Solving the Modified Diffusion Equation

E.H.F.

May, 2005

1 Introduction

We're going to use the discrete Fourier transform discussed last time to solve the modified diffusion equation. After covering the basics of the algorithm, we'll talk about how to implement it in the C programming language with the help of the *fftw* packages.

2 The Algorithm

We are trying to solve the modified diffusion equation

$$\frac{\partial}{\partial t}q(x,t) = \frac{\partial^2}{\partial x^2}q(x,t) - w(x)q(x,t) \tag{1}$$

for the initial condition $q(x, 0) = 1$ where $w(x)$ is the external field. Since the propagator $q(x, t)$ describes a single polymer chain, $w(x)$ is the external field that acts on the polymer. For today's purposes, we'll assume the field is independent of the contour variable t, but in general (and in particular for a diblock copolymer) this is not the case.

Formally, we solve the diffusion equation by defining the linear operator

$$\mathcal{L} \equiv \frac{\partial^2}{\partial x^2} - w(x) \tag{2}$$

and writing the solution of a linear first order differential equation in time as

$$q(x,t) = e^{\mathcal{L}t}q(x,0). \tag{3}$$

Consider a small contour step Δt. Then at any point along the contour t, we can move to $t + \Delta t$ by

$$q(x, t + \Delta t) = e^{\mathcal{L}\Delta t} q(x, t) \tag{4}$$

It turns out that an efficient way to implement this exponential operator is to "split" the two parts.

$$e^{\mathcal{L}\Delta t} = e^{-w(x)\Delta t/2} e^{(\partial^2/\partial x^2)\Delta t} e^{-w(x)\Delta t/2} + O(\Delta t^3) \tag{5}$$

It is a good exercise to convince oneself that the two expressions are equivalent up to second order in Δt. In this exercise, it is not necessary to assume that the two operators $w(x)$ and $\partial^2/\partial x^2$ commute.

Since these operators are finite dimensional matrices in a computer, it is convenient to represent them as such.

$$D_{jk} = \left(e^{(\partial^2/\partial x^2)\Delta t} \right)_{jk} \tag{6}$$

$$W_{jk} = \left(e^{-w(x)\Delta t/2} \right)_{jk} \tag{7}$$

So then the total operator for moving along the contour is

$$e^{\mathcal{L}\Delta t} \to \sum_{k,l} W_{jk} D_{kl} W_{lm} \tag{8}$$

where all sums run from 0 to $n - 1$. This matrix acts on the vector

$$q_j(t) \tag{9}$$

that is the discrete representation of the propagator in position space. In position space, W_{jk} is diagonal, so this step only requires n multiplications. Unfortunately, this is not the case for D_{jk}. To handle this, recall discrete Fourier transform matrices I defined last time

$$F^\dagger_{kj} = n^{-1/2} e^{-i(2\pi/n)kj} \tag{10}$$

$$F_{kj} = n^{-1/2} e^{i(2\pi/n)kj} \tag{11}$$

We showed last time that

$$F_{lk} F^\dagger_{kj} = \delta_{lj} \tag{12}$$

so these matricies give a resolution of the identity matrix. Also,

$$\hat{f}_k = \sum_j F^\dagger_{kj} f_j \tag{13}$$

is the discrete Fourier transform of a vector.

Then we can claim

$$\begin{aligned} e^{\mathcal{L}\Delta t} q(x,t) &= \sum_{k,l,m} W_{jk} D_{kl} W_{lm} q_m(t) \\ &= \sum_{k,l,m,o,p} W_{jk} \delta_{kl} D_{lm} \delta_{mo} W_{op} q_p(t) \\ &= \sum_{k,l,m,o,p,r} W_{jk} F_{kl} F^\dagger_{lm} D_{mo} F_{op} F^\dagger_{pq} W_{qr} q_r(t) \end{aligned}$$

We've basically inserted Fourier transform inverse transform pairs where they are appropriate. So then

$$\sum_r W_{qr} q_r(t) \tag{14}$$

is n multiplications in position space,

$$\sum_{q,r} F^\dagger_{pq} W_{qr} q_r(t) \tag{15}$$

is the Fourier transform of the resulting vector. The matrix

$$\sum_{m,o} F^\dagger_{lm} D_{mo} F_{op} \tag{16}$$

is diagonal in Fourier space. In fact, if you're in the mood for a serious challenge, show that

$$\sum_{m,o} F^\dagger_{lm} D_{mo} F_{op} = \exp\left(-4 \left[\sin \frac{\pi p}{n} \right]^2 \Delta t \frac{n^2}{L^2} \right) \delta_{lp} \tag{17}$$

After performing n multiplications in Fourier space, take an inverse Fourier transform and do another n multiplications; these two steps correspond to

$$W_{jk} F_{kl} \tag{18}$$

at the very left the expression for $e^{\mathcal{L}\Delta t}$.

The previous analysis gives the following algorithm for solving the diffusion equation starting at any point $q_j(t)$:

- at each point is position space, multiply by

$$q''_j(t+\Delta t) = e^{-w(jL/n)\Delta t/2} q_j(t) \tag{19}$$

- take the forward Fourier transform of this vector

$$\hat{q}''_k(t+\Delta t) = \sum_{j=0}^{n} F^{\dagger}_{kj} q''_j(t+\Delta t) \tag{20}$$

- multiply this vector by the exponential of the diffusion operator in Fourier space:

$$\hat{q}'_k(t+\Delta t) = e^{-(2\pi/L)^2 k^2 \Delta t} \hat{q}''_k(t+\Delta t) \tag{21}$$

  for $k = 0, 1, \ldots n/2$ and

$$\hat{q}'_k(t+\Delta t) = e^{-(2\pi/L)^2 (n-k)^2 \Delta t} \hat{q}''_k(t+\Delta t) \tag{22}$$

  for $k = (n/2)+1, \ldots, n-1$.

- take the inverse Fourier transform

$$q'_j(t+\Delta t) = \sum_{k=0}^{n-1} F_{jk} \hat{q}'_k(t+\Delta t) \tag{23}$$

- multiply by the exponential of the field again in position space

$$q_j(t+\Delta t) = e^{-w(jL/n)\Delta t/2} q'_j(t+\delta t) \tag{24}$$

Here, I've used the number primes to denote the number of steps away from the actual $q_j(t+\Delta t)$.

Everything in the algorithm should seem straightforward expect for the part about applying the diffusion operator in Fourier space. To understand the form of exponential operator, consider eq. 17 for large n:

$$e^{-4\pi^2 k^2 \Delta t/n^2} \tag{25}$$

This looks very similiar to what we apply in the algorithm; we replace the n with a L in the algorithm. To understand this, consider that the basis function can be written

$$e^{-i(2\pi/n)kj} = e^{-i(2\pi k/L)(jL/n)} \approx e^{-i(2\pi k/L)x} \tag{26}$$

where $x_j = jL/n$ and then $2\pi k/L$ is the wave vector. The operator in the algorithm contains the square of this wave vector. In some sense, we're taking the continuous limit of what we state in eq. 17.

A strange thing happens as well in this step. We stop in the middle at $k = n/2$ and suddenly start using $n - k$ instead of k. To understand this, think back to how we represent function in the Fourier basis using plane waves

$$f_j = n^{-1/2} \sum_{k=0}^{n-1} \hat{f}_k e^{i(2\pi/n)jk} \tag{27}$$

In reality, this discrete Fourier transform should be written

$$f_j = n^{-1/2} \sum_{k=-(n/2)+1}^{n/2} \hat{f}_k e^{i(2\pi/n)jk} \tag{28}$$

where the wave vectors of interest are distributed symmetrically around 0. The previous equation is also correct due to the periodicity of the plane waves as a function of k, and I believe that this form is convenient in implementing the recursive fast Fourier transform. But, in reality, we take a chunk of wave vectors symmetric around the origin called the first Brillouin zone. This implies that $\hat{f}_k$ for $k > n/2$ really correspond to $\hat{f}_{-k}$, and we must account for this in the algorithm.