6.581J / 20.482J
Foundations of Algorithms and Computational Techniques in Systems Biology
Professor Bruce Tidor
Professor Jacob K. White

6.581/20.482J Problem Set #2

Due 5:00 p.m. Tuesday, March 14th, 2006

1. Modeling the dynamics of a trapped ion. Here, we will explore the implementation of several integration schemes in molecular dynamics. Figure 1 shows the system of interest— a single mobile ion trapped by a square of like-charged ions.

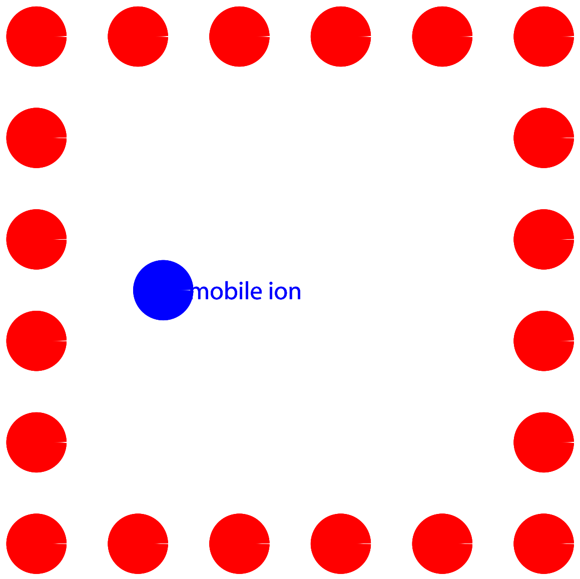

Figure 1: Physical system for molecular dynamics simulation.

The MATLAB script `moldyn.m` contains the description of the system, as well as the force and energy functions for both electrostatic interactions:

$$E_{ij}^{elec} = \frac{q_i q_j}{4\pi\varepsilon_0 r_{ij}} \tag{1}$$

and van der Waals interactions:

$$E_{ij}^{vdw} = 4\sqrt{\varepsilon_i\varepsilon_j}\left[\left(\frac{\sqrt{\sigma_i\sigma_j}}{r_{ij}}\right)^{12} - \left(\frac{\sqrt{\sigma_i\sigma_j}}{r_{ij}}\right)^{6}\right] \tag{2}$$

The van der Waals parameters ε and σ describe the energy of the most favorable interaction (ε) and the distance at which the interaction becomes unfavorable (σ). The parameters for three different ions are displayed below.

Ion	m (g/mol)	q (e)	σ (Å)	ε (kcal/mol)
F^-	18.998	-1.0	2.73295	0.72000
Cl^-	35.453	-1.0	4.41724	0.11779
Br^-	79.904	-1.0	4.62376	0.09000

Please note that in molecular level simulations, distance is measured in Å (1×10^{-10} m), charges in units of proton charge (e), masses in units of g/mol, and energies in kcal/mol; in these units, ε_0 is 2.397×10^{-4} $e^2 \cdot Å \cdot (\text{kcal/mol})^{-1}$. Although this choice of units leads to a unit of time equal to 4.888821×10^{-14}s, a conversion to a ps time unit is generally implemented (as in the MATLAB script `moldyn.m`).

(a) Implement the Forward Euler integration scheme, and propagate the dyamics of the ion with a 0.01 ps timestep.

(b) Describe the dynamic trajectory of the mobile ion.

(c) What happens to the energy of the system (kinetic, potential, and total) as a function of time?

(d) Repeat the simulation with a 0.001 ps time step, and repeat (b) and (c), and comment on the differences.

(e) Implement the Velocity Verlet integration scheme, and again propagate the dynamics of the ion with a 0.01 ps timestep. Repeat (b) and (c) for this trajectory. How could you test the validity of this trajectory? Verify your idea.

(f) Repeat (d) using a 0.1 ps time step, and comment on the differences.

(g) Repeat the simulation using both Cl^- and Br^- as the mobile ion (choose the integration method you feel is most appropriate). Compare the trajectories with the results from the F^- simulation. How does the trajectory of the ion change with ion type? How do the energetics of the ion change?

(h) Repeat the simulation with an initial position of -9.0 Å, again for each ion type. Discuss your observations of the trajectory and the energetics.

2. **Long-range electrostatic interactions.** In this problem, we will use eigendecomposition, via MATLAB's `eig` function, to illustrate that sometimes simple approximations can significantly reduce computational requirements (memory and time) while not sacrificing much accuracy. Recall that for a matrix A, an eigenvalue λ and a corresponding eigenvector x satisfy

$$Ax = \lambda x, \tag{3}$$

and that a symmetric matrix has a complete set of orthogonal eigenvectors.

Figure 2 shows two clusters of point charges. Each atom has a partial charge on it, which we model as a point charge at the atom center. Let the set of atomic charges in cluster 1 be denoted by the vector q; the set in cluster 2, by s. The charges q produce a potential field in cluster 2; let the vector of potentials at the atom centers in cluster 2 be φ. This vector of potentials produced by q can be found by the matrix multiplication

$$\varphi = Pq \tag{4}$$

where P is the potential matrix; the i^{th} column of P is the set of potentials produced at the atom centers in cluster 2, due to a unit charge at point i in cluster 1. (Thus an arbitrary charge distribution q in cluster 1 produces a scaled sum of the responses due to each charge separately.) We have set up a symmetric physical system, so P will be symmetric. In general P is unsymmetric, so the singular value decomposition (SVD) is used for analysis rather than eigendecomposition. Trefethen and Bau's Numerical Linear Algebra contains a very good exposition of the SVD.

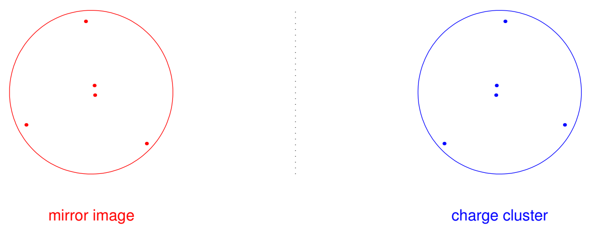

Figure 2: Electrostatic system: two well-separated, symmetric clusters of charges.

If the clusters are large, the matrix P becomes similarly large, so it becomes expensive to store the matrix and to multiply a vector by P. Download the script `estatic.m` and open it in your favorite editor. Several lines of this script have been left blank, for you to fill in the details.

(a) Fill in the matrix P: the (i, j) entry should equal the electrostatic potential induced at destination point i due to a unit charge at source point j. Equation (1) tells you the energy of the interactions between i and j, which equals the charge at i times the potential at i due to j. Check that P is symmetric.

(b) Use MATLAB's `eig` function to decompose P into a matrix V of eigenvectors and a diagonal matrix D of the eigenvalues. Check that

$$P = VDV^{-1} \tag{5}$$

and that the columns of V are orthonormal (that is, $V_i^T V_j = 1$ if $i = j$, or 0 otherwise). Hence $V^{-1} = V^T$ (in other words, V is *unitary*). Look at the entries of D when the separation distance is small and when the separation distance becomes very large. How does the spectrum (the set of eigenvalues) change?

(c) Imagine that one of the diagonal entries in D was actually zero. Then no product Pq would ever have any component along the direction of the corresponding eigenvector. Verify this, if you like; refer to Trefethen and Bau for more details. Similarly, if an eigenvalue λ_i were extremely small, say $O(\varepsilon_{machine})$ (the computer's precision) then most likely Pq would have an extremely small component in the direction v_i (the corresponding eigenvector).

If we know "ahead of time" that we will have a few dominant eigenvalues and everything else will be very small, we might design an approximation to P in the following way: throw away all the eigenvalue/eigenvector pairs when the eigenvalue is "small enough." Then, to multiply by $\hat{P}$, our *low-rank approximation* to Pq is

$$Pq \approx \hat{P}q = \hat{V}\hat{D}\hat{V}^T q \tag{6}$$

where $\hat{V}$ denotes the eigenvectors corresponding to the largest k eigenvalues and $\hat{D}$ is the k by k matrix of those eigenvalues. Calculate $\hat{P}$ for $k = \{1, 2, 8\}$.

(d) Vary the separation between the clusters and plot the relative errors $||Pq - \hat{P}q||$ as the separation increases.

(e) Assume there are n charges in each cluster. How much memory is required to store P? How many floating point operations are required to find the product Pq?

(f) If we don't store P but instead a rank k approximation $\hat{P}$, how much memory is needed? How many floating point operations are needed to calculate $\hat{P}q$? In the limit as n becomes very large, how do P and $\hat{P}$ compare?