10.34 Final Exam - Fall 2006
Solution

Overall: Average = 62.5 St. Dev. = 11.25

1) (20 points total) **Average = 14.2 St. Dev. = 3.8**

You are using the Gillespie Algorithm in a Kinetic Monte Carlo simulation of a biochemical process forming a protein. This process occurs at only one active site in a cell, and it initiates quickly, so there is almost always a growing protein attached to the site. This process runs correctly most of the time, but in about one case in a million the protein folds the wrong way. Ideally, the simulation could precisely determine what fraction of the cells will contain N={0, 1, 2, …} copies of the misfolded protein after two days.

The simulation includes two processes: addition of an amino acid to the growing protein (time constant approximately 1 millisecond) and misfolding of the growing protein (time constant approximately 10^5 seconds). In the simulation, the protein detaches from the active site immediately when the final amino acid is added, and a new protein immediately starts growing at the site. The rates of both processes vary a little bit depending on the size of the growing chain; these details are included in your simulation. The protein consists of 100 amino acids.

a) (10 points) Approximately how many random numbers will the computer have to generate in order to simulate what happens in a single cell over the course of two days?

In the Gillespie algorithm, one needs to generate 2 random numbers for each reaction event; one determines the time until the event, while the other determines the reaction that will take place at the given time. So we need to estimate the tau for the system in order to estimate the number of time steps that will be taken to reach 2 days (172800 seconds).

$$\tau = \left(\frac{1}{0.001s} + \frac{1}{1\times10^5 s}\right)^{-1} = 0.001 \ \ {}^{\text{sec}}\!/_{event}$$

So, in order to reach our goal of 2 days of simulation time, we need to divide the total time by the time per event (on average) to get to total number of events that will take place (on average). This is:

$$N_{events} = \frac{t_{Total}}{-\tau \cdot \ln\left(rand\left[0,1\right]\right)} \qquad note: \ \left\langle \ln\left(rand\left[0,1\right]\right)\right\rangle = -1 \quad \neq \ln\left(\left\langle rand\left[0,1\right]\right\rangle\right) = -0.693$$

$$N_{events} == \frac{t_{Total}}{\tau} = \frac{172800 \text{ sec}}{0.001 \ {}^{\text{sec}}\!/_{event}} = 1.728\times10^8 \ events$$

Since we said that each event requires the generation of 2 random numbers, the answer is:

$$\boxed{N_{rand\,\#} = 3.456\times10^8}$$

b) (10 points) Approximately how many cells will you need to simulate to compute the ensemble average $\left\langle \mathrm{N}_{\text{unfolded}}\left(t=2\ days\right)\right\rangle$ to a standard error of less than $\pm 1.0\%$? If the CPU time for the simulation is dominated by the cost of generating random numbers, and generating each random number takes 1 microsecond of CPU time, approximately how many CPU ***hours*** will be required for to compute this ensemble average to this level of accuracy using Gillespie's Algorithm?

There are several ways to solve this problem, all of which have "minor" aspects that must be taken care of to get the right answer. Many of you were fortunate that you got approximately the correct answer, but this was only because <N> ~ O(1) and σ_N ~ O(1).

Method #1:
It was mentioned that the Gillespie algorithm (and other MC methods) do not work very efficiently for systems in which there are very minor channels, as is the case in this problem. This part will illustrate this fact. The equation of important for this part is for the standard deviation in the fraction of events:

$$\frac{\sigma_{f_i}}{f_i}=\sqrt{\frac{1-f_i}{f_i\cdot N_{proteins}}} \qquad \text{where } f_i \text{ is the fraction of times event } i \text{ happens}$$

It is also important to remember here that we are talking about the entire protein of 100 amino acids, which means it takes 100 reaction events to generate a protein. However, one misfolding event will ruin the entire protein. The fraction of misfolding can be calculated as:

$$f_{unfold}=\frac{\frac{1\ misfold}{10^5\ \text{sec}}}{\frac{1\ a.acid}{10^{-3}\ \text{sec}}}\cdot 100\,\frac{a.acid}{protein}=1\times10^{-6}\ \frac{misfolded\ \ proteins}{total\ \ proteins\ created}$$

The problem asked us to calculate the number of trajectories needed to achieve an error of less than 1% in the number of misfolded proteins after 2 days. This is done by applying the above two results:

$$\frac{\sigma_{f_i}}{f_i}=0.01=\sqrt{\frac{1-f_i}{f_i\cdot N_{proteins}}} \qquad \Rightarrow \qquad N_{proteins}=\frac{1-f_i}{f_i}\cdot\frac{1}{\left(0.01\right)^2}=1\times10^{10}\ \ proteins\ created$$

Since it takes the number of proteins is the number of events divided by 100 ($1.728\text{x}10^6$), we have:

$$N_{cells}=N_{proteins\ total}\big/N_{proteins\ per\ cell}=1\times10^{10}\big/1.728\times10^6 \qquad \Rightarrow \qquad \boxed{N_{cells}>5787\ cells}$$

The problem also asked for the CPU hours necessary to perform this simulation if the CPU time is governed by generating random numbers and that each random number take 1.0 microsecond to generate. This is a simple calculation since we have already calculated the number of random numbers per trajectory and the total number of trajectories. The result would be:

$$Wall\ Time = 5787\ cells \cdot 3.456\times10^{8}\ \frac{rand\,\#}{cell} \cdot 1.0\times10^{-6}\ \frac{\text{sec}}{rand\,\#} \cdot \frac{1\ hr}{3600\ \text{sec}} = 555.5\ hrs$$

$$\boxed{Wall\ Time > \ 555.5\ hrs}$$

Method #2:
You could also realize directly that the uncertainty in Monte Carlo simulations typically go as $N^{-½}$. In this case it is important to realize that the N you calculate here is number of misfolded protein events (not necessarily the number of cells) that must occur to have an error in that number of less than 1%. You can determine the time necessary as follows:

$$\frac{\delta N_{misfold}}{\left\langle N_{misfold}\right\rangle} \approx \frac{1}{\sqrt{N_{misfold}}} \quad \Rightarrow \quad N_{misfold} \approx \left(\frac{\left\langle N_{misfold}\right\rangle}{\delta N_{misfold}}\right)^{2} = \left(\frac{1}{0.01}\right)^{2} = 1\times10^{4}\ misfold\ events$$

However, what we want is the number of cells needed...

$$N_{misfold\,/\,cell} = N_{misfold\ \ per\ \ protein} \cdot N_{proteins\,per\,cell} = 1\times10^{-6}\cdot1.728\times10^{6} = 1.728\ misfolds\ \ per\ cell$$

$$N_{cells} = N_{misfold\ \ needed} \,/\, N_{misfold\ \ per\ \ cell} = 1\times10^{4}/1.728 \quad \Rightarrow \quad \boxed{N_{cells} > 5787\ cells}$$

The rest of this is the same as above, and the total time will be 555.5 hours.

Method #3:
You could also approach this problem from a confidence interval approach. In this it was important to understand the difference between the sigma of the data and the sigma of the mean. This is the way you would go about this (not dealing with the log-normal aspects of the distribution). This is a somewhat weaker argument because it makes some assumptions about what you get from the analysis. The major points of ambiguity are:

1. Technically, a X% confidence interval is the percentage of the time that the actual mean will fall within the interval $<N_{misfold}>_{measured}$ +/- CI, for the given number of degrees of freedom. What we will estimate is the number of cells needed in order to be X% confidence that the true mean will fall with the $<N_{misfold}>_{measured}$ +/- 1% interval.
2. Since we don't have any actual data, assumptions will need to be made regarding the mean and standard deviation of the data. It will be assumed that $<N_{misfold}>_{measured} = <N_{misfold}>_{Ncells \to \infty}$. It will also be assumed that the sigma of the data is related to the expected mean as is the case for a true Poisson distribution with $N_{samples} \to \infty$.

$$\left\langle N_{misfold}\right\rangle_{data} + CI = 1.01\cdot\left\langle N_{misfold}\right\rangle_{data} \quad \Rightarrow \quad \frac{CI}{\left\langle N_{misfold}\right\rangle_{data}} = 0.01$$

From Method #2, we found that the $<N_{misfold}>$ = 1.728, which leads to the following expression:

$$CI = 0.01728 \quad \Rightarrow \quad CI = t_{\alpha,\nu}\frac{\sigma_{N_{misfold}}}{\sqrt{N_{cells}}} = 0.01728 \quad \Rightarrow \quad N_{cells} > \left(t_{\alpha,\nu}\frac{\sigma_{N_{misfold}}}{0.01728}\right)^2$$

Now, to get a numerical answer, we need to make some assumption about how confident we want to be in our 1% error, and get an estimate of the standard deviation in the $N_{misfold}$ per cell. Since ν will likely be large, the t-distribution will basically be identical to the normal (z) distribution. If we want to be 99% confident that the true mean lies in our 1% interval, then $t_{\alpha,\nu} \sim 2.6$, and if we wanted to be 67% confident, then $t_{\alpha,\nu} \sim 1.0$. Also, for the Poisson distribution, the sigma can be approximately related to the expected mean value: $\sigma_{N_{misfold}} \approx \sqrt{\langle N_{misfold}\rangle} = \sqrt{1.728} = 1.315$

This will results in the following values of N_{cells} for 99% and 70% confidence that the true mean will fall within the +/- 1% of the expected value interval:

$$\boxed{N_{cells}^{99\%} = \left(2.6\cdot\frac{1.315}{0.01728}\right)^2 > 2.9\times10^4\ cells \qquad N_{cells}^{67\%} = \left(1.0\cdot\frac{1.315}{0.01728}\right)^2 > 5787\ cells}$$

$$Time\left(N_{cells}^{99\%}\right) > 2800\ hrs \qquad Time\left(N_{cells}^{67\%}\right) > 555.5\ hrs$$

You will notice that the other methods imply that a 67% confidence (which equates to about 1 standard deviation) in the true mean being in the +/- 1% interval. If you wanted to have higher confidence that the true mean would fall within the 1% error bounds, you would have to perform more simulations.

Method #4:
This method is somewhat similar to Method #2, and it starts with the equation for the uncertainty in the evaluation of a Monte Carlo integral:

$$\delta I = \sqrt{\frac{\langle f^2\rangle - \langle f\rangle^2}{N_{samples}}} \quad \xrightarrow{\text{similar to}} \quad \delta N_{misfold} = \sqrt{\frac{\langle N_{misfold}^2\rangle - \langle N_{misfold}\rangle^2}{N_{cells}}}$$

However, this is the absolute uncertainty in the $N_{misfold}$, so we must divide by the expected value in order to equate it to a relative error of 1%. From before, the expected value is 1.728 misfolds per cell.

$$\frac{\delta N_{misfold}}{\langle N_{misfold}\rangle} = \frac{1}{\langle N_{misfold}\rangle}\sqrt{\frac{\langle N_{misfold}^2\rangle - \langle N_{misfold}\rangle^2}{N_{cells}}} \quad \Rightarrow \quad 0.01728 = \sqrt{\frac{\langle N_{misfold}^2\rangle - \langle N_{misfold}\rangle^2}{N_{cells}}}$$

So this will give us an expression for N_{cells} in terms of the fluctuation in the number of misfolds per cell:

$$N_{cells} = \frac{\langle N_{misfold}^2\rangle - \langle N_{misfold}\rangle^2}{(0.01728)^2} = 3349\cdot\left[\langle N_{misfold}^2\rangle - \langle N_{misfold}\rangle^2\right]\ cells$$

The problem with this is getting an estimate of the fluctuation in the number of misfolds, which is required to estimate the number of cells needed. It turns out that the above fluctuation is equivalent to the variance in the data (for a large number of samples), as is seen below:

$$\sigma^2 = \frac{1}{\nu}\sum\left(f_i - \overline{f}\right)^2 = \frac{1}{\nu}\sum\left(f_i^2 - 2 f_i \overline{f} + \overline{f}^2\right) \qquad note: \overline{f} = \langle f \rangle \approx \frac{1}{\nu}\sum\left(f_i\right)$$

$$\sigma^2 = \left\langle f_i^2 \right\rangle + \langle f \rangle^2 - 2\frac{1}{\nu}\sum\left(f_i \overline{f}\right) \qquad where: \; 2\frac{1}{\nu}\sum\left(f_i \langle f \rangle\right) = 2\langle f \rangle \frac{1}{\nu}\sum\left(f_i\right) = 2\langle f \rangle^2$$

$$\therefore \; \sigma^2 = \left\langle f_i^2 \right\rangle - \langle f \rangle^2$$

As mentioned in Method #3, we can estimate the sigma in the data for a Poisson distribution as the sqrt(expected value), which means that the variance can be approximated as the expected value (1.728). Therefore, we have the following expression:

$$\boxed{N_{cells} > 3349 \cdot \left[\left\langle N_{misfold}^2 \right\rangle - \left\langle N_{misfold} \right\rangle^2\right] = 3349 \cdot \sigma_{N_{misfold}}^2 \approx 3349 \cdot 1.728 = 5787 \; cells}$$

This would again yield the same amount of CPU time, which was 555.5 hours.

From all of these analyses, you should see that if the expected value of the number of misfolded proteins per cell was not O(1), then the answer would be significantly different (and many of you who happened upon a reasonable answer would have been far off). As an example, consider two other cases where the $<N_{misfold}> = 10^{-3}$ or 10^3 per cell. In these cases, one would need to simulate $1\text{x}10^7$ and 10 cells, respectively, to ensure an error of < 1%. This would translate into CPU times of $9.6\text{x}10^5$ and 0.96 hours. As you can see, it was fortunate that $<N_{misfold}> = O(1)$.

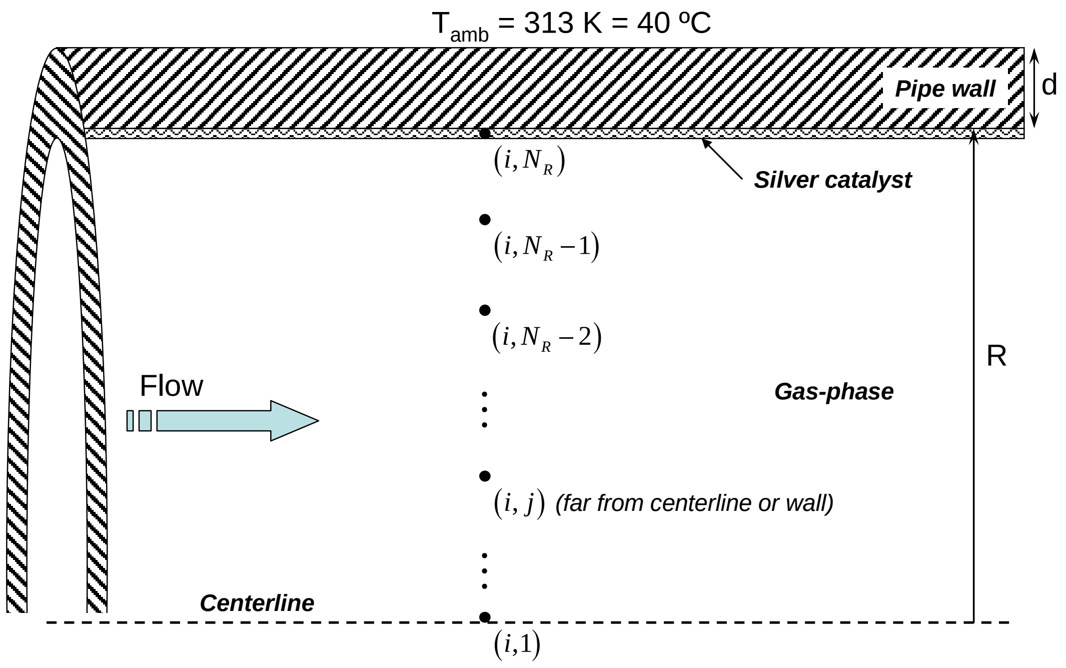
2) (30 points total) **Average = 21.9** **St. Dev. = 3.9**

The inside of a pipe, inner radius ***R***, wall thickness ***d***, is coated with a very thin layer of silver, which catalyzes the highly exothermic isomerization reaction ($\Delta H_{rxn} < 0$):

$$ethylene\ oxide(E) \rightarrow acetaldehyde(A) \qquad rate = k_{rxn}(T)\cdot\left(C^E - \frac{1}{K_{eq}}C^A\right)_{wall} \quad [=] \ \frac{moles}{m^2\cdot s}$$

(Exothermicity ΔH_{rxn}; Entropy change ΔS_{rxn}.). A mixture of ethylene oxide and acetaldehyde is flowing slowly down the pipe (so everything is laminar). The pressure at the inlet is $\mathbf{P_o}$, and at the output it is 1.00 bar. The length of the pipe is **L**. The gas viscosity is η, and its heat capacity is $\mathbf{C_P}$, both independent of temperature and composition. The thermal conductivity of the gas is κ_1, and the thermal conductivity of the pipe material is κ_2. The diffusivity of both isomers is **D** and the molecular weight is **MW**. The outside of the pipe is maintained at T = 40 ºC = 313 K. Because the pipe is thin-walled and the fluid is a gas, you can neglect axial temperature conduction in the pipe wall and gas phase. Make the ideal-gas approximation. You are simulating this reactor using *finite differences*. Be sure to include all appropriate indices when writing equations. The radial direction has been discretized using N_R number of points, with point $j=N_R$ on the boundary. *Do not use Method of Lines to solve this problem.*

Hint: In the full simulation there would be additional differential equations for the P and velocity fields, but you are not asked to derive these here. You can act as if the P and velocity field equations are already written out in finite difference form, and assume that all equations will be solved simultaneously with your T and C equations. Note that P and velocity will vary with position, and should include the appropriate indices when writing out the finite difference equations. The heat flux through the pipe wall can be taken to be: $\dot{q} = -\kappa_2\left(T_{amb} - T_{i,N_R}\right)/d$.



a) (10 points) Write the finite difference equations for the steady-state temperature (T) and concentration of ethylene oxide (C^E) at a generic mesh point ***(i,j)*** somewhere in the interior of the pipe (not near the walls nor the centerline). Let ***i*** be the discretization index in the axial direction, and ***j*** be the index in the radial direction.

In this part, we must derive the finite difference equations for a cylindrical geometry with diffusion, convection, heat transfer, and a reaction at a surface. This is very similar to the problem tackled in HW Set #8, insofar as the geometry is concerned. The governing equations for this system are:

$$\underline{v}\cdot\underline{\nabla}\left(\rho C_P T\right)=\kappa_1\cdot\nabla^2 T \quad\Rightarrow\quad v_z\frac{\partial\left(\rho T\right)}{\partial z}=\frac{\kappa_1}{C_P}\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T}{\partial r}\right)\right)=\frac{\kappa_1}{C_P}\left(\frac{1}{r}\frac{\partial T}{\partial r}+\frac{\partial^2 T}{\partial r^2}\right)$$

$$\underline{v}\cdot\underline{\nabla}\left(C^E\right)=D\cdot\nabla^2 T \quad\Rightarrow\quad v_z\frac{\partial C^E}{\partial z}=D\left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial C^E}{\partial r}\right)+\frac{\partial^2 C^E}{\partial z^2}\right)=D\left(\frac{1}{r}\frac{\partial C^E}{\partial r}+\frac{\partial^2 C^E}{\partial r^2}+\frac{\partial^2 C^E}{\partial z^2}\right)$$

Since we are dealing with a gas (and the constant density approximation would be invalid), we must write out the density as a function of T and P (using the ideal gas law) in the equation.

$$v_z\frac{\partial\left(\frac{P\cdot MW}{RT}\cdot T\right)}{\partial z}=\frac{v_z\cdot MW}{R}\frac{\partial P}{\partial z} \qquad\Rightarrow\qquad \frac{v_z\cdot MW}{R}\frac{\partial P}{\partial z}=\frac{\kappa_1}{C_P}\left(\frac{1}{r}\frac{\partial T}{\partial r}+\frac{\partial^2 T}{\partial r^2}\right)$$

Since we are only concerned with the interior points right now, and we never have to deal with boundary conditions at the entrance, exit, or centerline, we will ignore these for now. The discretization equations for the above equations are as follows (note the upwind difference was used in the convective term):

Heat equation terms:

$$\left.\frac{\partial P}{\partial z}\right|_{z_i}=\frac{P_{i,j}-P_{i-1,j}}{\Delta z} \qquad \left.\frac{\partial^2 T}{\partial r^2}\right|_{r_j}=\frac{T_{i,j+1}-2T_{i,j}+T_{i,j-1}}{\left(\Delta r\right)^2} \qquad \left.\frac{\partial T}{\partial r}\right|_{r_j}=\frac{T_{i,j+1}-T_{i,j-1}}{2\Delta r}$$

Diffusion equation terms:

$$\left.\frac{\partial C^E}{\partial z}\right|_{z_i}=\frac{C^E_{i,j}-C^E_{i-1,j}}{\Delta z} \qquad \left.\frac{\partial^2 C^E}{\partial z^2}\right|_{z_i}=\frac{C^E_{i+1,j}-2C^E_{i,j}+C^E_{i-1,j}}{\left(\Delta z\right)^2}$$

$$\left.\frac{\partial^2 C^E}{\partial r^2}\right|_{r_j}=\frac{C^E_{i,j+1}-2C^E_{i,j}+C^E_{i,j-1}}{\left(\Delta r\right)^2} \qquad \left.\frac{\partial C^E}{\partial r}\right|_{r_j}=\frac{C^E_{i,j+1}-C^E_{i,j-1}}{2\Delta r}$$

Now that we have the discretized form of the derivatives, we can substitute them into the PDEs above:

Heat equation:

$$v_z\cdot\frac{\partial P}{\partial z}=\beta\cdot\left(\frac{1}{r}\frac{\partial T}{\partial r}+\frac{\partial^2 T}{\partial r^2}\right) \qquad where:\ \beta=\frac{R\cdot\kappa_1}{MW\cdot C_P}$$

$$v_{i,j}\cdot\left(\frac{P_{i,j}-P_{i-1,j}}{\Delta z}\right)-\frac{\beta}{r_j}\cdot\left(\frac{T_{i,j+1}-T_{i,j-1}}{2\Delta r}\right)-\beta\cdot\left(\frac{T_{i,j+1}-2T_{i,j}+T_{i,j-1}}{\left(\Delta r\right)^2}\right)=0$$

$$\boxed{\left(\frac{v_{i,j}}{\Delta z}\right)P_{i,j}-\left(\frac{v_{i,j}}{\Delta z}\right)P_{i-1,j}-\left(\frac{1}{r_j}\frac{\beta}{2\Delta r}+\frac{\beta}{(\Delta r)^2}\right)T_{i,j+1}+\left(\frac{2\beta}{(\Delta r)^2}\right)T_{i,j}+\left(\frac{1}{r_j}\frac{\beta}{2\Delta r}-\frac{\beta}{(\Delta r)^2}\right)T_{i,j-1}=0}$$

Diffusion equation terms:

$$v_z\frac{\partial C^E}{\partial z}=D\left(\frac{1}{r}\frac{\partial C^E}{\partial r}+\frac{\partial^2 C^E}{\partial r^2}+\frac{\partial^2 C^E}{\partial z^2}\right)$$

$$v_{i,j}\left(\frac{C^E_{i,j}-C^E_{i-1,j}}{\Delta z}\right)-\frac{D}{r_j}\left(\frac{C^E_{i,j+1}-C^E_{i,j-1}}{2\Delta r}\right)-D\left(\frac{C^E_{i,j+1}-2C^E_{i,j}+C^E_{i,j-1}}{(\Delta r)^2}\right)-D\left(\frac{C^E_{i+1,j}-2C^E_{i,j}+C^E_{i-1,j}}{(\Delta z)^2}\right)=0$$

$$\boxed{-\left(\frac{D}{(\Delta z)^2}\right)C^E_{i+1,j}+\left(\frac{v_{i,j}}{\Delta z}+\frac{2D}{(\Delta r)^2}+\frac{2D}{(\Delta z)^2}\right)C^E_{i,j}-\left(\frac{v_{i,j}}{\Delta z}+\frac{D}{(\Delta z)^2}\right)C^E_{i-1,j}-\left(\frac{1}{r_j}\frac{D}{2\Delta r}+\frac{D}{(\Delta r)^2}\right)C^E_{i,j+1}+\left(\frac{1}{r_j}\frac{D}{2\Delta r}-\frac{D}{(\Delta r)^2}\right)C^E_{i,j-1}=0}$$

b) (20 points) Write the steady-state boundary conditions for T and C^E at the wall ***(i,N_R)***. What are the finite-difference equations for T and C^E mesh points in the gas phase immediately adjacent to the wall ***(i,N_R-1)***?

For this part, we are asked to write out the finite difference equations for the boundary at the wall (and no other boundary conditions). The correct way to go about this is to equate the fluxes at the interface. The concentration-field BC is something you should have seen already, with the need to match the diffusive flux to the wall and the consumption rate/flux of the surface reaction. The temperature-field BC is slightly different, since you have a flux from the wall to the gas, generation of heat at the wall, and a flux from the wall to the ambient (through the pipe wall). However, the same idea of equating fluxes at the boundary holds true because we are at steady-state and there is no accumulation of heat at the interface. The species and thermal flux balances are as follows:

$$-D\frac{\partial C^E}{\partial r}\bigg|_{i=N_R}=rate\ of\ reaction=k_{rxn}\left(T_{i,N_R}\right.\cdot\left[C^E_{i,N_R}-\frac{1}{K_{Eq}}\cdot C^A_{i,N_R}\right]$$

We can write out the quadratic approximation to the first derivative at the wall with the following:

$$\frac{\partial C^E}{\partial r}\bigg|_{i=N_R}=\frac{3C^E_{i,N_R}-4C^E_{i,N_R-1}+C^E_{i,N_R-2}}{2\Delta r}$$

This is then substituted into the flux balance to yield the final equation which is used to determine the concentration of ethylene oxide at the wall (when solved simultaneously with the interior points).

$$\boxed{\overbrace{\left(\frac{3C^E_{i,N_R}-4C^E_{i,N_R-1}+C^E_{i,N_R-2}}{2\Delta r}\right)}^{will\ be<0}+\overbrace{\frac{k_{rxn}\left(T_{i,N_R}\right)}{D}\cdot\left[C^E_{i,N_R}-\frac{1}{K_{Eq}}\cdot C^A_{i,N_R}\right]}^{will\ be>0}=0}$$

For the temperature-field boundary condition, we have three flux terms and the following equations relating them (see eqn 2.5-1 in Deen's book):

$$\begin{array}{ccccc} \textit{Heat generated by Rxn} & = & \textit{Flux to gas} & - & \textit{Flux to ambient through pipe} \\ \dot{Q}_{Rxn} & = & \vec{\boldsymbol{n}}_i \bullet \vec{\boldsymbol{q}}_{gas} & - & \vec{\boldsymbol{n}}_i \bullet \vec{\boldsymbol{q}}_{pipe} \end{array}$$

Since the flux in the pipe and gas are both defined in the same direction (in terms of increasing **r**), the **n*q** dot product of the unit normal vector of the interface with the flux vectors will both result in +q.

$$\overbrace{\left(k_{rxn}\left(T_{i,N_R}\right) \cdot \left[C^E_{i,N_R} - \frac{1}{K_{Eq}} \cdot C^A_{i,N_R} \right] \cdot \Delta H_{rxn} \right)}^{will\ be\ <0} = \overbrace{\overbrace{\left(-\kappa_1 \left.\frac{\partial T}{\partial r}\right|^{gas}_{i=N_R} \right)}^{will\ be\ \le 0} - \overbrace{\left(-\kappa_2 \left.\frac{\partial T}{\partial r}\right|^{pipe\ wall}_{i=N_R} \right)}^{will\ be\ \ge 0}}^{will\ be\ \le 0}$$

The above equation shows how the terms are related. The flux to the gas will be (-) because the temperature should increase as you get closer to the wall since there is no source term in the gas-phase. The flux to the ambient will be (+) because the ambient is the heat sink (T will decrease as you approach T_{amb} with increasing **r**). Therefore, the above equation makes qualitative sense. For a given amount of heat generated by the reaction, part must go to the gas and part must go to the ambient, the distribution will depend on the κ-values and the solution to the convection-conduction-diffusion problem in the interior points.

Since the formula for the flux through the pipe wall was given, we simply need to insert that term and write out the finite difference approximation for the dT/dr in the gas at the wall (similar to above for C^E):

$$\boxed{\overbrace{\left(k_{rxn}\left(T_{i,N_R}\right) \cdot \left[C^E_{i,N_R} - \frac{1}{K_{Eq}} \cdot C^A_{i,N_R} \right] \cdot \Delta H_{rxn} \right)}^{will\ be\ \le 0} + \overbrace{\kappa_1 \left(\frac{3T_{i,N_R} - 4T_{i,N_R-1} + T_{i,N_R-2}}{2\Delta r} \right)}^{will\ be\ \ge 0} - \overbrace{\kappa_2 \left(\frac{T_{amb} - T_{i,N_R}}{d} \right)}^{will\ be\ \le 0} = 0}$$

We were also asked to write out the finite difference equations for the grid point directly adjacent to the wall, point (*i , j=N_R-1*). This is very straightforward and simply involves re-writing the general equations for interior grid points, but using a specific index. This will show how the boundary condition equations are necessary to specify the values of the state variables at the boundary, which show up in the finite difference equations near the boundaries. They would be as follows:

$$\boxed{\left(\frac{v_{i,N_R-1}}{\Delta z} \right) P_{i,N_R-1} - \left(\frac{v_{i,N_R-1}}{\Delta z} \right) P_{i-1,N_R-1} - \left(\frac{1}{r_{N_R-1}} \frac{\beta}{2\Delta r} + \frac{\beta}{(\Delta r)^2} \right) \boldsymbol{T_{i,\ R}} + \left(\frac{2\beta}{(\Delta r)^2} \right) T_{i,N_R-1} + \left(\frac{1}{r_{N_R-1}} \frac{\beta}{2\Delta r} - \frac{\beta}{(\Delta r)^2} \right) T_{i,N_R-2} = 0}$$

and

$$\boxed{-\left(\frac{D}{(\Delta z)^2} \right) C^E_{i+1,N_R-1} + \left(\frac{v_{i,N_R-1}}{\Delta z} + \frac{2D}{(\Delta r)^2} + \frac{2D}{(\Delta z)^2} \right) C^E_{i,N_R-1} - \left(\frac{v_{i,N_R-1}}{\Delta z} + \frac{D}{(\Delta z)^2} \right) C^E_{i-1,N_R-1} - \left(\frac{1}{r_{N_R-1}} \frac{D}{2\Delta r} + \frac{D}{(\Delta r)^2} \right) \boldsymbol{C^E_{i,N_R}} + \left(\frac{1}{r_{N_R-1}} \frac{D}{2\Delta r} - \frac{D}{(\Delta r)^2} \right) C^E_{i,N_R-2} = 0}$$

The state variables that are highlighted in bold in the above equations are the ones that are determined by simultaneously solving the BC equations along with the interior points.

3) (50 points total) **Average = 26.3** **St. Dev. = 7.0**
Frosty and Claus have published a model for the shape of mist droplets resting on mistletoe leaves; of course the shapes depend on the angle at which the leaf is suspended from the ceiling.

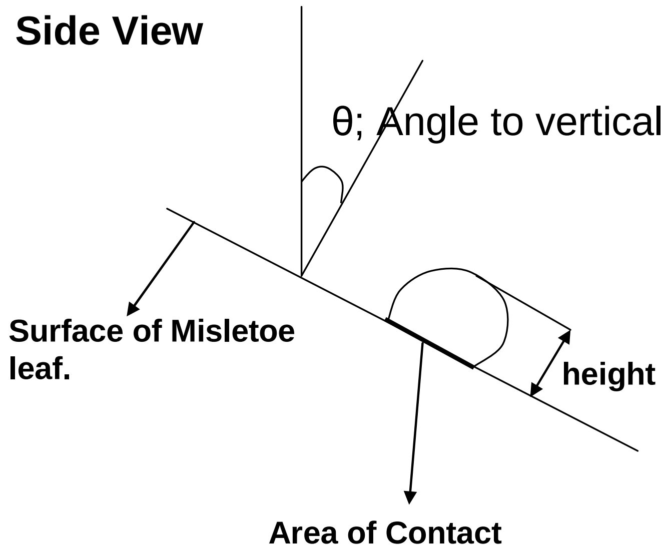
In particular, their model predicts how the height of each drop depends on

1. the angle at which the leaf is suspended and
2. the contact area between the droplet and the leaf.

Height = Y_{model}(angle, area; liquid parameters)

A postdoc in your research group has written a Matlab function that evaluates their model predictions, and also a Matlab function that computes certain derivatives of how their model predictions depend on the input parameters to the model (p1=the leaf-droplet surface tension (N/m), p2=the droplet-air surface tension (N/m), and p3=the density of the liquid in the droplet (g/cc)). The last two input parameters are expected to have values similar to those for pure water (air/water surface tension= 0.72 N/m; density of water = 1.00 g/cc), but slightly changed due to some of the biomolecules in the leaf dissolving into the droplet. For details on the Matlab functions, which the postdoc has given to you, see the header files below. The postdoc is now in northern Norway doing fieldwork and is unavailable for questions.

A hardworking UROP in your laboratory has put mist droplets on a mistletoe leaf, and then photographed the leaf from different perspectives. Using a Matlab image-analysis program she has recorded data on the leaf-droplet contact area and the height of 100 droplets, with the leaf suspended at three different angles theta. (See figure). At your request, while you were busy filming the 1st year skit, she repeated each experiment 400 times. She has now left for an extended vacation to recuperate. Before she left town she gave you all the data as 120,000 rows. (300 blocks of 400 rows each). You notice that there is perfect reproducibility for the angle and area measurements, but large fluctuations in the measured heights.



Trial Number	*Angle*	*Droplet Number*	*Area*	*height*	
1	Angle1	Droplet 1	A1	h1	400 repeats
2	Angle1	Droplet 2	A2	h2	400 repeats
⋮	⋮	⋮	⋮	⋮	
101	Angle2	Droplet1	A1	h101	400 repeats
⋮	⋮	⋮	⋮	⋮	
300	Angle3	Droplet100	A100	h300	400 repeats

You have sorted the data (as shown in the table) so the first 400 entries are data measured with the leaf at angle#1 for droplet #1; the next 400 are for droplet#2, etc. through all 100 droplets. Then there are 400 numbers measured with the leaf at angle#2 for droplet#1, then 400 for droplet#2, etc. Finally there are the 40,000 entries measured with the leaf at angle#3. The sorted data is available in Matlab as matrix SortedRawData (120000 rows and 5 columns). You of course will immediately condense each set of 400 replicates to a single point <height>$_n$ with an estimated variance σ_n.

a) (20 points) Write a Matlab function which takes SortedRawData and values of the 3 parameters as inputs, and returns as its output the corresponding value of χ^2.

Each of the 300 experiments is repeated 400 times. First we will calculate the average of those 400 values and then calculate the standard deviation of the 400 values σ. Since the average is calculated using 400 different data points, the average will have a standard deviation of $\sigma_{av} = \frac{\sigma}{\sqrt{400}} = \frac{\sigma}{20}$. First let us convert the 120,000 height values into 300 average heights and corresponding standard deviations. The average height $\underline{h}$ and its standard deviation σ_{av} are given by the expression below

$$\underline{h} = \frac{1}{400}\sum_{i=1}^{400} h_i \qquad \text{and} \qquad \sigma_{av} = \frac{1}{20}\sqrt{\frac{1}{399}\sum_{i=1}^{400}(h_i - \underline{h})^2}$$

This is done by the matlab program "calc_avg_std".

Once the average values and standard deviation of the 300 entries are known we can calculate the χ^2 value given by the model. The value of χ^2 is given by the expression below, where A is the contact area of the drop and θ is the angle of the leaf. This calculation is performed by the matlab function "calc_chi2".

$$\chi^2 = \sum_{i=1}^{300} \frac{\left(\underline{h_i} - f_i(A,\theta;p_1,p_2,p_3)\right)^2}{\sigma_{av,i}^2}$$

In the above expression $f_i(A,\theta;p_1,p_2,p_3)$ is just short for $f(A_i,\theta_i;p_1,p_2,p_3)$.

```
function chi2 = calc_chi2(SortedRawData, p1,p2,p3)
%calculates the chi2 for the data using the function "thickness"
%input data include the sortedRawData, p1 is leaf_tension, p2
%drop_air_tension, p3 is density of water

%output is the chi2

%first obtain the average heights and standard deviations from calc_avg_std
%notice only the 5th column is passed to the function
[height, std] = calc_avg_std(SortedRawData(:,5);

area = zeros(300,1); %extract area from SortedRawData
angle = zeros(300,1); % extract angle from SortedRawData

for i=1:300
    area(i) = SortedRawData((i-1)*400 + 1,4);
    angle(i) = SortedRawData((i-1)*400 + 1,2);
end

chi2 = 0; %initialize the chi2 and start adding terms to it

for i=1:300
    chi2 = chi2+(height(i)- thickness(area(i), angle(i), p1, p2, p3))^2 ...
    /std(i)^2;
end

return;




function [avg, std] =  calc_avg_std(height)
% takes in a vector of length 120,000 and returns a vector of length 300
% of the averages and vector of length 300 of the standard deviations.
avg = zeros(300,1);
std = zeros(300,1);

%first calculate the average
for i=1:3 %iterate over angles
    for j=1:100 %iterate over drops
        for k=1:400 %iterate over all the repeats
            avg((i-1)*100 + j) = avg((i-1)*100+j) + height((i-1)*100*400...
            + (j-1)*400 + k);
        end
    end
end

avg = avg/400;

%now calculate the standard deviation
```

```
for i=1:3 %iterate over angles
    for j=1:100 %iterate over drops
        for k=1:400 %iterate over all the repeats
            std((i-1)*100 + j) = std((i-1)*100 + j) + ...
                1/20 * ((avg((i-1)*100+j) - height((i-1)*100*400 + ...
                (j-1)*400 + k))^2/399)^0.5;
        end
    end
end

return;
```

b) (10 points) In their journal paper, Frosty and Claus suggest these values of the three parameters: p1 = 0.95 N/m, p2 = 0.73 N/m, p3 = 1.02 g/cc.
Suppose your function from part (a) is working and is bug-free. When you give it these values for p1,p2,p3 the function returns as its output the number 682. Is Frosty & Claus's model, using their suggested parameter values, consistent with the experimental data measured by your UROP? If you cannot determine this numerically right now, write down an inequality that, if true, would indicate that there was a high probability that the model and the data were consistent.

The χ^2 value should ideally be very close to the number of experimental values – number of fitted parameters, in this case this is 300-3 = 297. Thus the χ^2 obtained, 682, is too high and the model is most likely incorrect. The rigorous test is given by the p-value of the model. If the p-value if greater than 0.05 then the model is likely correct or if it is less than 0.05 then the model is most likely incorrect. The matlab function which can be used to perform this test is chi2cdf and is given below.
p-value = 1-chi2cdf(χ^2 ,297) > 0.05.

We can also write the criteria as follows

$$\int_{\chi^2}^{\infty} \frac{(1/2)^{\nu/2}}{\Gamma(\nu/2)} x^{\nu/2-1} e^{-x/2} dx \geq 0.05$$

Or

$$\frac{\Gamma(\nu/2, x/2)}{\Gamma(\nu/2)} \geq 0.05$$

c) (10 points) Your research advisor asks you to refine Frosty & Claus's suggested parameters by taking a single Newton-type step. Write a short Matlab function which would accomplish this, returning p_refined.

The parameters will be refined by taking a Newton-type step so that the χ^2 value is minimized.

$$\chi^2 = \sum_{i=1}^{300} \frac{\left(\underline{h_i} - f_i(A,\theta; p_1, p_2, p_3)\right)^2}{\sigma_{av,i}^2}$$

Now let us calculate the gradient and the Hessian of this cost function.

$$\nabla_{\underline{p}}\chi^2 = \begin{bmatrix} \frac{\partial \chi^2}{\partial p_1} \\ \frac{\partial \chi^2}{\partial p_2} \\ \frac{\partial \chi^2}{\partial p_3} \end{bmatrix}$$

The kth element of the vector is given below.

$$\frac{\partial \chi^2}{\partial p_k} = -\sum_{i=1}^{300} 2\frac{\left(\underline{h_i} - f_i(A,\theta;p_1,p_2,p_3)\right)}{\sigma_{av,i}^2}\frac{\partial f_i}{\partial p_k}$$

The Hessian of the cost function is a 3 X 3 matrix and l,k element of the matrix is given below

$$H_{lk} = \frac{\partial^2 \chi^2}{\partial p_l \partial p_k} = \sum_{i=1}^{300} 2\frac{1}{\sigma_{av,i}^2}\frac{\partial f_i}{\partial p_l}\frac{\partial f_i}{\partial p_k} - \sum_{i=1}^{300} 2\frac{\left(\underline{h_i} - f_i(A,\theta;p_1,p_2,p_3)\right)}{\sigma_{av,i}^2}\frac{\partial^2 f_i}{\partial p_l \partial p_k}$$

In the above expression we will ignore the second term and thus the Hessian becomes

$$H_{lk} = \sum_{i=1}^{300} 2\frac{1}{\sigma_{av,i}^2}\frac{\partial f_i}{\partial p_l}\frac{\partial f_i}{\partial p_k} = 2X^T X$$

Where

$$X_{i,j} = \frac{1}{\sigma_i}\frac{\partial f_i}{\partial p_j}$$

A Newton step in given below

$$\underline{p}^{m+1} = \underline{p}^m - \left[X^T X \,|_{\underline{p}^m} \right]^{-1} \nabla_{\underline{p}}\chi^2$$

For our problem the matrix X will have 300 rows and 3 columns. All the partial derivatives with respect to the parameters required to calculate X and $\nabla_{\underline{p}}\chi^2$ are given by the function “dh_dp”. This single step of the Newton method is carried out by the matlab function “p_refined”.

```
function p_new = p_refined(p_old, height, sigma, area, angle)
%p_old is a vector containing  leaf_tension, drop_air_tension and density
%of water. area is the vector of areas of the droplet and angle is the
%vector of angles made by the maple tree. These three vectors are of length
%300 and are extracted from SortedRawData in the same was as was done in
%the last problem.
```

```
X = zeros(300,3); % X is a matrix of 300 rows and 3 columns

for i=1:300 % fill all the rows
    X(i,:) = dh_dp(angle(i), area(i), p_old(1), p_old(2), p_old(3))...
      sigma(i);
end

g = [0; 0; 0];
error = 0; %calculate the error

for i=1:300
    g(1) = g(1) - (h(i) - thickness(angle(i), area(i), p_old(1), ...
        p_old(2), p_old(3))*X(i,1)/sigma(i)^2;

    g(2) = g(2) - (h(i) - thickness(angle(i), area(i), p_old(1), ...
        p_old(2), p_old(3))*X(i,2) /sigma(i)^2;

    g(3) = g(3) - (h(i) - thickness(angle(i), area(i), p_old(1), ...
        p_old(2), p_old(3))*X(i,3) /sigma(i)^2;
end

p_new = p_old - (X'*X)^(-1)*g;

return;
```

d) (10 points) Suppose that when you run the Matlab function asked for in part (c), everything works wonderfully: χ^2(p_refined) is very small, and the fit to the data looks wonderful. You decide to take a second Newton-type step, starting from p_refined. But this time you get the error message "Matrix singular or ill-conditioned". What does this mean? Your goal is to publish a good fit with information about the uncertainties in the parameters. You have the good fit; what would you do next so that you will be able to say something definitive about the uncertainties in p_refined, without performing more experiments? Explain clearly.

If the Newton-type step starting from p_refined gives the error that the matrix is singular or ill-conditioned then at least one of the three parameters cannot be determined accurately. The way to diagnose the problem is to perform the singular value decomposition of the Hessian matrix $X^T X$ (this Hessian matrix is a positive definite square matrix and SVD is the same as eigenvalue decomposition). This will give three singular values one of which will be very close to 0. The column of right singular matrix (V) corresponding to the 0 singular value is the direction which cannot be resolved by these sets of experiments. There can be two problems;

1. The experimental design is such that you are unable to decouple the dependence of two parameters.
2. The model is such that two parameters always occur in the model with a definite relation.

However in this case it is not practical to do any more experiments. What you should do next is to change variables to θ such that

$$\theta = V^T p$$

Again coming back to the equation that we had to solve

$$\left[X^T X \,|_{\underline{p}^m}\right] \Delta p = -\nabla_{\underline{p}} \chi^2 \qquad \text{Eqn 1}$$

Here Δp is the update in the parameter values. We also know that $\left[X^T X \,|_{\underline{p}^m}\right]$ is a symmetric square matrix. Thus the SVD of $\left[X^T X \,|_{\underline{p}^m}\right]$ is just the eigenvalue decomposition and can be written as

$$\left[X^T X \,|_{\underline{p}^m}\right] = V \Sigma V^T \qquad \text{Eqn 2}$$

Substituting this equation 2 back into the equation 1 we obtain the following

$$V \Sigma V^T \Delta p = -\nabla_{\underline{p}} \chi^2$$

Now since we are transforming the variables from p to θ, we can write down the following

$$V \Sigma \Delta \theta = -\nabla_{\underline{p}} \chi^2$$

In the above equation we will ignore the values of $\Delta\theta$ that correspond to the 0 eigenalues. The rest of the eigenvalues are solved easily as follows.

$$\Delta\theta_i = \frac{1}{\sigma_i}[V^T(-\nabla_{\underline{p}} \chi^2)]_i \qquad \text{Eqn 3}$$

The above equation shows that the data that we have is sufficient to determine the values of θ that correspond to the columns of V which have a corresponding non-zero singular value. The error in these directions is proportional to $\frac{1}{\sqrt{\sigma_i}}$. The 95% confidence intervals on the parameters θ is given below

$$\theta_i = \theta_{i,M} \pm \frac{Z_{2.5}}{\sqrt{\sigma_i}},$$

Where $\theta_{i,M}$ is best value of θ_i calculated using equation 3.

Functions given in the problem statements:

```
function h=thickness(angle,area,leaf_tension,drop_air_tension,rho)
% computes the droplet height above a mistletoe leaf predicted by the model %  of
Frosty and Claus.
%     Written by Peter Postdoc, Dec. 15, 2006
% inputs:
%    angle [=] radians that the leaf normal makes with vertical. For a leaf lying
%        flat, angle=0.
%   area [=] cm2    area of the leaf wetted by the droplet
%   leaf_tension [=] N/m     surface tension at the droplet/leaf interface
%  drop_air_tension [=] N/m   surface tension at droplet/air interface
%  rho [=] g/cm3    density of the liquid in the droplet
% output:
%    h [=] cm    maximum thickness of the droplet

function derivs=dh_dp(angle,area,leaf_tension,drop_air_tension,rho)
% computes the partial derivatives of the droplet thickness
% with respect to surface tensions and density,
% as predicted by the model of Frosty and Claus.
%     Written by Peter Postdoc, Dec. 15, 2006
% inputs:
%    angle [=] radians that the leaf normal makes with vertical. For a leaf lying
%        flat, angle=0.
%   area [=] cm2    area of the leaf wetted by the droplet
%   leaf_tension [=] N/m     surface tension at the droplet/leaf interface
%   drop_air_tension [=] N/m   surface tension at droplet/air interface
%   rho [=] g/cm3    density of the liquid in the droplet
% output:
%   derivs is a 3-vector with components:
%    derivs(1) [=] cm/(N/m)  d(thickness of the droplet)/d(leaf-drop tension)
%    derivs(2) [=] cm/(N/m)  d(thickness of the droplet)/d(air-drop tension)
%    derivs(3) [=] cm/(g/cm3)  d(thickness of the droplet)/d(rho)
```