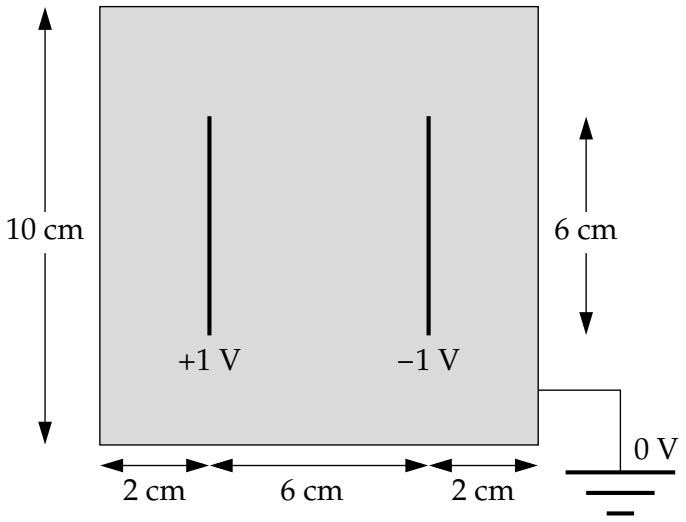
COMPUTATIONAL PHYSICS – PH 354

HOMEWORK DUE ON 1ST APRIL 2020

Exercise 1: Consider the following simple model of an electronic capacitor, consisting of two flat metal plates enclosed in a square metal box:



For simplicity let us model the system in two dimensions. Using any of the methods we have studied, write a program to calculate the electrostatic potential in the box on a grid of 100×100 points, where the walls of the box are at voltage zero and the two plates (which are of negligible thickness) are at voltages ± 1 V as shown. Have your program calculate the value of the potential at each grid point to a precision of 10^{-6} volts and then make a density plot of the result.

Hint: The capacitor plates are part of the boundary condition in this case: they behave the same way as the walls of the box, with potentials that are fixed at a certain value and cannot change.

Exercise 2: Thermal diffusion in the Earth's crust

A classic example of a diffusion problem with a time-varying boundary condition is the diffusion of heat into the crust of the Earth, as surface temperature

varies with the seasons. Suppose the mean daily temperature at a particular point on the surface varies as:

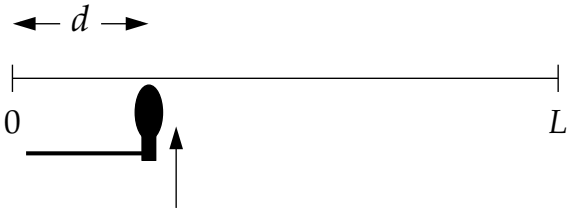
$$T_0(t) = A + B \sin \frac{2\pi t}{\tau},$$

where $\tau = 365$ days, $A = 10°$C and $B = 12°$C. At a depth of 20 m below the surface almost all annual temperature variation is ironed out and the temperature is, to a good approximation, a constant 11°C (which is higher than the mean surface temperature of 10°C—temperature increases with depth, due to heating from the hot core of the planet). The thermal diffusivity of the Earth's crust varies somewhat from place to place, but for our purposes we will treat it as constant with value $D = 0.1\,\mathrm{m}^2\,\mathrm{day}^{-1}$.

Write a program to calculate the temperature profile of the crust as a function of depth up to 20 m and time up to 10 years. Start with temperature everywhere equal to 10°C, except at the surface and the deepest point, choose values for the number of grid points and the time-step h, then run your program for the first nine simulated years, to allow it to settle down into whatever pattern it reaches. Then for the tenth and final year plot four temperature profiles taken at 3-month intervals on a single graph to illustrate how the temperature changes as a function of depth and time.

Exercise 3: FTCS solution of the wave equation

Consider a piano string of length L, initially at rest. At time $t = 0$ the string is struck by the piano hammer a distance d from the end of the string:



The string vibrates as a result of being struck, except at the ends, $x = 0$ and $x = L$, where it is held fixed. The equations for transverse displacement and velocity are

$$\frac{\partial \xi}{\partial t} = u, \quad \frac{\partial u}{\partial t} = v^2 \frac{\partial^2 \xi}{\partial x^2}.$$

a) Write a program that uses the FTCS method to solve the complete set of simultaneous first-order equations, Eq. (9.28), for the case $v = 100\,\mathrm{ms}^{-1}$, with the initial condition that $\xi(x) = 0$ (transverse displacement) everywhere but the transverse velocity $u(x)$ is nonzero, with profile

$$u(x) = C\frac{x(L-x)}{L^2}\exp\left[-\frac{(x-d)^2}{2\sigma^2}\right],$$

where $L = 1\,\mathrm{m}$, $d = 10\,\mathrm{cm}$, $C = 1\,\mathrm{ms}^{-1}$, and $\sigma = 0.3\,\mathrm{m}$. You will also need to choose a value for the time-step h. A reasonable choice is $h = 10^{-6}\,\mathrm{s}$.

b) Plot the transverse displacement as a function of x for different times, until numerical instabilities start to appear (after all this is unconditionally unstable!).

Exercise 4: The relaxation method for ordinary differential equations

There is no reason why the relaxation method must be restricted to the solution of differential equations with two or more independent variables. It can also be applied to those with one independent variable, i.e., to ordinary differential equations.

A ball of mass $m = 1\,\mathrm{kg}$ is thrown from height $x = 0$ into the air and lands back at $x = 0$ ten seconds later. The problem is to calculate the trajectory of the ball, but using the relaxation method. Ignoring friction effects, the trajectory is the solution of the ordinary differential equation

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = -g,$$

where g is the acceleration due to gravity.

a) Replacing the second derivative in this equation with its finite-difference approximation, Eq. (5.109), derive a relaxation-method equation for solving this problem on a time-like "grid" of points with separation h.

b) Taking the boundary conditions to be that $x = 0$ at $t = 0$ and $t = 10$, write a program to solve for the height of the ball as a function of time using the relaxation method with 100 points and make a plot of the result from $t = 0$ to $t = 10$. Run the relaxation method until the answers change by 10^{-6} or less at every point on each step.

Exercise 5: The Schrödinger equation and the Crank–Nicolson method

One of the most important PDEs, at least for physicists, is the Schrödinger equation. This exercise uses the Crank–Nicolson method to solve the full time-dependent Schrödinger equation and hence develop a picture of how a wavefunction evolves over time. The following exercise, Exercise 6, solves the same problem again, but using the spectral method.

We will look at the Schrödinger equation in one dimension. The techniques for calculating solutions in two or three dimensions are basically the same as for one dimension, but the calculations take much longer on the computer, so in the interests of speed we'll stick with one dimension. In one dimension the Schrödinger equation for a particle of mass M with no potential energy reads

$$-\frac{\hbar^2}{2M}\frac{\partial^2\psi}{\partial x^2} = \mathrm{i}\hbar\frac{\partial\psi}{\partial t}.$$

For simplicity, let's put our particle in a box with impenetrable walls, so that we only have to solve the equation in a finite-sized space. The box forces the wavefunction ψ to be zero at the walls, which we'll put at $x = 0$ and $x = L$.

Replacing the second derivative in the Schrödinger equation with a finite difference and applying Euler's method, we get the FTCS equation

$$\psi(x,t+h) = \psi(x,t) + h\frac{\mathrm{i}\hbar}{2ma^2}\bigl[\psi(x+a,t) + \psi(x-a,t) - 2\psi(x,t)\bigr],$$

where a is the spacing of the spatial grid points and h is the size of the time-step. (Be careful not to confuse the time-step h with Planck's constant $\hbar$.) Performing a similar step in reverse, we get the implicit equation

$$\psi(x,t+h) - h\frac{\mathrm{i}\hbar}{2ma^2}\bigl[\psi(x+a,t+h) + \psi(x-a,t+h) - 2\psi(x,t+h)\bigr] = \psi(x,t).$$

And taking the average of these two, we get the Crank–Nicolson equation for the Schrödinger equation:

$$\begin{aligned}\psi(x,t+h) - h\frac{\mathrm{i}\hbar}{4ma^2}&\bigl[\psi(x+a,t+h) + \psi(x-a,t+h) - 2\psi(x,t+h)\bigr]\\ &= \psi(x,t) + h\frac{\mathrm{i}\hbar}{4ma^2}\bigl[\psi(x+a,t) + \psi(x-a,t) - 2\psi(x,t)\bigr].\end{aligned}$$

This gives us a set of simultaneous equations, one for each grid point.

The boundary conditions on our problem tell us that $\psi = 0$ at $x = 0$ and $x = L$ for all t. In between these points we have grid points at a, $2a$, $3a$, and so forth. Let us arrange the values of ψ at these interior points into a vector

$$\boldsymbol{\psi}(t) = \begin{pmatrix} \psi(a,t) \\ \psi(2a,t) \\ \psi(3a,t) \\ \vdots \end{pmatrix}.$$

Then the Crank–Nicolson equations can be written in the form

$$\mathbf{A}\boldsymbol{\psi}(t+h) = \mathbf{B}\boldsymbol{\psi}(t),$$

where the matrices $\mathbf{A}$ and $\mathbf{B}$ are both symmetric and tridiagonal:

$$\mathbf{A} = \begin{pmatrix} a_1 & a_2 & & & \\ a_2 & a_1 & a_2 & & \\ & a_2 & a_1 & a_2 & \\ & & a_2 & a_1 & \\ & & & & \ddots \end{pmatrix}, \qquad \mathbf{B} = \begin{pmatrix} b_1 & b_2 & & & \\ b_2 & b_1 & b_2 & & \\ & b_2 & b_1 & b_2 & \\ & & b_2 & b_1 & \\ & & & & \ddots \end{pmatrix},$$

with

$$a_1 = 1 + h\frac{\mathrm{i}\hbar}{2ma^2}, \qquad a_2 = -h\frac{\mathrm{i}\hbar}{4ma^2}, \qquad b_1 = 1 - h\frac{\mathrm{i}\hbar}{2ma^2}, \qquad b_2 = h\frac{\mathrm{i}\hbar}{4ma^2}.$$

(Note the different signs and the factors of 2 and 4 in the denominators.)

The equation $\mathbf{A}\boldsymbol{\psi}(t+h) = \mathbf{B}\boldsymbol{\psi}(t)$ has precisely the form $\mathbf{A}\mathbf{x} = \mathbf{v}$ of simultaneous equations, where $\mathbf{A}$ is a tridiagonal matrix. Use the fast tridiagonal version of Gaussian elimination to solve for $\psi(x, t+h)$ for a given $\psi(x,t)$.

Consider an electron (mass $M = 9.109 \times 10^{-31}\,\mathrm{kg}$) in a box of length $L = 10^{-8}\,\mathrm{m}$. Suppose that at time $t = 0$ the wavefunction of the electron has the form

$$\psi(x,0) = \exp\left[-\frac{(x-x_0)^2}{2\sigma^2}\right]\mathrm{e}^{\mathrm{i}\kappa x},$$

where

$$x_0 = \frac{L}{2}, \qquad \sigma = 1 \times 10^{-10}\,\mathrm{m}, \qquad \kappa = 5 \times 10^{10}\,\mathrm{m}^{-1},$$

and $\psi = 0$ on the walls at $x = 0$ and $x = L$. (This expression for $\psi(x,0)$ is not normalized—there should really be an overall multiplying coefficient to make sure that the probability density for the electron integrates to unity. It's safe

to drop the constant, however, because the Schrödinger equation is linear, so the constant cancels out on both sides of the equation and plays no part in the solution.)

a) Write a program to perform a single step of the Crank–Nicolson method for this electron, calculating the vector $\boldsymbol{\psi}(t)$ of values of the wavefunction, given the initial wavefunction above and using $N = 1000$ spatial slices with $a = L/N$. Your program will have to perform the following steps. First, given the vector $\boldsymbol{\psi}(0)$ at $t = 0$, you will have to multiply by the matrix $\mathbf{B}$ to get a vector $\mathbf{v} = \mathbf{B}\boldsymbol{\psi}$. Because of the tridiagonal form of $\mathbf{B}$, this is fairly simple. The ith component of $\mathbf{v}$ is given by

$$v_i = b_1\psi_i + b_2(\psi_{i+1} + \psi_{i-1}).$$

You will also have to choose a value for the time-step h. A reasonable choice is $h = 10^{-18}$ s.

Second you will have to solve the linear system $\mathbf{A}\mathbf{x} = \mathbf{v}$ for $\mathbf{x}$, which gives you the new value of $\boldsymbol{\psi}$. You could do this using a standard linear equation solver like the function `solve` in `numpy.linalg`, but since the matrix $\mathbf{A}$ is tridiagonal a better approach would be to use the fast solver for banded matrices.

Third, once you have the code in place to perform a single step of the calculation, extend your program to perform repeated steps and hence solve for ψ at a sequence of times a separation h apart. Note that the matrix $\mathbf{A}$ is independent of time, so it doesn't change from one step to another. You can set up the matrix just once and then keep on reusing it for every step.

b) Extend your program to make an animation of the solution by displaying the real part of the wavefunction at each time-step.

c) Run your animation for a while and describe what you see. Write a few sentences explaining in physics terms what is going on in the system.

Exercise 6: The Schrödinger equation and the spectral method

This exercise uses the spectral method to solve the time-dependent Schödinger equation

$$-\frac{\hbar^2}{2M}\frac{\partial^2\psi}{\partial x^2} = \mathrm{i}\hbar\frac{\partial\psi}{\partial t}$$

for the same system as in Exercise 5, a single particle in one dimension in a box of length L with impenetrable walls. The wavefunction in such a box necessarily goes to zero on the walls and hence one possible (unnormalized) solution of the equation is

$$\psi_k(x,t) = \sin\left(\frac{\pi k x}{L}\right) \mathrm{e}^{\mathrm{i}Et/\hbar},$$

where the energy E can be found by substituting into the Schrödinger equation, giving

$$E = \frac{\pi^2\hbar^2 k^2}{2ML^2}.$$

As with the vibrating string of Section 9.3.4, we can write a full solution as a linear combination of such individual solutions, which on the grid points $x_n = nL/N$ takes the value

$$\psi(x_n,t) = \frac{1}{N}\sum_{k=1}^{N-1} b_k \sin\left(\frac{\pi k n}{N}\right) \exp\left(\mathrm{i}\frac{\pi^2\hbar k^2}{2ML^2}t\right),$$

where the b_k are some set of (possibly complex) coefficients that specify the exact shape of the wavefunction and the leading factor of $1/N$ is optional but convenient.

Since the Schrödinger equation (unlike the wave equation) is first order in time, we need only a single initial condition on the value of $\psi(x,t)$ to specify the coefficients b_k, although, since the coefficients are in general complex, we will need to calculate both real and imaginary parts of each coefficient.

Again, we consider an electron (mass $M = 9.109 \times 10^{-31}\,\mathrm{kg}$) in a box of length $L = 10^{-8}\,\mathrm{m}$. At time $t = 0$ the wavefunction of the electron has the form

$$\psi(x,0) = \exp\left[-\frac{(x-x_0)^2}{2\sigma^2}\right] \mathrm{e}^{\mathrm{i}\kappa x},$$

where

$$x_0 = \frac{L}{2}, \qquad \sigma = 1 \times 10^{-10}\,\mathrm{m}, \qquad \kappa = 5 \times 10^{10}\,\mathrm{m}^{-1},$$

and $\psi = 0$ on the walls at $x = 0$ and $x = L$.

a) Write a program to calculate the values of the coefficients b_k, which for convenience can be broken down into their real and imaginary parts as $b_k = \alpha_k + \mathrm{i}\eta_k$. Divide the box into $N = 1000$ slices and create two arrays

containing the real and imaginary parts of $\psi(x_n, 0)$ at each grid point. Perform discrete sine transforms on each array separately and hence calculate the values of the α_k and η_k for all $k = 1 \ldots N-1$.

To perform the discrete sine transforms, you can use the fast transform function `dst` from the package `dcst`, which you can find in the on-line resources in the file named `dcst.py`. The function takes an array of N real numbers and returns the discrete sine transform as another array of N numbers.

(Note that the first element of the input array should in principle always be zero for a sine transform, but if it is not the `dst` function will simply pretend that it is. Similarly the first element of the returned array is always zero, since the $k = 0$ coefficient of a sine transform is always zero. So in effect, the sine transform really only takes $N-1$ real numbers and transforms them into another $N-1$ real numbers. In some implementations of the discrete sine transform, therefore, though not the one in the package `dsct` used here, the first element of each array is simply omitted, since it's always zero anyway, and the arrays are only $N-1$ elements long.)

b) Putting $b_k = \alpha_k + \mathrm{i}\eta_k$ in the solution above and taking the real part we get

$$\operatorname{Re}\psi(x_n, t) = \frac{1}{N}\sum_{k=1}^{N-1}\left[\alpha_k \cos\left(\frac{\pi^2\hbar k^2}{2ML^2}t\right) - \eta_k \sin\left(\frac{\pi^2\hbar k^2}{2ML^2}t\right)\right]\sin\left(\frac{\pi k n}{N}\right)$$

for the real part of the wavefunction. This is an inverse sine transform with coefficients equal to the quantities in the square brackets. Extend your program to calculate the real part of the wavefunction $\psi(x, t)$ at an arbitrary time t using this formula and the inverse discrete sine transform function `idst`, also from the package `dcst`. Test your program by making a graph of the wavefunction at time $t = 10^{-16}$ s.

c) Extend your program further to make an animation of the wavefunction over time, similar to that described in part (b) of Exercise 5 above. A suitable time interval for each frame of the animation is about 10^{-18} s.

d) Run your animation for a while and describe what you see. Write a few sentences explaining in physics terms what is going on in the system.

Exercise 7: Almost tridiagonal system

Write a program to solve a tridiagonal system of equations $Ax = d$, where A is a tridiagonal matrix,

$$A = \begin{bmatrix} b_1 & c_1 & & & & & \\ a_2 & b_2 & c_2 & & & & \\ & a_3 & b_3 & c_3 & & & \\ & & & ... & & & \\ & & & a_{n-2} & b_{n-2} & c_{n-2} & \\ & & & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & & & a_n & b_n \end{bmatrix}.$$

Use Thomas algorithm with forward elimination and back-substitution as discussed in class.

Another situation where an almost-tridiagonal system results is when writing the implicit finite-difference formula (first order accurate in time and second ordered accurate in space) for the heat diffusion equation,

$$\frac{\partial f}{\partial t} = D\frac{\partial^2 f}{\partial x^2}, \tag{1}$$

given by

$$\frac{f_i^{n+1} - f_i^n}{\Delta t} = D\frac{f_{i+1}^{n+1} - 2f_i^{n+1} + f_{i-1}^{n+1}}{\Delta x^2}, \text{or} \tag{2}$$

$$Af^{n+1} = f^n, \tag{3}$$

where

$$A = \begin{bmatrix} b_1 & c_1 & & & ... & & a_1 \\ a_2 & b_2 & c_2 & & & & \\ & a_3 & b_3 & c_3 & & & \\ & & & ... & & & \\ & & & a_{n-2} & b_{n-2} & c_{n-2} & \\ & & & & a_{n-1} & b_{n-1} & c_{n-1} \\ c_n & & & ... & & a_n & b_n \end{bmatrix};$$

f^{n+1} and f^n are vectors at times t^{n+1} and t^n respectively; $b_i = 1 + 2D\Delta t/\Delta x^2$, $a_i = -D\Delta t/\Delta x^2$, and $c_i = -D\Delta t/\Delta x^2$. Notice that we have used periodic boundary conditions; this results in appearance of a_1 and c_n in corners apart from the tridiagonal structure. Thomas method can be modified to solve Eq. 3

using the **Sherman-Morrison** formula. Sherman-Morrison formula states that the solution of the matrix equation

$$(A + uv^T)x = d, \tag{4}$$

where u and v are column vectors (v^T is the transpose of v), is given by solving

$$Ay = d, Aq = u,$$

and computing $x = y - q(v^T y)/(1 + v^T q)$. To solve Eq. 2 you will need to choose column vectors u and v appropriately and apply the tridiagonal method twice. Hence the solution is $\mathcal{O}(n)$, where n is the number of grid points.

Write a code to solve the diffusion equation (Eq. 1) using periodic boundary conditions with the method discussed above and the tridiagonal code that you wrote. Use $D = 1$ and a domain going from 0 to 1. The initial condition is $f = 1$ for $0.4 < x < 0.6$; outside this f vanishes. Find the solution (f) at time $t = 0.025$; choose the timestep $\Delta t = 4\Delta x^2/D$. Use 256 grid points such that $\Delta x = 1/256$. See how the solution at $t = 1$ converges with increasing resolution (try $n = 128$, 256, 512, 1024). Plot L1 Richardson error as a function of Δx on log-log scale. What is is order of convergence, as deduced from the slope of this plot? L1 Richardson error is defined as

$$E_1 = \frac{1}{n}\sum_{i=1}^{n} |f_i^n - f_i^{2n}|,$$

where f_i^n is the numerical solution obtained at grid point i using n grid points and f_i^{2n} is the numerical solution obtained at the same point but using double the number of grid points.

Exercise 8: Different methods for hyperbolic equations

We have discussed several methods for solving hyperbolic equations. We will apply some of the methods that we discussed to two prototype hyperbolic equations: the advection equation and the Burger's equation.

The constant velocity advection equation is

$$\frac{\partial f}{\partial t} + v\frac{\partial f}{\partial x} = 0, \tag{5}$$

where v is the constant advection velocity. Assume $v = 1$, periodic boundary conditions, domain $0 \leq x \leq 1$, and assume that the initial condition is

$$f(x, t = 0) = e^{-200(x-0.3)^2} + \text{sqr}[0.6, 0.8], \tag{6}$$

where sqr$[a, b]$ is a square wave for $a \leq x \leq b$; i.e., it is zero everywhere except when x lies between a and b, where it is 1. Plot $f(x, t)$ at $t = 0,\ 1,\ 2,\ 3$. Compare the results with the following methods (use $\Delta t = 0.5\Delta x/v$; use 128 grid points): upwind, Lax, and Lax-Wendroff. Which method is the best and why? Try combining Lax-Wendroff with artificial viscosity (discussed later) and see if the oscillations go away. Perform convergence analysis for all these methods at $t = 1$ (use 32, 64, 128, 256, 512 grid points) and plot L1 error (you can either use the analytic results or use Richardson error; i.e., treating the higher resolution result as the true solution) as a function of resolution (Δx). What's the order of convergence?

The Burger's equation, a model for supersonic flows, is given by

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0, \tag{7}$$

where u is the fluid velocity. Its better to numerically evolve the conservative form,

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u^2}{2}\right) = 0. \tag{8}$$

Use periodic boundary conditions, $0 \leq x \leq 1$, and with the initial condition given by

$$u(x, t = 0) = \sin(2\pi x). \tag{9}$$

Show results at $t = 0,\ 0.1,\ 0.25,\ 0.5$ using Lax, upwind, Lax-Wendroff methods; use $\Delta t = 0.5\Delta x$ and resolution of 128 grid-points. You will see that the Lax-Wendroff method becomes unstable once a shock forms. Lax-Wendroff method can be stabilized by using artificial viscosity which is implemented as follows. Instead of solving Eq. 8, we solve

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u^2}{2}\right) = -\frac{\partial q}{\partial x}, \tag{10}$$

where q is the (artificial) viscous flux given by

$$\begin{aligned} q \ &= \ l^2\left(\frac{\partial u}{\partial x}\right)^2 \text{ when } \frac{\partial u}{\partial x} < 0, \\ &\quad\ 0, \text{ otherwise,} \end{aligned} \tag{11}$$

where l is a length scale over which a shock is smoothened. While both the transport and diffusion terms in Eq. 10 can be applied in a single step with both

terms treated explicitly (forward in time), remember that the diffusion term has a restrictive stability limit on Δt. Thus it is better to use operator splitting and applying the transport and viscous diffusion steps independently. Since the main timestep ($\Delta t = 0.5\Delta x$) can be much shorter than the viscous timestep ($\Delta t_{\rm vis} = 0.5\Delta x^2/(l^2 \max[\partial u/\partial x]) \approx 0.25\Delta x/(\text{qcon } \max[u])$, where qcon=$l/\Delta x$), we have to subcycle the artificial viscosity step; i.e., apply artificial viscosity for multiple (nsub) times with a smaller timestep ($\Delta t_{\rm sub}$) such that $\Delta t_{\rm sub} \leq \Delta t_{\rm vis}$ and nsub $\times\ \Delta t_{\rm sub} = 0.5\Delta x$ (the main timestep). What value of qcon is able to make the scheme stable? Plot the results with this method at different times. Which scheme seems to work best?

Exercise 9: Solving Poisson equation for gravitational potential

The gravitational potential ($\Phi(x, y)$) due to a mass density distribution $\rho(x, y)$ is the solution of the following Poisson equation,

$$\frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} = \rho \tag{12}$$

in appropriate units. The finite difference form for this equation in two dimensions is

$$\frac{\Phi_{i+1,j} - 2\Phi_{i,j} + \Phi_{i-1,j}}{\Delta x^2} + \frac{\Phi_{i,j+1} - 2\Phi_{i,j} + \Phi_{i,j-1}}{\Delta y^2} = \rho_{i,j}. \tag{13}$$

The matrix equation corresponding to this equation is not easy to solve directly (its easy in 1D because we have a tridiagonal matrix), so its solved iteratively. Recall Jacobi relaxation which solves the following equation iteratively until the desired accuracy is obtained

$$\Phi^{n+1}_{i,j} = \frac{\left(\Phi^n_{i+1,j} + \Phi^n_{i-1,j} + \Phi^n_{i,j+1} + \Phi^n_{i,j-1}\right)}{4} - \frac{\Delta^2 \rho_{i,j}}{4}, \tag{14}$$

where $\Delta = \Delta x = \Delta y$. Solve the above equation using periodic boundary conditions; the density distribution is given by

$$\begin{aligned} \rho(x, y) \ &= \ e^{-y/0.1} \text{ if } -0.3 \leq x \leq 0.3, \\ &\quad\ 0 \text{ otherwise.} \end{aligned} \tag{15}$$

Iterate Eq. 14 and make contourplots of the potential (Φ) at $10^{\rm th}$, $100^{\rm th}$ and $1000^{\rm th}$ iteration on a 40×40 grid. Plot L1 error (using consecutive iterations) as a function of number of iterations. From these contourplots can you see

from that the large k (small scale) modes converge quickly but the large scale modes require many iterations to converge. This can be seen by realizing that Jacobi method is equivalent to solving the diffusion equation explicitly with a stable timestep; thus diffusion at large scales takes many timesteps to capture as compared to small scales.

Try to solve the same problem with Gauss-Seidel relaxation (its only very slightly different from Jacobi in that the Φ values are updated right away); it corresponds to the matrix decomposition (see class-slides and NR for details)

$$(\mathbf{L} + \mathbf{D}) \cdot \mathbf{x}^{(r)} = -\mathbf{U} \cdot \mathbf{x}^{(r-1)} + \mathbf{b}, \tag{16}$$

where matrix $\mathbf{A}$ is written as the sum of upper/lower triangular and diagonal matrices. Successive over-relaxation (SOR) method is equivalent to

$$\mathbf{x}^{(r+1)} = \mathbf{x}^{(r)} + \omega(\mathbf{L} + \mathbf{D})^{-1} \cdot \mathbf{r}^{(r-1)}, \tag{17}$$

where $\mathbf{r} \equiv \mathbf{b} - \mathbf{A} \cdot \mathbf{x}$ is the residual, and ω is the over-relaxation parameter; $(\mathbf{L} + \mathbf{D})^{-1} \cdot \mathbf{r}^{(r-1)}$ can be easily evaluated via forward-substitution because $(\mathbf{L} + \mathbf{D})$ is a very simple lower-triangular matrix. The optimum choice of ω for Poisson equation in Cartesian coordinates is $\omega \approx 2/(1 + \pi/J)$ where J is the number of grid points in each direction. Apply SOR for the above problem and plot L1 error as a function of number of iterations. Does SOR converge faster than Jacobi, as advertized? Try $\omega = 0.5, 1.5$ and see how these compare to the optimum case.