Implementation of energy and force grids in molecular simulations of porous materials

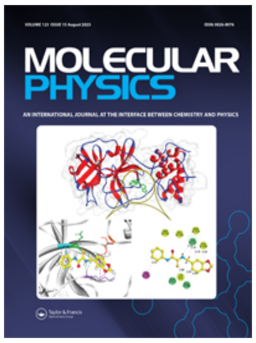
Ran, Youri A.; Tapia, Joseph; Sharma, Shrinjay; Bai, Peng; Calero, Sofia; Vlugt, Thijs J.H.; Dubbeldam, David

DOI
10.1080/00268976.2025.2542486

Publication date
2025

Document Version
Final published version

Published in
Molecular Physics



Molecular Physics

An International Journal at the Interface Between Chemistry and Physics

ISSN: 0026-8976 (Print) 1362-3028 (Online) Journal homepage: www.tandfonline.com/journals/tmph20

Implementation of energy and force grids in molecular simulations of porous materials

Youri A. Ran, Joseph Tapia, Shrinjay Sharma, Peng Bai, Sofia Calero, Thijs J. H. Vlugt & David Dubbeldam

To cite this article: Youri A. Ran, Joseph Tapia, Shrinjay Sharma, Peng Bai, Sofia Calero, Thijs J. H. Vlugt & David Dubbeldam (06 Aug 2025): Implementation of energy and force grids in molecular simulations of porous materials, Molecular Physics, DOI: 10.1080/00268976.2025.2542486

To link to this article: https://doi.org/10.1080/00268976.2025.2542486

MOLECULAR PHYSICS e2542486
https://doi.org/10.1080/00268976.2025.2542486

THERMODYNAMICS 2024 CONFERENCE (BY INVITATION ONLY)

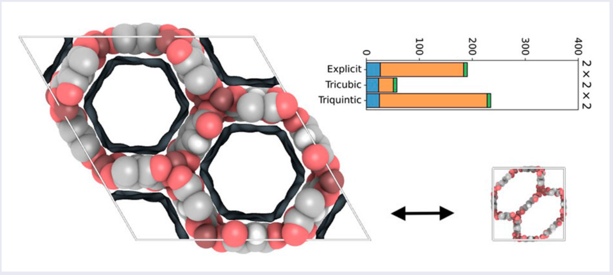
OPEN ACCESS Check for updates

Implementation of energy and force grids in molecular simulations of porous materials

Youri A. Ran [a], Joseph Tapia [b], Shrinjay Sharma [c], Peng Bai [b], Sofia Calero [c], Thijs J. H. Vlugt [d] and David Dubbeldam [a]

[a]Van 't Hoff Institute for Molecular Sciences, University of Amsterdam, Amsterdam, The Netherlands; [b]Chemical Engineering Department, University of Massachusetts Amherst, Amherst, MA, USA; [c]Department of Applied Physics and Science Education, Eindhoven University of Technology, Eindhoven, The Netherlands; [d]Engineering Thermodynamics, Process & Energy Department, Faculty of Mechanical, Maritime and Materials Engineering, Delft University of Technology, Delft, The Netherlands

ABSTRACT
Adsorption simulations often assume a rigid framework, which can be exploited by replacing the expensive framework-adsorbate energy/force evaluation by interpolation of a precomputed energy grid. We present the implementation in RASPA3 of a triquintic interpolation algorithm by Boateng and Bradach and compare it to the tricubic algorithm of Lekien and Marsden. We extended the scheme to interpolation in fractional space to facilitate interpolation of non-rectangular frameworks and evaluated the accuracy. We find that the use of grids is advantageous for larger systems and/or large cutoffs, but generally the efficiency gains are modest (a factor of 2–5).



ARTICLE HISTORY
Received 2 June 2025
Accepted 28 July 2025

KEYWORDS
Molecular simulation; adsorption; diffusion; interpolation

1. Introduction

Crystalline nanoporous materials like zeolites are considered stable and rigid materials. Since the very early work of Bezus et al. [1], the zeolite is usually modelled as a rigid crystal (known as the Kiselev model). Rigid-framework models enable the use of grid-interpolation techniques to determine the interaction of an adsorbate atom with the framework and avoid having to consider all framework atoms [2]. This can potentially speed up the energy calculation because the costly summation with all framework atoms is replaced with a simple interpolation that depends on system size and loading. However, the guest-guest interactions cannot be pre-tabulated.

June et al. used pretabulation of the sorbate atom-zeolite potentials on a fine three-dimensional grid of 0.2 Å spacing over the asymmetric unit of the unit cell [3]. Because of the orthorhombic symmetry (space group 'Pnma') of silicalite, this computation needed to be performed only over one-eighth of the unit cell. Sorbate-zeolite interaction potentials needed for the evaluation of the configurational integrals were then interpolated at run time from the tabulated potential data with a three-dimensional cubic spline (see Ref. [4]; The spline routines had been made available as QCPE 322 from the Quantum Chemistry Program Exchange, Indiana University, Bloomington, Indiana). This pretabulation scheme allowed for 'a 100-fold reduction' in computational effort over the direct pairwise summation of zeolite-sorbate interactions at every step of the simulation, using four-point Lagrangian interpolation [3].

CONTACT David Dubbeldam d.dubbeldam@uva.nl Van 't Hoff Institute for Molecular Sciences, University of Amsterdam, Science Park 904 Amsterdam 1098 XH, The Netherlands

Supplemental data for this article can be accessed at https://doi.org/10.1080/00268976.2025.2542486

Vlugt et al. developed a force field for united atom alkanes in zeolites using a fixed Lennard–Jones $\sigma = 3.6$ Å size-parameter for the framework atoms, and only allowed the Lennard–Jones strength parameter ϵ to change [5]. The strength-parameter ϵ is just a scaling parameter, and therefore only a single grid needed to be stored for $\epsilon = 1$ and for each united atom (CH_4, CH_3, CH_2, CH and C) the interpolated value was post-multiplied with its ϵ-value. The fixed σ-parameter was to avoid the huge memory requirements of using a separate grid for each of the five alkane sorbate atoms. To interpolate, a fifth order 3D polynomial scheme was used that interpolated first in z, then in y, and then in x [6]. To achieve fifth-order interpolation, two additional energy grid points to either side of the interpolation grid point were used (six in total).

Bai et al. developed the TraPPE-zeo model for both polar and nonpolar molecules in zeolites [7, 8]. This model explicitly included the interaction with the silicon atoms. Most models up to this point in time only used the interaction with the oxygen, assuming the oxygen parameters implicitly included the silicon interactions. Using a probe atom with Lennard–Jones parameters $\varepsilon = 1$, $\sigma = 1$, and unit charge $q = 1$, Bai created two grids for the silicon, tabulating $4\varepsilon(\sigma/r)^{12}$ and $4\varepsilon(\sigma/r)^{6}$ separately, and two grids for the oxygen framework atoms, and one grid for the Coulomb interactions including both real-space and reciprocal space contributions of the Ewald summation method. Therefore, instead of a separate grid for different pseudo-atoms, only five grid files were needed that could be scaled by $\varepsilon\sigma^{12}$, $\varepsilon\sigma^{6}$, and q and used for screening calculations of any sorbate functional groups.

In RASPA2 [9], we implemented the triclinic grid interpolation scheme in three dimensions of Lekien and Marsden [10, 11]. The algorithm is based on a specific 64×64 matrix that provides the relationship between the derivatives at the corners of the elements and the coefficients of the tricubic interpolant for this element. In sharp contrast to polynomial interpolation, the cubic interpolant and its first three derivatives are continuous and consistent. The same grids can therefore be used for both Monte Carlo (MC) and Molecular Dynamics (MD) with no additional energy drift besides the drift due to the integration scheme, i.e. the energy gradients are the exact derivatives of the energy at each point in the element.

In 2019, Walker extended triclinic interpolation to a quadcubic algorithm [12]. The quadcubic interpolation scheme can be applied to situations where there is one output, modelled as a function of a four-variable polynomial. For example, it can be applied to model ocean temperatures as they fluctuate over time. While both the tricubic and quadcubic interpolation schemes produce C^1 continuous polynomials, the lack of required free coefficients prevents them from achieving C^2 continuity. C^1 ensures continuous energy and forces, while C^2 ensures continuous energy, forces, and Hessians. In 2023, the tricubic algorithm was extended to a triquintic algorithm by Boateng and Bradach that involves a three-variate polynomial of degree 5 in each variable [13]. This method ensures C^2 continuity and, at interpolation, returns the energy, force, and the Hessian values. C^2 continuity ensures continuity in the Hessian, without which interpolated Hessian values cannot be deemed reliable.

In this article, we compare the triquintic algorithm by Boateng and Bradach with the tricubic algorithm by Lekien and Marsden. Both schemes have been implemented in the RASPA3 code [14]. For the triquintic scheme, we need derivatives up to order six. Up to order three can be found in the literature, and here we have derived expressions for the fourth, fifth, and sixth order derivatives for pair-potentials. In addition, we perform our interpolation in fractional space, instead of Cartesian space, to facilitate easy interpolation on non-rectangular frameworks. After explaining the methods and details of the implementation, we show and discuss the results for the efficiency and accuracy of the methods.

2. Methodology

2.1. Three-dimensional polynomial grid interpolation

The most common strategy is to break up the problem into a succession of one-dimensional interpolation. For an interpolation of order $k-1$ in x, y, and z [15]:

- We first locate the $k \times k \times k$ sub-block that contains our point x, y, z,
- Loop over the points in x-direction and interpolate into temporary storage for y
- Loop over the temporary storage points in y-direction and interpolate into temporary storage for z
- Loop over the temporary storage points in z-direction and do the final interpolation

Note that the interpolation might be order-dependent. The tricubic scheme, discussed next, has a cheaper computational cost and allows for a much easier and more accurate computation of higher derivatives of the extrapolated data [10].

2.2. Tricubic interpolation energy/force grids

Lekien and Marsden have developed a tricubic interpolation scheme in three dimensions which is both C^1 and isotropic [10]. The algorithm relies on a specific 64×64

matrix B that establishes the relationship between the derivatives at the element corners, stored in vector b, and the coefficients of the tricubic interpolant a_{ijk} for the respective element. Unlike global interpolation, where the interpolated function typically depends on the entire data set, this tricubic local interpolation utilises only the data within the direct vicinity of an individual element.

The algorithm is defined on a unit-cube with fractional coordinates $\{x, y, z\}$. The energy value at this point is approximated through a summation of polynomials of degree 3:

$$U(x,y,z) = \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} a_{ijk}x^i y^j z^k, \tag{1}$$

The coefficients a_{ijk} are determined using

$$\bar{a} = B^{-1}\bar{b} \tag{2}$$

where the matrix B^{-1} is constructed in such a way that the energy and the gradients are continuous (this 64×64 matrix forms the core of the tricubic interpolator), and the vector $\bar{b}$ contains for each of the eight corner vertices: the energy, and various energy derivatives, in units of energy and energy per distance to the power i, j, and k, respectively. The Supporting information S-I describes in more detail how the coefficients a_{ijk} are defined. The gradients can be obtained as

$$\frac{\partial U(x,y,z)}{\partial x} = \sum_{i=1}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} i a_{ijk}x^{i-1} y^j z^k \tag{3}$$

$$\frac{\partial U(x,y,z)}{\partial y} = \sum_{i=0}^{3}\sum_{j=1}^{3}\sum_{k=0}^{3} j a_{ijk}x^{i} y^{j-1} z^k \tag{4}$$

$$\frac{\partial U(x,y,z)}{\partial z} = \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=1}^{3} k a_{ijk}x^{i} y^j z^{k-1} \tag{5}$$

At each of the corner points, 8 values are stored for tricubic interpolation: U, $\partial U/\partial x$, $\partial U/\partial y$, $\partial U/\partial z$, $\partial^2 U/\partial x\partial y$, $\partial^2 U/\partial x\partial z$, $\partial^2 U/\partial y\partial z$, and $\partial^3 U/\partial x\partial y\partial z$. The tricubic method also provides continuous first derivatives and is symmetric in all three variables (isotropic).

2.3. *Triquintic interpolation energy/force grids*

Boateng and Bradach developed a triquintic algorithm that involves a trivariate polynomial of degree 5 in each variable [13]. This yields a polynomial of degree 15. By increasing the degree of the polynomial, we ensure that global C^2 continuity is met. The resulting triquintic polynomial, like the tricubic, is also isotropic but achieves global C^2 continuity. Triquintic interpolation, in three dimensions, results in a 216 term polynomial:

$$U\left(x,y,z\right) = \sum_{i=0}^{5}\sum_{j=0}^{5}\sum_{k=0}^{5} a_{ijk}x^i y^j z^k \tag{6}$$

At each of the corner points, 27 values are stored for triquintic interpolation: U, $\partial U/\partial x$, $\partial U/\partial y$, $\partial U/\partial z$, $\partial^2 U/\partial x^2$, $\partial^2 U/\partial x\partial y$, $\partial^2 U/\partial x\partial z$, $\partial^2 U/\partial y^2$, $\partial^2 U/\partial y\partial z$, $\partial^2 U/\partial z^2$, $\partial^3 U/\partial x^2\partial y$, $\partial^3 U/\partial x^2\partial z$, $\partial^3 U/\partial x\partial y^2$, $\partial^3 U/\partial x\partial y\partial z$, $\partial^3 U/\partial y^2\partial z$, $\partial^3 U/\partial x\partial z^2$, $\partial^3 U/\partial y\partial z^2$, $\partial^4 U/\partial x^2\partial y^2$, $\partial^4 U/\partial x^2\partial z^2$, $\partial^4 U/\partial y^2\partial z^2$, $\partial^4 U/\partial x^2\partial y\partial z$, $\partial^4 U/\partial x\partial y^2\partial z$, $\partial^4 U/\partial x\partial y\partial z^2$, $\partial^5 U/\partial x^2\partial y^2\partial z$, $\partial^5 U/\partial x^2\partial y\partial z^2$, $\partial^5 U/\partial x\partial y^2\partial z^2$, and $\partial^6 U/\partial x^2\partial y^2\partial z^2$. These, like in the tricubic interpolation scheme, are used to determine polynomial coefficients as described in Section S-I.

The algorithm is based on a 216×216 matrix, which provides the relationship between the undetermined polynomial coefficients and the observed data that we wish to interpolate. The algorithm provides increased accuracy and smoothness compared to the tricubic interpolation scheme with global C^1 continuity. The gradients can be obtained as

$$\frac{\partial U(x,y,z)}{\partial x} = \sum_{i=1}^{5}\sum_{j=0}^{5}\sum_{k=0}^{5} i a_{ijk}x^{i-1} y^j z^k \tag{7}$$

$$\frac{\partial U(x,y,z)}{\partial y} = \sum_{i=0}^{5}\sum_{j=1}^{5}\sum_{k=0}^{5} j a_{ijk}x^{i} y^{j-1} z^k \tag{8}$$

$$\frac{\partial U(x,y,z)}{\partial z} = \sum_{i=0}^{5}\sum_{j=0}^{5}\sum_{k=1}^{5} k a_{ijk}x^{i} y^j z^{k-1} \tag{9}$$

and the second derivatives can be obtained as

$$\frac{\partial^2 U(x,y,z)}{\partial x^2} = \sum_{i=2}^{5}\sum_{j=0}^{5}\sum_{k=0}^{5} i(i-1) a_{ijk}x^{i-2} y^j z^k \tag{10}$$

$$\frac{\partial^2 U(x,y,z)}{\partial x\partial y} = \sum_{i=1}^{5}\sum_{j=1}^{5}\sum_{k=0}^{5} ij a_{ijk}x^{i-1} y^{j-1} z^k \tag{11}$$

$$\frac{\partial^2 U(x,y,z)}{\partial x\partial z} = \sum_{i=1}^{5}\sum_{j=0}^{5}\sum_{k=1}^{5} ik a_{ijk}x^{i-1} y^j z^{k-1} \tag{12}$$

$$\frac{\partial^2 U(x,y,z)}{\partial y^2} = \sum_{i=0}^{5}\sum_{j=2}^{5}\sum_{k=0}^{5} j(j-1) a_{ijk}x^{i} y^{j-2} z^k \tag{13}$$

$$\frac{\partial^2 U(x,y,z)}{\partial y\partial z} = \sum_{i=0}^{5}\sum_{j=1}^{5}\sum_{k=1}^{5} jk a_{ijk}x^{i} y^{j-1} z^{k-1} \tag{14}$$

$$\frac{\partial^2 U(x,y,z)}{\partial z^2} = \sum_{i=0}^{5}\sum_{j=0}^{5}\sum_{k=2}^{5} k(k-1) a_{ijk}x^{i} y^j z^{k-2} \tag{15}$$

Note that taking the derivative of the function increases the error by approximately one order of magnitude. This is expected, since taking the derivative of a polynomial reduces the degree by one, via the power rule [16]. To produce a global C^3 continuous polynomial, a polynomial of degree 7 or higher would be required [16]. To compute the higher order derivatives for pair potentials, it is convenient to define

$$f_1^\dagger(r) = \frac{1}{r}\frac{\partial U(r)}{\partial r} = \frac{1}{r}f_1 \tag{16}$$

$$f_2^\dagger(r) = \frac{1}{r}\frac{\partial f_1^\dagger(r)}{\partial r} = \frac{f_2}{r^2} - \frac{f_1}{r^3} \tag{17}$$

$$f_3^\dagger(r) = \frac{1}{r}\frac{\partial f_2^\dagger(r)}{\partial r} = \frac{f_3}{r^3} - \frac{3f_2}{r^4} + \frac{3f_1}{r^5} \tag{18}$$

$$f_4^\dagger(r) = \frac{1}{r}\frac{\partial f_3^\dagger(r)}{\partial r} = \frac{f_4}{r^4} - \frac{6f_3}{r^5} + \frac{15f_2}{r^6} - \frac{15f_1}{r^7} \tag{19}$$

$$f_5^\dagger(r) = \frac{1}{r}\frac{\partial f_4^\dagger(r)}{\partial r} = \frac{f_5}{r^5} - \frac{10f_4}{r^6} + \frac{45f_3}{r^7} - \frac{105f_2}{r^8} + \frac{105f_1}{r^9} \tag{20}$$

$$f_6^\dagger(r) = \frac{1}{r}\frac{\partial f_5^\dagger(r)}{\partial r} = \frac{f_6}{r^6} - \frac{15f_5}{r^7} + \frac{105f_4}{r^8} - \frac{420f_3}{r^9} + \frac{945f_2}{r^{10}} - \frac{945f_1}{r^{11}} \tag{21}$$

For the Lennard–Jones potentials, these are given by:

$$U(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] \tag{22}$$

$$f_1^\dagger(r) = \frac{1}{r}\frac{\partial U(r)}{\partial r} = -24\varepsilon\left[2\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]\frac{1}{r^2} \tag{23}$$

$$f_2^\dagger(r) = \frac{1}{r}\frac{\partial f_1^\dagger(r)}{\partial r} = 96\varepsilon\left[7\left(\frac{\sigma}{r}\right)^{12} - 2\left(\frac{\sigma}{r}\right)^{6}\right]\frac{1}{r^4} \tag{24}$$

$$f_3^\dagger(r) = \frac{1}{r}\frac{\partial f_2^\dagger(r)}{\partial r} = -384\varepsilon\left[28\left(\frac{\sigma}{r}\right)^{12} - 5\left(\frac{\sigma}{r}\right)^{6}\right]\frac{1}{r^6} \tag{25}$$

$$f_4^\dagger(r) = \frac{1}{r}\frac{\partial f_3^\dagger(r)}{\partial r} = 4608\varepsilon\left[42\left(\frac{\sigma}{r}\right)^{12} - 5\left(\frac{\sigma}{r}\right)^{6}\right]\frac{1}{r^8} \tag{26}$$

$$f_5^\dagger(r) = \frac{1}{r}\frac{\partial f_4^\dagger(r)}{\partial r} = -322{,}560\varepsilon\left[12\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]\frac{1}{r^{10}} \tag{27}$$

$$f_6^\dagger(r) = \frac{1}{r}\frac{\partial f_5^\dagger(r)}{\partial r} = 2{,}580{,}480\varepsilon\left[33\left(\frac{\sigma}{r}\right)^{12} - 2\left(\frac{\sigma}{r}\right)^{6}\right]\frac{1}{r^{12}} \tag{28}$$

For the real part of the Ewald summation, these expressions read

$$U(r) = q_1 q_2 \frac{\mathrm{erfc}(\alpha r)}{r} \tag{29}$$

$$f_1^\dagger(r) = -q_1 q_2\left(\frac{2\alpha r e^{-(\alpha r)^2}}{\sqrt{\pi}} + \mathrm{erfc}(\alpha r)\right)\frac{1}{r^3} \tag{30}$$

$$f_2^\dagger(r) = q_1 q_2\left(\frac{2\alpha r e^{-(\alpha r)^2}\left(2(\alpha r)^2 + 3\right)}{\sqrt{\pi}} + 3\mathrm{erfc}(\alpha r)\right)\frac{1}{r^5} \tag{31}$$

$$f_3^\dagger(r) = -q_1 q_2\left(\frac{2\alpha r e^{-(\alpha r)^2}\left(4(\alpha r)^4 + 10(\alpha r)^2 + 15\right)}{\sqrt{\pi}} + 15\mathrm{erfc}(\alpha r)\right)\frac{1}{r^7} \tag{32}$$

$$f_4^\dagger(r) = q_1 q_2\left(\frac{2\alpha r e^{-(\alpha r)^2}\left(8(\alpha r)^6 + 28(\alpha r)^4 + 70(\alpha r)^2 + 105\right)}{\sqrt{\pi}} + 105\mathrm{erfc}(\alpha r)\right)\frac{1}{r^9} \tag{33}$$

$$f_5^\dagger(r) = -q_1 q_2\left(\frac{2\alpha r e^{-(\alpha r)^2}\left(16(\alpha r)^8 + 72(\alpha r)^6 + 252(\alpha r)^4 + 630(\alpha r)^2 + 945\right)}{\sqrt{\pi}} + 945\mathrm{erfc}(\alpha r)\right)\frac{1}{r^{11}} \tag{34}$$

$$f_6^\dagger(r) = q_1 q_2\left(\frac{2\alpha r e^{-(\alpha r)^2}\left(32(\alpha r)^{10} + 176(\alpha r)^8 + 792(\alpha r)^6 + 2772(\alpha r)^4 + 6930(\alpha r)^2 + 10{,}395\right)}{\sqrt{\pi}} + 10{,}395\mathrm{erfc}(\alpha r)\right)\frac{1}{r^{13}} \tag{35}$$

The derivatives of $U(r)$ with respect to Cartesian coordinates are given by

$$\frac{\partial U(r)}{\partial r_i} = f_1^\dagger r_i \tag{36}$$

The second derivatives are easily derived as [17]

$$\frac{\partial^2 U(r)}{\partial r_i \partial r_j} = f_2^\dagger r_i r_j + \delta_{ij} f_1^\dagger \tag{37}$$

where δ_{ij} is the Kronecker delta function ($\delta_{ij} = 1$ if $i = j$, and 0 otherwise). The third derivatives can be found in the manual of GULP [18, 19]:

$$\frac{\partial^3 U(r)}{\partial r_i \partial r_j \partial r_k} = f_3^\dagger r_i r_j r_k + f_2^\dagger \left(\delta_{jk} r_i + \delta_{ik} r_j + \delta_{ij} r_k\right) \tag{38}$$

To the best of our knowledge, the fourth, fifth, and sixth order derivatives have not appeared in the literature and are derived here. The fourth derivatives read:

$$\begin{aligned}\frac{\partial^4 U(r)}{\partial r_i \partial r_j \partial r_k \partial r_l} &= f_4^\dagger r_i r_j r_k r_l \\ &+ f_3^\dagger \left(\delta_{ij} r_k r_l + \delta_{ik} r_j r_l + \delta_{il} r_j r_k + \delta_{jk} r_i r_l \right.\\ &\left. + \delta_{jl} r_i r_k + \delta_{kl} r_i r_j\right) \\ &+ f_2^\dagger \left(\delta_{ij}\delta_{kl} + \delta_{il}\delta_{jk} + \delta_{ik}\delta_{jl}\right)\end{aligned} \tag{39}$$

The expressions for the fifth and sixth-order derivatives become more tedious. The fifth derivatives of an isotropic (radial) function must be a linear combination of the fully symmetrised products of $\{r_i r_j r_k r_l r_m\}$, $\{\delta_{ij} r_k r_l r_m\}$, and $\{\delta_{ij}\delta_{kl} r_m\}$

$$\begin{aligned}\frac{\partial^5 U(r)}{\partial r_i \partial r_j \partial r_k \partial r_l \partial r_m} &= f_5^\dagger r_i r_j r_k r_l r_m \\ &+ f_4^\dagger (\delta_{ij} r_k r_l r_m + \delta_{ik} r_j r_l r_m + \delta_{il} r_j r_k r_m \\ &+ \delta_{im} r_j r_k r_l + \delta_{jk} r_i r_l r_m \\ &+ \delta_{jl} r_i r_k r_m + \delta_{jm} r_i r_k r_l + \delta_{kl} r_i r_j r_m \\ &+ \delta_{km} r_i r_j r_l + \delta_{lm} r_i r_j r_k) \\ &+ f_3^\dagger (\delta_{jk}\delta_{lm} r_i + \delta_{jm}\delta_{kl} r_i + \delta_{jl}\delta_{km} r_i + \delta_{kl}\delta_{mi} r_j \\ &+ \delta_{ki}\delta_{lm} r_j + \delta_{km}\delta_{li} r_j + \delta_{lm}\delta_{ij} r_k + \delta_{lj}\delta_{mi} r_k \\ &+ \delta_{li}\delta_{mj} r_k + \delta_{mi}\delta_{jk} r_l \\ &+ \delta_{mk}\delta_{ij} r_l + \delta_{mj}\delta_{ik} r_l + \delta_{ij}\delta_{kl} r_m + \delta_{il}\delta_{jk} r_m + \delta_{ik}\delta_{jl} r_m)\end{aligned} \tag{40}$$

A more compact way to write it is:

$$\frac{\partial^5 U(r)}{\partial r_i \partial r_j \partial r_k \partial r_l \partial r_m} = f_5^\dagger r_i r_j r_k r_l r_m + f_4^\dagger \sum_{\text{sym}} \delta_{ij} r_k r_l r_m + f_3^\dagger \sum_{\text{sym}} \delta_{ij}\delta_{kl} r_m \tag{41}$$

where the notation $\sum_{\text{sym}}$ means the sum over all distinct ways of contracting the indicated pairs of indices (i.e. over all permutations of i, j, k, l, m consistent with making the term symmetric in all five indices). Similarly, the sixth-order derivatives are derived as

$$\begin{aligned}\frac{\partial^6 U(r)}{\partial r_i \partial r_j \partial r_k \partial r_l \partial r_m \partial r_n} &= f_6^\dagger r_i r_j r_k r_l r_m r_n + f_5^\dagger (\delta_{ij} r_k r_l r_m r_n \\ &+ \delta_{ik} r_j r_l r_m r_n + \delta_{il} r_j r_k r_m r_n + \delta_{im} r_j r_k r_l r_n + \delta_{in} r_j r_k r_l r_m \\ &+ \delta_{jk} r_i r_l r_m r_n + \delta_{jl} r_i r_k r_m r_n + \delta_{jm} r_i r_k r_l r_n + \delta_{jn} r_i r_k r_l r_m \\ &+ \delta_{kl} r_i r_j r_m r_n + \delta_{km} r_i r_j r_l r_n + \delta_{kn} r_i r_j r_l r_m + \delta_{lm} r_i r_j r_k r_n \\ &+ \delta_{ln} r_i r_j r_k r_m + \delta_{mn} r_i r_j r_k r_l) \\ &+ f_4^\dagger (\delta_{kl}\delta_{mn} r_i r_j + \delta_{km}\delta_{ln} r_i r_j + \delta_{lm}\delta_{kn} r_i r_j \\ &+ \delta_{jl}\delta_{mn} r_i r_k + \delta_{jm}\delta_{ln} r_i r_k + \delta_{jn}\delta_{lm} r_i r_k \\ &+ \delta_{jk}\delta_{mn} r_i r_l + \delta_{jm}\delta_{kn} r_i r_l + \delta_{jn}\delta_{km} r_i r_l + \delta_{jk}\delta_{ln} r_i r_m \\ &+ \delta_{jl}\delta_{kn} r_i r_m + \delta_{jn}\delta_{kl} r_i r_m \\ &+ \delta_{jk}\delta_{lm} r_i r_n + \delta_{jl}\delta_{km} r_i r_n + \delta_{jm}\delta_{kl} r_i r_n \\ &+ \delta_{il}\delta_{mn} r_j r_k + \delta_{im}\delta_{ln} r_j r_k + \delta_{in}\delta_{lm} r_j r_k \\ &+ \delta_{ik}\delta_{mn} r_j r_l + \delta_{im}\delta_{kn} r_j r_l + \delta_{in}\delta_{km} r_j r_l \\ &+ \delta_{ik}\delta_{ln} r_j r_m + \delta_{il}\delta_{kn} r_j r_m + \delta_{in}\delta_{kl} r_j r_m \\ &+ \delta_{ik}\delta_{lm} r_j r_n + \delta_{il}\delta_{km} r_j r_n + \delta_{im}\delta_{kl} r_j r_n \\ &+ \delta_{ij}\delta_{mn} r_k r_l + \delta_{im}\delta_{jn} r_k r_l + \delta_{in}\delta_{jm} r_k r_l \\ &+ \delta_{ij}\delta_{ln} r_k r_m + \delta_{il}\delta_{jn} r_k r_m + \delta_{in}\delta_{jl} r_k r_m \\ &+ \delta_{ij}\delta_{lm} r_k r_n + \delta_{il}\delta_{jm} r_k r_n + \delta_{im}\delta_{jl} r_k r_n \\ &+ \delta_{ij}\delta_{kn} r_l r_m + \delta_{ik}\delta_{jn} r_l r_m + \delta_{in}\delta_{jk} r_l r_m + \delta_{ij}\delta_{km} r_l r_n \\ &+ \delta_{ik}\delta_{jm} r_l r_n + \delta_{im}\delta_{jk} r_l r_n \\ &+ \delta_{ij}\delta_{kl} r_m r_n + \delta_{ik}\delta_{jl} r_m r_n + \delta_{il}\delta_{jk} r_m r_n) \\ &+ f_3^\dagger (\delta_{ij}\delta_{kl}\delta_{mn} + \delta_{ij}\delta_{kn}\delta_{lm} + \delta_{ij}\delta_{km}\delta_{ln} \\ &+ \delta_{ik}\delta_{jl}\delta_{mn} + \delta_{ik}\delta_{jm}\delta_{ln} \\ &+ \delta_{ik}\delta_{jn}\delta_{lm} + \delta_{il}\delta_{jk}\delta_{mn} \\ &+ \delta_{il}\delta_{jm}\delta_{kn} + \delta_{il}\delta_{jn}\delta_{km} + \delta_{im}\delta_{jk}\delta_{ln} \\ &+ \delta_{im}\delta_{jl}\delta_{kn} + \delta_{im}\delta_{jn}\delta_{kl} \\ &+ \delta_{in}\delta_{jk}\delta_{lm} + \delta_{in}\delta_{jl}\delta_{km} + \delta_{in}\delta_{jm}\delta_{kl})\end{aligned} \tag{42}$$

written more succinctly as

$$\begin{aligned}\frac{\partial^6 U(r)}{\partial r_i \partial r_j \partial r_k \partial r_l \partial r_m \partial r_n} &= f_6^\dagger r_i r_j r_k r_l r_m r_n + f_5^\dagger \sum_{\text{sym}} \delta_{ij} r_k r_l r_m r_n \\ &+ f_4^\dagger \sum_{\text{sym}} \delta_{ij}\delta_{kl} r_m r_n\end{aligned}$$

$$+f_3^\dagger \sum_{\text{sym}} \delta_{ij}\delta_{kl}\delta_{mn} \tag{43}$$

where the first term is just all indices 'carried' by r's, the second term is the sum over all ways of picking one pair of indices δ_{ij} with the remaining four indices carried by r, the third term is the sum over all ways of picking two disjoint pairs of indices for two Kronecker deltas $\delta_{ij}\delta_{kl}$, and the last term is the sum over all ways of picking three disjoint pairs of indices to form three Kronecker delta's $\delta_{ij}\delta_{kl}\delta_{mn}$.

2.4. Cartesian vs. fractional space

In general, the unit cell is defined by the cell lengths a, b, c and angles α, β, γ, and by the fractional coordinates of the atoms within the unit cell. Fractional coordinates lie in a dimensionless, orthonormal space. The transformation from fractional space to Cartesian space can be carried out by the matrix $\mathbf{h}$ [20]:

$$\mathbf{h} = \begin{pmatrix} a & b\cos(\gamma) & c\cos(\beta) \\ 0 & b\sin(\gamma) & c\zeta \\ 0 & 0 & c\sqrt{1-\cos^2\beta-\zeta^2} \end{pmatrix} \tag{44}$$

with

$$\zeta = \frac{\cos\alpha - \cos\gamma\,\cos\beta}{\sin\gamma} \tag{45}$$

This aligns the a cell vector along the x axis, b in the xy-plane. Conversely, the inverse of the box matrix $\mathbf{h}^{-1}$ transforms real coordinates to fractional coordinates, with a total box length of 1. Our potential force field is defined in Cartesian space. Therefore, it is convenient to store positions in Cartesian space, transform them to fractional space, apply periodic boundary conditions, and transform back to Cartesian space to compute distances within the simulation box [20]

$$\begin{aligned} \mathbf{s} &= \mathbf{h}^{-1}\mathbf{r} \\ \mathbf{s}' &= \mathbf{s} - \text{rint}(\mathbf{s}) \\ \mathbf{r}' &= \mathbf{h}\mathbf{s}' \end{aligned} \tag{46}$$

where the 'rint'-function returns the rounded integer value of its argument. The smallest perpendicular width of the unit cell has to be larger than twice the spherical cutoff in Cartesian space to be consistent with the minimum image convention.

The gradient in fractional space $\mathbf{g}'$ can be obtained from the gradient $\mathbf{g}$ in Cartesian space using

$$\mathbf{g}' = \mathbf{h}^T\mathbf{g} \tag{47}$$

where subscript T denotes the transpose of the matrix. Likewise, the Hessian matrix in fractional space $\mathcal{H}'$ can be obtained from the Hessian matrix in Cartesian space $\mathcal{H}$ by using

$$\mathcal{H}' = \mathbf{h}^T\mathcal{H}\mathbf{h} \tag{48}$$

In general, the transformations of the first-, second-, third-, fourth-, fifth-, and sixth-order tensors, converted from Cartesian space T to fractional space T', read [21]

$$T'_i = h_{ip}T_p \tag{49}$$
$$T'_{ij} = h_{ip}h_{jq}T_{pq} \tag{50}$$
$$T'_{ijk} = h_{ip}h_{jq}h_{kr}T_{pqr} \tag{51}$$
$$T'_{ijkl} = h_{ip}h_{jq}h_{kr}h_{ls}T_{pqrs} \tag{52}$$
$$T'_{ijklm} = h_{ip}h_{jq}h_{kr}h_{ls}h_{mt}T_{pqrst} \tag{53}$$
$$T'_{ijklmn} = h_{ip}h_{jq}h_{kr}h_{ls}h_{mt}h_{nu}T_{pqrstu} \tag{54}$$

where the Einstein-summation convention has been used (repeated indices are implicitly summed over) [22]. For example, the sixth derivative in fractional space is explicitly written out as:

$$\begin{aligned} \frac{\partial^6 U}{\partial s_x^2 \partial s_y^2 \partial s_z^2} = {} & h_{xx}^2 h_{yx}^2 h_{zx}^2 T_{xxxxxx} + 2h_{xx}^2 h_{yx} h_{yy} h_{zx}^2 T_{xxxxxy} \\ & + h_{xx}^2 h_{yx}^2 h_{zx} h_{zy} T_{xxxxxy} \\ & + 2h_{xx}^2 h_{yx}^2 h_{zx} h_{zz} T_{xxxxxz} \\ & + h_{xx}^2 h_{yx}^2 h_{zx} h_{zy} T_{xxxxyx} \\ & + h_{xx}^2 h_{yy}^2 h_{zx}^2 T_{xxxxyy} \\ & + 4h_{xx}^2 h_{yx} h_{yy} h_{zx} h_{zy} T_{xxxxyy} \\ & + h_{xx}^2 h_{yx}^2 h_{zy}^2 T_{xxxxyy} \\ & + 4h_{xx}^2 h_{yx} h_{yy} h_{zx} h_{zz} T_{xxxxyz} \\ & + 2h_{xx}^2 h_{yx}^2 h_{zy} h_{zz} T_{xxxxyz} \\ & + h_{xx}^2 h_{yx}^2 h_{zz}^2 T_{xxxxzz} \\ & + 2h_{xx}^2 h_{yy}^2 h_{zx} h_{zy} T_{xxxyyy} \\ & + 2h_{xx}^2 h_{yx} h_{yy} h_{zy}^2 T_{xxxyyy} \\ & + 2h_{xx}^2 h_{yy}^2 h_{zx} h_{zz} T_{xxxyyz} \\ & + 4h_{xx}^2 h_{yx} h_{yy} h_{zy} h_{zz} T_{xxxyyz} \\ & + 2h_{xx}^2 h_{yx} h_{yy} h_{zz}^2 T_{xxxyzz} \\ & + h_{xx}^2 h_{yy}^2 h_{zy}^2 T_{xxyyyy} \\ & + 2h_{xx}^2 h_{yy}^2 h_{zy} h_{zz} T_{xxyyyz} \\ & + h_{xx}^2 h_{yy}^2 h_{zz}^2 T_{xxyyzz} \end{aligned} \tag{55}$$

This means that to convert a derivative from Cartesian to fractional, in general, many elements of the tensor are needed. However, for rectangular cells, the expressions simplify significantly.

The gradient in Cartesian space $\mathbf{g}$ can be obtained from the gradient $\mathbf{g}'$ in fractional space using

$$\mathbf{g} = \mathbf{h}^{-T}\mathbf{g}' \tag{56}$$

where the subscript $-T$ denotes the transpose-inverse of the matrix. Likewise, the Hessian matrix in Cartesian space $\mathcal{H}$ can be obtained from the Hessian matrix in fractional space $\mathcal{H}'$ by using

$$\mathcal{H} = \mathbf{h}^{-T}\mathcal{H}'\mathbf{h}^{-1} \tag{57}$$

In general, the transformations of the first- and second-order tensors, converted from fractional space T' to Cartesian space T, read

$$T_i = h_{ip}^{-1}T'_p \tag{58}$$

$$T_{ij} = h_{ip}^{-1}h_{jq}^{-1}T'_{pq} \tag{59}$$

2.5. Multiple cells in fractional space

The interpolation can be improved by using higher-order derivatives, but this has a limit and only works for polynomial data. It can also be improved by dividing the unit cube into many rectangular cuboids (see Figure 1). If we choose to divide the unit cube into N_{cells} cells in each direction, we need $N_{\text{grid-points}} + 1$ grid points in each direction.

It is convenient to start counting from zero. In that case, we can find the lower bound indices $\mathbf{g_l}$ and upper bound indices $\mathbf{g_u}$ of the grid cuboid as

$$\mathbf{g_l} = \lfloor \mathbf{s} * N_{\text{cells}} \rfloor \tag{60}$$

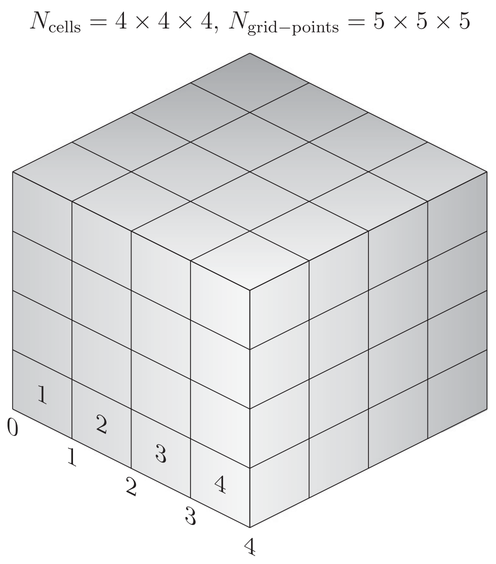

$N_{\text{cells}} = 4 \times 4 \times 4,\ N_{\text{grid-points}} = 5 \times 5 \times 5$
1
2
3
4
0
1
2
3
4

Figure 1. Rectilinear grid: the fractional unit cell is divided up into many rectangular cuboids.

$$\mathbf{g_u} = \lceil \mathbf{s} * N_{\text{cells}} \rceil = \mathbf{g_l} + \{1, 1, 1\} \tag{61}$$

where $\lfloor\ \ \rfloor$ is the floor of the number (i.e. the largest integer that does not exceed the argument floating point number), and $\lceil\ \ \rceil$ is the ceiling of the number (the smallest integer greater than or equal to the argument floating point number). The fractional position $\mathbf{t}$ within that cuboid is now

$$\mathbf{t} = \mathbf{s} * N_{\text{cells}} - \mathbf{g_l} \tag{62}$$

The spacing of the lattice is $\delta = 1/N_{\text{cells}}$ and the derivatives need to be corrected for this transformation:

$$\frac{\partial U}{\partial t_x} = \delta_x \frac{\partial U}{\partial s_x} \tag{63}$$

$$\frac{\partial U}{\partial t_y} = \delta_y \frac{\partial U}{\partial s_y} \tag{64}$$

$$\frac{\partial U}{\partial t_z} = \delta_z \frac{\partial U}{\partial s_z} \tag{65}$$

$$\frac{\partial^2 U}{\partial t_x \partial t_y} = \delta_x\delta_y \frac{\partial^2 U}{\partial s_x \partial s_y} \tag{66}$$

$$\frac{\partial^2 U}{\partial t_x \partial t_z} = \delta_x\delta_z \frac{\partial^2 U}{\partial s_x \partial s_z} \tag{67}$$

$$\frac{\partial^2 U}{\partial t_y \partial t_z} = \delta_y\delta_z \frac{\partial^2 U}{\partial s_y \partial s_z} \tag{68}$$

$$\frac{\partial^3 U}{\partial t_x \partial t_y \partial t_z} = \delta_x\delta_y\delta_z \frac{\partial^3 U}{\partial s_x \partial s_y \partial s_z} \tag{69}$$

and similarly, all the derivatives of the triquintic scheme can be transformed.

2.6. Tabulating the Lennard–Jones potential

In principle, we need to tabulate $4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$ for each ε, σ-pair. As pointed out by Bai [8], we can avoid storing a separate grid for each different σ-value when using Jorgensen-Berthelot mixing rules:

$$\sigma_{ij} = \sqrt{\sigma_{ii} \times \sigma_{jj}} \tag{70}$$

$$\varepsilon_{ij} = \sqrt{\varepsilon_{ii} \times \varepsilon_{jj}} \tag{71}$$

We only have to store two Lennard–Jones grids: $\left(\frac{\sigma^0}{r}\right)^{12}$ and $\left(\frac{\sigma^0}{r}\right)^{6}$, where σ^0 is a fixed, chosen reference σ value. In this work, the reference σ^0 has been chosen as $\sigma^0 = 1.0$ Å. The interpolated value for any $\varepsilon_{ii}, \sigma_{ii}$ pair is then

$$\mathcal{P}_{LJ_{12-6}} = \sqrt{\varepsilon_{ii}}\left[\left(\sigma_{ii}/\sigma^0\right)^6 \mathcal{P}_{LJ_{12}} - \left(\sigma_{ii}/\sigma^0\right)^3 \mathcal{P}_{LJ_6}\right] \tag{72}$$

For a shifted Lennard–Jones, we have to tabulate $\left(\frac{\sigma^0}{r}\right)^{12} - \left(\frac{\sigma^0}{r_c}\right)^{12}$ and $\left(\frac{\sigma^0}{r}\right)^{6} - \left(\frac{\sigma^0}{r_c}\right)^{6}$, where r_c is the cutoff distance. When using Lorentz–Berthelot mixing rules [23]

$$\sigma_{ij} = \left(\sigma_{ii} + \sigma_{jj}\right)/2 \tag{73}$$

$$\varepsilon_{ij} = \sqrt{\varepsilon_{ii} \times \varepsilon_{jj}} \tag{74}$$

a separate grid is required for each of the σ values.

2.7. Tabulating the electrostatics

A minor complication arises when the framework has a non-zero net electric charge compensated by counter ions. Let us assume the framework is kept fixed, but non-framework cations are allowed to move. Although the system as a whole may be electrically neutral, the omission of framework-framework interactions from the calculation also means that the Ewald Fourier contributions should be split into separate sums which are each net charged. Not only should the interactions of a framework atom with other framework atoms be omitted, but also the interactions with all its images.

Splitting of the potential energy into separate contributions involves computing cross terms between components of types A and B [24, 25], for example A as the framework, and B the adsorbates. In real space, this is trivially accomplished, but the reciprocal separation is more difficult. Cross term interaction energies in reciprocal space are given by

$$U_{A,B}^{\text{rec}} = \frac{2\pi}{V}\sum_{\mathbf{k}\neq 0}\frac{1}{k^2}e^{-\frac{k^2}{4\eta^2}}\left[\left(\sum_{i\in A} q_i\cos\left(\mathbf{k}\cdot\mathbf{r}_i\right)\right)\times\left(\sum_{i\in B} q_i\cos\left(\mathbf{k}\cdot\mathbf{r}_i\right)\right) + \left(\sum_{i\in A} q_i\sin\left(\mathbf{k}\cdot\mathbf{r}_i\right)\right)\left(\sum_{i\in B} q_i\sin\left(\mathbf{k}\cdot\mathbf{r}_i\right)\right)\right]. \tag{75}$$

The sums in Equation (75) are stored in memory. It is important to note that Equation (75) only applies when both the separate sums over species A and B are charge neutral. This is, in general, not the case. For example, in MC adsorption simulations, one often encounters a negatively charged zeolite charge-compensated by either cations or protons. All-silica zeolites devoid of cations are neutral, but framework ions like aluminum induce a net charge.

It is customary to treat charged periodic systems via a uniform neutralising background plasma [25]. However, this leads to serious artifacts in both system energy and pressure and leads to unrealistic behaviour. Bogusz et al. corrected these artifacts by instituting a net-charge correction term that consists of subtracting off the Ewald sum for a single particle with charge equal to the net charge [25]. This correction implicitly restores η-independence in net-charged systems. We define $\mathcal{U}_\zeta$ as the reciprocal energy of a single ion placed at the centre of charge $\mathbf{o}$

$$\mathcal{U}_\zeta = \frac{2\pi}{V}\sum_{\mathbf{k}\neq 0}\frac{1}{k^2}e^{-\frac{k^2}{4\eta^2}}\left[\left(\cos\left(\mathbf{k}\cdot\mathbf{o}\right)\right)^2 + \left(\sin\left(\mathbf{k}\cdot\mathbf{o}\right)\right)^2\right] - \frac{\eta}{\sqrt{\pi}}. \tag{76}$$

If only the energy is needed, the particle can simply be placed at the origin. Note that the term then only depends on the shape and size of the simulation cell and has to be computed only once if the cell does not change. The charge-charge interaction energy $U_{A,B}$ of component A and B is given by

$$U_{A,B} = U_{A,B}^{\text{real}} + U_{A,B}^{\text{rec}} + U_{A,B}^{\text{self}} + U^{\text{corr}} \tag{77}$$

$$U_{A,B}^{\text{self}} = -\left(1-\delta_B\right)\frac{\eta}{\sqrt{\pi}}\sum_{i\in A,B} q_i^2 \tag{78}$$

$$U^{\text{corr}} = \left(\sum_{i\in A} q_i\right)\left(\sum_{i\in B} q_i\right)\mathcal{U}_\zeta \tag{79}$$

where $\delta_{ij} = 1$ if $i = j$, zero otherwise, denotes the Kronecker delta. These expressions are valid in any kind of net-charge arrangement, including a net charge of the total system.

Note that it is also possible to precompute the electrostatic potential and use that for the adsorbate-framework grid interpolation. However, this scheme is more difficult to combine with the Ewald summation for adsorbate–adsorbate interactions when the framework has a net charge (compensated by mobile cations).

2.8. Expanded ensembles

In the original Continuous Fractional Component Monte Carlo (CFCMC) method [26, 27], and also in this work, Lennard–Jones interactions $u_{\text{LJ}}(r)$ and charge-charge interactions u_{Coul} are scaled as

$$u_{\text{LJ}}\left(r,\lambda\right) = \lambda 4\epsilon\left[\frac{1}{\left[\frac{1}{2}\left(1-\lambda\right)^2 + \left(\frac{r}{\sigma}\right)^6\right]^2} - \frac{1}{\left[\frac{1}{2}\left(1-\lambda\right)^2 + \left(\frac{r}{\sigma}\right)^6\right]}\right] \tag{80}$$

$$u_{\mathrm{Coul}}(r,\lambda) = \lambda^5 \frac{1}{4\pi\varepsilon_0}\frac{q_i q_j}{r}. \tag{81}$$

where ϵ_0 is the dielectric constant in vacuum, r is the interatomic distance, q is the atomic charge, ϵ is the LJ strength parameter and σ is the LJ size parameter. For an expanded-ensemble scheme on a discrete λ-space, the values for each λ at a spatial grid point could be tabulated. However, it will be better to compute all fractional particles using the explicit computation (i.e. looping over all framework atoms), as the number of fractional particles is much lower than the number of integer particles.

Special care is needed when transforming fractional molecules into integer molecules, and vice versa [27]. Here, the integer molecules are computed using grids and the fractional molecule using full computation. The energy difference ΔU between these two must be taken into account for the Monte Carlo energy drift. Also, this energy difference comes back as an additional Boltzmann factor $\exp[-\beta\Delta U]$ term in the acceptance rule [28].

3. Results

3.1. Polynomial tests

Table 1 reports the maximum absolute error $|f(x,y,z) - \mathcal{P}(x,y,z)|$ of the interpolated $\mathcal{P}(x,y,z)$ compared to the reference function $f(x,y,z)$ for various polynomials (order 2–6) and nonpolynomial forms. For the tricubic scheme, polynomials up to order 3 can be represented exactly (i.e. with zero error to within floating-point precision on a uniform $\{1,1,1\}$ grid). For higher order polynomials, the tricubic interpolation becomes approximate, yielding nonzero errors that decrease as the grid resolution increases. The triquintic interpolation scheme polynomials extend exactness to fifth-order polynomials and, like tricubic, treats the exponential function only approximately. However, the errors for the triquintic scheme are consistently smaller than those of tricubic interpolation for identical resolutions.

3.2. Benchmark

Figure 2 reports the wall-clock time required for a single evaluation of the Framework-Molecule energy routine. All simulations are carried out on a AMD EPYC 9554 64-core processor (with 256 MB L3 cache) as single-core jobs. Benchmarks for the Coulomb energy and gradient routines are shown in Figures S1–S3. Each test computed the total interaction energy or its gradient for N_{test} probe positions distributed throughout a triclinic simulation box of $k \times k \times k$ unit cells of CHA-type zeolite.

The interpolated methods have a $\mathcal{O}(N_{\mathrm{test}})$ scaling, whereas the full energy computation has a $\mathcal{O}(N_{\mathrm{test}}N_{\mathrm{fw}})$ scaling, for N_{fw} framework atoms. Figure 2 shows that increasing the number of unit cells $k \times k \times k$ does not increase the computational cost in the interpolated routine. The cost of these routines scales linearly with the amount of probe positions. Notably, it is shown that the tricubic and triquintic interpolation methods show a minimum execution time of 2 ms (tricubic) and 10 ms (triquintic), regardless of the number of probe particles used. Therefore, the speedup is only achieved for a sufficiently large amount of framework atoms N_{fw}.

Figure 3 shows the results of the micro-benchmarks for grand-canonical Monte Carlo simulations of methane in $k \times k \times k$ MFI boxes at 300 K and 100 kPa. Interpolating the framework–molecule interaction reduces wall-clock time by up to a factor of two, with the benefit increasing steadily from $k = 1$ to $k = 5$. Minor improvements are observed for the different implementations

Table 1. Maximum absolute error $|f(x,y,z) - \mathcal{P}(x,y,z)|$ of the interpolated value $\mathcal{P}(x,y,z)$ compared to the function value $f(x,y,z)$ in the domain $[-3.0,3.0] \times [-3.0,3.0] \times [-3.0,3.0]$, evaluated at a 100^3 mesh for the triclinic and triquintic scheme and various polynomial (order 2–6) and non-polynomial forms.

			N_{cells}		
function $f(x,y,z)$	order	scheme	$\{1,1,1\}$	$\{8,8,8\}$	$\{64,64,64\}$
$x^2+y^2+z^2$	2	tricubic	7.46×10^{-14}	2.17×10^{-13}	2.04×10^{-13}
	2	triquintic	1.45×10^{-12}	2.27×10^{-11}	1.25×10^{-11}
$x^3+2x^2yz^2-y^2z+1$	3	tricubic	1.22×10^{-11}	1.85×10^{-12}	3.30×10^{-12}
	3	triquintic	2.57×10^{-9}	4.64×10^{-10}	1.96×10^{-10}
$(x^2+y^2+z^2)^2$	4	tricubic	243.0	0.0591	1.44×10^{-5}
	4	triquintic	1.34×10^{-9}	4.08×10^{-10}	2.79×10^{-10}
$x^5y^3z+2xy^3+4x^2yz^2$	5	tricubic	5631.771	20.980	0.005250
	5	triquintic	6.00×10^{-6}	1.17×10^{-8}	1.57×10^{-8}
$(x^2+y^2+z^2)^3$	6	tricubic	7290.0	8.616	0.00226
	6	triquintic	2187.0	0.0083	3.17×10^{-8}
$(x^2+y^2+z^2)e^{-(x^2+y^2+z^2)}$	–	tricubic	0.3678793	0.0083915	1.1920×10^{-5}
	–	triquintic	0.3678750	0.0001741	1.2037×10^{-8}

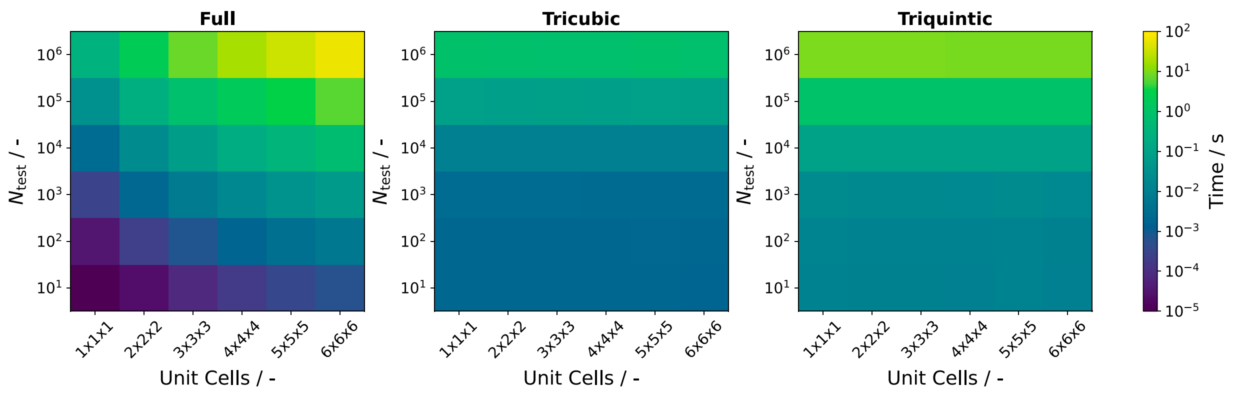


Figure 2. Benchmark for calculating the framework-particle Lennard–Jones energy. Benchmarks are performed by calculating the energy for N_{test} randomly placed probe methane molecules in CHA zeolite.

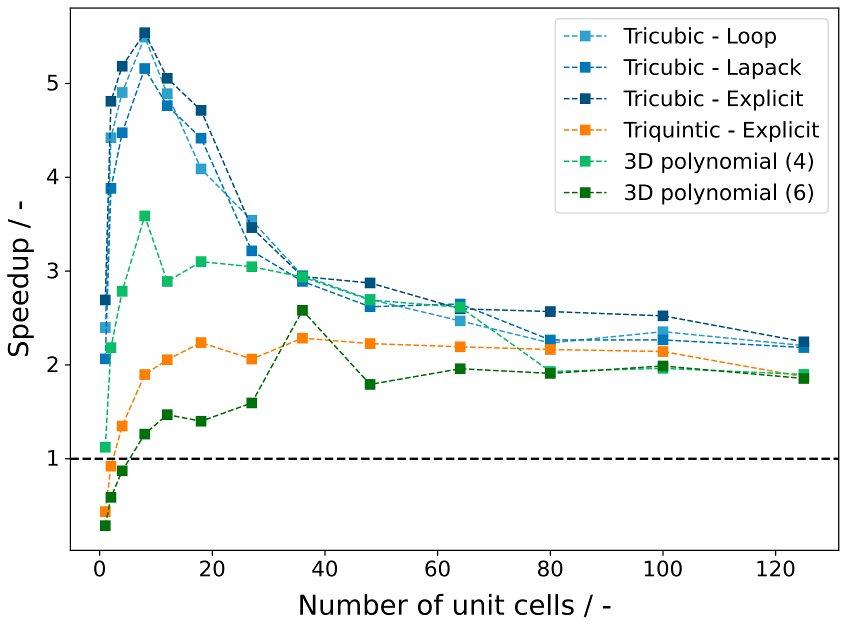


Figure 3. Relative speedup of tricubic (blues), triquintic (orange) and 3D polynomial (greens) simulations using interpolation schemes relative to simulations using the full molecule framework energy computation routines. Simulations are performed for methane adsorption using CBMC insertions and deletions for a $k \times k \times k$ unit cells of MFI zeolite at 300 K and 100 kPa. Three different implementations of tricubic interpolation are shown, where the matrix-vector product is computed using a conventional loop, using Lapack, or by writing out the non-zero terms terms explicitly.

of the interpolation routine. The implementation where the matrix-vector product is explicitly written out, using only the non-zero terms in the sparse matrix, is the most efficient. For the smallest simulation boxes, the overhead of the higher per-call cost of the interpolated routine offsets any gain, whereas for larger boxes the fraction of CPU time spent in the framework energy calculation becomes dominant, making interpolation computationally favourable. That is, a sufficient amount of framework atoms is needed before interpolation pays off. After reaching an optimum, the speedup for larger cells becomes less, because more and more relative computational time is spent in computing intermolecular interactions. Figure S5 shows the speedup of Widom insertions of methane in MFI in an empty framework. Since there are no molecule-molecule and only framework-molecule interactions, the speedup goes up with larger systems in that case. Splitting the total run time into pressure-sampling and non-Ewald energy calculations, plotted in Figure S6, shows that pressure sampling is inexpensive in small boxes. In RASPA3, the energy decomposition computation per component and the computation of the pressure (computed using the same routines) are done periodically every 10 or so cycles. But its share of CPU time grows with the box size.

When an entire adsorption isotherm is computed (Figure 4), tricubic interpolation shortens the calculation from 24 CPU h to 4 CPU h for Configurational-Bias

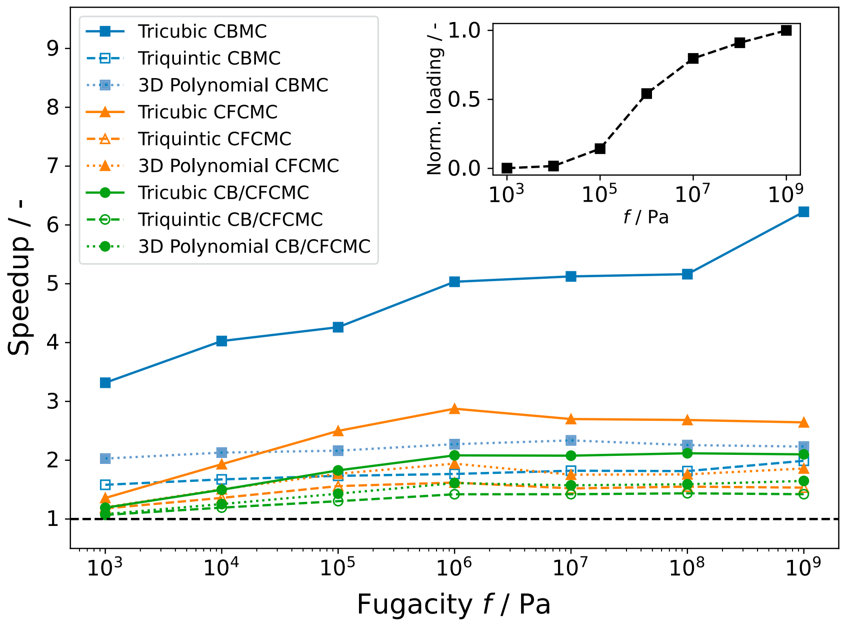


Figure 4. Relative speedup of GCMC adsorption simulations using polynomial, tricubic and triquintic interpolation methods of methane in $2 \times 2 \times 2$ MFI zeolite. The inset shows the normalised loading isotherm of the CBMC simulation with explicit energy routines, representative for all loading isotherm (which all lie within the error bar).

Monte Carlo (CBMC) insertions [29]. Triquintic interpolation results in a 25% increase in efficiency for all protocols at $k = 2$, consistent with the similar benchmark timings in Figure S4. Greater speedup is expected for both schemes as system size increases. The more marginal efficiency benefits observed for the schemes involving fractional molecules are attributed to the fact that all energy routines involving the fractional molecule must be computed explicitly. Overall, the simpler tricubic scheme offers the best cost–accuracy trade-off for routine GCMC production runs.

3.3. Precision

The highest precision for the interpolation needs to be achieved for the lowest energies (since these are exponentially preferred in the Boltzmann ensemble). Hence, we employ an error measure through Boltzmann weighting of the energy estimates [5]. The Boltzmann weighted mean squared error is defined as:

$$\text{error} = \sqrt{\frac{\sum_i^{\text{ntrials}} \left(U^{\text{full}}(r_i) - U^{\text{interp.}}(r_i)\right)^2 \times \exp\left[-\beta U^{\text{full}}(r_i)\right]}{\sum_i^{\text{ntrials}} \left(U^{\text{full}}(r_i)\right)^2 \exp\left[-\beta U^{\text{full}}(r_i)\right]}}, \quad (82)$$

where U^{full} is the explicit energy method, $U^{\text{interp.}}$ the interpolated energy, and r_i a uniformly sampled random position inside the simulation box. Figure 5 shows the error computation for 10^5 random probe positions for the energy grid of Lennard–Jones particles in IRMOF-1 as a function of ε and σ. The estimated weighted relative error for the energy calculation does not exceed 0.9% for tricubic and 0.03% for triquintic interpolation. Higher errors are seen for lower values of σ, where the highest error is observed for values of σ lower than the grid spacing of 0.15 Å. Although these values are rarely used in simulations, it is important to be aware of high errors when the particle size approaches the grid spacing.

To compare whether the computed properties lie within the error bar, we define the Z-score based on the estimated mean and variance of the observable:

$$Z = \frac{\mu_{\text{full}} - \mu_{\text{interp.}}}{\sqrt{\sigma^2_{\text{full}} + \sigma^2_{\text{interp.}}}}, \quad (83)$$

for the mean μ and variance σ^2. In Figure 6 we show the Z-score for the estimation of observables in a GCMC simulation of methane in MFI for 2×10^5 cycles. For low grid spacings (≤ 0.2 Å) only some deviations within one σ are observed, which is to be expected. At higher grid spacings, the error increases and the triquintic interpolation scheme shows a structural underestimation of the framework-molecule energy (i.e. more negative), leading to an overestimation of the loading. Clearly, it is important that appropriate grid spacing is selected when employing interpolation methods to prevent structural errors occurring.

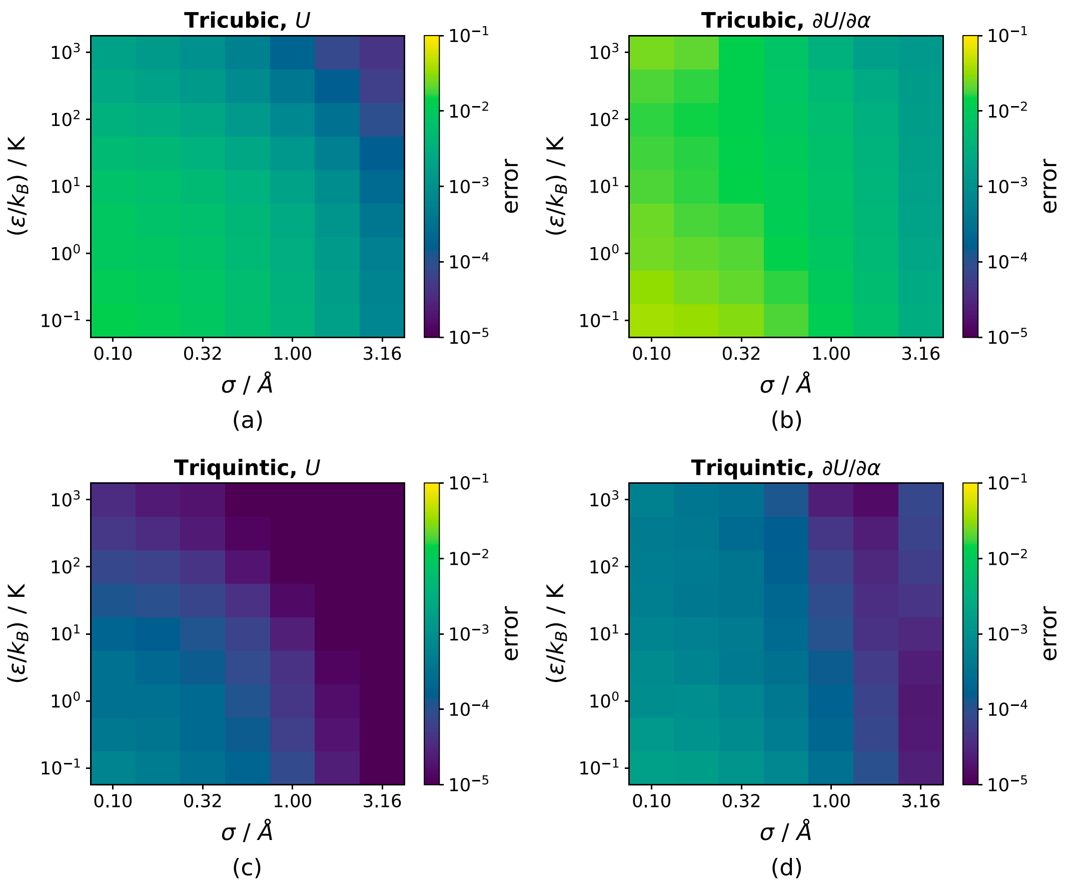


Figure 5. Lennard–Jones particles (ε, σ) in IRMOF-1: relative error of the measured energy (a,c) and gradient (b,d) of the tested grids, computed by 10^5 random trial points for tricubic (a,b) and triquintic (c,d) interpolation with a grid spacing of 0.15 Å.

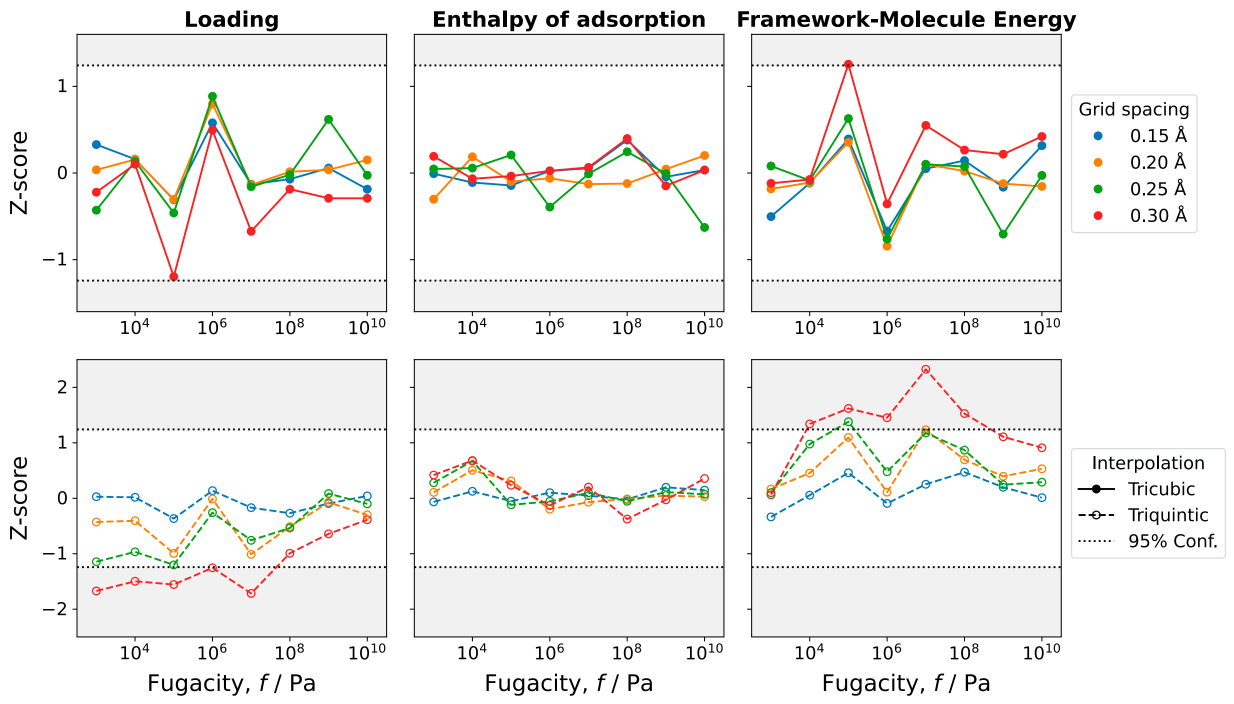


Figure 6. Z-score error for tricubic (top) and triquintic (bottom) interpolation, for three observables: loading (left), enthalpy (middle) and framework-molecule energy (right). The unshaded area shows the 95% confidence interval. Isotherm simulations are carried out using CBMC simulation of methane in 2 × 2 × 2 MFI at 300 K.

4. Conclusions

In this work, we have implemented and benchmarked two local interpolation schemes – tricubic and triquintic – for rapid evaluation of framework-molecule interactions in nanoporous materials. We have extended RASPA3 code to support the tricubic interpolation scheme by Lekien and Marsden [10] and the more recent triquintic extension of Boateng and Bradach [13]. Across polynomial test functions and realistic methane-zeolite systems, we observed that triquintic interpolation achieves global C^2 continuity and lower interpolation error at fine grid resolutions. The simpler tricubic interpolation, also implemented in the earlier RASPA2, offers superior computational speed and efficiency.

The benefits of the interpolation methods lie in the scaling behaviour of the computational efficiency. Every full computation of the energy is associated with a few to a dozen multiplications and a division (per framework-molecule interaction), whereas every interpolated energy calculation involves calculating a sparse matrix-vector product of size 64×64 and 216×216 respectively. While not cheap, the benefit lies in the fact that the interpolation cost shows no scaling with the total number of unit cells. The interpolation schemes, especially the triquintic scheme, therefore only have added value starting from a sufficiently large number of framework atoms. Note that using a larger cutoff also leads to an increased number of unit cells. In addition, the more complex and computationally expensive the computed potential is, the higher the benefit of grid interpolation. The accuracy of grid interpolation is dependent on both the strength-parameter ε and size-parameter σ of the Lennard–Jones potential, and on the details of the framework. For most adsorbates and framework structures, a grid-spacing of 0.15–0.2 Å will be sufficient.

Although triquintic interpolation requires additional computational effort and is significantly slower than tricubic interpolation, it offers distinct advantages in certain contexts. In tasks such as force-field fitting or machine-learning potential training, the Hessian is necessary for the optimisation schemes and therefore the C^2 continuity is required. This full continuity guarantees well-defined, continuous first and second derivatives throughout the simulation cell, enabling accurate Hessian evaluation at arbitrary points. By contrast, explicit models for direct Hessian computation impose a substantial overhead. The Hessian is also of use in advanced energy minimisation techniques [30], transition state optimizations [31], and the computation of vibrational properties (infra-red spectra and mode analysis). Looking forward, we expect that tricubic interpolation will remain the method of choice for high-throughput screening and large-scale adsorption studies, where millions of energy evaluations are required. The triquintic scheme will enable fast force field development, automatic differentiation, and hybrid ML-physics workflows that rely on smooth, differentiable potential energy surfaces.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work is part of the Advanced Research Center for Chemical Building Blocks, ARC CBBC, which is co-founded and co-financed by the Dutch Research Council (NWO) and the Netherlands Ministry of Economic Affairs and Climate Policy. J.T. and P.B. acknowledge the support from the National Science Foundation under Award CBET-2144360 and the computational resources provided by the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) through Allocations CTS190069 and CHM250041.

ORCID

Youri A. Ran http://orcid.org/0009-0001-5065-0314
Joseph Tapia http://orcid.org/0000-0001-5762-8905
Shrinjay Sharma http://orcid.org/0000-0001-8345-7433
Peng Bai http://orcid.org/0000-0002-6881-4663
Sofia Calero http://orcid.org/0000-0001-9535-057X
David Dubbeldam http://orcid.org/0000-0002-4382-1509
Thijs J. H. Vlugt http://orcid.org/0000-0003-3059-8712

References

[1] A.G. Bezus, A.V. Kiselev, A.A. Lopatkin and P.Q. Du, J. Chem. Soc. Faraday Trans. II **74**, 367–379 (1978). doi:10.1039/F29787400367.
[2] B. Smit and J.I. Siepmann, Science **264** (5162), 1118–1120 (1994). doi:10.1126/science.264.5162.1118.
[3] R.L. June, A.T. Bell and D.N. Theodorou, J. Phys. Chem. **94** (4), 1508–1516 (1990). doi:10.1021/j100367a056.
[4] N. Sathyamurthy and L.M. Raffn, J. Chem. Phys. **63**, 464–473 (1975). doi:10.1063/1.431126.
[5] T.J.H. Vlugt, Ph. D. thesis, University of Amsterdam, The Netherlands, 2000.
[6] W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery, *Numerical Recipes, the Art of Scientific Computing*, 3rd ed. (Cambridge University Press, New York, NY, USA, 2007).
[7] P. Bai, M. Tsapatsis and J.I. Siepmann, J. Phys. Chem. C **117** (46), 24375–24387 (2013). doi:10.1021/jp4074224.
[8] P. Bai, Ph. D. thesis, University of Minnesota, USA, 2014.
[9] D. Dubbeldam, S. Calero, D.E. Ellis and R.Q. Snurr, Mol. Simul. **42** (2), 81–101 (2016). doi:10.1080/08927022.2015.1010082.
[10] F. Lekien and J. Marsden, Int. J. Numer. Methods Eng. **63** (3), 455–471 (2005). doi:10.1002/(ISSN)1097-0207.

[11] A. Torres-Knoop, S.P. Balaji, T.J.H. Vlugt and D. Dubbeldam, J. Chem. Theor. Comp. **10** (3), 942–952 (2014). doi:10.1021/ct4009766.
[12] P. Walker, eprint arXiv:1904.09869 (2019).
[13] H.A. Boateng and K. Bradach, J. Comp. Appl. Math. **430**, 115254 (2023). doi:10.1016/j.cam.2023.115254.
[14] Y.A. Ran, S. Sharma, S.R.G. Balestra, Z. Li, S. Calero, T.J.H. Vlugt, R.Q. Snurr and D. Dubbeldam, J. Chem. Phys. **161** (11), 114106 (2024). doi:10.1063/5.0226249.
[15] W. Press, S. Teukolsky, W. Vetterling and B. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed. (Cambridge University Press, New York, USA, 2007).
[16] K. Bradach, Ph. D. thesis, San Francisco State University USA, 2022.
[17] C.R.A. Catlow and W.C. Mackrodt, in *Computer Simulation of Solids*, edited by C.R.A. Catlow and W.C. Mackrodt (Springer-Verlag, Berlin, 1982), Vol. 166, Chap. 1.
[18] J.D. Gale, J. Chem. Soc. Faraday Trans. **93** (4), 629–637 (1997). doi:10.1039/a606455h.
[19] J.D. Gale and A.L. Rohl, Mol. Simulat. **29** (5), 291–341 (2003). doi:10.1080/0892702031000104887.
[20] M.E. Tuckerman, *Statistical Mechanics: Theory and Molecular Simulation* (Oxford University Press, New Oxford, UK, 2010).
[21] T.T.C. Ting, *Anisotropic Elasticity: Theory and Applications* (Oxford University Press, New York, USA, 1996).
[22] J.F. Nye, *Physical Properties of Crystals: Their Representation by Tensors and Matrices*, 1st ed. (Oxford Science Publications, Oxford, UK, 1985).
[23] M. Allen and D. Tildesley, *Computer Simulation of Liquids*, 2nd ed. (Clarendon Press, Oxford, 2017).
[24] D.J. Adams, J. Chem. Phys. **78** (5), 2585–2590 (1983). doi:10.1063/1.445014.
[25] S. Bogusz, T.E. Cheatham and B.R. Brooks, J. Chem. Phys. **108** (17), 7070–7084 (1998). doi:10.1063/1.476320.
[26] W. Shi and E.J. Maginn, J. Chem. Theory Comput. **3** (4), 1451–1463 (2007). doi:10.1021/ct7000039.
[27] A. Rahbari, R. Hens, M. Ramdin, O.A. Moultos, D. Dubbeldam and T.J.H. Vlugt, Mol. Simul. **47** (10–11), 804–823 (2021). doi:10.1080/08927022.2020.1828585.
[28] D. Frenkel and B. Smit, *Understanding Molecular Simulation*, 3rd ed. (Academic Press, London, UK, 2023).
[29] T.J.H. Vlugt, R. Krishna and B. Smit, J. Phys. Chem. B **103** (7), 1102–1118 (1999). doi:10.1021/jp982736c.
[30] D. Dubbeldam, R. Krishna and R. Snurr, J. Phys. Chem. C **113** (44), 19317–19327 (2009). doi:10.1021/jp906635f.
[31] J. Baker, J. Comp. Chem. **7** (4), 385–395 (1986). doi:10.1002/jcc.v7:4.