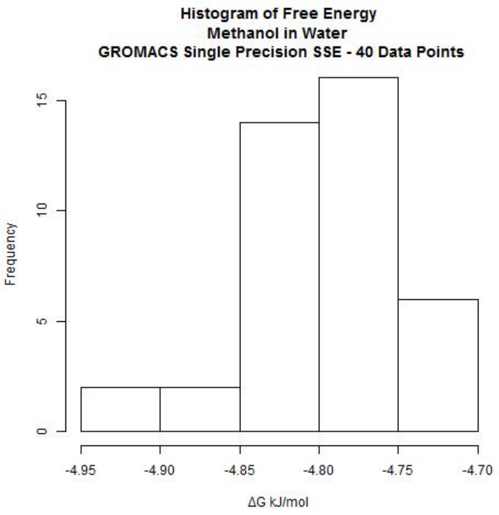

Histogram of Free Energy
Methane in Water
Variable Precision 3.31 - 40 Data Points
Frequency
0
2
4
6
8
10
-9.10
-9.05
-9.00
-8.95
ΔG kJ/mol

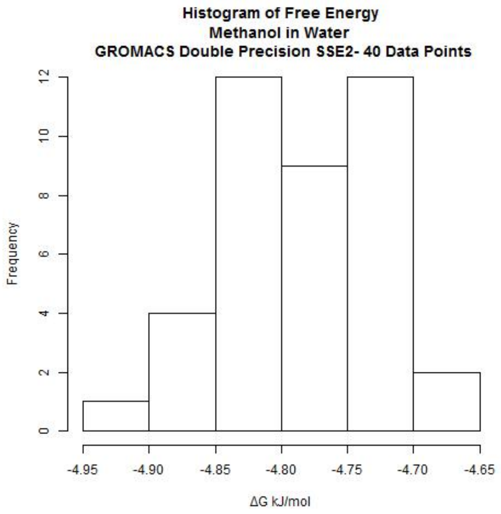

Histogram of Free Energy
Methane in Water
Variable Precision 3.61 - 40 Data Points
Frequency
0
2
4
6
8
10
-9.10
-9.05
-9.00
-8.95
-8.90
ΔG kJ/mol

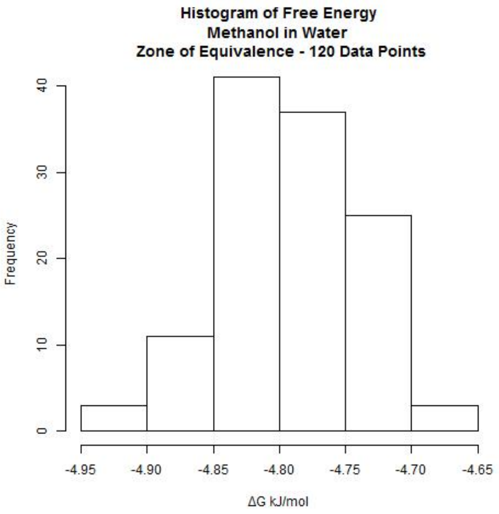

Histogram of Free Energy
Methane in Water
Variable Precision 3.91 – 40 Data Points
Frequency
0
2
4
6
8
10
12
-9.10
-9.05
-9.00
-8.95
-8.90
ΔG kJ/mol

4.1.5 Methanol in Water

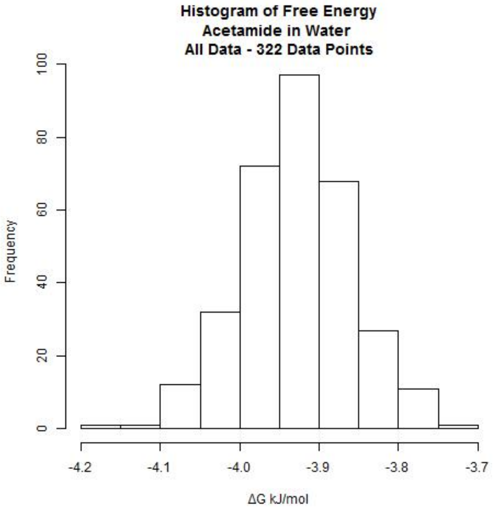


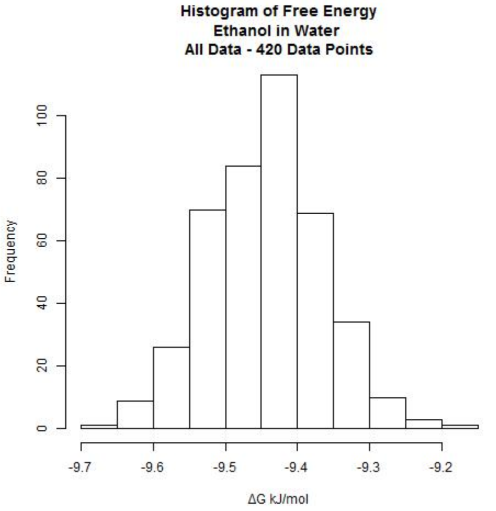


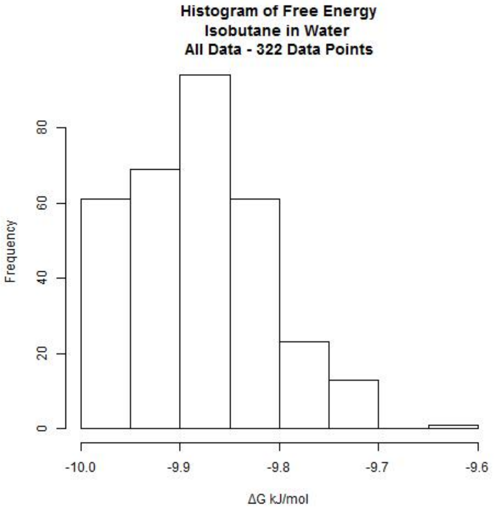


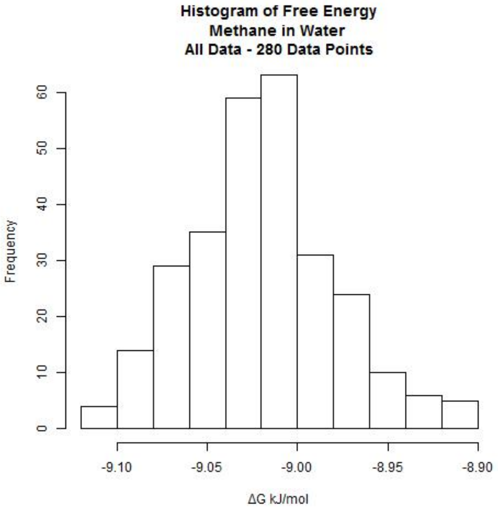


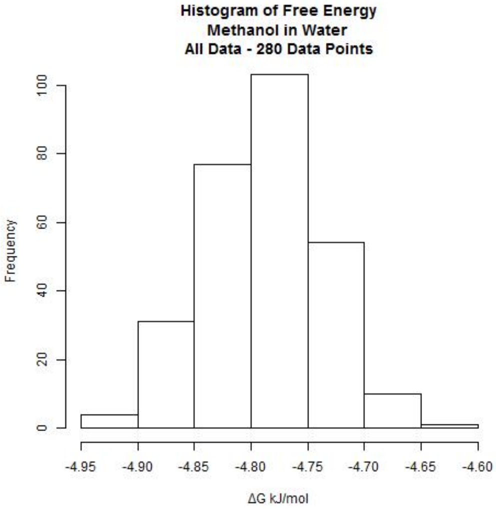
4.1.6 Histograms of All Data





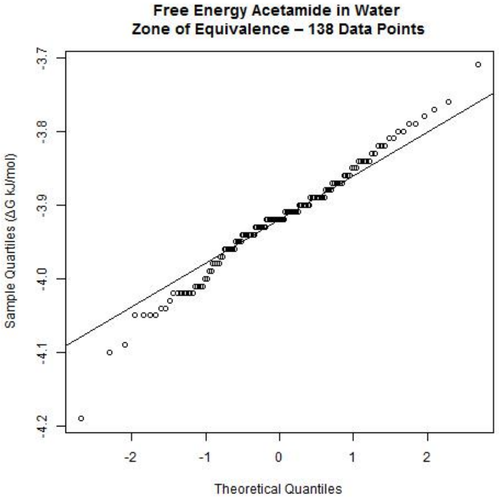


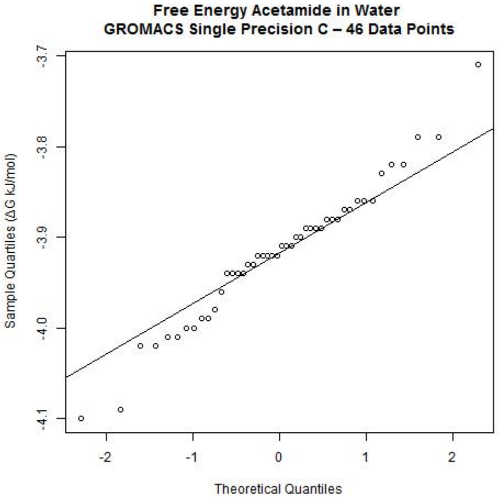


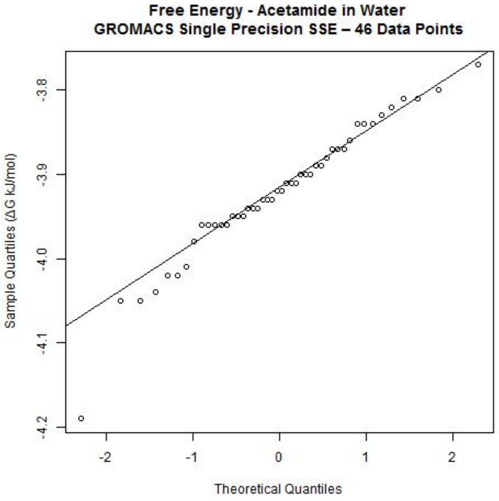

Histogram of Free Energy
Methanol in Water
All Data - 280 Data Points
Frequency
0
20
40
60
80
100
-4.95
-4.90
-4.85
-4.80
-4.75
-4.70
-4.65
-4.60
ΔG kJ/mol

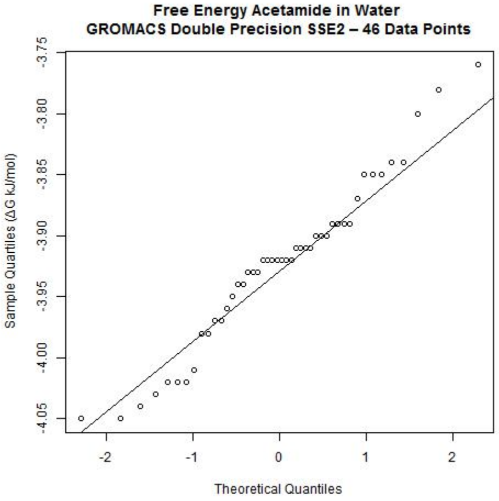
5.1 Quantile to Quantile Plots

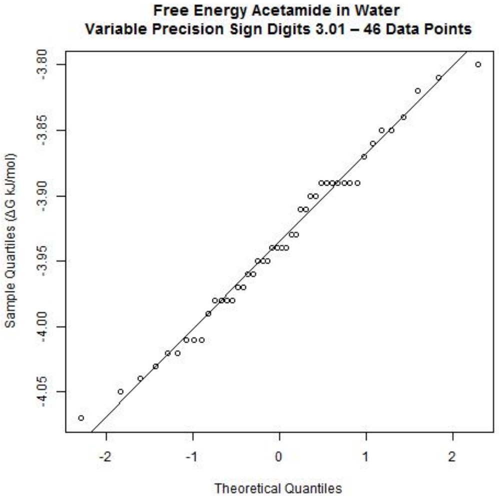
5.1.1 Acetamide in Water

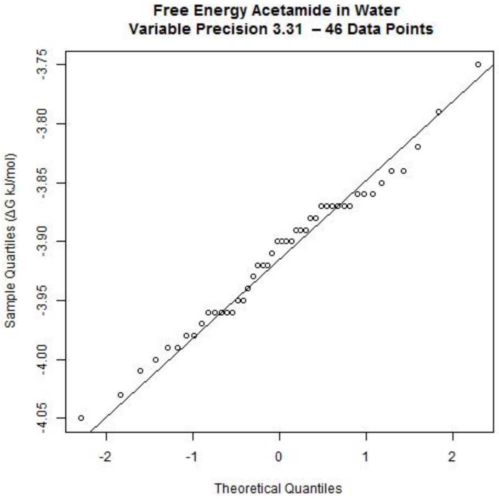


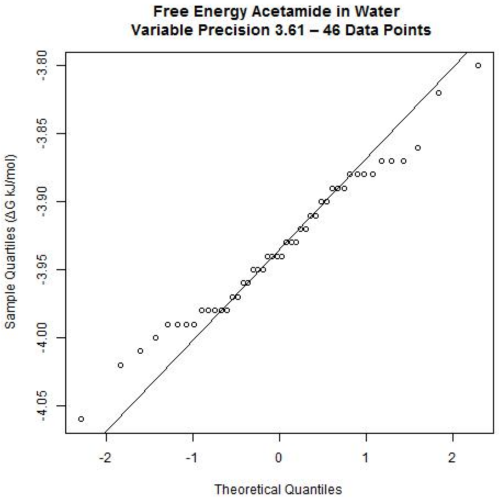


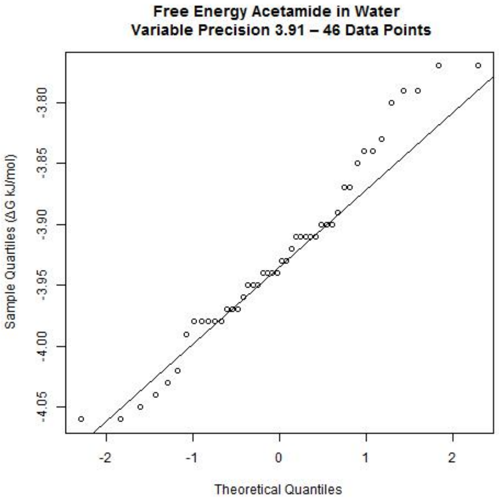




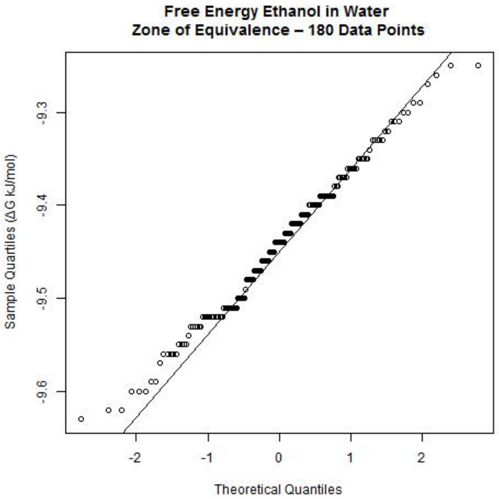

Free Energy Acetamide in Water
Variable Precision Sign Digits 3.01 – 46 Data Points
Sample Quartiles (ΔG kJ/mol)
-3.80
-3.85
-3.90
-3.95
-4.00
-4.05
Theoretical Quantiles
-2
-1
0
1
2

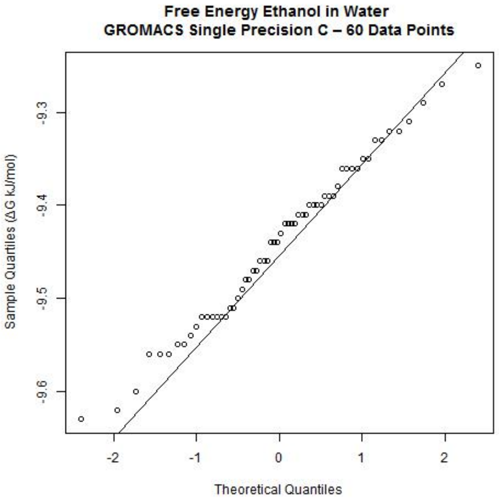

Free Energy Acetamide in Water
Variable Precision 3.31 – 46 Data Points
Sample Quartiles (ΔG kJ/mol)
-3.75
-3.80
-3.85
-3.90
-3.95
-4.00
-4.05
Theoretical Quantiles
-2
-1
0
1
2

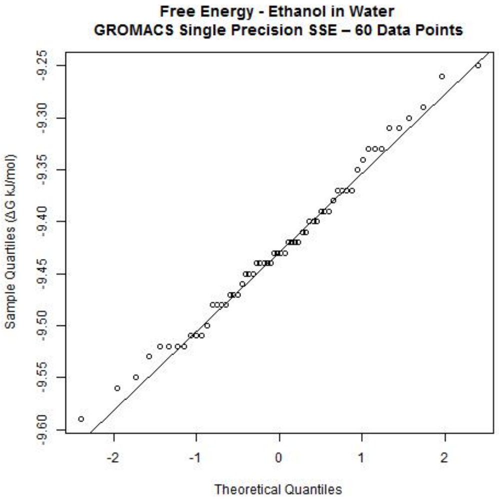

Free Energy Acetamide in Water
Variable Precision 3.61 – 46 Data Points
Sample Quartiles (ΔG kJ/mol)
-3.80
-3.85
-3.90
-3.95
-4.00
-4.05
Theoretical Quantiles
-2
-1
0
1
2

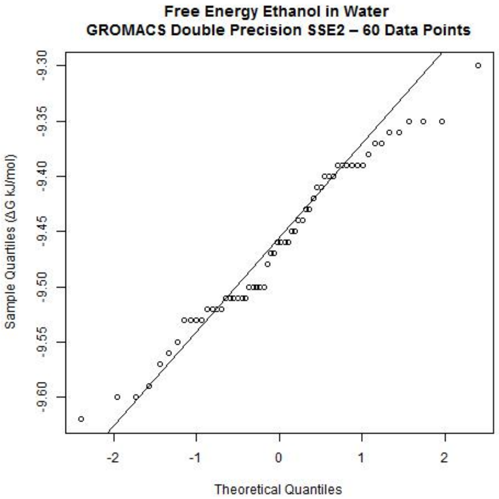

Free Energy Acetamide in Water
Variable Precision 3.91 – 46 Data Points
Sample Quartiles (ΔG kJ/mol)
-3.80
-3.85
-3.90
-3.95
-4.00
-4.05
Theoretical Quantiles
-2
-1
0
1
2

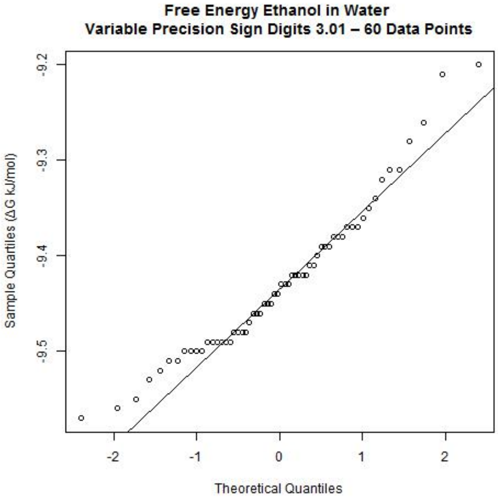
5.1.2 Ethanol in Water

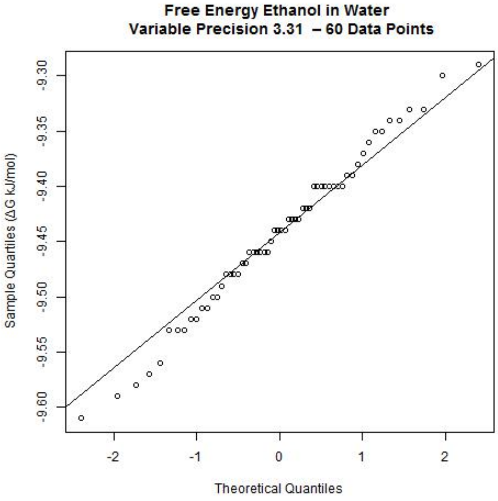


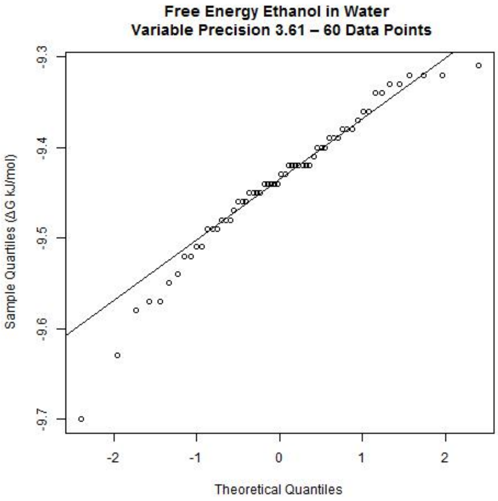


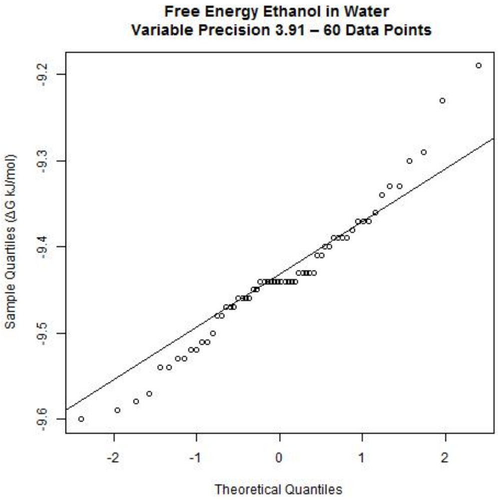




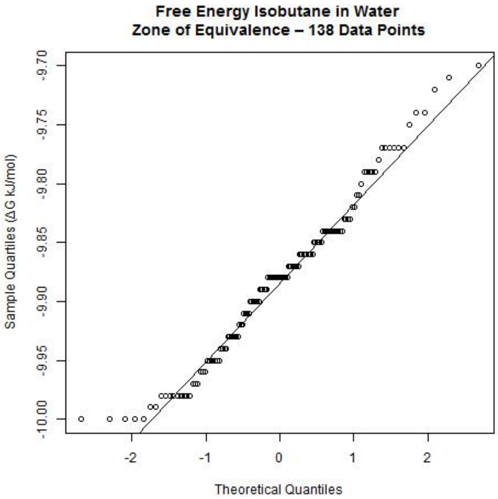

Free Energy Ethanol in Water
Variable Precision Sign Digits 3.01 – 60 Data Points

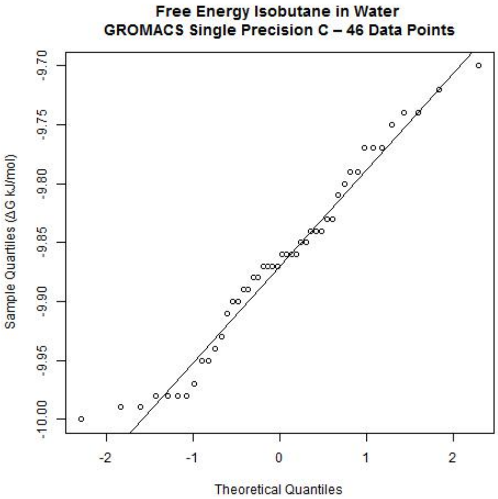

Free Energy Ethanol in Water
Variable Precision 3.31 – 60 Data Points

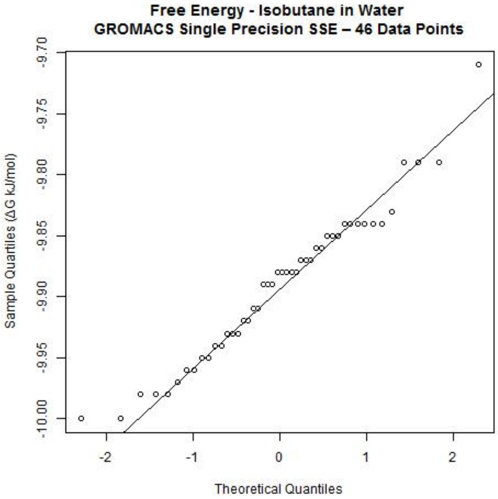

Free Energy Ethanol in Water
Variable Precision 3.61 – 60 Data Points

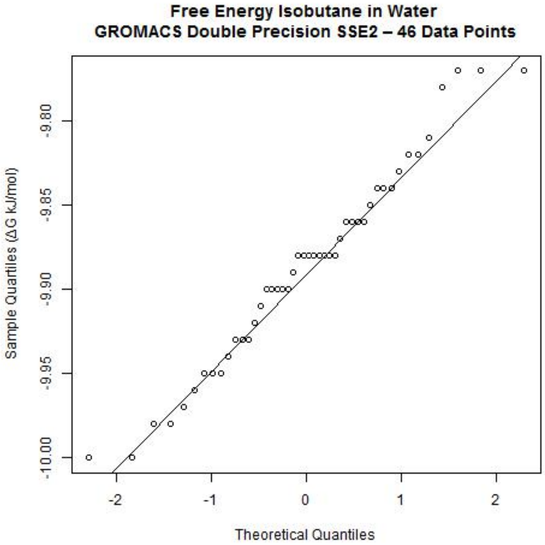

Free Energy Ethanol in Water
Variable Precision 3.91 – 60 Data Points

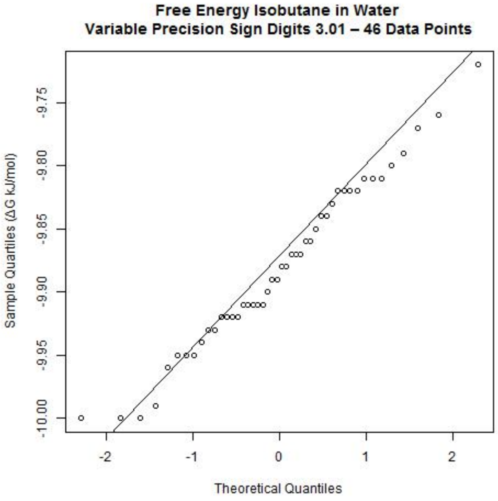
5.1.3 Isobutane in Water



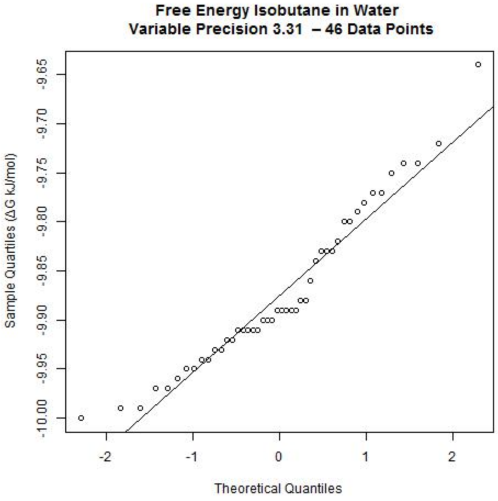




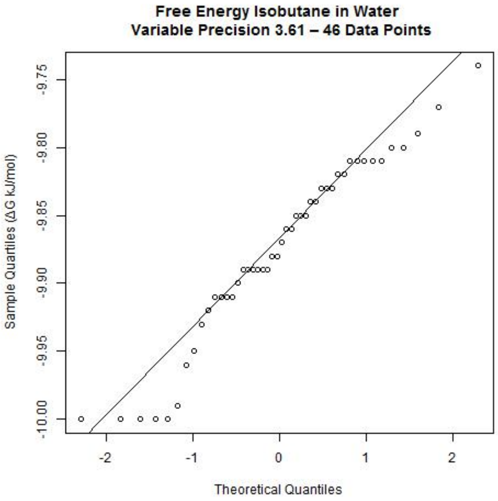




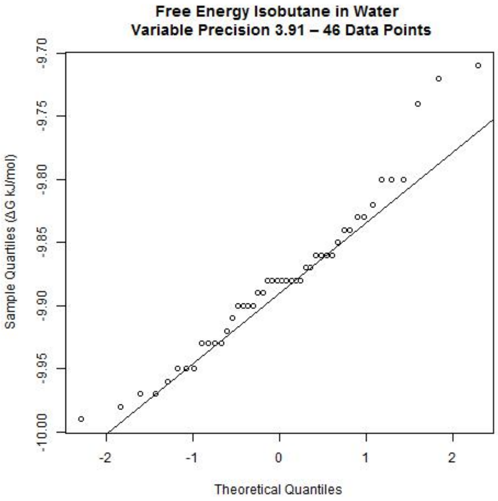
Free Energy Isobutane in Water
Variable Precision Sign Digits 3.01 – 46 Data Points



Free Energy Isobutane in Water
Variable Precision 3.31 – 46 Data Points

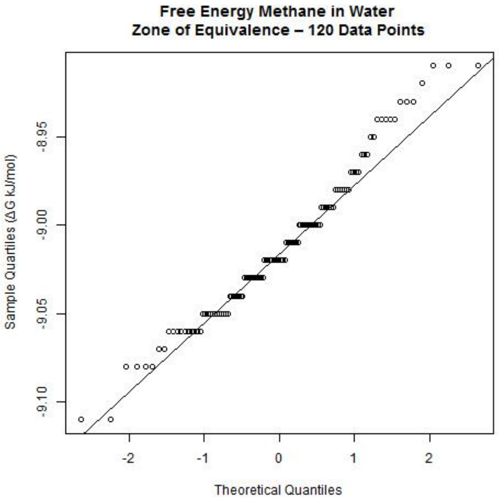


Free Energy Isobutane in Water
Variable Precision 3.61 – 46 Data Points

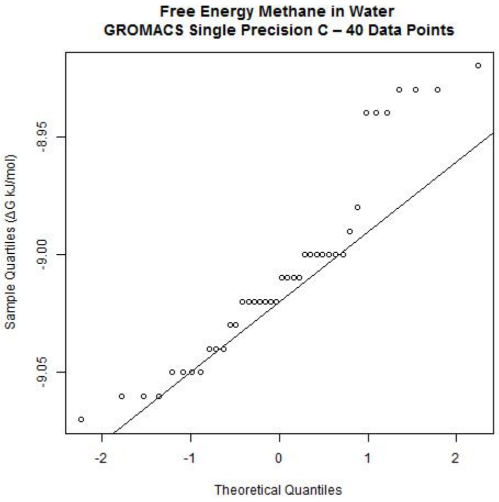


Free Energy Isobutane in Water
Variable Precision 3.91 – 46 Data Points

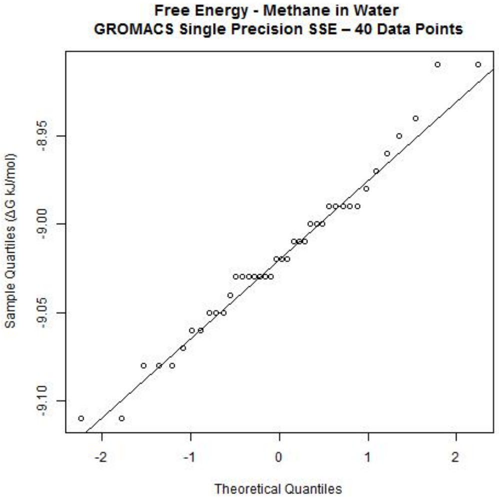
5.1.4 Methane in Water



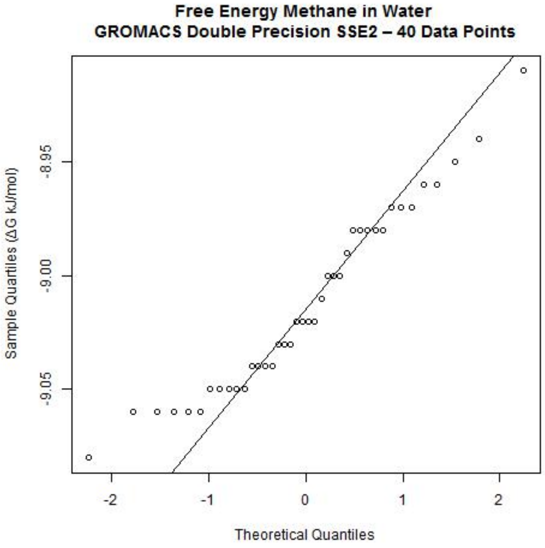
Free Energy Methane in Water
Zone of Equivalence – 120 Data Points



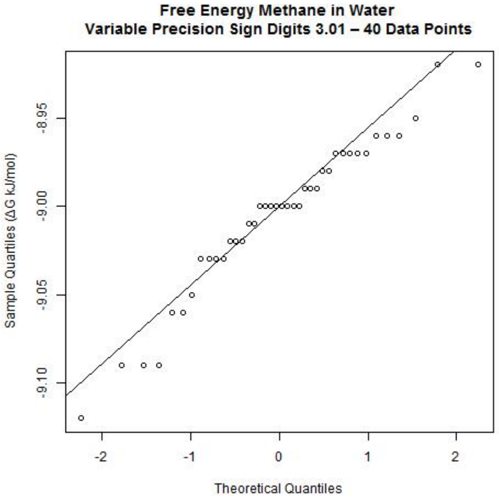
Free Energy Methane in Water
GROMACS Single Precision C – 40 Data Points

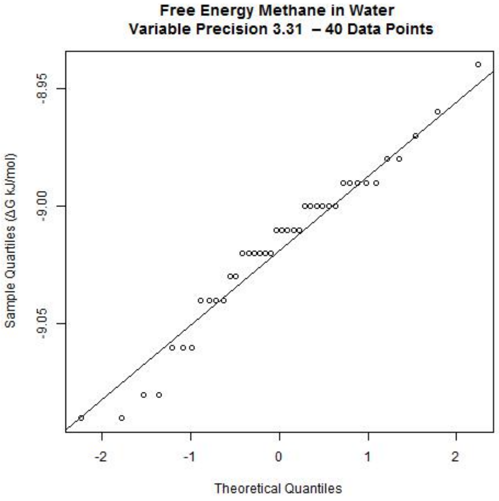


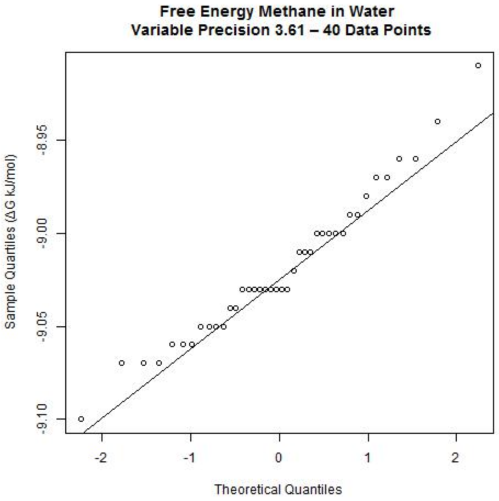
Free Energy - Methane in Water
GROMACS Single Precision SSE – 40 Data Points

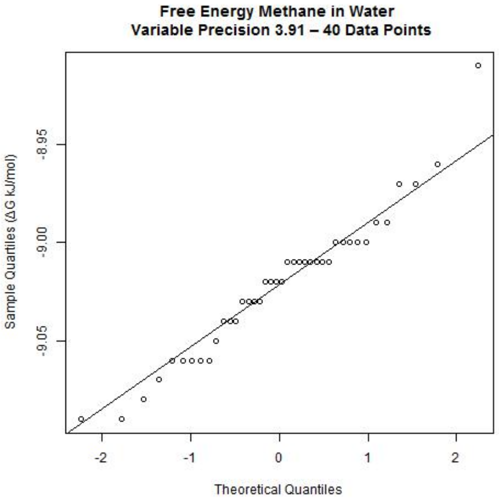


Free Energy Methane in Water
GROMACS Double Precision SSE2 – 40 Data Points

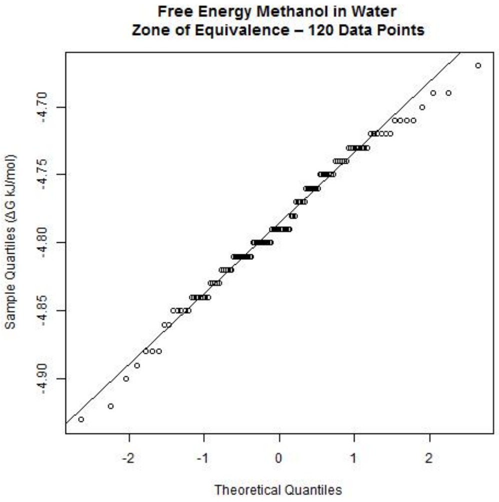

Free Energy Methane in Water
Variable Precision Sign Digits 3.01 – 40 Data Points

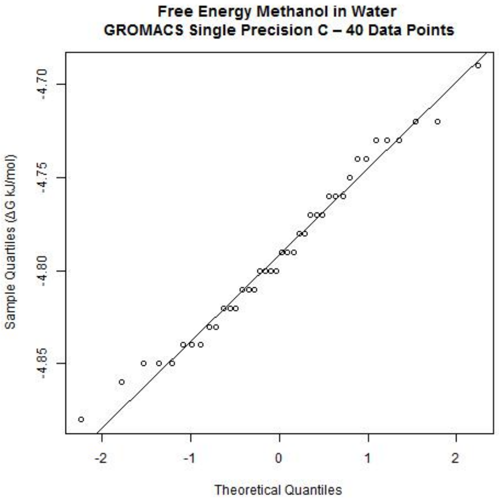

Free Energy Methane in Water
Variable Precision 3.31 – 40 Data Points

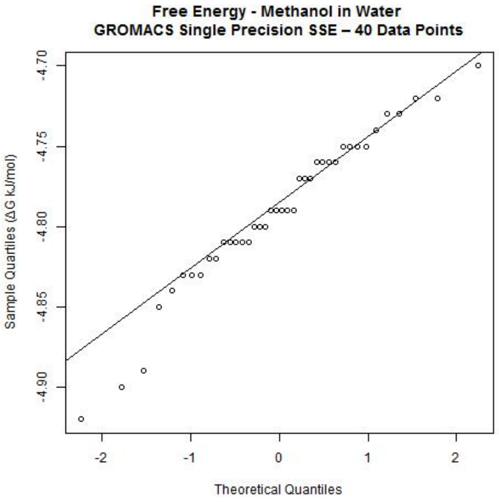

Free Energy Methane in Water
Variable Precision 3.61 – 40 Data Points

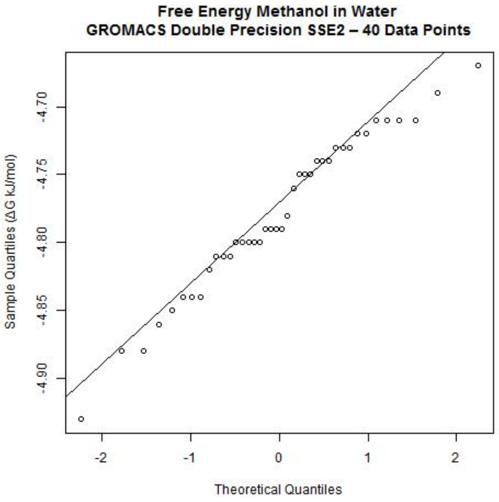

Free Energy Methane in Water
Variable Precision 3.91 – 40 Data Points

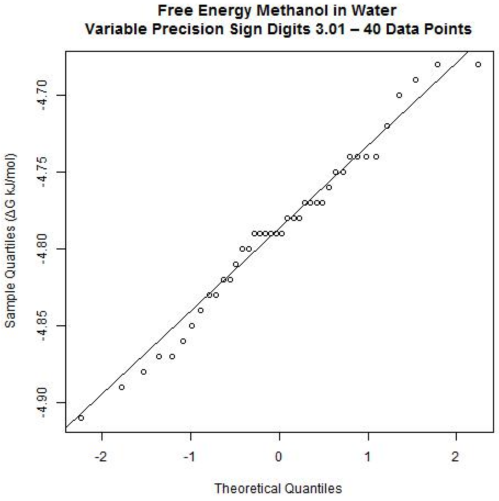
5.1.5 Methanol in Water

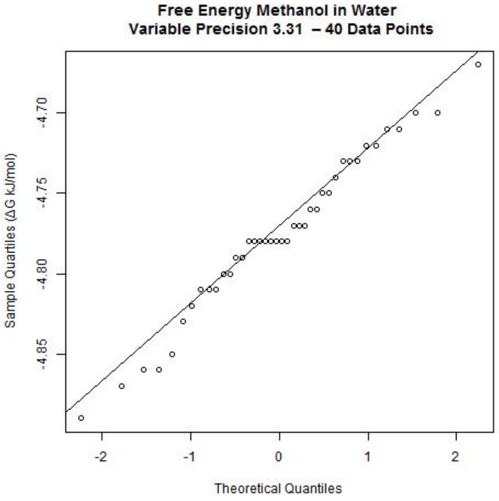

Free Energy Methanol in Water
Zone of Equivalence – 120 Data Points

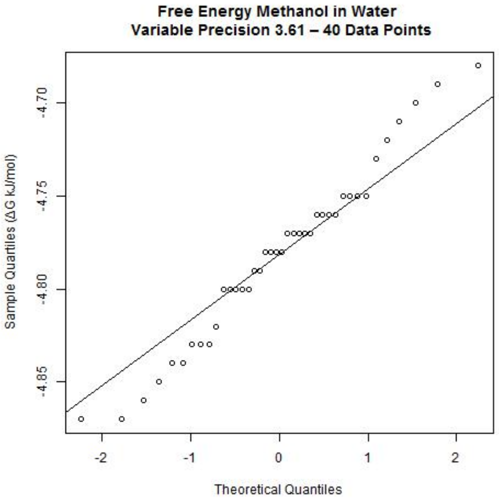

Free Energy Methanol in Water
GROMACS Single Precision C – 40 Data Points

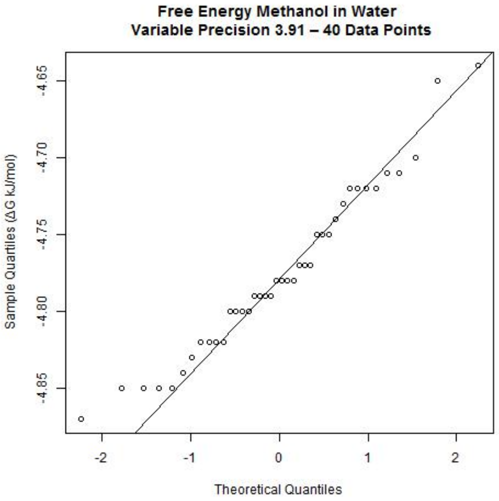

Free Energy - Methanol in Water
GROMACS Single Precision SSE – 40 Data Points


Free Energy Methanol in Water
GROMACS Double Precision SSE2 – 40 Data Points

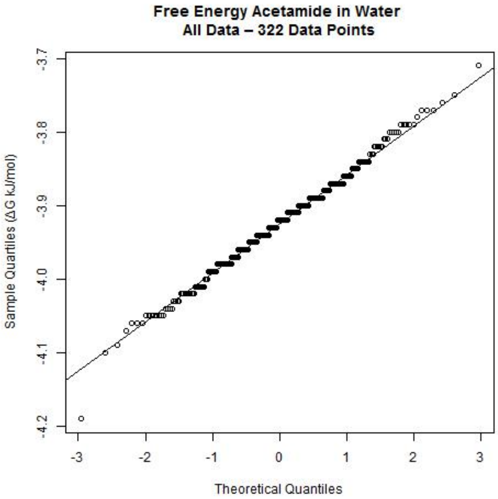

Free Energy Methanol in Water
Variable Precision Sign Digits 3.01 – 40 Data Points
Sample Quartiles (ΔG kJ/mol)
-4.70
-4.75
-4.80
-4.85
-4.90
-2
-1
0
1
2
Theoretical Quantiles

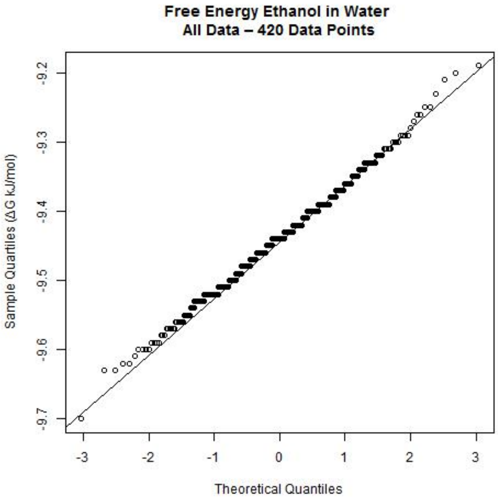

Free Energy Methanol in Water
Variable Precision 3.31 – 40 Data Points
Sample Quartiles (ΔG kJ/mol)
-4.70
-4.75
-4.80
-4.85
-2
-1
0
1
2
Theoretical Quantiles

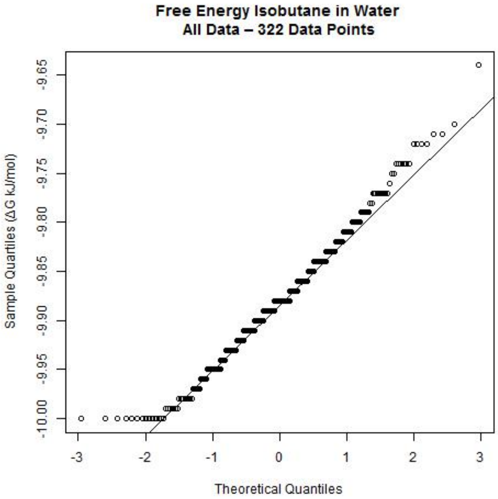

Free Energy Methanol in Water
Variable Precision 3.61 – 40 Data Points
Sample Quartiles (ΔG kJ/mol)
-4.70
-4.75
-4.80
-4.85
-2
-1
0
1
2
Theoretical Quantiles

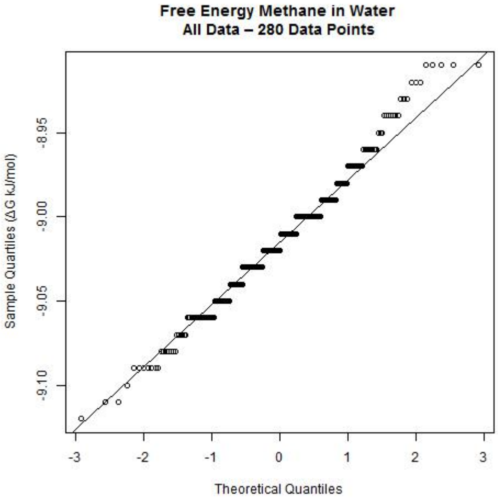

Free Energy Methanol in Water
Variable Precision 3.91 – 40 Data Points
Sample Quartiles (ΔG kJ/mol)
-4.65
-4.70
-4.75
-4.80
-4.85
-2
-1
0
1
2
Theoretical Quantiles

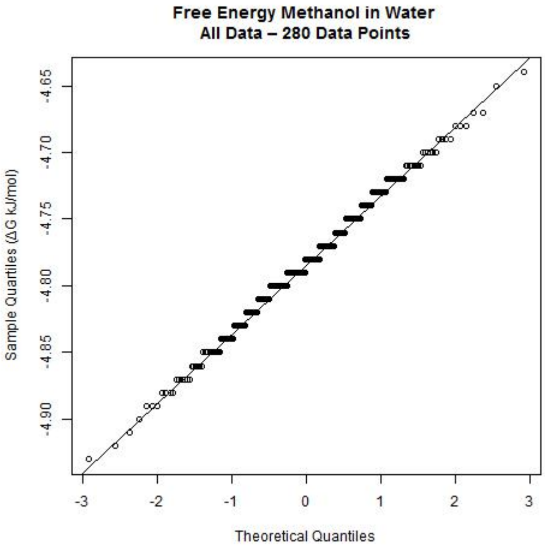
5.1.6 Q-Q Plots of All Data










Free Energy Methanol in Water
All Data – 280 Data Points
Sample Quartiles (ΔG kJ/mol)
-4.65
-4.70
-4.75
-4.80
-4.85
-4.90
-3
-2
-1
0
1
2
3
Theoretical Quantiles

6.1 R Language Commands

The following R language commands were used to perform normality testing and to determine sample size. These commands were also used to generate quantile to quantile plots and histograms for each of the GROMACS 4.5.3 builds, the four variable precision algorithms, the defined zone of equivalence and to analyze all of the data for all versions when combined. The only changes from one build/algorithm/zone of equivalence to the other was the plot/chart headings and the data input to the script. The following is the example for methane in water.

```
#   Methane in Water GROMACS SSE Single Precision Data Points

x<-c(   Insert data series here )

sink("Methane_NORM.txt", append=FALSE, split=FALSE) # Output sample size and test results
#           run normality tests
s <- sd(x)                          # Standard Deviation

#           Sample Size Determination
#
#                       based on zscore, standard deviation, margin of error
#                       .95/2 lookup in zscore table = 1.96 for 95%
#                       margin of error 0.02

n <- (( (s*1.96)/0.02))                         # Determine estimated sample size based on the data set
n <- n*n
print(paste("Methane in Water - GROMACS SSE - Sample size 95% with margin of error 0.02: ",n))

#           Normality Tests

ad.test(x)                          # Anderson-Darling Normality Test
shapiro.test(x)                                 # Shapiro-Wilk Normality Test
pearson.test(x)                                 # Pearson chi-square Normality Test
lillie.test(x)                                  # Lilliefors (Kolmogorov-Smirnov) Normality Test
cvm.test(x)                                     # Cramer-von Mises Normality
# CI of Non normal data
wilcox.test(x,conf.int=TRUE)   # Wilcox Non Parametric Test

sink()      # close the output file

#           Output Q-Q plot using file naming convention

jpeg('Methane_QQ_GROMACS_SSE.jpg')
qqnorm(x, main="Free Energy - Methane in Water \nGROMACS Single Precision SSE – 40 Data Points",
ylab='Sample Quantiles (ΔG kJ/mol)'); qqline(x)
dev.off()

#           Output Histograms using file naming convention

jpeg('Methane_HIST_GROMACS_SSE.jpg')
hist(x, main='Histogram of Free Energy\nMethane in Water \n GROMACS Single Precision SSE - 40 Data
Points', xlab='ΔG kJ/mol')
dev.off()
```

7.1 Raw Data

7.1.1 Acetamide in Water

Raw Data Acetamide in Water - 46 Samples														
	GROMACS Double Precision SSE2		GROMACS Single Precision SSE		GROMACS Single Precision C		Variable Precision Algorithms							
Significant Digits	≈ 15.95		≈ 7.22		≈ 7.22		≈ 3.01		≈ 3.31		≈ 3.61		≈ 3.91	
	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-
	-4.05	0.09	-3.84	0.04	-3.92	0.05	-4.05	0.09	-3.88	0.06	-3.90	0.04	-3.98	0.11
	-3.95	0.06	-3.90	0.08	-3.82	0.09	-3.95	0.06	-3.85	0.06	-3.88	0.05	-3.79	0.08
	-3.87	0.04	-3.95	0.03	-3.93	0.09	-3.87	0.04	-3.90	0.08	-3.97	0.11	-3.95	0.05
	-3.93	0.03	-3.87	0.05	-3.99	0.06	-3.93	0.03	-3.90	0.03	-3.88	0.06	-3.98	0.10
	-3.89	0.06	-3.92	0.07	-3.89	0.03	-3.89	0.06	-4.01	0.08	-3.94	0.08	-3.87	0.09
	-4.02	0.09	-3.88	0.03	-3.96	0.05	-4.02	0.09	-3.98	0.10	-3.98	0.05	-3.79	0.06
	-3.98	0.04	-3.92	0.05	-3.94	0.07	-3.98	0.04	-3.86	0.06	-3.96	0.05	-3.94	0.06
	-3.85	0.12	-3.96	0.08	-3.98	0.09	-3.85	0.12	-3.92	0.05	-4.01	0.08	-3.98	0.05
	-3.89	0.08	-3.83	0.08	-3.91	0.06	-3.89	0.08	-3.75	0.03	-3.93	0.08	-3.84	0.07
	-3.90	0.08	-3.82	0.03	-3.88	0.05	-3.90	0.08	-3.87	0.04	-3.87	0.11	-3.93	0.02
	-3.76	0.09	-3.89	0.07	-3.71	0.05	-3.85	0.05	-3.99	0.03	-3.94	0.07	-4.04	0.11
	-3.92	0.08	-3.96	0.04	-3.87	0.08	-3.91	0.05	-3.92	0.06	-3.98	0.05	-3.94	0.02
	-3.84	0.04	-4.05	0.05	-3.90	0.06	-3.98	0.05	-3.84	0.09	-3.95	0.09	-4.06	0.09
	-3.92	0.07	-3.94	0.06	-4.02	0.05	-4.01	0.09	-3.89	0.09	-4.00	0.02	-4.06	0.05
	-3.91	0.12	-3.86	0.04	-4.00	0.05	-3.89	0.04	-3.90	0.06	-3.89	0.05	-3.91	0.03
	-4.04	0.08	-3.89	0.09	-3.86	0.12	-3.96	0.05	-3.87	0.07	-3.91	0.05	-3.91	0.05
	-3.92	0.06	-3.91	0.08	-3.99	0.07	-3.99	0.09	-4.05	0.09	-3.99	0.08	-4.02	0.05
	-3.94	0.10	-3.81	0.08	-3.83	0.09	-3.94	0.10	-3.86	0.04	-3.96	0.08	-3.95	0.05
	-3.91	0.07	-3.87	0.09	-3.92	0.09	-3.91	0.03	-3.79	0.06	-3.92	0.09	-3.97	0.07
	-3.92	0.08	-4.04	0.03	-3.92	0.06	-4.04	0.03	-3.90	0.03	-3.82	0.07	-3.94	0.12
	-3.92	0.07	-3.94	0.04	-3.86	0.09	-3.98	0.08	-3.93	0.06	-3.94	0.06	-3.92	0.07
	-3.89	0.09	-3.95	0.04	-4.02	0.04	-4.01	0.09	-3.96	0.09	-3.99	0.08	-3.89	0.09
	-4.05	0.10	-3.91	0.04	-3.87	0.07	-3.90	0.08	-4.03	0.11	-3.86	0.02	-4.05	0.10
	-3.85	0.03	-4.01	0.07	-3.82	0.03	-3.94	0.05	-3.94	0.10	-3.93	0.04	-3.85	0.03
	-3.80	0.05	-4.05	0.06	-3.86	0.03	-4.02	0.06	-3.98	0.07	-4.02	0.02	-3.80	0.05
	-3.97	0.09	-3.91	0.08	-3.88	0.04	-3.95	0.06	-3.88	0.11	-3.89	0.01	-3.97	0.09
	-3.96	0.03	-3.96	0.08	-3.94	0.04	-4.01	0.05	-3.97	0.06	-3.97	0.07	-3.96	0.03
	-3.94	0.05	-3.95	0.08	-3.90	0.03	-3.98	0.08	-3.87	0.03	-3.99	0.09	-3.94	0.05
	-3.93	0.05	-3.77	0.07	-3.91	0.05	-3.89	0.07	-3.87	0.05	-3.90	0.08	-3.93	0.05
	-3.90	0.14	-3.90	0.05	-3.93	0.04	-3.89	0.08	-3.96	0.04	-3.93	0.05	-3.90	0.14
	-3.90	0.03	-3.84	0.07	-3.88	0.06	-3.95	0.04	-3.82	0.08	-3.98	0.04	-3.97	0.07
	-3.98	0.04	-3.93	0.04	-3.79	0.10	-3.89	0.07	-3.87	0.03	-4.06	0.09	-4.03	0.04
	-3.91	0.11	-3.98	0.08	-3.92	0.05	-3.97	0.04	-3.86	0.07	-3.80	0.09	-3.95	0.10
	-3.97	0.07	-3.93	0.07	-4.10	0.02	-3.94	0.07	-3.89	0.07	-3.95	0.04	-3.90	0.03
	-3.92	0.06	-3.80	0.05	-3.79	0.07	-4.03	0.07	-3.99	0.04	-3.95	0.04	-3.83	0.04
	-3.91	0.09	-3.87	0.03	-4.09	0.06	-3.80	0.05	-3.96	0.05	-3.87	0.07	-3.98	0.05
	-4.03	0.05	-3.84	0.02	-4.01	0.09	-3.94	0.07	-3.96	0.04	-3.89	0.03	-3.87	0.04
	-3.93	0.02	-3.96	0.09	-3.89	0.04	-3.84	0.08	-4.00	0.05	-3.94	0.05	-3.77	0.05
	-3.84	0.05	-3.90	0.05	-3.89	0.06	-3.96	0.04	-3.87	0.04	-3.92	0.08	-3.90	0.05
	-4.02	0.04	-4.19	0.05	-3.89	0.08	-3.93	0.05	-3.96	0.06	-3.88	0.06	-3.91	0.08
	-3.78	0.05	-3.81	0.08	-3.94	0.07	-3.89	0.08	-3.89	0.05	-3.91	0.05	-3.98	0.12
	-3.92	0.06	-3.96	0.05	-3.94	0.1	-3.81	0.06	-3.95	0.08	-3.87	0.03	-3.99	0.05
	-3.85	0.09	-4.02	0.06	-4.01	0.07	-3.86	0.04	-3.84	0.07	-3.98	0.09	-3.91	0.07
	-4.02	0.02	-3.94	0.07	-4.00	0.10	-3.82	0.05	-3.92	0.07	-3.98	0.09	-3.84	0.06
	-3.89	0.04	-3.93	0.04	-3.91	0.06	-3.97	0.04	-3.91	0.07	-3.99	0.05	-3.77	0.08
	-4.01	0.04	-4.02	0.06	-3.92	0.05	-4.07	0.11	-3.95	0.07	-3.88	0.05	-3.91	0.05
Mean	-3.92	0.07	-3.92	0.06	-3.92	0.06	-3.94	0.06	-3.91	0.06	-3.93	0.06	-3.93	0.07

7.1.2 Ethanol in Water

Raw Data Ethanol in Water														
	GROMACS Double Precision SSE2		GROMACS Single Precision SSE		GROMACS Single Precision C		Variable Precision Algorithms							
Approx Sign Digits	≈ 15.95		≈ 7.22		≈ 7.22		≈ 3.01		≈ 3.31		≈ 3.61		≈ 3.91	
	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-
	-9.50	0.06	-9.29	0.09	-9.60	0.13	-9.41	0.05	-9.46	0.06	-9.38	0.09	-9.34	0.07
	-9.53	0.06	-9.44	0.06	-9.49	0.05	-9.46	0.07	-9.42	0.08	-9.40	0.03	-9.58	0.17
	-9.36	0.04	-9.48	0.09	-9.56	0.08	-9.31	0.08	-9.52	0.06	-9.70	0.05	-9.54	0.06
	-9.43	0.11	-9.43	0.05	-9.38	0.08	-9.50	0.10	-9.58	0.08	-9.39	0.05	-9.36	0.08
	-9.52	0.10	-9.47	0.09	-9.33	0.08	-9.56	0.07	-9.44	0.09	-9.44	0.04	-9.44	0.03
	-9.43	0.07	-9.48	0.09	-9.62	0.06	-9.37	0.06	-9.40	0.08	-9.52	0.04	-9.44	0.08
	-9.45	0.13	-9.52	0.03	-9.52	0.07	-9.43	0.09	-9.40	0.06	-9.38	0.08	-9.37	0.10
	-9.46	0.06	-9.33	0.14	-9.44	0.05	-9.57	0.07	-9.40	0.06	-9.42	0.11	-9.39	0.02
	-9.55	0.10	-9.40	0.07	-9.44	0.04	-9.50	0.04	-9.48	0.09	-9.57	0.03	-9.43	0.07
	-9.46	0.03	-9.52	0.08	-9.41	0.11	-9.45	0.11	-9.48	0.08	-9.43	0.09	-9.37	0.08
	-9.37	0.08	-9.45	0.04	-9.52	0.04	-9.43	0.05	-9.61	0.03	-9.32	0.06	-9.40	0.06
	-9.50	0.09	-9.51	0.10	-9.39	0.05	-9.42	0.05	-9.47	0.08	-9.40	0.10	-9.40	0.06
	-9.40	0.07	-9.38	0.10	-9.56	0.04	-9.36	0.09	-9.51	0.10	-9.42	0.04	-9.39	0.11
	-9.60	0.09	-9.25	0.05	-9.56	0.11	-9.49	0.07	-9.43	0.16	-9.36	0.04	-9.47	0.09
	-9.39	0.05	-9.55	0.07	-9.52	0.07	-9.21	0.12	-9.48	0.05	-9.42	0.08	-9.44	0.07
	-9.51	0.03	-9.51	0.03	-9.48	0.08	-9.38	0.01	-9.46	0.07	-9.45	0.07	-9.52	0.08
	-9.44	0.09	-9.26	0.09	-9.36	0.07	-9.26	0.06	-9.40	0.07	-9.45	0.07	-9.48	0.11
	-9.48	0.16	-9.43	0.08	-9.36	0.06	-9.47	0.05	-9.49	0.07	-9.41	0.09	-9.33	0.09
	-9.39	0.06	-9.45	0.05	-9.36	0.06	-9.48	0.05	-9.50	0.05	-9.52	0.05	-9.48	0.09
	-9.38	0.08	-9.51	0.11	-9.54	0.07	-9.38	0.05	-9.59	0.05	-9.48	0.12	-9.23	0.07
	-9.30	0.03	-9.43	0.03	-9.33	0.07	-9.50	0.11	-9.52	0.10	-9.44	0.05	-9.44	0.03
	-9.40	0.16	-9.37	0.06	-9.35	0.08	-9.51	0.10	-9.34	0.08	-9.54	0.09	-9.59	0.03
	-9.35	0.07	-9.39	0.07	-9.42	0.05	-9.51	0.14	-9.40	0.08	-9.45	0.03	-9.60	0.09
	-9.50	0.07	-9.35	0.07	-9.52	0.06	-9.49	0.10	-9.47	0.10	-9.38	0.12	-9.46	0.10
	-9.47	0.08	-9.43	0.07	-9.41	0.09	-9.37	0.05	-9.33	0.10	-9.44	0.12	-9.44	0.11
	-9.35	0.07	-9.41	0.02	-9.45	0.05	-9.46	0.08	-9.42	0.10	-9.46	0.04	-9.44	0.04
	-9.53	0.04	-9.33	0.07	-9.53	0.04	-9.52	0.11	-9.45	0.13	-9.48	0.05	-9.57	0.11
	-9.51	0.06	-9.45	0.03	-9.63	0.09	-9.48	0.09	-9.39	0.07	-9.51	0.12	-9.47	0.08
	-9.59	0.06	-9.31	0.06	-9.52	0.08	-9.45	0.12	-9.57	0.10	-9.34	0.09	-9.53	0.10
	-9.46	0.02	-9.39	0.07	-9.39	0.08	-9.48	0.07	-9.34	0.12	-9.63	0.08	-9.44	0.05
	-9.35	0.06	-9.34	0.01	-9.51	0.11	-9.45	0.04	-9.30	0.08	-9.55	0.08	-9.30	0.13
	-9.53	0.05	-9.42	0.09	-9.46	0.06	-9.44	0.15	-9.51	0.10	-9.33	0.09	-9.53	0.04
	-9.39	0.05	-9.30	0.07	-9.52	0.04	-9.38	0.08	-9.43	0.07	-9.33	0.09	-9.37	0.12
	-9.39	0.06	-9.44	0.07	-9.43	0.09	-9.44	0.04	-9.50	0.08	-9.58	0.07	-9.52	0.04
	-9.56	0.11	-9.37	0.04	-9.27	0.05	-9.35	0.08	-9.53	0.05	-9.40	0.09	-9.51	0.06
	-9.39	0.09	-9.31	0.05	-9.40	0.08	-9.37	0.11	-9.39	0.02	-9.49	0.12	-9.44	0.06
	-9.52	0.06	-9.56	0.06	-9.46	0.06	-9.34	0.13	-9.53	0.05	-9.34	0.05	-9.19	0.03
	-9.52	0.06	-9.42	0.06	-9.50	0.04	-9.48	0.10	-9.37	0.05	-9.42	0.07	-9.45	0.07
	-9.62	0.04	-9.37	0.05	-9.40	0.08	-9.42	0.11	-9.44	0.05	-9.39	0.11	-9.44	0.05
	-9.40	0.12	-9.53	0.04	-9.47	0.03	-9.55	0.08	-9.43	0.05	-9.57	0.06	-9.54	0.02
	-9.39	0.06	-9.48	0.07	-9.36	0.10	-9.39	0.05	-9.44	0.10	-9.32	0.11	-9.47	0.06
	-9.36	0.10	-9.44	0.10	-9.44	0.04	-9.50	0.08	-9.46	0.08	-9.36	0.11	-9.41	0.07
	-9.51	0.09	-9.52	0.02	-9.42	0.07	-9.40	0.02	-9.56	0.02	-9.45	0.11	-9.43	0.05
	-9.52	0.09	-9.39	0.03	-9.42	0.04	-9.46	0.05	-9.48	0.04	-9.42	0.09	-9.39	0.09
	-9.53	0.04	-9.59	0.08	-9.48	0.10	-9.42	0.04	-9.40	0.03	-9.49	0.07	-9.29	0.07
	-9.41	0.05	-9.42	0.04	-9.55	0.10	-9.39	0.08	-9.46	0.03	-9.47	0.06	-9.50	0.08
	-9.44	0.07	-9.40	0.05	-9.47	0.06	-9.49	0.12	-9.40	0.03	-9.43	0.03	-9.33	0.14
	-9.37	0.05	-9.47	0.10	-9.51	0.06	-9.53	0.07	-9.33	0.05	-9.37	0.13	-9.44	0.06
	-9.51	0.08	-9.48	0.08	-9.42	0.04	-9.32	0.08	-9.36	0.07	-9.44	0.06	-9.51	0.04
	-9.47	0.14	-9.33	0.07	-9.35	0.07	-9.41	0.07	-9.42	0.15	-9.42	0.06	-9.39	0.08
	-9.46	0.07	-9.37	0.08	-9.25	0.04	-9.49	0.11	-9.44	0.04	-9.42	0.12	-9.46	0.12
	-9.42	0.08	-9.41	0.04	-9.40	0.05	-9.20	0.10	-9.53	0.08	-9.49	0.09	-9.43	0.08
	-9.60	0.09	-9.46	0.08	-9.32	0.10	-9.42	0.08	-9.35	0.06	-9.44	0.05	-9.43	0.06
	-9.51	0.10	-9.50	0.06	-9.29	0.10	-9.31	0.05	-9.35	0.12	-9.48	0.03	-9.38	0.08
	-9.57	0.08	-9.44	0.09	-9.32	0.02	-9.49	0.07	-9.46	0.06	-9.31	0.06	-9.43	0.05
	-9.50	0.03	-9.42	0.06	-9.39	0.09	-9.49	0.07	-9.29	0.08	-9.32	0.06	-9.44	0.09
	-9.50	0.05	-9.40	0.07	-9.31	0.05	-9.28	0.05	-9.43	0.02	-9.46	0.07	-9.45	0.06
	-9.51	0.06	-9.47	0.10	-9.41	0.04	-9.42	0.04	-9.40	0.08	-9.51	0.15	-9.46	0.12
	-9.41	0.14	-9.52	0.15	-9.46	0.05	-9.39	0.09	-9.46	0.04	-9.46	0.09	-9.41	0.07
Mean	-9.46	0.07	-9.43	0.07	-9.44	0.07	-9.43	0.08	-9.44	0.07	-9.44	0.08	-9.44	0.07

7.1.3 Isobutane in Water

Raw Data Isobutane in Water 46 Samples														
	GROMACS Double Precision SSE2		GROMACS Single Precision SSE		GROMACS Single Precision C		Variable Precision Algorithms							
Approx Sign Digits	≈ 15.95		≈ 7.22		≈ 7.22		≈ 3.01		≈ 3.31		≈ 3.61		≈ 3.91	
	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-
	-9.90	0.10	-9.71	0.09	-9.90	0.03	-9.81	0.09	-9.91	0.08	-9.82	0.05	-9.83	0.08
	-9.84	0.07	-9.93	0.12	-9.98	0.07	-9.85	0.03	-9.94	0.05	-9.81	0.07	-9.80	0.04
	-9.94	0.12	-9.96	0.08	-9.74	0.08	-9.79	0.06	-9.89	0.08	-10.00	0.09	-9.80	0.04
	-9.90	0.06	-9.87	0.10	-9.77	0.06	-9.91	0.09	-9.89	0.08	-9.77	0.05	-9.84	0.08
	-9.98	0.10	-9.94	0.10	-9.75	0.08	-9.86	0.10	-9.91	0.08	-9.89	0.03	-9.92	0.04
	-9.77	0.04	-9.85	0.07	-9.97	0.10	-9.90	0.05	-9.88	0.11	-9.81	0.11	-9.84	0.10
	-9.83	0.05	-9.89	0.08	-9.93	0.04	-9.83	0.10	-9.90	0.04	-9.91	0.10	-9.86	0.08
	-9.78	0.04	-9.84	0.08	-9.87	0.11	-9.95	0.07	-9.93	0.06	-9.86	0.14	-9.80	0.04
	-9.98	0.06	-9.89	0.07	-9.87	0.05	-9.89	0.09	-9.91	0.12	-9.85	0.08	-9.74	0.11
	-9.95	0.11	-9.85	0.10	-9.77	0.07	-9.82	0.05	-9.94	0.08	-9.81	0.04	-9.72	0.09
	-9.91	0.04	-9.88	0.09	-9.83	0.05	-10.00	0.03	-9.77	0.09	-9.88	0.13	-9.88	0.04
	-9.97	0.05	-9.94	0.05	-9.89	0.04	-9.87	0.09	-9.97	0.05	-9.90	0.06	-9.93	0.07
	-9.84	0.04	-10.00	0.14	-9.89	0.06	-9.84	0.08	-9.84	0.08	-9.89	0.10	-9.90	0.13
	-9.90	0.03	-9.93	0.06	-9.88	0.08	-9.86	0.05	-9.74	0.12	-9.99	0.09	-9.88	0.14
	-9.90	0.11	-9.87	0.06	-9.85	0.07	-9.92	0.05	-9.95	0.03	-9.92	0.08	-9.97	0.07
	-9.95	0.10	-9.84	0.06	-9.99	0.05	-9.96	0.06	-9.78	0.05	-10.00	0.11	-9.95	0.10
	-9.92	0.05	-9.84	0.03	-9.81	0.05	-9.91	0.05	-9.75	0.05	-10.00	0.07	-9.86	0.07
	-9.88	0.06	-9.95	0.14	-9.90	0.06	-9.88	0.05	-9.96	0.05	-9.89	0.05	-9.95	0.04
	-9.93	0.09	-9.92	0.08	-9.79	0.12	-9.76	0.07	-9.86	0.10	-9.87	0.11	-9.71	0.06
	-9.93	0.06	-9.84	0.10	-9.94	0.06	-9.92	0.04	-9.83	0.03	-9.83	0.09	-9.97	0.09
	-9.88	0.07	-9.93	0.09	-9.95	0.09	-9.82	0.05	-9.80	0.07	-9.91	0.05	-9.98	0.07
	-9.88	0.07	-9.91	0.08	-9.87	0.11	-9.81	0.10	-9.82	0.05	-9.80	0.04	-9.93	0.04
	-9.86	0.04	-9.97	0.05	-9.79	0.06	-9.80	0.06	-9.95	0.07	-9.89	0.03	-9.90	0.04
	-9.77	0.08	-9.84	0.10	-9.72	0.06	-9.91	0.10	-10.00	0.06	-9.95	0.07	-9.83	0.13
	-9.86	0.03	-9.86	0.05	-9.86	0.05	-9.88	0.08	-9.92	0.08	-10.00	0.04	-9.88	0.04
	-9.88	0.08	-9.88	0.07	-9.74	0.08	-9.99	0.07	-9.89	0.07	-9.86	0.08	-9.90	0.12
	-10.00	0.06	-9.84	0.07	-9.87	0.03	-10.00	0.05	-9.83	0.05	-9.74	0.07	-9.91	0.01
	-9.88	0.03	-9.79	0.05	-9.84	0.12	-9.91	0.05	-9.97	0.06	-9.83	0.07	-9.99	0.09
	-9.95	0.08	-9.92	0.05	-9.99	0.03	-9.95	0.05	-9.93	0.09	-9.83	0.08	-9.88	0.09
	-9.85	0.10	-9.98	0.08	-9.91	0.02	-9.77	0.07	-9.79	0.06	-10.00	0.11	-9.93	0.08
	-9.84	0.05	-9.79	0.09	-9.85	0.10	-10.00	0.04	-9.91	0.02	-9.85	0.12	-9.88	0.04
	-9.86	0.05	-9.98	0.06	-9.70	0.04	-9.72	0.11	-9.90	0.09	-9.91	0.07	-9.86	0.05
	-9.89	0.08	-9.88	0.04	-9.77	0.04	-9.87	0.09	-9.91	0.04	-9.81	0.04	-9.89	0.08
	-9.86	0.04	-9.88	0.09	-9.84	0.08	-9.94	0.06	-9.99	0.04	-9.85	0.05	-9.86	0.04
	-9.96	0.04	-9.88	0.13	-9.83	0.09	-9.87	0.12	-9.74	0.07	-9.89	0.04	-9.96	0.04
	-9.93	0.05	-9.89	0.11	-10.00	0.10	-9.81	0.07	-9.92	0.09	-9.84	0.05	-9.93	0.05
	-9.82	0.09	-9.95	0.07	-9.84	0.04	-9.92	0.05	-9.80	0.06	-9.82	0.09	-9.82	0.09
	-9.90	0.07	-9.85	0.09	-9.86	0.07	-9.89	0.06	-9.77	0.06	-9.88	0.05	-9.90	0.07
	-9.88	0.10	-9.83	0.05	-9.86	0.05	-9.91	0.05	-9.90	0.05	-9.96	0.09	-9.88	0.10
	-9.87	0.05	-9.79	0.04	-9.86	0.10	-9.92	0.11	-9.64	0.07	-9.79	0.10	-9.87	0.05
	-9.88	0.05	-9.91	0.09	-9.80	0.10	-9.95	0.08	-9.88	0.02	-9.93	0.10	-9.87	0.12
	-9.88	0.07	-10.00	0.05	-9.98	0.06	-9.82	0.09	-9.72	0.06	-9.80	0.11	-9.88	0.12
	-10.00	0.09	-9.87	0.03	-9.88	0.08	-9.93	0.04	-9.89	0.07	-9.84	0.05	-9.88	0.03
	-9.82	0.04	-9.96	0.09	-9.95	0.08	-9.93	0.05	-9.83	0.05	-9.81	0.03	-9.95	0.12
	-9.77	0.06	-9.86	0.10	-9.98	0.08	-9.84	0.06	-9.99	0.07	-9.91	0.11	-9.89	0.04
	-9.81	0.07	-9.98	0.06	-9.98	0.09	-9.82	0.06	-9.89	0.07	-9.89	0.02	-9.85	0.03
Mean	-9.89	0.07	-9.89	0.08	-9.87	0.07	-9.88	0.07	-9.87	0.07	-9.88	0.07	-9.88	0.07

7.1.4 Methane in Water

Raw Data Methane in Water - 40 Samples														
	GROMACS Double Precision SSE2		GROMACS Single Precision SSE		GROMACS Single Precision C		Variable Precision Algorithms							
Approx Sign Digits	≈ 15.95		≈ 7.22		≈ 7.22		≈ 3.01		≈ 3.31		≈ 3.61		≈ 3.91	
	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-
	-8.94	0.05	-9.08	-9.08	-9.04	0.06	-9.02	-9.02	-8.97	0.04	-9.05	0.02	-9.02	0.05
	-8.96	0.03	-9.08	-9.08	-9.00	0.02	-8.99	-8.99	-9.04	0.09	-9.01	0.04	-9.09	0.05
	-8.95	0.05	-9.00	-9.00	-8.99	0.03	-9.00	-9.00	-9.00	0.02	-8.98	0.03	-9.01	0.06
	-9.03	0.06	-9.03	-9.03	-9.00	0.06	-8.96	-8.96	-9.01	0.02	-9.04	0.03	-8.99	0.08
	-9.01	0.04	-9.01	-9.01	-9.02	0.04	-8.96	-8.96	-9.06	0.01	-8.97	0.02	-9.06	0.02
	-9.06	0.05	-9.03	-9.03	-9.02	0.05	-9.12	-9.12	-8.98	0.05	-9.00	0.04	-9.04	0.03
	-9.06	0.04	-8.99	-8.99	-9.06	0.05	-8.96	-8.96	-9.04	0.05	-9.00	0.05	-8.97	0.02
	-9.04	0.02	-9.01	-9.01	-8.92	0.03	-8.97	-8.97	-9.00	0.05	-9.04	0.05	-9.06	0.06
	-9.03	0.05	-9.03	-9.03	-9.04	0.03	-9.03	-9.03	-9.00	0.04	-9.01	0.03	-8.96	0.06
	-8.91	0.04	-8.95	-8.95	-9.00	0.02	-8.97	-8.97	-8.94	0.05	-9.00	0.06	-8.91	0.02
	-9.04	0.04	-9.03	-9.03	-9.00	0.05	-8.99	-8.99	-8.98	0.03	-9.00	0.05	-9.00	0.03
	-9.04	0.01	-9.07	-9.07	-9.07	0.06	-9.00	-9.00	-9.09	0.04	-8.96	0.08	-9.01	0.04
	-8.97	0.05	-9.02	-9.02	-8.93	0.05	-9.02	-9.02	-9.02	0.04	-9.05	0.04	-9.00	0.05
	-9.02	0.03	-9.11	-9.11	-9.03	0.02	-9.06	-9.06	-9.00	0.02	-9.03	0.04	-9.02	0.06
	-9.06	0.03	-8.96	-8.96	-8.94	0.06	-8.92	-8.92	-9.08	0.04	-9.03	0.04	-9.00	0.06
	-9.08	0.03	-8.98	-8.98	-9.00	0.05	-9.03	-9.03	-9.02	0.05	-9.10	0.05	-9.04	0.05
	-9.02	0.04	-9.03	-9.03	-9.02	0.04	-9.01	-9.01	-9.03	0.04	-8.91	0.03	-9.00	0.03
	-9.05	0.04	-9.05	-9.05	-9.02	0.04	-9.09	-9.09	-9.04	0.03	-9.06	0.04	-9.01	0.04
	-9.05	0.04	-9.06	-9.06	-9.00	0.05	-9.00	-9.00	-8.99	0.04	-9.06	0.07	-9.06	0.03
	-9.00	0.03	-9.08	-9.08	-9.05	0.02	-9.05	-9.05	-8.99	0.03	-9.03	0.04	-9.00	0.04
	-8.97	0.08	-9.05	-9.05	-9.02	0.03	-9.01	-9.01	-8.99	0.01	-8.99	0.03	-9.06	0.02
	-8.98	0.06	-8.99	-8.99	-8.98	0.07	-9.00	-9.00	-9.01	0.03	-9.05	0.07	-9.03	0.03
	-9.05	0.03	-9.04	-9.04	-8.93	0.02	-8.99	-8.99	-9.01	0.04	-9.02	0.01	-9.02	0.03
	-9.03	0.04	-9.02	-9.02	-9.05	0.06	-8.97	-8.97	-8.96	0.02	-9.07	0.03	-9.09	0.05
	-9.02	0.04	-8.99	-8.99	-9.05	0.05	-9.02	-9.02	-8.99	0.08	-9.07	0.04	-9.01	0.04
	-9.05	0.04	-8.99	-8.99	-9.02	0.04	-9.00	-9.00	-9.04	0.03	-9.05	0.04	-9.08	0.05
	-8.99	0.02	-9.00	-9.00	-9.01	0.02	-8.98	-8.98	-9.06	0.04	-9.03	0.05	-9.01	0.04
	-8.96	0.04	-9.06	-9.06	-8.94	0.03	-9.03	-9.03	-9.08	0.01	-9.03	0.06	-9.01	0.06
	-9.00	0.04	-8.97	-8.97	-9.01	0.03	-8.97	-8.97	-9.01	0.02	-9.01	0.02	-8.99	0.04
	-8.98	0.04	-8.91	-8.91	-9.04	0.06	-9.09	-9.09	-9.03	0.05	-9.03	0.02	-9.04	0.03
	-9.02	0.05	-9.03	-9.03	-9.01	0.07	-8.92	-8.92	-9.00	0.05	-9.00	0.03	-9.03	0.04
	-9.05	0.05	-9.01	-9.01	-9.05	0.04	-8.95	-8.95	-9.02	0.05	-8.99	0.04	-9.07	0.05
	-9.00	0.04	-8.94	-8.94	-9.06	0.05	-9.06	-9.06	-9.01	0.02	-8.94	0.04	-9.02	0.04
	-9.04	0.05	-8.99	-8.99	-9.03	0.05	-9.03	-9.03	-9.02	0.03	-9.03	0.04	-9.05	0.05
	-8.98	0.05	-9.00	-9.03	-9.02	0.06	-8.98	-9.00	-9.09	0.05	-8.97	0.04	-9.03	0.02
	-9.06	0.04	-9.02	-9.02	-9.06	0.05	-9.00	-9.00	-9.02	0.04	-9.03	0.04	-9.01	0.04
	-8.97	0.05	-9.11	-9.11	-8.94	0.06	-9.00	-9.00	-8.99	0.01	-9.06	0.07	-9.06	0.03
	-8.98	0.06	-9.05	-9.05	-8.93	0.02	-8.97	-8.97	-9.06	0.04	-9.07	0.04	-9.01	0.06
	-8.98	0.04	-8.91	-8.91	-9.01	0.07	-9.09	-9.09	-9.02	0.05	-9.03	0.02	-9.03	0.04
	-8.98	0.05	-9.00	-9.00	-9.02	0.05	-8.98	-8.98	-9.09	0.04	-8.97	0.04	-9.03	0.05
Mean	-9.01	0.04	-9.02	-9.02	-9.01	0.04	-9.00	-9.01	-9.02	0.04	-9.02	0.04	-9.02	0.04

7.1.5 Methanol in Water

Raw Data Methanol in Water - 40 Samples														
	GROMACS Double Precision SSE2		GROMACS Single Precision SSE		GROMACS Single Precision C		Variable Precision Algorithms							
Approx Sign Digits	≈ 15.95		≈ 7.22		≈ 7.22		≈ 3.01		≈ 3.31		≈ 3.61		≈ 3.91	
	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-	Δ G kJ/mol	+/-
	-4.84	0.06	-4.80	0.05	-4.84	0.05	-4.70	0.04	-4.77	0.04	-4.72	0.05	-4.72	0.03
	-4.67	0.05	-4.76	0.03	-4.86	0.03	-4.79	0.02	-4.78	0.03	-4.83	0.04	-4.75	0.05
	-4.76	0.06	-4.79	0.04	-4.74	0.04	-4.81	0.05	-4.71	0.04	-4.75	0.03	-4.79	0.06
	-4.75	0.05	-4.73	0.05	-4.72	0.02	-4.68	0.02	-4.75	0.07	-4.84	0.03	-4.85	0.07
	-4.71	0.02	-4.75	0.08	-4.77	0.03	-4.80	0.04	-4.70	0.04	-4.87	0.06	-4.72	0.03
	-4.74	0.07	-4.75	0.07	-4.75	0.05	-4.77	0.02	-4.67	0.09	-4.77	0.05	-4.83	0.04
	-4.84	0.02	-4.85	0.05	-4.79	0.08	-4.77	0.02	-4.73	0.07	-4.80	0.06	-4.85	0.04
	-4.85	0.04	-4.79	0.05	-4.76	0.05	-4.87	0.05	-4.81	0.05	-4.76	0.05	-4.78	0.04
	-4.79	0.04	-4.75	0.04	-4.80	0.04	-4.85	0.02	-4.76	0.04	-4.86	0.05	-4.77	0.04
	-4.81	0.04	-4.79	0.06	-4.74	0.04	-4.86	0.06	-4.70	0.06	-4.68	0.03	-4.75	0.03
	-4.73	0.05	-4.81	0.05	-4.85	0.07	-4.78	0.06	-4.80	0.03	-4.85	0.07	-4.71	0.04
	-4.88	0.07	-4.89	0.04	-4.72	0.04	-4.80	0.05	-4.74	0.04	-4.71	0.03	-4.72	0.04
	-4.73	0.06	-4.90	0.07	-4.84	0.05	-4.83	0.03	-4.78	0.06	-4.80	0.03	-4.78	0.03
	-4.74	0.08	-4.79	0.05	-4.69	0.01	-4.78	0.03	-4.77	0.01	-4.80	0.04	-4.74	0.04
	-4.84	0.05	-4.79	0.05	-4.83	0.04	-4.76	0.05	-4.72	0.04	-4.77	0.03	-4.75	0.06
	-4.82	0.00	-4.72	0.05	-4.88	0.04	-4.68	0.01	-4.82	0.03	-4.87	0.03	-4.80	0.06
	-4.72	0.06	-4.77	0.06	-4.77	0.05	-4.75	0.04	-4.89	0.03	-4.80	0.06	-4.82	0.05
	-4.80	0.02	-4.77	0.06	-4.73	0.04	-4.69	0.04	-4.86	0.04	-4.69	0.04	-4.82	0.06
	-4.71	0.05	-4.74	0.07	-4.73	0.03	-4.79	0.02	-4.85	0.05	-4.83	0.04	-4.79	0.05
	-4.69	0.02	-4.70	0.03	-4.76	0.03	-4.87	0.05	-4.78	0.02	-4.78	0.04	-4.78	0.03
	-4.80	0.04	-4.81	0.06	-4.81	0.04	-4.77	0.05	-4.79	0.06	-4.76	0.07	-4.64	0.05
	-4.79	0.05	-4.82	0.05	-4.80	0.07	-4.79	0.03	-4.83	0.06	-4.76	0.04	-4.77	0.03
	-4.80	0.05	-4.83	0.05	-4.82	0.06	-4.75	0.08	-4.81	0.04	-4.80	0.04	-4.85	0.04
	-4.88	0.05	-4.81	0.05	-4.73	0.05	-4.91	0.05	-4.86	0.04	-4.78	0.05	-4.80	0.06
	-4.71	0.02	-4.80	0.08	-4.85	0.03	-4.82	0.05	-4.78	0.06	-4.77	0.03	-4.82	0.03
	-4.79	0.02	-4.83	0.08	-4.77	0.05	-4.74	0.04	-4.76	0.06	-4.70	0.07	-4.78	0.03
	-4.86	0.06	-4.84	0.04	-4.78	0.03	-4.88	0.04	-4.79	0.06	-4.77	0.06	-4.77	0.03
	-4.72	0.04	-4.75	0.09	-4.81	0.04	-4.74	0.04	-4.78	0.01	-4.78	0.05	-4.85	0.07
	-4.73	0.05	-4.82	0.04	-4.79	0.07	-4.74	0.05	-4.75	0.05	-4.73	0.04	-4.70	0.04
	-4.81	0.02	-4.76	0.03	-4.78	0.04	-4.77	0.06	-4.78	0.04	-4.75	0.03	-4.87	0.03
	-4.74	0.04	-4.81	0.07	-4.76	0.06	-4.78	0.05	-4.72	0.05	-4.79	0.06	-4.71	0.07
	-4.75	0.01	-4.76	0.06	-4.81	0.02	-4.83	0.02	-4.71	0.06	-4.78	0.06	-4.80	0.06
	-4.80	0.05	-4.83	0.02	-4.85	0.02	-4.74	0.05	-4.81	0.06	-4.77	0.05	-4.73	0.05
	-4.75	0.04	-4.76	0.03	-4.83	0.04	-4.82	0.01	-4.78	0.04	-4.82	0.02	-4.84	0.05
	-4.81	0.03	-4.77	0.08	-4.84	0.04	-4.89	0.04	-4.73	0.07	-4.76	0.05	-4.72	0.06
	-4.80	0.07	-4.72	0.04	-4.79	0.05	-4.72	0.07	-4.87	0.03	-4.75	0.07	-4.80	0.06
	-4.93	0.05	-4.81	0.05	-4.82	0.03	-4.84	0.03	-4.80	0.04	-4.79	0.08	-4.82	0.04
	-4.78	0.06	-4.80	0.04	-4.80	0.06	-4.79	0.04	-4.77	0.06	-4.84	0.06	-4.65	0.03
	-4.79	0.02	-4.73	0.05	-4.82	0.04	-4.79	0.02	-4.78	0.04	-4.83	0.03	-4.79	0.06
	-4.71	0.04	-4.92	0.02	-4.80	0.05	-4.79	0.04	-4.73	0.02	-4.75	0.04	-4.79	0.05
Mean	-4.78	0.04	-4.79	0.05	-4.79	0.04	-4.79	0.04	-4.78	0.05	-4.78	0.05	-4.78	0.05

8.1 Free Energy Configuration in the GROMACS .mdp File

.mdp File Simulation	Comments
integrator = sd	Simulation integrator
rlist = 1.0	Short range neighbor list cut-off distance (nm)
coulombtype = Reaction-Field	Use Reaction Field for Electrostatics
rcoulomb = 1.0	Coulomb cut-off Distance (nm)
epsilon_r = 1	Relative dielectric constant.
epsilon_rf = 54.0	Relative dielectric constant of the reaction field.
pcoupl = Parrinello-Rahman	Pressure coupling
tcoupl = No	No temperature coupling because it is provided through
ref_t = 298	Use a temperature of 298 K
tau_t=1.0	Avoid over-damping the water dynamics
free_energy =yes	Do a free energy calculation interpolating between the
iInit_lambda = 0.00	Starting λ value
delta_lambda = 0	No time-dependent changes to our λ values.
foreign_lambda = 0.05	Additional values of λ for which ΔH will be written to dhdl.xvg (with frequency nstdhdl). The configurations
sc-alpha = 0.5	The α scaling factor used in the "soft-core" Lennard-
sc_power = 1.0	Power for λ used in the soft-core equation.
sc_sigma = 0.3	The value of σ assigned to any atom types that have C6 or C12 parameters equal to zero or σ < sc-sigma
couple-moltype = ANALOGUE_NAME	The name of the [moleculetype] in that will have its topology interpolated from state A to state B. Note that the name given here must match a [moleculetype] name,
couple-lambda0 = vdw	The types of nonbonded interactions that are present in state A between the interpolated [moleculetype] and the
Couple-lambda1 = none	The types of nonbonded interactions that are present in state B between the interpolated [moleculetype] and the remainder of the system. The value "none" indicates that
couple-intramol = no	Do not decouple intramolecular interactions. That is, the λ factor is applied to only solute-solvent nonbonded
nstdhdl = 10	Frequency that ∂H/∂λ and ΔH are written to dhdl.xvg