Create account Log in

Article Talk Read Edit More Search

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Add links

Constraint algorithm

From Wikipedia, the free encyclopedia

In computational chemistry, a **constraint algorithm** is a method for satisfying a Newtonian motion of a rigid which consists of mass points. By this algorithm, the distance between mass points is maintained constant. In general, the algorithm is constructed by following procedures; (i) choosing novel unconstrained coordinates (internal coordinates), (ii) introducing explicit constraint forces, (iii) minimizing constraint forces implicitly by the technique of Lagrange multipliers or projection methods.

Constraint algorithms are often applied to molecular dynamics simulations. Although such simulations are sometimes performed using internal coordinates that automatically satisfy the bond-length, bond-angle and torsion-angle constraints, simulations may also be performed using explicit or implicit constraint forces for these three constraints. However, explicit constraint forces give a cause of the inefficient computation; more computational power is required to get a trajectory of a given length. Therefore, internal coordinates and implicit-force constraint solvers are generally preferred.

Constraint algorithms achieve efficient computational time by neglecting motions of intramolecular atoms. If intramolecular behavior is important, e.g. hydrogen bonding and its behavior, constraint algorithms should not be used.

Contents [hide]

Mathematical background [edit]

The motion of a set of N particles can be described by a set of second-order ordinary differential equations, Newton's second law, which can be written in matrix form

$$\mathbf{M} \cdot \frac{d^2\mathbf{q}}{dt^2} = \mathbf{f} = -\frac{\partial V}{\partial \mathbf{q}}$$

where **M** is a *mass matrix* and **q** is the vector of generalized coordinates that describe the particles' positions. For example, the vector **q** may be a $3N$ Cartesian coordinates of the particle positions $\mathbf{r}_k$, where k runs from 1 to N; in the absence of constraints, **M** would be the $3N$x$3N$ diagonal square matrix of the particle masses. The vector **f** represents the generalized forces and the scalar $V(\mathbf{q})$ represents the potential energy, both of which are functions of the generalized coordinates **q**.

If M constraints are present, the coordinates must also satisfy M time-independent algebraic equations

$$g_j(\mathbf{q}) = 0$$

where the index j runs from 1 to M. For brevity, these functions g_i are grouped into an M-dimensional vector **g** below. The task is to solve the combined set of differential-algebraic (DAE) equations, instead of just the ordinary differential equations (ODE) of Newton's second law.

This problem was studied in detail by Joseph Louis Lagrange, who laid out most of the methods for solving it.[1] The simplest approach is to define new generalized coordinates that are unconstrained; this approach eliminates the algebraic equations and reduces the problem once again to solving an ordinary differential equation. Such an approach is used, for example, in describing the motion of a rigid body; the position and orientation of a rigid body can be described by six independent, unconstrained coordinates, rather than describing the positions of the particles that make it up and the constraints among them that maintain their relative distances. The drawback of this approach is that the equations may become unwieldy and complex; for example, the mass matrix **M** may become non-diagonal and depend on the generalized coordinates.

A second approach is to introduce explicit forces that work to maintain the constraint; for example, one could introduce strong spring forces that enforce the distances among mass points within a "rigid" body. The two difficulties of this approach are that the constraints are not satisfied exactly, and the strong forces may require very short time-steps, making simulations inefficient computationally.

A third approach is to use a method such as Lagrange multipliers or projection to the constraint manifold to determine the coordinate adjustments necessary to satisfy the constraints. Finally, there are various hybrid approaches in which different sets of constraints are satisfied by different methods, e.g., internal coordinates, explicit forces and implicit-force solutions.

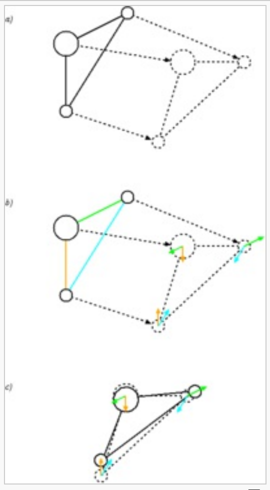
Internal coordinate methods [edit]

The simplest approach to satisfying constraints in energy minimization and molecular dynamics is to represent the mechanical system in so-called *internal coordinates* corresponding to unconstrained independent degrees of freedom of the system. For example, the dihedral angles of a protein are an independent set of coordinates that specify the positions of all the atoms without requiring any constraints. The difficulty of such internal-coordinate approaches is twofold: the Newtonian equations of motion become much more complex and the internal coordinates may be difficult to define for cyclic systems of constraints, e.g., in ring puckering or when a protein has a disulfide bond.

The original methods for efficient recursive energy minimization in internal coordinates were developed by Gō and coworkers.[2][3]

Efficient recursive, internal-coordinate constraint solvers were extended to molecular dynamics.[4][5] Analogous methods were applied later to other systems.[6][7][8]

Lagrange multiplier-based methods [edit]



Resolving the constraints of a rigid water molecule using Lagrange multipliers: a) the unconstrained positions are obtained after a simulation time-step, b) the gradients of each constraint over each particle are computed and c) the Lagrange multipliers are computed for each gradient such that the constraints are satisfied.

In most of molecular dynamics simulations that use constraint algorithms, constraints are enforced using the method of Lagrange multipliers. Given a set of *n* linear (holonomic) constraints at the time *t*,

$$\sigma_k(t) := \|\mathbf{x}_{k\alpha}(t) - \mathbf{x}_{k\beta}(t)\|^2 - d_k^2 = 0, \quad k = 1 \ldots n$$

where $\mathbf{x}_{k\alpha}(t)$ and $\mathbf{x}_{k\beta}(t)$ are the positions of the two particles involved in the *k*th constraint at the time *t* and d_k is the prescribed inter-particle distance.

The forces due to these constraints are added in the equations of motion, resulting in, for each of the *N* particles in the system

$$\frac{\partial^2 \mathbf{x}_i(t)}{\partial t^2} m_i = -\frac{\partial}{\partial \mathbf{x}_i}\left[V(\mathbf{x}_i(t)) + \sum_{k=1}^{n} \lambda_k \sigma_k(t)\right], \quad i = 1 \ldots N.$$

Adding the constraint forces does not change the total energy, as the net work done by the constraint forces (taken over the set of particles that the constraints act on) is zero.

From integrating both sides of the equation with respect to the time, the constrainted coordinates of particles at the time, $t + \Delta t$, are given,

$$\mathbf{x}_i(t+\Delta t) = \hat{\mathbf{x}}_i(t+\Delta t) + \sum_{k=1}^{n} \lambda_k \frac{\partial \sigma_k(t)}{\partial \mathbf{x}_i} (\Delta t)^2 m_i^{-1}, \quad i = 1 \ldots N$$

where $\hat{\mathbf{x}}_i(t+\Delta t)$ is the unconstrained (or uncorrected) position of the *i*th particle after integrating the unconstrained equations of motion.

To satisfy the constraints $\sigma_k(t+\Delta t)$ in the next timestep, the Lagrange multipliers should be determined as the following equation,

$$\sigma_k(t+\Delta t) := \|\mathbf{x}_{k\alpha}(t+\Delta t) - \mathbf{x}_{k\beta}(t+\Delta t)\|^2 - d_k^2 = 0.$$

This implies solving a system of n non-linear equations

$$\sigma_j(t+\Delta t) := \left\|\hat{\mathbf{x}}_{j\alpha}(t+\Delta t) - \hat{\mathbf{x}}_{j\beta}(t+\Delta t) + \sum_{k=1}^{n} \lambda_k (\Delta t)^2 \left[\frac{\partial \sigma_k(t)}{\partial \mathbf{x}_{j\alpha}} m_{j\alpha}^{-1} - \frac{\partial \sigma_k(t)}{\partial \mathbf{x}_{j\beta}} m_{j\beta}^{-1}\right]\right\|^2 - d_j^2 = 0, \quad j = 1 \ldots n$$

simultaneously for the n unknown Lagrange multipliers λ_k.

This system of n non-linear equations in n unknowns is commonly solved using Newton–Raphson method where the solution vector $\underline{\lambda}$ is updated using

$$\underline{\lambda}^{(l+1)} \leftarrow \underline{\lambda}^{(l)} - \mathbf{J}_\sigma^{-1} \underline{\sigma}(t+\Delta t)$$

where $\mathbf{J}_\sigma$ is the Jacobian of the equations σ*k*:

$$\mathbf{J} = \begin{pmatrix} \frac{\partial \sigma_1}{\partial \lambda_1} & \frac{\partial \sigma_1}{\partial \lambda_2} & \cdots & \frac{\partial \sigma_1}{\partial \lambda_n} \\ \frac{\partial \sigma_2}{\partial \lambda_1} & \frac{\partial \sigma_2}{\partial \lambda_2} & \cdots & \frac{\partial \sigma_2}{\partial \lambda_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \sigma_n}{\partial \lambda_1} & \frac{\partial \sigma_n}{\partial \lambda_2} & \cdots & \frac{\partial \sigma_n}{\partial \lambda_n} \end{pmatrix}.$$

Since not all particles contribute to all of constraints, $\mathbf{J}_\sigma$ is a block matrix and can be solved individually to block-unit of the matrix. In other words, $\mathbf{J}_\sigma$ can be solved individually for each molecule.

Instead of constantly updating the vector $\underline{\lambda}$, the iteration can be started with $\underline{\lambda}^{(0)} = \mathbf{0}$, resulting in simpler expressions for $\sigma_k(t)$ and $\frac{\partial \sigma_k(t)}{\partial \lambda_j}$. In this case

$$J_{ij} = \left.\frac{\partial \sigma_j}{\partial \lambda_i}\right|_{\lambda=0} = 2\left[\hat{x}_{j\alpha} - \hat{x}_{j\beta}\right]\left[\frac{\partial \sigma_i}{\partial x_{j\alpha}} m_{j\alpha}^{-1} - \frac{\partial \sigma_i}{\partial x_{j\beta}} m_{j\beta}^{-1}\right].$$

then λ is updated to

$$\lambda_j = -\mathbf{J}^{-1}\left[\|\hat{\mathbf{x}}_{j\alpha}(t+\Delta t) - \hat{\mathbf{x}}_{j\beta}(t+\Delta t)\|^2 - d_j^2\right].$$

After each iteration, the unconstrained particle positions are updated using

$$\hat{\mathbf{x}}_i(t+\Delta t) \leftarrow \hat{\mathbf{x}}_i(t+\Delta t) + \sum_{k=1}^{n} \lambda_k \frac{\partial \sigma_k}{\partial \mathbf{x}_i}.$$

The vector is then reset to

$$\underline{\lambda} = \mathbf{0}.$$

The above procedure is repeated until the solution of constraint equations, $\sigma_k(t+\Delta t)$, converges to a prescribed tolerance of a numerical error.

Although there are a number of algorithms to compute the Lagrange multipliers, these difference is rely only on the methods to solve the system of equations. For this methods, quasi-Newton methods are commonly used.

The SETTLE algorithm [edit]

The SETTLE algorithm[9] solves the system of non-linear equations analytically for $n = 3$ constraints in constant time. Although it does not scale to larger numbers of constraints, it is very often used to constrain rigid water molecules, which are present in almost all biological simulations and are usually modelled using three constraints (e.g. SPC/E and TIP3P water models).

The SHAKE algorithm [edit]

The SHAKE algorithm was first developed for satisfying a bond geometry constraint during molecular dynamics simulations.[10] In SHAKE algorithm, the system of non-linear constraint equations is solved using the Gauss-Seidel method which approximates the solution of the linear system of equations using the Newton-Raphson method;

$$\underline{\lambda} = -\mathbf{J}_\sigma^{-1}\underline{\sigma}.$$

This amounts to assuming that $\mathbf{J}_\sigma$ is diagonally dominant and solving the kth equation only for the k unknown. In practice, we compute

$$\begin{aligned}\lambda_k &\leftarrow \frac{\sigma_k(t)}{\partial\sigma_k(t)/\partial\lambda_k},\\ \mathbf{x}_{k\alpha} &\leftarrow \mathbf{x}_{k\alpha} + \lambda_k\frac{\partial\sigma_k(t)}{\partial\mathbf{x}_{k\alpha}},\\ \mathbf{x}_{k\beta} &\leftarrow \mathbf{x}_{k\beta} + \lambda_k\frac{\partial\sigma_k(t)}{\partial\mathbf{x}_{k\beta}},\end{aligned}$$

for all $k = 1\ldots n$ iteratively until the constraint equations $\sigma_k(t+\Delta t)$ are solved to a given tolerance.

The calculation cost of each iteration is $\mathcal{O}(n)$, and the iterations themselves converge linearly.

A noniterative form of SHAKE was developed later.[11]

Several variants of the SHAKE algorithm exist. Although they differ in how they compute or apply the constraints themselves, the constraints are still modelled using Lagrange multipliers which are computed using the Gauss-Seidel method.

The original SHAKE algorithm is limited to mechanical systems with a tree structure, i.e., no closed loops of constraints. A later extension of the method, QSHAKE (Quaternion SHAKE) was developed to amend this.[12] It works satisfactorily for *rigid* loops such as aromatic ring systems but fails for flexible loops, such as when a protein has a disulfide bond.[13]

Further extensions include RATTLE,[14] WIGGLE[15] and MSHAKE.[16] RATTLE works the same way as SHAKE,[17] yet using the Velocity Verlet time integration scheme. WIGGLE extends SHAKE and RATTLE by using an initial estimate for the Lagrange multipliers λ_k based on the particle velocities. Finally, MSHAKE computes corrections on the constraint *forces*, achieving better convergence.

A final modification is the P-SHAKE algorithm[18] for rigid or semi-rigid molecules. P-SHAKE computes and updates a pre-conditioner which is applied to the constraint gradients before the SHAKE iteration, causing the Jacobian $\mathbf{J}_\sigma$ to become diagonal or strongly diagonally dominant. The thus de-coupled constraints converge much faster (quadratically as opposed to linearly) at a cost of $\mathcal{O}(n^2)$.

The M-SHAKE algorithm [edit]

The M-SHAKE algorithm[19] solves the non-linear system of equations using Newton's method directly. In each iteration, the linear system of equations

$$\underline{\lambda} = -\mathbf{J}_\sigma^{-1}\underline{\sigma}$$

is solved exactly using an LU decomposition. Each iteration costs $\mathcal{O}(n^3)$ operations, yet the solution converges quadratically, requiring fewer iterations than SHAKE.

This solution was first proposed in 1986 by Ciccotti and Ryckaert[20] under the title "the matrix method", yet differed in the solution of the linear system of equations. Ciccotti and Ryckaert suggest inverting the matrix $\mathbf{J}_\sigma$ directly, yet doing so only once, in the first iteration. The first iteration then costs $\mathcal{O}(n^3)$ operations, whereas the following iterations cost only $\mathcal{O}(n^2)$ operations (for the matrix-vector multiplication). This improvement comes at a cost though, since the Jacobian is no longer updated, convergence is only linear, albeit at a much faster rate than for the SHAKE algorithm.

Several variants of this approach based on sparse matrix techniques were studied by Barth *et al.*.[21]

The SHAPE algorithm [edit]

The SHAPE algorithm[22] is a multicenter analog of SHAKE for constraining rigid bodies of three or more centers. Like SHAKE, an unconstrained step is taken and then corrected by directly calculating and applying the rigid body rotation matrix that satisfies:

$$L^{rigid}\left(t+\frac{\Delta t}{2}\right) = L^{nonrigid}\left(t+\frac{\Delta t}{2}\right)$$

This approach involves a single 3x3 matrix diagonalization followed by three or four rapid Newton iterations to determine the rotation matrix. SHAPE provides the identical trajectory that is provided with fully converged iterative SHAKE, yet it is found to be more efficient and more accurate than SHAKE when applied to systems involving three or more centers. It extends the ability of SHAKE like constraints to linear systems with three or more atoms, planar systems with four or more atoms, and to significantly larger rigid structures where SHAKE is intractable. It also allows rigid bodies to be linked with one or two common centers (e.g. peptide planes) by solving rigid body constraints iteratively in the same basic manner that SHAKE is used for atoms involving more than one SHAKE constraint.

The LINCS algorithm [edit]

An alternative constraint method, LINCS (Linear Constraint Solver) was developed in 1997 by Hess, Bekker, Berendsen and Fraaije,[23] and was based on the 1986 method of Edberg, Evans and Morriss (EEM),[24] and a modification thereof by Baranyai and Evans (BE).[25]

LINCS applies Lagrange multipliers to the constraint forces and solves for the multipliers by using a series expansion to approximate the inverse of the Jacobian $\mathbf{J}_\sigma$:

$$(\mathbf{I}-\mathbf{J}_\sigma)^{-1} = \mathbf{I} + \mathbf{J}_\sigma + \mathbf{J}_\sigma^2 + \mathbf{J}_\sigma^3 + \ldots$$

in each step of the Newton iteration. This approximation only works for matrices with Eigenvalues smaller than 1, making the LINCS algorithm suitable only for molecules with low connectivity.

LINCS has been reported to be 3-4 times faster than SHAKE.[23]

Hybrid methods [edit]

Hybrid methods have also been introduced in which the constraints are divided into two groups; the constraints of the first group are solved using internal coordinates whereas those of the second group are solved using constraint forces, e.g., by a Lagrange multiplier or projection method.[26][27][28] This approach was pioneered by Lagrange,[1] and result in *Lagrange equations of the mixed type*.[29]

See also [edit]

- Molecular dynamics
- Software for molecular mechanics modeling

References and footnotes [edit]

1. ^ *a* *b* Lagrange, GL (1788). *Mécanique analytique*.
2. ^ Noguti T, Toshiyuki; Gō N (1983). "A Method of Rapid Calculation of a 2nd Derivative Matrix of Conformational Energy for Large Molecules". *Journal of the Physical Society of Japan* **52** (10): 3685–3690. Bibcode:1983JPSJ...52.3685N. doi:10.1143/JPSJ.52.3685.
3. ^ Abe, H; Braun W; Noguti T; Gō N (1984). "Rapid Calculation of 1st and 2nd Derivatives of Conformational Energy with respect to Dihedral Angles for Proteins: General Recurrent Equations". *Computers and Chemistry* **8** (4): 239–247. doi:10.1016/0097-8485(84)85015-9.
4. ^ Bae, D-S; Haug EJ (1988). "A Recursive Formulation for Constrained Mechanical System Dynamics: Part I. Open Loop Systems". *Mechanics of Structures and Machines* **15**: 359–382.
5. ^ Jain, A; Vaidehi N; Rodriguez G (1993). "A Fast Recursive Algorithm for Molecular Dynamics Simulation". *Journal of Computational Physics* **106** (2): 258–268. Bibcode:1993JCoPh.106..258J. doi:10.1006/jcph.1993.1106.
6. ^ Rice, LM; Brünger AT (1994). "Torsion Angle Dynamics: Reduced Variable Conformational Sampling Enhances Crystallographic Structure Refinement". *Proteins: Structure, Function, and Genetics* **19** (4): 277–290. doi:10.1002/prot.340190403. PMID 7984624.
7. ^ Mathiowetz, AM; Jain A; Karasawa N; Goddard III, WA (1994). "Protein Simulations Using Techniques Suitable for Very Large Systems: The Cell Multipole Method for Nonbond Interactions and the Newton-Euler Inverse Mass Operator Method for Internal Coordinate Dynamics". *Proteins: Structure, Function, and Genetics* **20** (3): 227–247. doi:10.1002/prot.340200304. PMID 7892172.
8. ^ Mazur, AK (1997). "Quasi-Hamiltonian Equations of Motion for Internal Coordinate Molecular Dynamics of Polymers". *Journal of Computational Chemistry* **18** (11): 1354–1364. doi:10.1002/(SICI)1096-987X(199708)18:11<1354::AID-JCC3>3.0.CO;2-K.
9. ^ Miyamoto, S; Kollman PA (1992). "SETTLE: An Analytical Version of the SHAKE and RATTLE Algorithm for Rigid Water Models". *Journal of Computational Chemistry* **13** (8): 952–962. doi:10.1002/jcc.540130805.
10. ^ Ryckaert, J-P; Ciccotti G; Berendsen HJC (1977). "Numerical Integration of the Cartesian Equations of Motion of a System with Constraints: Molecular Dynamics of *n*-Alkanes". *Journal of Computational Physics* **23** (3): 327–341. Bibcode:1977JCoPh..23..327R. doi:10.1016/0021-9991(77)90098-5.
11. ^ Yoneya, M; Berendsen HJC; Hirasawa K (1994). "A Noniterative Matrix Method for Constraint Molecular-Dynamics Simulations". *Molecular Simulations* **13** (6): 395–405. doi:10.1080/08927029408022001.
12. ^ Forester, TR; Smith W (1998). "SHAKE, Rattle, and Roll: Efficient Constraint Algorithms for Linked Rigid Bodies". *Journal of Computational Chemistry* **19**: 102–111. doi:10.1002/(SICI)1096-987X(19980115)19:1<102::AID-JCC9>3.0.CO;2-T.
13. ^ McBride, C; Wilson MR; Howard JAK (1998). "Molecular dynamics simulations of liquid crystal phases using atomistic potentials". *Molecular Physics* **93** (6): 955–964. doi:10.1080/002689798168655.
14. ^ Andersen, Hans C. (1983). "RATTLE: A "Velocity" Version of the SHAKE Algorithm for Molecular Dynamics Calculations". *Journal of Computational Physics* **52**: 24–34. Bibcode:1983JCoPh..52...24A. doi:10.1016/0021-9991(83)90014-1.
15. ^ Lee, Sang-Ho; Kim Palmo; Samuel Krimm (2005). "WIGGLE: A new constrained molecular dynamics algorithm in Cartesian coordinates". *Journal of Computational Physics* **210**: 171–182. Bibcode:2005JCoPh.210..171L. doi:10.1016/j.jcp.2005.04.006.
16. ^ Lambrakos, S. G.; J. P. Boris; E. S. Oran; I. Chandrasekhar; M. Nagumo (1989). "A Modified SHAKE algorithm for Maintaining Rigid Bonds in Molecular Dynamics Simulations of Large Molecules". *Journal of Computational Physics* **85** (2): 473–486. Bibcode:1989JCoPh..85..473L. doi:10.1016/0021-9991(89)90160-5.
17. ^ Leimkuhler, Benedict; Robert Skeel (1994). "Symplectic numerical integrators in constrained Hamiltonian systems". *Journal of Computational Physics* **112**: 117–125. Bibcode:1994JCoPh.112..117L. doi:10.1006/jcph.1994.1085.
18. ^ Gonnet, Pedro (2007). "P-SHAKE: A quadratically convergent SHAKE in $\mathcal{O}(n^2)$". *Journal of Computational Physics* **220** (2): 740–750. Bibcode:2007JCoPh.220..740G. doi:10.1016/j.jcp.2006.05.032.
19. ^ Kräutler, Vincent; W. F. van Gunsteren; P. H. Hünenberger (2001). "A Fast SHAKE Algorithm to Solve Distance Constraint Equations for Small Molecules in Molecular Dynamics Simulations". *Journal of Computational Chemistry* **22** (5): 501–508. doi:10.1002/1096-987X(20010415)22:5<501::AID-JCC1021>3.0.CO;2-V.
20. ^ Ciccotti, G.; J. P. Ryckaert (1986). "Molecular Dynamics Simulation of Rigid Molecules". *Computer Physics Reports* **4** (6): 345–392. Bibcode:1986CoPhR...4..346C. doi:10.1016/0167-7977(86)90022-5.
21. ^ Barth, Eric; K. Kuczera; B. Leimkuhler; R. Skeel (1995). "Algorithms for constrained molecular dynamics". *Journal of Computational Chemistry* **16** (10): 1192–1209. doi:10.1002/jcc.540161003.
22. ^ Tao, Peng; Xiongwu Wu; Bernard R. Brooks (2012). "Maintain rigid structures in Verlet based Cartesian molecular dynamics simulations". *The Journal of Chemical Physics* **137**: 134110. Bibcode:2012JChPh.137m4110T. doi:10.1063/1.4756796.
23. ^ *a* *b* Hess, B; Bekker H; Berendsen HJC; Fraaije JGEM (1997). "LINCS: A Linear Constraint Solver for Molecular Simulations". *Journal of Computational Chemistry* **18** (12): 1463–1472. doi:10.1002/(SICI)1096-987X(199709)18:12<1463::AID-JCC4>3.0.CO;2-H.
24. ^ Edberg, R; Evans DJ; Morriss GP (1986). "Constrained Molecular-Dynamics Simulations of Liquid Alkanes with a New Algorithm". *Journal of Chemical Physics* **84** (12): 6933–6939. Bibcode:1986JChPh..84.6933E. doi:10.1063/1.450613.
25. ^ Baranyai, A; Evans DJ (1990). "New Algorithm for Constrained Molecular-Dynamics Simulation of Liquid Benzene and Naphthalene". *Molecular Physics* **70**: 53–63. Bibcode:1990MolPh..70...53B. doi:10.1080/00268979000100841.
26. ^ Mazur, AK (1999). "Symplectic integration of closed chain rigid body dynamics with internal coordinate equations of motion". *Journal of Chemical Physics* **111** (4): 1407–1414. Bibcode:1999JChPh.111.1407M. doi:10.1063/1.479399.
27. ^ Bae, D-S; Haug EJ (1988). "A Recursive Formulation for Constrained Mechanical System Dynamics: Part II. Closed Loop Systems". *Mechanics of Structures and Machines* **15**: 481–506.
28. ^ Rodriguez, G; Jain A; Kreutz-Delgado K (1991). "A Spatial Operator Algebra for Manipulator Modeling and Control". *The International Journal for Robotics Research* **10** (4): 371–381. doi:10.1177/027836499101000406.
29. ^ Sommerfeld, Arnold (1952). *Lectures on Theoretical Physics, Vol. I: Mechanics*. New York: Academic Press. ISBN 0-12-654670-3.

Categories: Molecular dynamics | Computational chemistry | Molecular physics | Computational physics | Numerical differential equations