single-spin-flip algorithm. Start in a magnetized state, and watch the spins rearrange until roughly half are pointing up. Start at high temperatures, and watch the up- and down-spin regions grow slowly. Run a large enough system that you get tired of waiting for equilibration.

The Wolff algorithm flips large clusters of spins at one time, largely bypassing the sluggishness near T_c. It can only be implemented at zero external field. It is described in detail in Exercise 8.8.

(b) *Implement the Wolff algorithm. A recursive implementation works only for small system sizes on most computers. Instead, put the spins that are destined to flip on a list* `toFlip`. *You will also need to keep track of the sign of the original triggering spin.*

```
While there are are spins toFlip,
  if the first spin remains parallel to the original,
      flip it, and
      for each neighbor of the flipped spin,
          if it is parallel to the original spin,
              add it to toFlip with probability $p$.
```

(c) *Estimate visually how many Wolff cluster flips it takes to reach the equilibrium state at* T_c. *Is Wolff faster than the single-spin-flip algorithms? How does it compare at high temperatures?*

(d) *Starting from a random configuration, change to a low temperature* $T = 1$ *and observe the equilibration using a single-spin flip algorithm. Compare with your Wolff algorithm.* (See also Exercise 12.3.) *Which reaches equilibrium faster? Is the dynamics changed qualitatively, though?*

(8.10) **Stochastic cells.**[35] (Biology, Computation) ④
Living cells are amazingly complex mixtures of a variety of complex molecules (RNA, DNA, proteins, lipids, . . .) that are constantly undergoing reactions with one another. This complex of reactions has been compared to computation; the cell gets input from external and internal sensors, and through an intricate series of reactions produces an appropriate response. Thus, for example, receptor cells in the retina 'listen' for light and respond by triggering a nerve impulse.

The kinetics of chemical reactions are usually described using differential equations for the concentrations of the various chemicals, and rarely are statistical fluctuations considered important. In a cell, the numbers of molecules of a given type can be rather small; indeed, there is (often) only one copy of the relevant part of DNA for a given reaction. It is an important question whether and when we may describe the dynamics inside the cell using continuous concentration variables, even though the actual numbers of molecules are always integers.

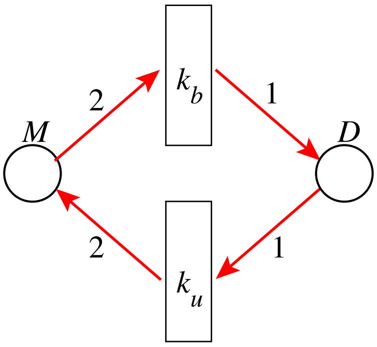

Fig. 8.11 Dimerization reaction. A Petri net diagram for a dimerization reaction, with dimerization rate k_b and dimer dissociation rate k_u.

Consider a dimerization reaction; a molecule M (called the 'monomer') joins up with another monomer and becomes a dimer D: $2M \longleftrightarrow D$. Proteins in cells often form dimers; sometimes (as here) both proteins are the same (homodimers) and sometimes they are different proteins (heterodimers). Suppose the forward reaction rate is k_d and the backward reaction rate is k_u. Figure 8.11 shows this as a Petri net [50] with each reaction shown as a box, with incoming arrows showing species that are consumed by the reaction, and outgoing arrows showing species that are produced by the reaction; the number consumed or produced (the *stoichiometry*) is given by a label on each arrow. There are thus two reactions: the backward unbinding reaction rate per unit volume is $k_u[D]$ (each dimer disassociates with rate k_u), and the forward binding reaction rate per unit volume is $k_b[M]^2$ (since each monomer must wait for a collision with another monomer before binding, the rate is proportional to the monomer concentration squared).

[35] This exercise and the associated software were developed in collaboration with Christopher Myers.

The brackets [.] denote concentrations. We assume that the volume per cell is such that one molecule per cell is 1 nM (10^{-9} moles per liter). For convenience, we shall pick nanomoles as our unit of concentration, so $[M]$ is also the number of monomers in the cell. Assume $k_b = 1\,\mathrm{nM}^{-1}\mathrm{s}^{-1}$ and $k_u = 2\,\mathrm{s}^{-1}$, and that at $t = 0$ all N monomers are unbound.

(a) Continuum dimerization. *Write the differential equation for* $\mathrm{d}M/\mathrm{d}t$ *treating* M *and* D *as continuous variables.* (Hint: Remember that two M molecules are consumed in each reaction.) *What are the equilibrium concentrations for* $[M]$ *and* $[D]$ *for* $N = 2$ *molecules in the cell, assuming these continuous equations and the values above for* k_b *and* k_u*? For* $N = 90$ *and* $N = 10\,100$ *molecules? Numerically solve your differential equation for* $M(t)$ *for* $N = 2$ *and* $N = 90$*, and verify that your solution settles down to the equilibrium values you found.*

For large numbers of molecules in the cell, we expect that the continuum equations may work well, but for just a few molecules there surely will be relatively large fluctuations. These fluctuations are called *shot noise*, named in early studies of electrical noise at low currents due to individual electrons in a resistor. We can implement a Monte Carlo algorithm to simulate this shot noise.[36] Suppose the reactions have rates Γ_i, with total rate $\Gamma_{\mathrm{tot}} = \sum_i \Gamma_i$. The idea is that the expected time to the next reaction is $1/\Gamma_{\mathrm{tot}}$, and the probability that the next reaction will be j is $\Gamma_j/\Gamma_{\mathrm{tot}}$. To simulate until a final time t_f, the algorithm runs as follows.

(1) Calculate a list of the rates of all reactions in the system.

(2) Find the total rate Γ_{tot}.

(3) Pick a random time t_{wait} with probability distribution $\rho(t) = \Gamma_{\mathrm{tot}} \exp(-\Gamma_{\mathrm{tot}}\, t)$.

(4) If the current time t plus t_{wait} is bigger than t_f, no further reactions will take place; return.

(5) Otherwise,

- increment t by t_{wait},
- pick a random number r uniformly distributed in the range $[0, \Gamma_{\mathrm{tot}})$,
- pick the reaction j for which $\sum_{i<j} \Gamma_i \leq r < \sum_{i<j+1} \Gamma_i$ (that is, r lands in the jth interval of the sum forming Γ_{tot}),
- execute that reaction, by incrementing each chemical involved by its stoichiometry.

(6) Repeat.

There is one important additional change:[37] the binding reaction rate for M total monomers binding is no longer $k_b M^2$ for discrete molecules; it is $k_b M(M-1)$.[38]

(b) Stochastic dimerization. *Implement this algorithm for the dimerization reaction of part (a). Simulate for* $N = 2$*,* $N = 90$*, and* $N = 10\,100$ *and compare a few stochastic realizations with the continuum solution. How large a value of* N *do you need for the individual reactions to be well described by the continuum equations (say, fluctuations less than* $\pm 20\%$ *at late times)?*

Measuring the concentrations in a single cell is often a challenge. Experiments often average over many cells. Such experiments will measure a smooth time evolution even though the individual cells are noisy. Let us investigate whether this ensemble average is well described by the continuum equations.

(c) Average stochastic dimerization. *Find the average of many realizations of your stochastic dimerization in part (b), for* $N = 2$ *and* $N = 90$*, and compare with your deterministic solution. How much is the long-term average shifted by the stochastic noise? How large a value of* N *do you need for the ensemble average of* $M(t)$ *to be well described by the continuum equations (say, shifted by less than 5% at late times)?*

(8.11) **The repressilator.**[39] (Biology, Computation) ④

The 'central dogma' of molecular biology is that the flow of information is from DNA to RNA to proteins; DNA is *transcribed* into RNA, which then is *translated* into protein.

Now that the genome is sequenced, it is thought that we have the parts list for the cell. All that

[36] In the context of chemical simulations, this algorithm is named after Gillespie [45]; the same basic approach was used just a bit earlier in the Ising model by Bortz, Kalos, and Lebowitz [19], and is called *continuous-time Monte Carlo* in that context.

[37] Without this change, if you start with an odd number of cells your concentrations can go negative!

[38] Again $[M] = M$, because we assume one molecule per cell gives a concentration of 1 nM.

[39] This exercise draws heavily on Elowitz and Leibler [37]; it and the associated software were developed in collaboration with Christopher Myers.

remains is to figure out how they work together! The proteins, RNA, and DNA form a complex network of interacting chemical reactions, which governs metabolism, responses to external stimuli, reproduction (*proliferation*), *differentiation* into different cell types, and (when the cell perceives itself to be breaking down in dangerous ways) programmed cell death, or *apoptosis*.

Our understanding of the structure of these interacting networks is growing rapidly, but our understanding of the dynamics is still rather primitive. Part of the difficulty is that the cellular networks are not neatly separated into different modules; a given protein may participate in what would seem to be several separate regulatory pathways. In this exercise, we will study a model gene regulatory network, the *repressilator*. This experimental system involves three proteins, each of which inhibits the formation of the next. They were added to the bacterium *E. coli*, with hopefully minimal interactions with the rest of the biological machinery of the cell. We will implement the stochastic model that the authors used to describe their experimental system [37]. In doing so, we will

- implement in a tangible system an example both of the central dogma and of *transcriptional regulation*: the control by proteins of DNA expression into RNA,
- introduce sophisticated Monte Carlo techniques for simulations of stochastic reactions,
- introduce methods for automatically generating continuum descriptions from reaction rates, and
- illustrate the *shot noise* fluctuations due to small numbers of molecules and the *telegraph noise* fluctuations due to finite rates of binding and unbinding of the regulating proteins.

Figure 8.12 shows the biologist's view of the repressilator network. Three proteins (TetR, λCI, and LacI) each repress the formation of the next. We shall see that, under appropriate circumstances, this can lead to spontaneous oscillations; each protein peaks in turn, suppressing the suppressor of its suppressor, leading to its own later decrease.

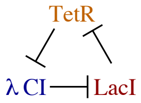

Fig. 8.12 Biology repressilator. The biologist's view of the repressilator network. The T-shapes are blunt arrows, signifying that the protein at the tail (bottom of the T) suppresses the production of the protein at the head. Thus LacI (pronounced lack-eye) suppresses TetR (tet-are), which suppresses λ CI (lambda-see-one). This condensed description summarizes a complex series of interactions (see Fig. 8.13).

The biologist's notation summarizes a much more complex picture. The LacI protein, for example, can bind to one or both of the *transcriptional regulation* or *operator* sites ahead of the gene that codes for the tetR mRNA.[40] When bound, it largely blocks the translation of DNA into tetR.[41] The level of tetR will gradually decrease as it degrades; hence less TetR protein will be translated from the tetR mRNA. The resulting network of ten reactions is depicted in Fig. 8.13, showing one-third of the total repressilator network. The biologist's shorthand (Fig. 8.12) does not specify the details of how one protein represses the production of the next. The larger diagram, for example, includes two operator sites for the repressor molecule to bind to, leading to three states (P_0, P_1, and P_2) of the promoter region depending upon how many LacI proteins are bound.

You may retrieve a simulation package for the repressilator from the computational exercises portion of the book web site [129].

(a) *Run the simulation for at least 6000 seconds and plot the protein, RNA, and promoter states as a function of time. Notice that*

- *the protein levels do oscillate, as in [37, figure 1(c)],*
- *there are significant noisy-looking fluctuations,*
- *there are many more proteins than RNA.*

[40] Messenger RNA (mRNA) codes for proteins. Other forms of RNA can serve as enzymes or parts of the machinery of the cell. Proteins in *E. coli* by convention have the same names as their mRNA, but start with capitals where the mRNA start with small letters.

[41] *RNA polymerase*, the molecular motor responsible for transcribing DNA into RNA, needs to attach to the DNA at a *promoter site*. By binding to the adjacent operator sites, our repressor protein inhibits this attachment and hence partly blocks transcription. The residual transcription is called 'leakiness'.

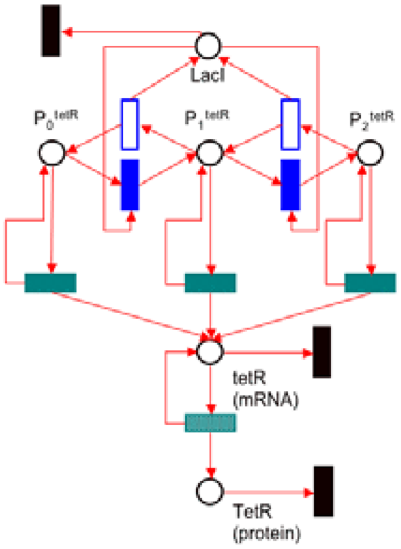

Fig. 8.13 Computational repressilator. The Petri net version [50] of *one-third* of the repressilator network (the LacI repression of TetR). The biologist's shorthand (Fig. 8.12) hides a lot of complexity! We have implemented these equations for you, so studying this figure is optional. The solid lighter vertical rectangles represent binding reactions $A + B \to C$, with rate $k_b[A][B]$; the open vertical rectangles represent unbinding $C \to A + B$, with rate $k_u[C]$. The horizontal rectangles represent catalyzed synthesis reactions $C \to C+P$, with rate $\gamma[C]$; the darker ones represent transcription (formation of mRNA), and the lighter one represent translation (formation of protein). The black vertical rectangles represent degradation reactions, $A \to$ nothing with rate $k_d[A]$. The LacI protein (top) can bind to the DNA in two *promoter sites* ahead of the gene coding for tetR; when bound, it largely blocks the transcription (formation) of tetR mRNA. P_0 represents the promoter without any LacI bound; P_1 represents the promoter with one site blocked, and P_2 represents the doubly-bound promoter. LacI can bind to one or both of the promoter sites, changing P_i to P_{i+1}, or correspondingly unbind. The unbound P_0 state transcribes tetR mRNA quickly, and the bound states transcribe it slowly (leaky repression). The tetR mRNA then catalyzes the formation of the TetR protein.

To see how important the fluctuations are, we should compare the stochastic simulation to the solution of the continuum reaction rate equations (as we did in Exercise 8.10). In [37], the authors write a set of six differential equations giving a continuum version of the stochastic simulation. These equations are simplified; they both 'integrate out' or coarse-grain away the promoter states from the system, deriving a Hill equation (Exercise 6.12) for the mRNA production, and they also rescale their variables in various ways. Rather than typing in their equations and sorting out these rescalings, it is convenient and illuminating to write a routine to generate the continuum differential equations directly from our reaction rates.

(b) *Write a* `DeterministicRepressilator`*, derived from* `Repressilator` *just as* `StochasticRepressilator` *was. Write a routine* `dcdt(c,t)` *that does the following.*

- *Sets the chemical amounts in the reaction network to the values in the array* `c`*.*
- *Sets a vector* `dcdt` *(of length the number of chemicals) to zero.*
- *For each reaction:*
  - *compute its rate;*
  - *for each chemical whose stoichiometry is changed by the reaction, add the stoichiometry change times the rate to the corresponding entry of* `dcdt`.

Call a routine to integrate the resulting differential equation (as described in the last part of Exercise 3.12, for example), and compare your results to those of the stochastic simulation.

The stochastic simulation has significant fluctuations away from the continuum equation. Part of these fluctuations are due to the fact that the numbers of proteins and mRNAs are small; in particular, the mRNA numbers are significantly smaller than the protein numbers.

(c) *Write a routine that creates a stochastic repressilator network that multiplies the mRNA concentrations by* `RNAFactor` *without otherwise affecting the continuum equations. (That is, multiply the initial concentrations and the transcription rates by* `RNAFactor`*, and divide the translation rate by* `RNAFactor`*.) Try boosting the* `RNAFactor` *by ten and one hundred. Do the RNA and protein fluctuations become significantly smaller?* This noise, due to the discrete, integer values of chemicals in the cell, is analogous to the *shot noise* seen in electrical circuits due to the discrete quantum of electric charge. It scales, as do most fluctuations, as the square root of the number of molecules.

A continuum description of the binding of the proteins to the operator sites on the DNA seems particularly dubious; a variable that must be zero or one is replaced by a continuous evolution between these extremes. (Such noise in other contexts is called *telegraph noise*—in analogy to the telegraph, which is either silent or sending as the operator taps the key.) The continuum description is accurate in the limit where the binding and unbinding rates are fast compared to all of the other changes in the system; the protein and mRNA variations then see the average, local equilibrium concentration. On the other hand, if the rates are slow compared to the response of the mRNA and protein, the latter can have a switching appearance.

(d) *Incorporate a* `telegraphFactor` *into your stochastic repressilator routine, that multiplies the binding and unbinding rates. Run for 1000 seconds with* `RNAFactor` $= 10$ *(to suppress the shot noise) and* `telegraphFactor` $= 0.001$. *Do you observe features in the mRNA curves that appear to switch as the relevant proteins unbind and bind?*

(8.12) **Entropy increases! Markov chains.** (Mathematics) ③

Convexity arguments are a basic tool in formal statistical mechanics. The function $f(x) = -x \log x$ is strictly concave (convex downward) for $x \geq 0$ (Fig. 5.9); this is easily shown by noting that its second derivative is negative in this region.

(a) Convexity for sums of many terms. *If* $\sum_\alpha \mu_\alpha = 1$, *and if for all* α *both* $\mu_\alpha \geq 0$ *and* $x_\alpha \geq 0$, *show by induction on the number of states* M *that if* $g(x)$ *is concave for* $x \geq 0$, *then*

$$g\left(\sum_{\alpha=1}^{M} \mu_\alpha x_\alpha\right) \geq \sum_{\alpha=1}^{M} \mu_\alpha g(x_\alpha). \qquad (8.25)$$

This is a generalization of Jensen's inequality (eqn 5.27), which was the special case of equal μ_α. (Hint: In the definition of concave, $f(\lambda a + (1-\lambda)b) \geq \lambda f(a) + (1-\lambda)f(b)$, take $(1-\lambda) = \mu_{M+1}$ and $b = x_{M+1}$. Then a is a sum of M terms, rescaled from their original values. Do the coefficients of x_α in a sum to one? Can we apply induction?)

In Exercise 5.7 you noticed that, formally speaking, entropy does not increase in Hamiltonian systems. Let us show that it does increase for Markov chains.[42]

The Markov chain is implicitly exchanging energy with a heat bath at the temperature T. Thus to show that the entropy for the world as a whole increases, we must show that $\Delta S - \Delta E/T$ increases, where ΔS is the entropy of our system and $\Delta E/T$ is the entropy flow from the heat bath. Hence, showing that entropy increases for our Markov chain is equivalent to showing that the free energy $E - TS$ decreases.

Let $P_{\alpha\beta}$ be the transition matrix for a Markov chain, satisfying detailed balance with energy E_α at temperature T. The current probability of being in state α is ρ_α. The free energy

$$F = E - TS = \sum_\alpha \rho_\alpha E_\alpha + k_B T \sum_\alpha \rho_\alpha \log \rho_\alpha. \qquad (8.26)$$

(b) *Show that the free energy decreases for a Markov chain. In particular, using eqn 8.25, show that the free energy for* $\rho_\beta^{(n+1)} = \sum_\alpha P_{\beta\alpha}\rho_\alpha^{(n)}$ *is less than or equal to the free energy for* $\rho^{(n)}$. *You may use the properties of the Markov transition matrix* P, *(*$0 \leq P_{\alpha\beta} \leq 1$ *and* $\sum_\alpha P_{\alpha\beta} = 1$*), and detailed balance (*$P_{\alpha\beta}\rho_\beta^* = P_{\beta\alpha}\rho_\alpha^*$, *where* $\rho_\alpha^* = \exp(-E_\alpha/k_B T)/Z$*).* (Hint: You will want to use $\mu_\alpha = P_{\alpha\beta}$ in eqn 8.25, but the entropy will involve $P_{\beta\alpha}$, which is *not* the same. Use detailed balance to convert from one to the other.)

(8.13) **Hysteresis and avalanches.**[43] (Complexity, Computation) ④

A piece of magnetic material exposed to an increasing external field $H(t)$ (Fig. 8.14) will magnetize (Fig. 8.15) in a series of sharp jumps, or *avalanches* (Fig. 8.16). These avalanches arise as magnetic domain walls in the material are pushed by the external field through a rugged potential energy landscape due to irregularities and impurities in the magnet. The magnetic signal resulting from these random avalanches is called *Barkhausen noise*.

We model this system with a non-equilibrium lattice model, the *random field Ising model*. The Hamiltonian or energy function for our system is

$$\mathcal{H} = -\sum_{\langle i,j \rangle} J s_i s_j - \sum_i \left(H(t) + h_i\right) s_i, \qquad (8.27)$$

[42] We know that the Markov chain eventually evolves to the equilibrium state, and we argued that the latter minimizes the free energy. What we are showing here is that the free energy goes continuously downhill for a Markov chain.

[43] This exercise is largely drawn from [69]. It and the associated software were developed in collaboration with Christopher Myers.