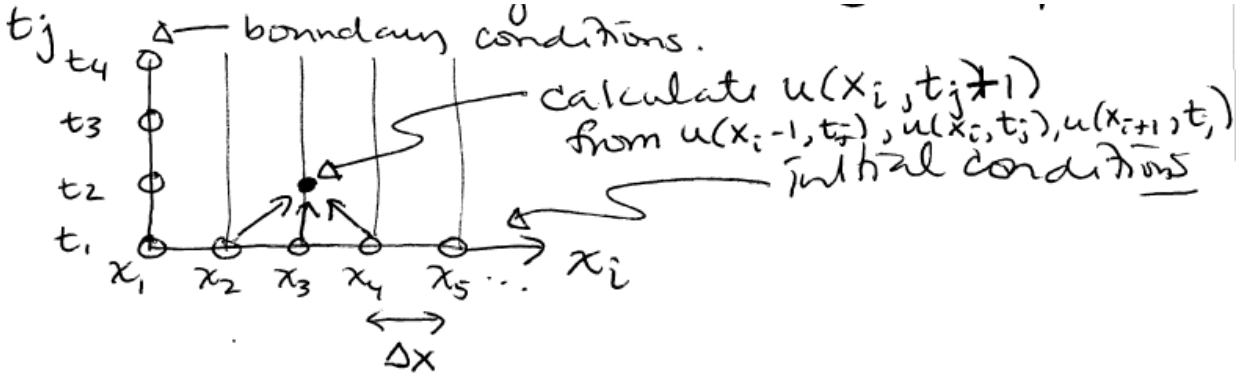
Lecture 8: PDE numeric methods and nondimensionalization

Solving PDE problems numerically:

PDEs can be solved using many algorithms, referred to as finite element methods (FEM), finite difference methods and finite volume methods. FEM are most common, and solve transient problems by turning the system into a discretized mesh that can be solved with ordinary differential equations, and thus uses the same solver algorithms as ODEs once this is set up.

We will not address these numerical methods here, but will go over the simplest possible method, to help you understand the general principle of how PDEs can be solved numerically, by describing the Finite Difference Method, originally proposed by Richardson and Schmidt in the 1920's.

In the finite difference approach, we break the space into a grid and solve at each node. I illustrate this here with a 1D simulation, so we just need to calculate u(x,t).



The initial conditions tell use $u(x_i, t_1 = 0)$ for each point x_i on our grid. The boundary conditions tell us $u(x_1, t_j)$ at all time points t_j, if x1 is our left boundary. Thus, the information in the open circles in the illustration above is known from our initial and boundary conditions.

Now we step forward in time, calculating $u(x_i, t_{j+1})$For any point x_i on the interior of our grid, as illustrated by the solid circle in the illustration above. (At the first iteration,$j = 1$). We calculate this by applying the equation of the PDE to all the points adjacent to x_i, using their known values at time t_j. That is, we use the known values at $u(x - \Delta x, t), u(x, t), u(x + \Delta x, t)$. We can obtain approximations for the gradient and the Laplacian in terms of these known values using first and second-order Taylor expansions:

$$\frac{\partial u}{\partial x} \sim \frac{u(x + \Delta x) - u(x)}{\Delta x}$$

$$\frac{\partial^2 u}{\partial x^2} \sim \frac{u(x + \Delta x) - 2u(x) + u(x - \Delta x)}{\Delta x^2}$$

This allows us to convert the PDE into a vector (for 1D, or an array for 2D or 3D) of ODEs. For example, the diffusion equation, which is $\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}$ in 1D, becomes an array defined for I = 2 to

N-1, where N is the number of nodes: $\frac{du(i)}{dt} = \frac{D}{\Delta x^2}[\,u(i+1) - 2u(i) + u(i-1)]$. This can be solved with ODE solvers. However, we can't use this equation to calculate $\frac{du(1)}{dt}$ since there is no $u(i-1)$, and $\frac{du(N)}{dt}$, since there is no $u(N+1)$. Instead, we use the boundary conditions to replace these unknown expressions with a prescribed value of u, or we replace $u(i+1) - u(i)$ with a proper scaling of the prescribed flux.

However, most people now use programs such as COMSOL, that have many choices for algorithms that break the space into meshes or nodes, and then solve the PDEs, just as we used ODE solvers in MATLAB. Thus, we will not dwell on solution algorithms.

Recommended PDE Numeric Model Solving method:

1. Equations: Build the PDE mathematical model, including PDEs, initial conditions, boundary conditions (and thus geometry), and parameter values. Even if you use software like COMSOL that writes the equations for you, you should understand the underlying physics to ensure you are using the software correctly.
2. Numeric Methods: Convert the mathematical model to a numeric model by selecting algorithms that determine how the problem is solved, including solving algorithm, spatial mesh, and (for time-dependent solutions) the temporal discretization. You have three goals, in order of importance when you select these: 1) running without errors 2) convergence (no significant numeric artifacts in the outcomes that affect your conclusions), and 3) efficiency.
    a. Solvers. There are multiple solver algorithms that have been developed over the years. Different solvers are used for stationary (steady state) versus transient (time dependent) solutions. Some are optimized for different problems, but in many cases, the newer ones are simply more efficient for the same accuracy. Indeed, solving algorithms have improved as quickly as hardware, following a Moore's law-type curve; numerical solutions to PDE benefit from increases in both hardware and algorithm speeds and so are much more efficient now than even a few years ago.
        i. Direct or factorization solving methods find an approximate answer through matrix factorization. They require more memory and are CPU time intensive, especially for larger problems. An example is Gaussian Elimination (GE). With direct methods, a problem that is not well-conditioned will give an error that the solution does not match the tolerance.
        ii. Iterative methods start with an approximate initial guess and then iteratively improve it. These are often better for larger problems but are harder to get working for some sets of equations (that is, some types of physics). Examples include Gauss-Seidel (GS), Conjugate Gradient (CG), Successive Over Relaxation (SOR) methods. With iterative methods, the error gets steadily smaller (converges) unless the problem is not well-conditioned.
        iii. Multigrid (MG) methods are a modification of iterative methods that address a major shortcoming. Iterative methods use fourier modes (sine waves) forms of solutions that are iteratively improved and do not work well when the waves

are larger than the grid size so have some shortcomings that are made worse by making finer grids. This is solved by MG methods.

b. Spatial Mesh. There are numerous meshing algorithms, each of which provides advantages in different situations. Meshes can be set shapes, or algorithms can create shapes to fit the geometry. Once a type of mesh is chosen, the size (fine vs course) needs to be selected as well. As with the timestep, there is a trade-off between accuracy and efficiency. However, good meshing techniques can improve both by using a fine mesh only at locations in the geometry where the solution has a steep spatial gradient. Try to find a setting that dynamically builds a mesh to adapt to the geometry and even the physics during a solution (e.g. physics-controlled mesh in COMSOL.) If this does not provide a satisfactory solution, then use help files to learn to manually improve the mesh by making it finer at the appropriate locations, such as a boundary layer mesh applied to the boundaries you indicate. If the geometry or spatial gradient changes dramatically during the simulation, see if your software has an adaptive mesh that will be revised as needed during the simulation.

c. Temporal Grid and Tolerance. Default in COMSOL is 0.001 relative tolerance. Tolerance will determine number of iterations or the integration time step, for stationary and transient solutions. Absolute tolerance is in the units of your dependent variables. If the total tolerance is the sum of absolute and relative tolerance, then the least restrictive (largest) tolerance will control the simulation. The default in COMSOL is to automatically determine the integration time step needed to meet the tolerance setting, and to save the solution at the times requested by the user. Remember that the integration step size is not necessarily the same as the output step size. Changing the output step size may be needed to make a smoother graph of an output of interest when the simulation has converged.

3. Verify your model results when possible in a simple situation, at least until you are familiar with how to use a software package for the type of equations you are solving.

   a. Convergence. The most important verification tool with complex PDE systems is to ensure that the model converges. If the algorithm, mesh, and tolerance are not causing artifacts, then changing each of these should not affect the outcome. If you compare two methods and get different outcomes, it does not mean that both are wrong. Rather, one or both may be wrong. Using more accurate meshes and tolerances verify your numerical methods to make sure your solution is not affected by grid sizes or time steps that are too course.

   b. Analytic Comparison and/or Intuition. Check that you constructed the model correctly by identifying some expected behaviors. If you are planning to use a complex geometry or too many different model components to make this possible, you should first simplify your model to test individual parts, such as using a simple geometry, or by eliminating some of the processes or variables (leave out the parts altogether or set some parameters to zero). For simple enough subsystems, you should be able to calculate the steady state solution and the characteristic times, so check if it reaches that equilibrium on that time scale. (Remember that you can

calculate the various characteristic times for your simulation by calculating combinations of variables that have units of time, and interpreting what each means.

4. Now solve your problem and do whatever analysis you need.

Nondimensionalization: Gravitational settling

Here we pose a general question as to when we need to include convection due to gravitational settling in a model. You can quickly determine whether diffusion will prevent settling in a chamber of height H, by solving the 1D diffusion-convection equation. The convection due to gravity is $-V_g$, since it moves downward. Since the velocity doesn't change with y, this equation is simply:

$$\frac{\partial u}{\partial t} = D\nabla^2 u - \nabla \cdot u\vec{V} = D\frac{\partial^2 u}{\partial y^2} + V_g \frac{\partial u}{\partial y}$$

IC are $u(y,0) = C_0$

BC are $\frac{\partial u}{\partial y}(0,t) = 0$ and $\frac{\partial u}{\partial y}(h,t) = 0$. That is, no flux boundary conditions.

We can calculate two time constants from the dimensions of the parameters: $C_t 1 = H^2/D$ and $C_t 2 = H/V_g$. The first is the time it takes the particle to diffuse the height of the chamber, and the second the time it takes to settle the height of the chamber. We might realize already that diffusion will dominate (so the particles won't settle) if diffusion occurs in less time than settling. That is, if $\frac{H^2}{D} \ll \frac{H}{V_g}$, or more concisely, if $\frac{HV_g}{D} \ll 1$, while we will get settling if $\frac{HV_g}{D} > 1$. If we don't realize this, we can calculate the steady state solution, given initial conditions $u(y) = C_0$ and no flux boundary conditions. For the steady state solution, $\frac{\partial^2 u}{\partial y^2} + V_g \frac{\partial u}{\partial y} = 0$, or $D\frac{\partial^2 u}{\partial y^2} = -V_g \frac{\partial u}{\partial y}$, or by integration, $\frac{\partial u}{\partial y} = -\frac{V_g}{D}u + C_1$. The no flux condition at u = 0 tells us that $C_1 = 0$. Solving the trivial resulting ODE provides the solution $u = u_0 \exp\left(-\frac{V_g}{D}t\right)$. We call D/V_g the 'atmospheric' height, because this situation is identical to the situation that causes oxygen concentration to decay exponentially from the earth's surface, which creates the atmosphere around the earth. The value of u_0 can be calculated by conservation of mass due to the no flux condition, but is not necessary for our analysis. Instead, we simply note that we don't need to worry about settling if the height of the chamber is much less than the characteristic decay distance of the solution, $H \ll D/V_g$, which is the same conclusion that we drew from comparing the two time constants.

For specific examples, assume $r = 2.5\ nm$ for a typical protein, and density $\rho = 1.3\ g/ml$, which is $\Delta\rho = 0.3\ g/ml$ greater than the water the protein will displace upon settling. Thus, $D = \frac{k_B T}{6\pi\mu r} \sim 1e-10\ m^2/s$, $F_g = mg = \frac{4}{3}\pi r^3 \Delta\rho g$, where $g = 9.8\ m/s^2$, or $F_g \sim 2e-22\ N$, and $V_g = \frac{F_g}{6\pi\mu r} = 4e-12\ m/s$. Finally, $\frac{D}{V_g} = 21\ m$

Thus, we have an atmosphere of $21\ m$. This is clearly something we don't worry about for a microfluidic chamber. On the other hand, D scales with r^{-1}, and V_g scales with r^2, so $\frac{D}{V_g}$ scales with r^{-3}. Thus, 2.5 nm, 25 nm, 250 nm, and 2.5 micron complexes with the same density as proteins will have an atmosphere 21 m, 21 mm, 21 microns, and 21 nm. Cells are typically are

less dense that proteins, for example with $\rho = 1.03$, so $\Delta\rho = 0.03$; since $\frac{D}{V_g}$ scales with the inverse of the relative density ($1/\Delta\rho$), a 2.5 micron cell would an atmosphere of 210 nm. Thus, we can neglect settling for large protein complexes, but not necessarily for viruses and definitely not for cells, even small ones like bacteria. Indeed, for sufficiently large particles, we may be able to neglect diffusion instead

Nondimensionalization: We follow the SPICE method. Recall that our original equations were:

PDE: $\frac{\partial u}{\partial t} = D\nabla^2 u - \nabla \cdot u\vec{V} = D\frac{\partial^2 u}{\partial y^2} + V_g\frac{\partial u}{\partial y}$

IC are $u(y, 0) = C_0$

BC are $\frac{\partial u}{\partial y}(0, t) = 0$ and $\frac{\partial u}{\partial y}(h, t) = 0$.

We thus have variables t, u, and y so we define 3 scaling parameters

$$T = \frac{t}{C_t}, U = \frac{u}{C_u}, Y = \frac{y}{C_y}$$

We insert these into the original equations:

PDE: $\frac{C_u}{C_t}\frac{\partial U}{\partial T} = \frac{DC_u}{C_y^2}\frac{\partial^2 U}{\partial Y^2} + \frac{V_g C_u}{C_y}\frac{\partial U}{\partial Y}$

IC are $C_u U(Y, 0) = C_0$

BC are $\frac{C_u}{C_y}\frac{\partial U}{\partial Y}(0, T) = 0$ and $\frac{C_u}{C_y}\frac{\partial U}{\partial Y}\left(\frac{h}{C_y}, T\right) = 0$.

And simplify:

PDE: $\frac{\partial U}{\partial T} = \frac{DC_t}{C_y^2}\frac{\partial^2 U}{\partial Y^2} + \frac{V_g C_t}{C_y}\frac{\partial U}{\partial Y}$

IC are $U(Y, 0) = C_0/C_u$

BC are $\frac{\partial U}{\partial Y}(0, T) = 0$ and $\frac{\partial U}{\partial Y}\left(\frac{h}{C_y}, T\right) = 0$.

We then set coefficients to one where we can. We have 4 coefficients, and three scaling parameters, so we know already that there will be one nondimensional parameter that determines system behavior.

$$\frac{DC_t}{C_y^2} = 1, \frac{V_g C_t}{C_y} = 1, \frac{C_0}{C_u} = 1, \frac{h}{C_y} = 1$$

I like to use the physics (PDES) rather than geometry when possible to scale my distances, so I will use the first two equations to identify C_t and C_y. That is, $C_t = C_y^2/D$ and $C_t = C_y/V_g$. Combining these gives $C_y = D/V_g$, which has units of length. With this method of nondimensionalization, we have thus scaled y to the atmosphere height. Then I use that to get $C_t = D/V_g^2$, which is easily interpreted from the two equations we used above ($C_t = C_y^2/D$ and $C_t = C_y/V_g$), which tell use that C_t is the time needed to settle or diffuse the characteristic length scale. Our only option for the last is $C_u = C_0$, so we scale concentration to the initial conditions

in this closed system. This leaves one nondimensional parameter to be defined: $\alpha = \frac{h}{C_y} = \frac{hV_g}{D}$, which is the nondimensional chamber height.

The obvious alternative would be to define $C_y = h$, which gives two choices for C_t: We can let $C_t = \frac{h^2}{D}$, which is the time to diffuse the height of the chamber. This leaves the last coefficient as $\beta = \frac{hV_g}{D}$, which is the nondimensional settling velocity. When $\beta > 1$, the particles settles rapidly enough that we need to consider settling. Alternatively, $C_t = \frac{h}{V_g}$, which is the time to settle the height of the chamber. This will result in the final coefficient being $\gamma = \frac{D}{hV_g}$, which is the nondimensional diffusion constant that determines whether diffusion through the chamber is significant.

Lab Example: Adhesion of particles in a microfluidic chamber.

There are many situations where a particle flows through a chamber to test for its adhesion. Examples range from surface Plasmon resonance (SPR) experiments that measure binding of a small molecule, to adhesion studies in which viruses, bacteria or eukaryotic cells are tested. This situation combines all the examples we gave above. However, we always seek to simplify the problem. For example, we would perform calculations such as the previous section to determine whether to include or neglect gravitational settling.

You can model the concentration of analyte in solution, $C(x, y, t)$, and the concentration of bound analyte in complex with a receptor on the surface: $B(x, t)$. Because there is no variation in the width of the channel, we ignore the third spatial dimension. The SPR chip has length L, height H. The inflow concentration being washed into the chamber is C_0 and the total concentration of receptor on the surface is R_0. The free receptor is at concentration $R_0 - B$, so does not need a variable. The diffusion coefficient in solution is D, and we assume that the bound analyte and receptors cannot diffuse since the receptor is immobilized on a solid surface, not a fluid membrane, so $D_B = 0$. The analyte binds to the surface with rate $(R_0 - B)Ck_a$, and detaches at rate Bk_d. Fluid is pumped into the chip at a prescribed volumetric flow rate, which creates a predictable parabolic flow profile, so that fluid moves at a velocity in the x-direction of $V_x(y) = S\left[y - \frac{y^2}{H}\right]$, where S is a constant (the wall shear rate) that depends on the overall flow rate. The 8 parameters are thus $L, H, R_0, C_0, D,\ k_a, k_d,\ S$. Note that C is the chemical species in solution (in $mole/m^3$), and R_0 and B are both in $mole/m^2$.

This system is modeled with coupled PDEs, one for the solution chemical (C, in 2D) and one for the bound chemical (B, in 1D). In solution we have:

$$\frac{\partial C}{\partial t} = D\nabla^2 C - \nabla \cdot C\vec{V}$$

With initial conditions $C(x, y, 0) = C_0$, or 0, depending on whether we start with the material in the chamber already or not, and boundary conditions:

- no flux on the upper boundary
- inflow on the left boundary ($C = C_0$)
- outflow on the right boundary

- adsorption on the lower boundary (as described below)

While on the lower surface we have a lower-order PDE:

$$\frac{\partial B}{\partial t} = k_a C(R_0 - B) - k_d B$$

with initial conditions $B(x, 0) = 0$, and no flux boundary conditions, since the bound form is immobile.

The convection velocity vector due to fluid velocity is $V_x(y) = S\left[y - \frac{y^2}{H}\right]$ and $V_y = 0$. This is usually expressed in experiments in terms of Q, the volumetric flow rate, but we don't do this in lab, since we don't know the width of the chamber in our 2D simulation. We ignore gravity due to our calculation above.

While we modeled the bound form with a lower dimension variable, we need to model the loss of solution species as a flux across the boundary that depends on both the density of available binding sites on the surface, and the concentration in solution near the boundary. That is, the *outward* flux at the boundary equals the concentration-dependent adsorption reaction:

$$\vec{n} \cdot \vec{J} = adsorption - desorption$$

In order to calculate the reaction rate, we need to keep track of the amount adsorbed to the surface to calculate both desorption, which is proportional to the adsorbed material, and adsorption, which is proportional to the open sites. In this model, the outward flux at the absorbing boundary is

$$\vec{n} \cdot \vec{J} = k_a C(R_0 - B) - k_d B$$

(Note however, that COMSOL asks you to enter *inward* flux normal to the boundary, or

$$-\vec{n} \cdot \vec{J} = -k_a C(R_0 - B) + k_d B$$

Summary of PDE systems:

1. PDE terms:
    a. Gradient is $\vec{\nabla}u = \left(\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}\right)$
    b. Divergence is $\nabla \cdot \vec{F} = \frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y}$
    c. Laplacian is $\nabla^2 u = \nabla \cdot \vec{\nabla} u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$
2. Boundary Conditions
    a. Dirichlet: value is prescribed u=C(t)
    b. Neumann: flux normal to boundary is prescribed: $\vec{n} \cdot \vec{J} = B(t)$, where : $\vec{n}$ is the unit vector pointing outward from the system, normal to that boundary.
    c. Generalized/Robin: flux normal to boundary is function of value: $\vec{n} \cdot \vec{J} = Au + B$.
3. Numeric methods
    a. May need to optimize mesh, solver, or tolerance to improve accuracy and efficiency.
    b. If available, as your default, use settings that allow the software to choose spatial and temporal grids that are appropriate for the equations (physics) being solved.
4. Verification of numeric solutions.
    a. Make sure your solution converges, meaning that you get essentially the same outcome (any differences do not affect your conclusions) when you change the solver, mesh, or tolerance.
    b. For complicated systems, make sure you are using software or writing equations correctly by separately verifying each component in simple conditions that allow analytic solutions.
    c. To make determine simulation times and verify the solution further, calculate time constants by combining parameters to create expressions that have unit time. You can do this conveniently in the parameter window of COMSOL, even if you never use the time constants you define.
5. In transport of chemical species
    a. Concentration is in mol/m^3 or $\frac{mol}{L} = M$.
    b. Diffusion coefficient, D, is in m^2/s
    c. Permeability P of a boundary is in m/s
    d. Flux J is in $\frac{mol}{m^2 s}$
    e. J is the flux, $J_{conv} = u\vec{V}$, $\overrightarrow{J_{diff}} = -D\vec{\nabla}u$, so $\vec{J} = -D\vec{\nabla}u + u\vec{V}$
    f. The transport equation is: $\frac{\partial u}{\partial t} = D\nabla^2 u - \nabla \cdot u\vec{V} + rxn$
6. Nondimensionalization. To reduce the number of variables to make analysis simpler, we convert it to nondimensional form using the SPICE procedure. You may then solve the nondimensional model, or simply define the nondimensional parameters in the model (e.g. the parameter window of COMSOL) to guide your systems analysis.