Science & Technology Facilities Council

Technical Report
RAL-TR-2018-003

Using mixed precision within DL_POLY's force and energy evaluations: long-range interactions and fast Fourier transforms

HS Thorne

May 2018

Enquiries concerning this report should be addressed to:

RAL Library
STFC Rutherford Appleton Laboratory
Harwell Oxford
Didcot
OX11 0QX

Tel: +44(0)1235 445384
Fax: +44(0)1235 446403
email: libraryral@stfc.ac.uk

Science and Technology Facilities Council reports are available online at: http://epubs.stfc.ac.uk

ISSN 1358-6254

Using mixed precision within DL_POLY's force and energy evaluations: long-range interactions and fast Fourier transforms

H. Sue Thorne[1,2]

ABSTRACT

DL_POLY 4 is a molecular dynamics simulation code developed by CCP5. In this report, we consider the use of a simple mixed-precision methodology within the Smoothed Particle Mesh Ewald (SPME) method for systems with long-ranged ionic interactions. We show that such a method can be successfully used within DL_POLY and that, for some of the test problems, the overall execution time was reduced by 14%.

This work forms part of the CoSeC-funded Software Outlook project.

Keywords: Molecular Dynamics, DL_POLY, Ewald Method, FFT, Mixed Precision, Intel Xeon (IvyBridge), Software Outlook, CCP, CCP5

May 16, 2018

[1]The Hartree Centre, Science and Technology Facilities Council, Rutherford Appleton Laboratory, Harwell Campus, Didcot, OX11 0QX, UK.

[2]Correspondence to: sue.thorne@stfc.ac.uk

1 Background and motivation

In the last two decades, huge developments were made in molecular dynamics simulation methods , particularly in the ability to use large-scale parallel platforms. In this report, we consider the use of a mixed-precision methodology within the Smoothed Particle Mesh Ewald (SPME) method [4] for systems with long-ranged forces. In particular, our investigation hinges on the SPME implementation within DL_POLY 4, which incorporates the domain decomposition method with a 3D Fast Fourier Transform, known as the Daresbury Advanced Fourier Transform (DAFT) [2]. In DL_POLY 4, any real values are stored in double precision and operated on using double precision arithmetic (IEEE Standard 754 is assumed). On modern high performance computing architectures, there is normally the provision to use single precision arithmetic that is significantly faster than the corresponding double precision operation [6]. Additionally, the communication of real data between processors in a distributed system will normally be significantly faster when passing single precision data compared to double precision data. However, the desire to speed-up the code must be balanced with the requirements of getting an accurate solution. Indeed, if all real values are stored in single precision and operated on using single precision arithmetic, then DL_POLY is unstable. Thus, it will be necessary to use single precision in a restricted manner, i.e., use mixed precision. In this report, we focus on using a mixed precision approach within part of the SPME calculation, namely, the DAFT component will use single precision as much as possible.

In Section 2, the Smoothed Particle Mesh Ewald Method is reviewed. The Discrete Fourier transform and DAFT are discussed in Section 3.1. Our mixed precision methodology is introduced in Section 4 and numerical results from a number of test problems are provided in Section 5. Section 6 contains the conclusions from this work.

2 The Smoothed Particle Mesh Ewald Method

In molecular dynamics, the conditionally convergent Coulomb sum is used to describe the system energy contribution from ionic interactions in periodic charged systems. The Ewald Sum replaces this sum by three distinct sums with guaranteed convergence:

$$E = E^S + E^L + E^{SELF},$$

where E^S, the *real space* sum, is cast in normal physical space and represents the short-range interactions; E^L, the *reciprocal space* sum, is cast in the reciprocal space of the unit cell and represents the long-range terms; E^{SELF} is the self interaction term. Suppose we have N ions in a vacuum at locations $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N$ with point charges $q_1, q_2, \ldots, q_N$, respectively. We define $\epsilon_0 = 8.854 \times 10^{-12}\mathrm{C}^2\mathrm{N}^{-1}\mathrm{m}^{-2}$ to be the electric constant (or vacuum permittivity). Let the ions be subjected to periodic boundary conditions, which we describe using three repeat vectors $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}_3$: these form the *supercell*. Thus, if there is an ion with charge q_i at location $\mathbf{r}_i$, then there are also ions with charge q_i at $\mathbf{r}_i + n_1\mathbf{c}_1 + n_2\mathbf{c}_2 + n_3\mathbf{c}_3$, where n_1, n_2, and n_3 are arbitrary integers. We simplify the notation by writing an arbitrary repeat vector as $\mathbf{n}L$, where L represents the supercell. If we further assume that the charge distribution is of Guassian form with standard deviation σ and the supercell has volume V, then E^S, E^L and E^{SELF} are defined as

$$E^S = \frac{1}{4\pi\epsilon_0}\frac{1}{2}\sum_{\mathbf{n}}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}\frac{q_i q_j}{|\mathbf{r}_i - \mathbf{r}_j + \mathbf{n}L|}\,\mathbf{erfc}\left(\frac{|\mathbf{r}_i - \mathbf{r}_j + \mathbf{n}L|}{\sqrt{2}\sigma}\right), \tag{1}$$

$$E^L = \frac{1}{2V\epsilon_0}\sum_{\mathbf{k}\neq 0}\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{q_i q_j}{k^2}e^{i\mathbf{k}\cdot(\mathbf{r}_i - \mathbf{r}_j)e^{-\sigma^2 k^2/2}}, \tag{2}$$

$$E^{SELF} = \frac{1}{4\pi\epsilon_0}\frac{1}{\sqrt{2\pi}\sigma}\sum_{i=1}^{N}q_i^2. \tag{3}$$

In this report, we focus on the calculation of E^L. The calculation of E^S is considered in [5]. The SPME method improves the performance of the Ewald reciprocal space sum calculation by using (complex) Fast Fourier Transforms (FFTs). See [2] for further details about the framework in which the FFT is used.

3 Discrete Fourier Transforms

Let $y_0, y_1, \ldots, y_{N-1}$ be complex numbers forming a one-dimensional array. The Discrete Fourier Transform (DFT) is defined as

$$Y_k = \sum_{n=0}^{N-1} y_n w_N^{k,n}, \quad k = 0, \ldots, N-1, \tag{4}$$

where

$$w_N^{k,n} = e^{-\mathbf{i}2\pi kn/N}. \tag{5}$$

Thus, a new one-dimensional array is formed. If we were to evaluate this Y_k directly for each k, then $O(N^2)$ operations would be required. A Fast Fourier Transform (FFT) computes all Y_k using just $O(N \log N)$ operations.

3.1 Fast Fourier Transforms

Let $a(x)$ be a polynomial of degree $N-1$, where $a(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{N-1} x^{N-1}$. Define the $b(x)$ and $c(x)$ to be polynomials of degrees $\lfloor (N-1)/2 \rfloor$ and $N-1-\lceil N/2 \rceil$, respectively, where $b(x) = a_0 + a_2 x + a_4 x^2 + \ldots$ and $c(x) = a_1 + a_3 x + a_5 x^3 + .$ Noting that $a(x) = b(x^2) + x c(x^2)$ and that (4) is a polynomial with $x = e^{-\mathbf{i}2\pi k/N}$, then the time $T(N)$ to calculate all Y_k satisfies

$$\begin{aligned} T(N) &= 2T(N/2) + O(N) \\ &= O(N \log N). \end{aligned}$$

In DL_POLY, a three-dimensional version of the DFT is used: for an array y_{n_1,n_2,n_3} with $n_l = 0, 1, \ldots, N_{l-1}$ and $l = 1, 2, 3$, the three-dimensional DFT is defined as

$$Y_{k_1,k_2,k_3} = \sum_{n_1=0}^{N_1-1} w_{N_1}^{k_1 n_1} \sum_{n_2=0}^{N_2-1} w_{N_2}^{k_2 n_2} \sum_{n_3=0}^{N_3-1} w_{N_3}^{k_3 n_3} y_{n_1,n_2,n_3}, \quad k_l = 0, 1, \ldots, N_{l-1}, \quad l = 1, 2, 3. \tag{6}$$

Note that this is a sequence of 3 one-dimensional DFTs, i.e., Y can be calculated by performing a DFT along the n_3 dimension, then apply a DFT along the n_2 directions and, finally, apply a DFT along the n_1 direction. Let $N = N_1 N_2 N_3$, then it can be shown that the complexity of this approach is $O(N \log N)$.

The Daresbury Advanced Fourier Transform (DAFT) [2] is a fully distributed parallel implementation of the 3D FFT, which makes use of the underlying domain decomposition, and is designed to maximise performance on architectures with a large number of processors. Unfortunately, the nature of the DFT means that there needs to be a lot of communication between the processors and if this communication can be done using single precision data instead of double precision, then there should be an appreciable time/energy saving in communication costs. This motivates our desire to investigate the use of mixed precision within the reciprocal SPME calculation and DAFT.

4 Mixed precision within the reciprocal SPME calculation

Our method for using mixed precision within the reciprocal SPME calculation can be attributed to the work presented in [3]. If the SPME method is called by DL_POLY, then the SPME reciprocal space sum is computed once during each time step. The DAFT is called twice during each calculation of the SPME reciprocal space sum. In the current version of DL_POLY, the values of $w_N^{k,n}$ are computed once in double precision, stored in double precision and reused during each call to DAFT. All calculations within DAFT are carried out using double precision arithmetic. Our proposed mixed precision method is as follows:

- Compute $w_N^{k,n}$ using double precision arithmetic but store the resulting value using single precision. These values are reused by each call to DAFT.
- Convert the input FFT data to single precision (complex) data.
- Form the FFT using single precision arithmetic, producing single precision (complex) output.
- Convert the output to double precision on-the-fly as needed by the SPME calculation.

In the original double precision version of DL_POLY, we note that the first call to the 3D FFT function within each reciprocal SPME call is preceded by the conversion of real double precision data to double precision complex data, which then forms the input data to the FFT. In our proposed method, the real double precision data is converted to single precision complex data.

5 Numerical results

All of our tests were run on the Hartree Centre's Napier Compute System [1], which consists of 360 nodes containing 2 x12 core Intel Xeon processors (IvyBridge E5-2697v2 2.7GHz) and 64GB RAM. The interconnect is Infiniband from Mellanox (FDR Connect-IB 56 GB/s). In all our runs, 24 processes per node were requested. DL_POLY 4.09 was used as our base code and all double precision runs use this version. DL_POLY was compiled using Intel MPI (version 5.1.1) and its `mpif90` wrapper, which points to the GNU compiler `gfortran`. The flag `-pg` was used to enable profiling and we always set the environment variable `GMON_OUT_PREFIX` to `gmon.out-‘/bin/uname -n‘` to save the separate profiles for each MPI process.

For each test problem, number of MPI processes, n_p, and choice of precision used within the 3D FFT method, five separate runs were performed and the `gprof` profiler was used to analyse the timing profiles across all of the processes. In the following, we report times that are averaged (mean) across all five runs and all MPI processes. We are particularly interested in

fft_3d the total time spent performing the 3D FFT kernel by an MPI process;

spme the total time spent performing the Ewald SPME kernel by an MPI process;

total the total DL_POLY execution time for an MPI process.

5.1 Standard DL_POLY test cases

As part of DL_POLY, 28 small test problems are provided. Of these, 14 use the Ewald SPME method and we also immediately discounted test problems 01 and 03 because total execution time for the SPME method on a single processor was a fraction of a second. In Table 1, we list the test problems used and their attributes: the problems are numbered as in the DL_POLY Manual [7]. When comparing the use of the mixed precision and double precision versions of DL_POLY, we found that the mixed precision version that there was very little or no loss in accuracy in the output from DL_POLY.

In Table 2, we list the (mean) time spent by each MPI process in the various kernels (listed in Section 5) when either the mixed precision or double precision versions of the DaFT are used within the DL_POLY run. The times in bold are those that are at most 2% larger than the best time.

Problem	Total system size	Temperature(K)	Description
02	51737 atoms	300	200 DMPC molecules in 9379 water molecules
04	99120 atoms	300	8 Gramacidin A molecules in 32096 water molecules
07	12428 atoms	300	Lipid bilayer in water
08 and 09	8000 charged points and 4000 shells	3000	MgO with adiabatic and with relaxed shell model
10	500 ions and 39000 atoms	300	Potential of mean force on K+ in water
18	34992 atoms	25	SPC IceVII Water with constraint bonds
19	34992 atoms	25	SPC IceVII Water with rigid bodies
20	28816 atoms	295	64 NaCl ion pairs with 4480 water molecules represented by constraint bonds and 4416 water molecules represented by rigid bodies
21	29052 particles	295	7263 TIP4P rigid body water molecules
22	44352 ions	400	Ionic liquid dimethylimidazolium
23	23712 ions	310	600 molecules of calcite in 6904 water molecules

Table 1: Standard DL_POLY test problems using the Ewald SPME Method and their attributes. Test problems 01 and 03 excluded due to them having very small execution times.

		fft_3d			*spme*			*total*			*fft_3d/total*	
Problem	n_p	mixed	double	ratio	mixed	double	ratio	mixed	double	ratio	mixed	double
02	1	**4.15**	5.26	0.79	**8.59**	9.67	0.89	**45.65**	46.85	0.97	0.091	0.112
02	2	**1.97**	2.31	0.85	**4.19**	4.55	0.92	**23.63**	**24.03**	0.98	0.083	0.096
02	4	**0.87**	1.10	0.79	**2.05**	2.29	0.90	**12.16**	12.44	0.98	0.072	0.088
02	8	**0.45**	0.52	0.87	**1.00**	1.08	0.92	**6.25**	**6.34**	0.99	0.072	0.082
04	1	0.77	**0.75**	1.03	**6.46**	**6.43**	1.00	**54.25**	**53.80**	1.01	0.014	0.014
04	2	0.37	**0.36**	1.03	**3.21**	**3.19**	1.01	**28.16**	**28.06**	1.00	0.013	0.013
04	4	0.18	**0.16**	1.13	**1.70**	**1.69**	1.01	**14.61**	**14.51**	1.01	0.012	0.011
04	8	0.09	**0.08**	1.13	**0.78**	**0.78**	1.00	**7.51**	**7.50**	1.01	0.012	0.011
07	1	**3.31**	**3.31**	1.00	**10.01**	**10.11**	0.99	**31.69**	**31.67**	1.00	0.104	0.105
07	2	**1.71**	**1.72**	0.99	**5.06**	**5.14**	0.98	**16.80**	**16.88**	1.00	0.102	0.102
07	4	**0.85**	0.95	0.89	**2.65**	2.78	0.95	**9.36**	**9.45**	0.99	0.091	0.101
07	8	**0.45**	0.57	0.79	**1.41**	1.54	0.91	**4.90**	5.02	0.98	0.092	0.114
08	1	**0.12**	0.14	0.86	**0.85**	0.89	0.96	**2.81**	**2.81**	1.00	0.043	0.050
08	2	**0.05**	0.06	0.83	**0.40**	0.42	0.95	**1.51**	1.55	0.97	0.033	0.039
08	4	**0.02**	0.03	0.67	**0.21**	**0.21**	1.00	**0.83**	**0.84**	0.99	0.024	0.036
08	8	**0.01**	0.02	0.50	**0.12**	**0.12**	1.00	**0.43**	**0.44**	0.98	0.023	0.046
09	1	**1.19**	1.49	0.80	**9.05**	9.29	0.97	**28.74**	**28.95**	0.99	0.0414	0.0515
09	2	**0.62**	**0.63**	0.98	**4.16**	**4.23**	0.98	**15.96**	**15.94**	1.00	0.039	0.040
09	4	**0.28**	0.33	0.85	**2.18**	2.24	0.97	**8.73**	**8.77**	1.01	0.032	0.038
09	8	0.15	**0.13**	1.15	**1.19**	1.21	0.98	**4.34**	**4.39**	0.99	0.035	0.030
10	1	0.24	**0.22**	1.09	**2.49**	**2.50**	1.00	**19.65**	**19.70**	0.99	0.012	0.011
10	2	0.12	**0.11**	1.09	**1.32**	**1.34**	0.99	**10.52**	**10.57**	0.99	0.011	0.010
10	4	**0.07**	0.07	1.00	**0.62**	**0.61**	1.02	**5.38**	**5.42**	0.99	0.013	0.013
10	8	**0.03**	0.04	0.75	**0.33**	**0.33**	1.00	**2.85**	**2.85**	1.00	0.011	0.014
18	1	1.59	**1.34**	1.19	**14.84**	**14.59**	1.02	**56.95**	**56.42**	1.01	0.028	0.024
18	2	0.77	**0.70**	1.10	**7.77**	**7.77**	1.00	**30.09**	**30.04**	1.00	0.026	0.023
18	4	0.38	**0.34**	1.12	**3.62**	**3.58**	1.01	**15.27**	**15.24**	1.00	0.025	0.022
18	8	0.19	**0.17**	1.12	**1.90**	**1.88**	1.01	**8.38**	**8.40**	1.00	0.023	0.020
19	1	**0.34**	**0.34**	1.00	3.37	**3.31**	1.02	**14.27**	**14.21**	1.00	0.024	0.024
19	2	0.19	**0.14**	1.35	1.78	**1.73**	1.03	**7.56**	**7.49**	1.01	0.025	0.019
19	4	0.09	**0.08**	1.13	**0.81**	**0.82**	0.98	**3.81**	**3.81**	1.01	0.024	0.021
19	8	0.05	**0.03**	1.67	0.44	**0.43**	1.02	**2.00**	**1.99**	1.01	0.025	0.015
20	1	**0.30**	0.35	0.86	**4.33**	**4.40**	0.98	**24.65**	**24.71**	1.00	0.012	0.014
20	2	**0.13**	0.16	0.81	**2.17**	**2.18**	1.00	**12.82**	**12.78**	1.00	0.010	0.013
20	4	**0.07**	0.08	0.88	**1.06**	**1.08**	0.98	**6.63**	**6.63**	1.00	0.011	0.012
20	8	**0.03**	0.03	1.00	**0.59**	**0.58**	1.02	**3.52**	**3.51**	1.00	0.009	0.009
21	1	0.51	**0.44**	1.16	**7.38**	**7.28**	1.01	**41.43**	**41.32**	1.01	0.012	0.011
21	2	0.27	**0.25**	1.08	**3.37**	**3.36**	1.00	**21.10**	**21.10**	1.00	0.013	0.012
21	4	**0.12**	0.14	0.86	**1.79**	**1.81**	0.99	**11.09**	**11.11**	1.00	0.011	0.013
21	8	0.07	**0.06**	1.17	**1.00**	**1.00**	1.00	**5.83**	**5.82**	1.00	0.012	0.010
22	1	**0.40**	0.43	0.93	**3.15**	**3.18**	0.99	**13.67**	**13.63**	1.01	0.029	0.032
22	2	0.21	**0.20**	1.05	**1.58**	**1.58**	1.00	**7.17**	**7.13**	1.01	0.029	0.028
22	4	**0.10**	0.11	0.91	**0.75**	**0.75**	1.00	**3.78**	**3.79**	1.00	0.027	0.029
22	8	**0.06**	0.06	1.00	**0.40**	**0.40**	1.00	**2.08**	**2.07**	1.00	0.029	0.029
23	1	**0.04**	0.05	0.80	**1.53**	**1.52**	1.01	**12.80**	**12.93**	0.99	0.003	0.004
23	2	**0.02**	0.04	0.50	**0.82**	**0.83**	0.99	**6.77**	**6.81**	0.99	0.003	0.006
23	4	**0.02**	**0.02**	1.00	**0.37**	**0.37**	1.00	**3.46**	**3.48**	0.99	0.006	0.006
23	8	**0.01**	**0.01**	0.99	**0.20**	0.21	0.95	**1.82**	**1.83**	0.99	0.006	0.006

Table 2: Average time (seconds) per MPI process spent doing FFT calculations (*fft_3d*), doing SPME calcululations (*spme*)and executing DL_POLY (*total*). Timings for both the mixed precision and double precision versions are given and for 1,2,4 and 8 MPI processes.

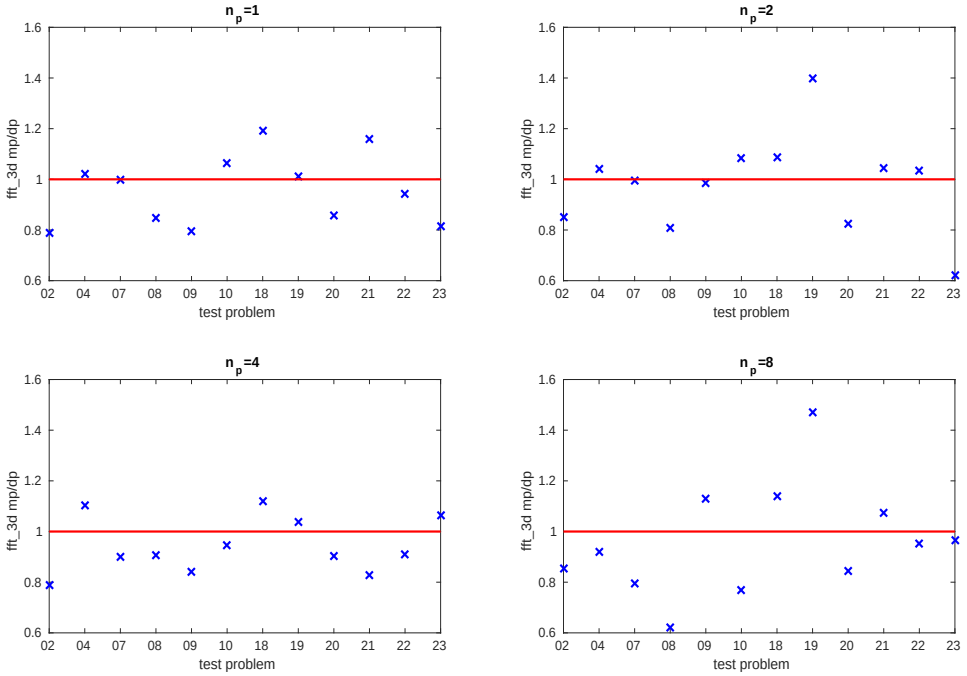

Figure 1: Ratio of the time spent performing the FFT calculations using mixed precision versus double precision.

In Figure 1, we look at the ratio of mean time spent by a process on the DaFT calculation when using mixed precision versus double precision. We observe that, for some problems using one MPI process, the time dropped by upto 20%. For one test problem, Test 08 with 8 MPI processes, there was a 40% drop in execution time when mixed precision was used instead of double precision. However, there were also problems where the reverse happened, for example, Test 19. For test problems 02 and 07, the DaFT kernel accounted for roughly 45% and 35% of the time in performing the Ewald SPME calculations. For the rest of the test problems, the DaFT kernel accounted for roughly 5-15% of the Ewald SMPE time. As a result, the savings or gains in using mixed precision for the SPME component are far more moderate when we consider the mean execution time across an MPI process, see Figure 2. In fact, the maximum savings is close to 10% (Test 02) and the largest increase is 3%.

The Ewald SPME calculations are just part of the overall force calculations being performed within a run of DL_POLY and, hence, the time spent performing the FFT computations is generally a small fraction of the total time. Therefore, the execution time gains/losses for the FFT computation will have an even smaller effect on the total execution time for DL_POLY. For test problems 02 and 07, roughly 10% of the time is spent on each MPI process is the FFT calculation; for five of the problems (04, 10, 20, 21 and 23), less than 2% of the time is spent in the FFT calculations.

It should be noted that, whilst these standard DL_POLY tests are not generally FFT dominant, for applications where the FFT is dominant, any savings in the FFT execution could be significant. Additionally, the size of these test problems do not replicate the typical nature of the molecular dynamics problems run using DL_POLY.

5.2 Enlarged DL_POLY test cases

In this section, we enlarge some of the test problems to enable us to examine the effect of communication between processes for the 3D FFT component and give a better idea of the mixed precision behaviour. To enlarge the test problems, we use DL_POLY's `nfold` facility. For test problem 02, we had to additionally alter the CONTROL

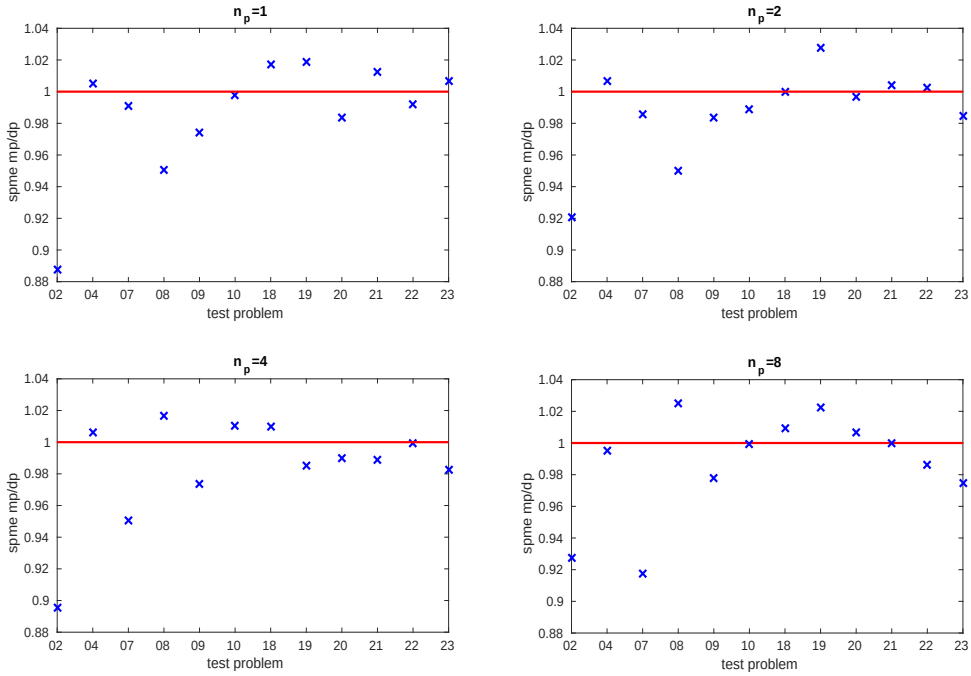

Figure 2: Ratio of the time spent performing the Ewald SPME calculations using mixed precision versus double precision.

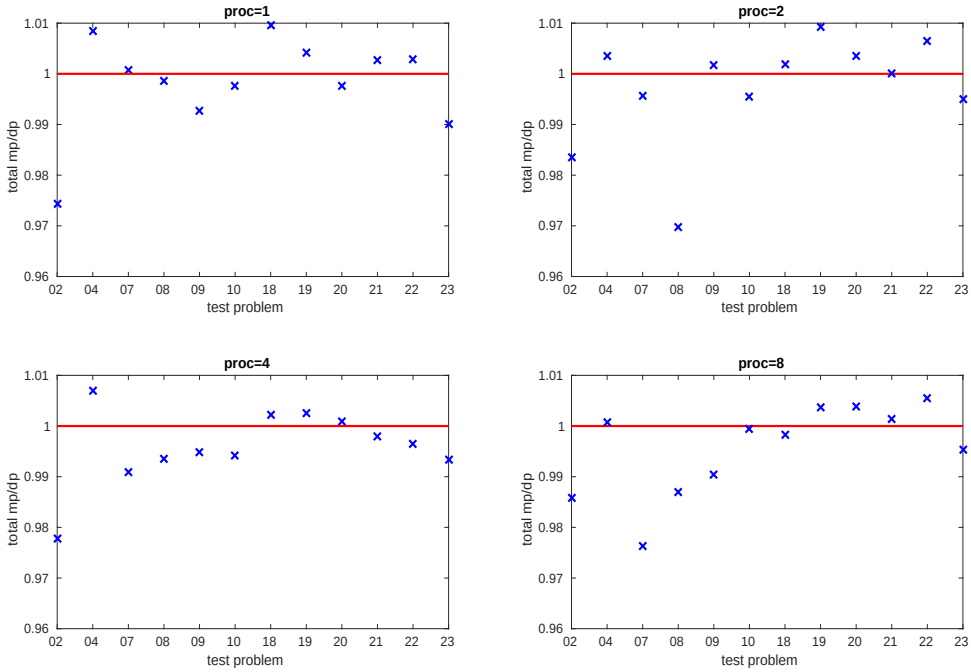

Figure 3: Ratio of the time spent running DL_POLY using mixed precision versus double precision.

file by appropriately scaling the maximum number of k-vector indices in the Ewald specification line. We append "b" to the test problem number for problems that were expanded by doubling the x, y and z dimensions; "c" is appended to the problem number for problems where the x, y and z dimensions are increased by a factor of 5. The original problem is replicated 8 (125) times and shifted appropriately to fill the enlarged space. With the enlarged systems, it is necessary to increase the number of MPI processes used to solve the problems.

		fft_3d			*spme*			*total*			*fft_3d/total*	
Problem	n_p	mixed	double	ratio	mixed	double	ratio	mixed	double	ratio	mixed	double
02b	8	**5.10**	6.77	0.75	**9.99**	11.76	0.85	**49.93**	51.69	0.97	0.102	0.132
02b	16	**2.75**	4.13	0.67	**5.48**	6.93	0.79	**26.36**	27.77	0.95	0.104	0.149
02b	24	**2.63**	5.05	0.52	**4.66**	7.17	0.65	**19.31**	21.77	0.89	0.136	0.232
02b	32	**1.38**	2.30	0.60	**2.81**	3.79	0.74	**13.78**	14.77	0.93	0.100	0.156
04b	8	**1.00**	1.25	0.80	**6.72**	7.00	0.96	**60.37**	**60.58**	1.00	0.017	0.021
04b	16	**0.49**	0.84	0.58	**3.45**	3.81	0.91	**32.49**	**32.72**	0.99	0.015	0.026
04b	24	**0.46**	1.06	0.43	**2.55**	3.16	0.81	**23.85**	**24.32**	0.98	0.019	0.043
04b	32	**0.24**	0.44	0.55	**1.80**	2.02	0.89	**16.76**	**16.89**	0.99	0.014	0.026
07b	8	**4.10**	4.72	0.87	**11.22**	11.94	0.94	**40.04**	**40.76**	0.98	0.102	0.116
07b	16	**2.51**	3.71	0.68	**6.46**	7.72	0.84	**23.3**	24.42	0.95	0.108	0.152
07b	24	**2.72**	4.84	0.56	**5.63**	7.85	0.72	**18.89**	21.04	0.90	0.144	0.230
07b	32	**1.40**	2.31	0.61	**3.55**	4.51	0.79	**12.74**	13.66	0.93	0.110	0.169
10b	8	**0.42**	0.47	0.88	**2.76**	2.83	0.98	**24.2**	**24.4**	0.99	0.017	0.019
10b	16	**0.23**	0.30	0.79	**1.50**	1.57	0.95	**13.1**	**13.2**	0.99	0.018	0.023
10b	24	**0.16**	0.25	0.62	**1.03**	1.13	0.91	**8.77**	**8.84**	0.99	0.018	0.028
10b	32	**0.11**	0.13	0.87	**0.72**	0.74	0.98	**6.65**	**6.76**	0.98	0.017	0.019
18b	8	**1.11**	1.15	0.96	**8.64**	**8.73**	0.99	**38.98**	**39.12**	1.00	0.029	0.029
18b	16	**0.58**	0.65	0.90	**4.66**	**4.75**	0.98	**20.76**	**20.88**	0.99	0.028	0.031
18b	24	**0.44**	0.63	0.69	**3.25**	3.44	0.94	**14.95**	**15.15**	0.99	0.029	0.042
18b	32	**0.31**	**0.34**	0.91	**2.29**	**2.32**	0.99	**11.12**	**11.13**	1.00	0.028	0.031
19b	8	**0.31**	0.33	**0.95**	**2.48**	**2.46**	1.01	**11.82**	**11.60**	1.02	0.026	0.028
19b	16	**0.17**	0.18	0.92	**1.29**	**1.30**	0.99	**6.19**	**6.16**	1.00	0.026	0.029
19b	24	**0.12**	0.18	0.66	**0.88**	0.95	0.92	**4.24**	4.3466	0.97	0.028	0.041
19b	32	**0.09**	0.10	0.86	**0.65**	0.67	0.98	**3.17**	**3.19**	0.99	0.028	0.031
02c	48	**27.66**	46.10	0.60	**45.00**	63.67	0.71	**174.11**	192.76	0.90	0.159	0.239
02c	96	**14.62**	29.07	0.50	**23.05**	38.65	0.60	**91.84**	106.61	0.86	0.159	0.273
02c	144	**9.83**	17.22	0.57	**15.00**	22.58	0.66	**62.68**	70.22	0.89	0.157	0.245
04c	96	**3.06**	5.32	0.58	**11.07**	13.42	0.82	**107.20**	109.20	0.98	0.029	0.049
04c	144	**2.37**	4.27	0.56	**7.76**	9.71	0.80	**72.34**	73.84	0.98	0.033	0.058
04c	192	**1.34**	2.60	0.52	**5.40**	6.70	0.81	**53.36**	54.42	0.98	0.025	0.048
18c	48	**2.16**	4.01	0.54	**8.76**	10.69	0.82	**56.48**	58.48	0.97	0.038	0.069
18c	96	**0.87**	1.66	0.53	**4.17**	5.02	0.83	**27.36**	28.22	0.97	0.032	0.059
18c	144	**0.57**	1.38	0.41	**2.81**	3.64	0.77	**17.07**	17.89	0.95	0.033	0.077

Table 3: Average time (seconds) per MPI process spent doing FFT calculations (*fft_3d*), doing SPME calculations (*spme*)and executing DL_POLY (*total*). Timings for both the mixed precision and double precision versions are given and for 1,2,4 and 8 MPI processes.

In Table 3, we list the (mean) time spent by each MPI process in the various kernels (listed in Section 5) when either the mixed precision or double precision versions of the DaFT are used within the DL_POLY run. The times in bold are those that are at most 2% larger than the best time. As the size of the test problem is scaled up by a factor 8 and then a factor of 125 (with respect to the original test problem), we observe that the mixed precision value of *fft_3d* approaches 0.5 times that when double precision is used. As noted in Section 5.1, the affect on the overall execution time depends on how dominant the FFT component is in the overall DL_POLY run. We note that, increasing the problem size has the effect of increasing the dominance of the FFT component. For the "b" problems, switching from 24 to 32 processes leads to a decrease in the dominance but this corresponds to

moving from using one to using two nodes. This need for off-node communication results in differing behaviours for different components within DL_POLY and, hence, different ratios. For the "c" test problem where the FFT is most dominant (Test 02c), the (wall-clock) execution time is decreased by 14% when mixed precision is used in the SPME calculation compared to using double precision; for the test problem where it is least dominant (Test 04c), there is a 2% decrease in execution: for large molecular dynamics problems taking many hours to run, this can still be a useful gain.

6 Conclusions

We conclude that a simple modification to the DAFT algorithm allowed us to turn it into a mixed precision method and halved the fast Fourier transform calculation in the larger test problems. There was little or no loss in accuracy when comparing the output from DL_POLY. For some of our test problems where the fast Fourier transform is a dominant component within DL_POLY, we were able to reduce the overall execution time of DL_POLY by 14%. The exact nature of the problem being solved will dictate the possible gains from using this mixed precision approach.

Acknowledgements The author would like to thank Alin Marin Elena and Michael Seaton for the discussions we had with regards using mixed precision within DL_POLY, and for proposing this work.

References

[1] See http://community.hartree.stfc.ac.uk/wiki/site/admin/resources.html.

[2] I. Bush, I. Todorov, and W. Smith, *A DAFT DL_POLY distributed memory adaptation of the Smoothed Particle Mesh Ewald method*, Computer Physics Communications, 175 (2006), pp. 323 – 329.

[3] S. Qi, S. Shi, and X. Wang, *Mixed Precision Method for GPU-based FFT*, 2013 IEEE 16th International Conference on Computational Science and Engineering, 00 (2011), pp. 580–586.

[4] W. Smith and I. T. Todorov, *DL_POLY_3: the CCP5 national UK code for molecular–dynamics simulations*, Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 362 (2004), pp. 1835–1852.

[5] H. S. Thorne, *Using mixed precision within DL_POLY's force and energy evaluations: short-range interactions*, Tech. Rep. RAL-TR-2018-004, 2018.

[6] H. S. Thorne, L. Mason, and A. D. Taylor, *Mixed precision: Is it the holy grail for software efficiency?*, in 2nd Conference of Research Software Engineers (RSE17), Manchester, U.K., September 7-8 2017. http://purl.org/net/epubs/work/34840371.

[7] I. Todorov and W. Smith, *The DL_POLY_4 User Manual (version 4.08)*. March 2016.