PARTIAL DIFFERENTIAL EQUATIONS

Ari Maller

PDES

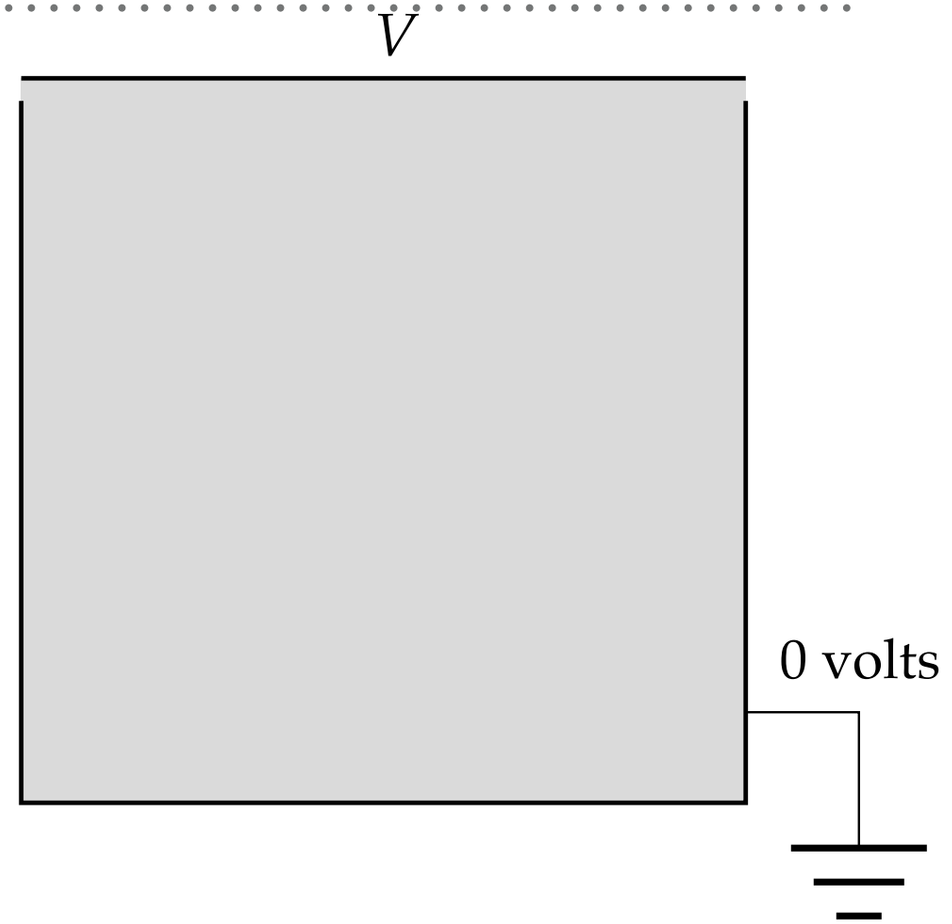
- Many important equations in physics are partial differential equations. These include the wave equation, the diffusion equation, the Laplace and Poisson equations, Maxwell's equations and the Schrodinger equation. That's a lot of physics.
- While the basic numerical techniques for solving PDEs is straight forward, solution are often computationally challenging. So a lot of effort has gone into solving PDEs efficiently.
- Like ODEs, PDEs can be boundary value or initial value problems. Unlike ODEs, boundary value PDEs are often easier to solve than PDE initial value problems.

BOUNDARY VALUE PDES

- Let's consider Laplace's equation as an example of a PDE.
- The electric potential φ is related to the electric field by E = -∇φ. In the absence of any charges ∇·∇φ=0, we get Laplace's equation:

$$\nabla^2 \phi = 0$$

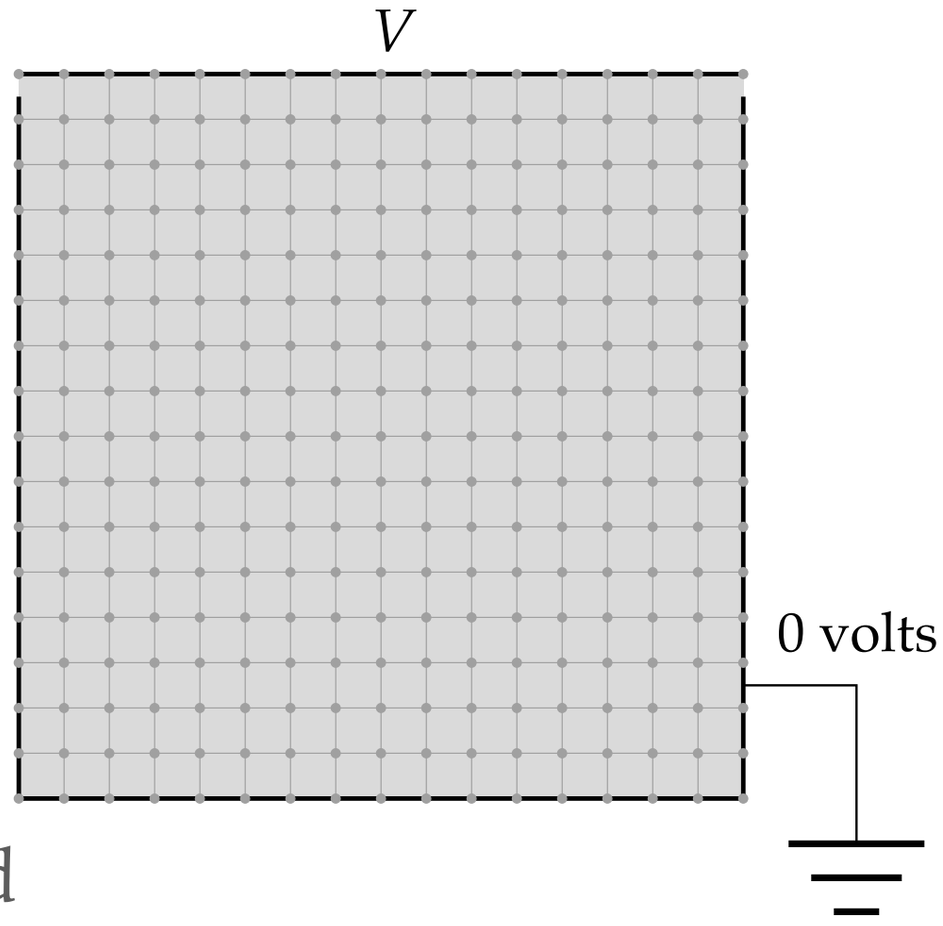
- Let's imagine the situation pictured to the right. An empty box with a voltage V on the top side and 0 on the other sides.



FINITE DIFFERENCE

- For simplicity let's treat our box as 2D instead of 3D. Techniques are basically the same in 2 or 3 dimensions and it is often a good idea to start with 2D before moving on to 3D. In 2D Laplace's equation becomes:

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0$$



- For the method of finite differences we need to divide the space into a grid of discrete points. Many kinds of grids are used, but the simplest choice and appropriate for our problem is a square grid.
- We put points at the boundary where we know φ and in the interior where we want to calculate it.

FINITE DIFFERENCE

- Let the space between point be a. Remember when we did numerical derivatives we could use a finite difference formula. Applying that here we get

$$\frac{\partial^2 \phi}{\partial x^2} = \frac{\phi(x+a, y) + \phi(x-a, y) - 2\phi(x, y)}{a^2}$$

- This formula gives us an expression of the 2nd derivative in terms of the values of 3 points on our grid. Similarly we have

$$\frac{\partial^2 \phi}{\partial y^2} = \frac{\phi(x, y+a) + \phi(x, y-a) - 2\phi(x, y)}{a^2}$$

- Thus we can write the Laplacian operator as

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial x^2} = \frac{\phi(x+a, y) + \phi(x-a, y) + \phi(x, y+a) + \phi(x, y-a) - 4\phi(x, y)}{a^2}$$

FINITE DIFFERENCE

- In other words we add the values of φ from the points adjacent to x,y and then we subtract 4 times the value at x,y and divide by a^2.
- So Laplace's equation at each point x,y becomes

$$\phi(x+a,y)+\phi(x-a,y)+\phi(x,y+a)+\phi(x,y-a)-4\phi(x,y)=0$$

- This is now a system of linear equations. It can be solved with methods we have discussed like Gaussian elimination or LU decomposition.
- It can also be solved with relaxation, which we introduced for nonlinear equations, but happens to be a good choice here.

JACOBI METHOD

- To use the relaxation method we rearrange the equation as

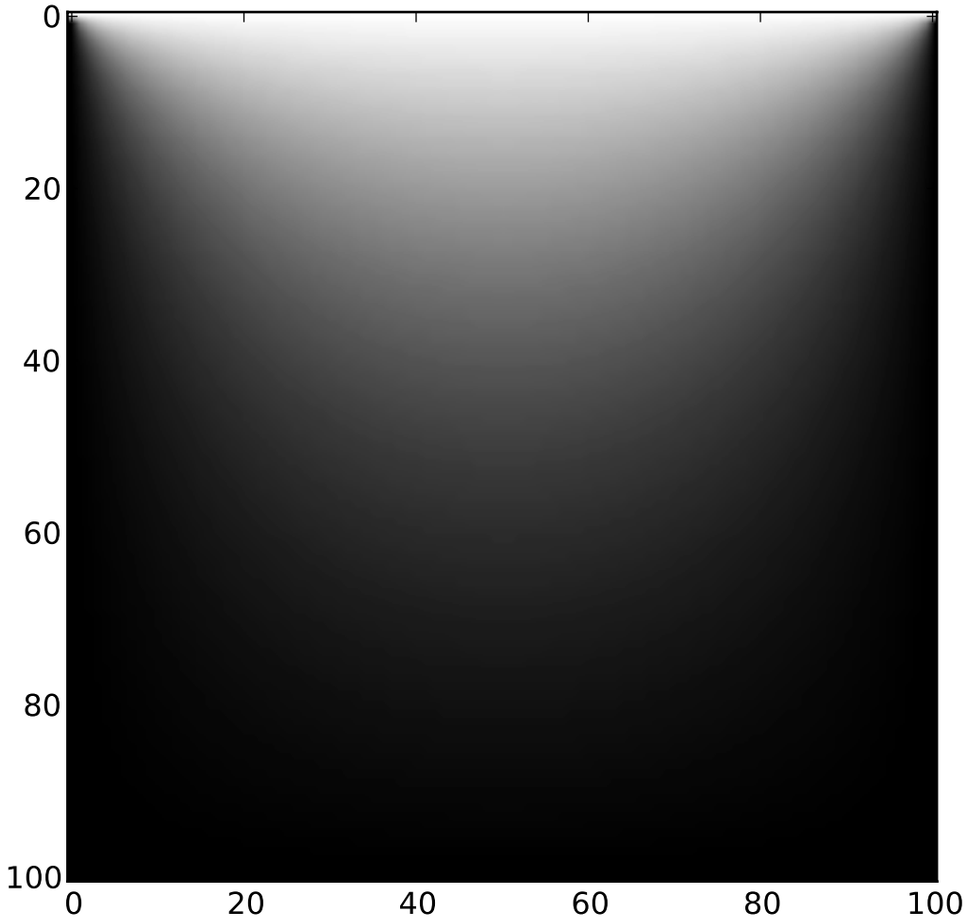
$$\phi(x,y) = \frac{1}{4}[\phi(x+a,y) + \phi(x-a,y) + \phi(x,y+a) + \phi(x,y-a)]$$

- That is the value of φ is just the average of the 4 adjacent points. We start with our boundary condition and some initial guess for the internal points, zero is fine.
- Then we use the above equation to generate new values for all the interior points. Then we repeat and repeat until the values of φ settle down to fixed values.
- This approach for solving Laplace's equation is called the Jacobi Method. Like any relaxation method we have to worry that the iteration process actually converges. Situation where it doesn't are called *numerically unstable*. Luckily in this case in can be proven that the Jacobi method is stable and always gives a solution.

```
from numpy import empty,zeros,max
M = 100         # Grid squares on a side
V = 1.0         # Voltage at top wall
target = 1e-6   # Target accuracy
# Create arrays to hold potential values
phi = zeros([M+1,M+1],float)
phi[0,:] = V
phiprime = empty([M+1,M+1],float)
# Main loop
delta = 1.0
while delta>target:
  for i in range(M+1):
    for j in range(M+1):
      if i==0 or i==M or j==0 or j==M:
        phiprime[i,j] = phi[i,j]
      else:
        phiprime[i,j] = (phi[i+1,j] + phi[i-1,j] + phi[i,j+1] + phi[i,j-1])/4

# Calculate maximum difference from old values
delta = max(abs(phi-phiprime))
# Swap the two arrays around
phi,phiprime = phiprime,phi
```

- The solution one gets from running the program is shown in the figure.
- It is important to remember that this is only a approximate solution.
- Even if we require very high accuracy from our relaxation, the fact that the finite difference approximation for a second derivative is not particularly accurate will limit the accuracy of our calculation.

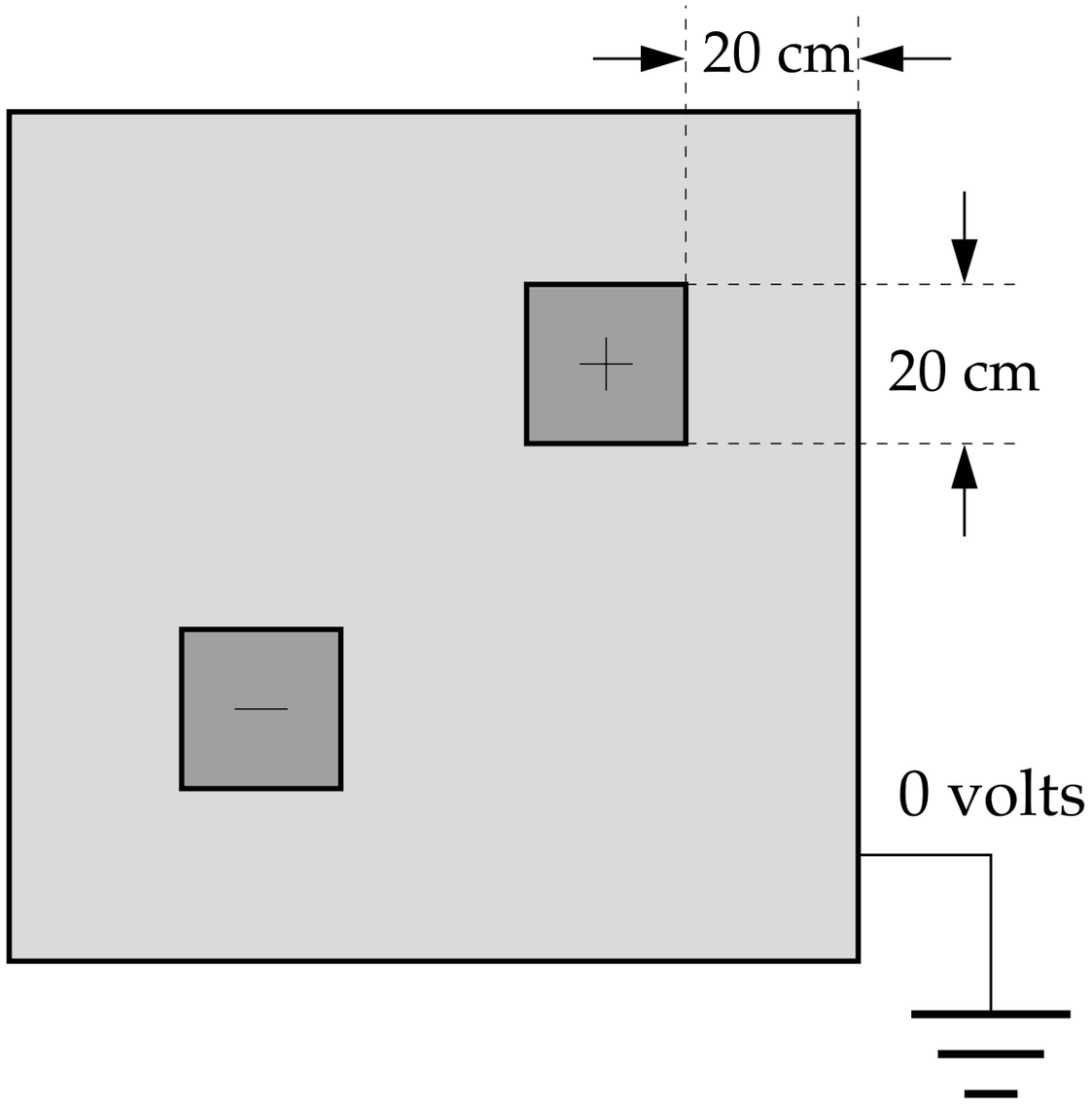


- One can use a higher order approximation for the derivative or increase the number of grid points.
- The greater the number of grid points, the longer the calculation will take.

SHORTCOMING OF JACOBI METHOD

- Also, we have only determined values of the potential on the grid points. If we want to know the value at some point off of the grid points we will have to perform some time of interpolation like we have discussed earlier.
- A more technical point, is that boundaries of the space may not always be simple and square like in the exercise. The could be diagonal or curved or have holes, all of which would make getting the grid points to fall on the boundary more difficult.
- One can always approximate a shape by its closest values on a square grid, but this introduces more error into the calculation.
- There are more complicated grids that can be used that may more closely fall on the boundaries, but this introduces more complexity into the equations.

EXAMPLE 9.2 THE POISSON EQUATION



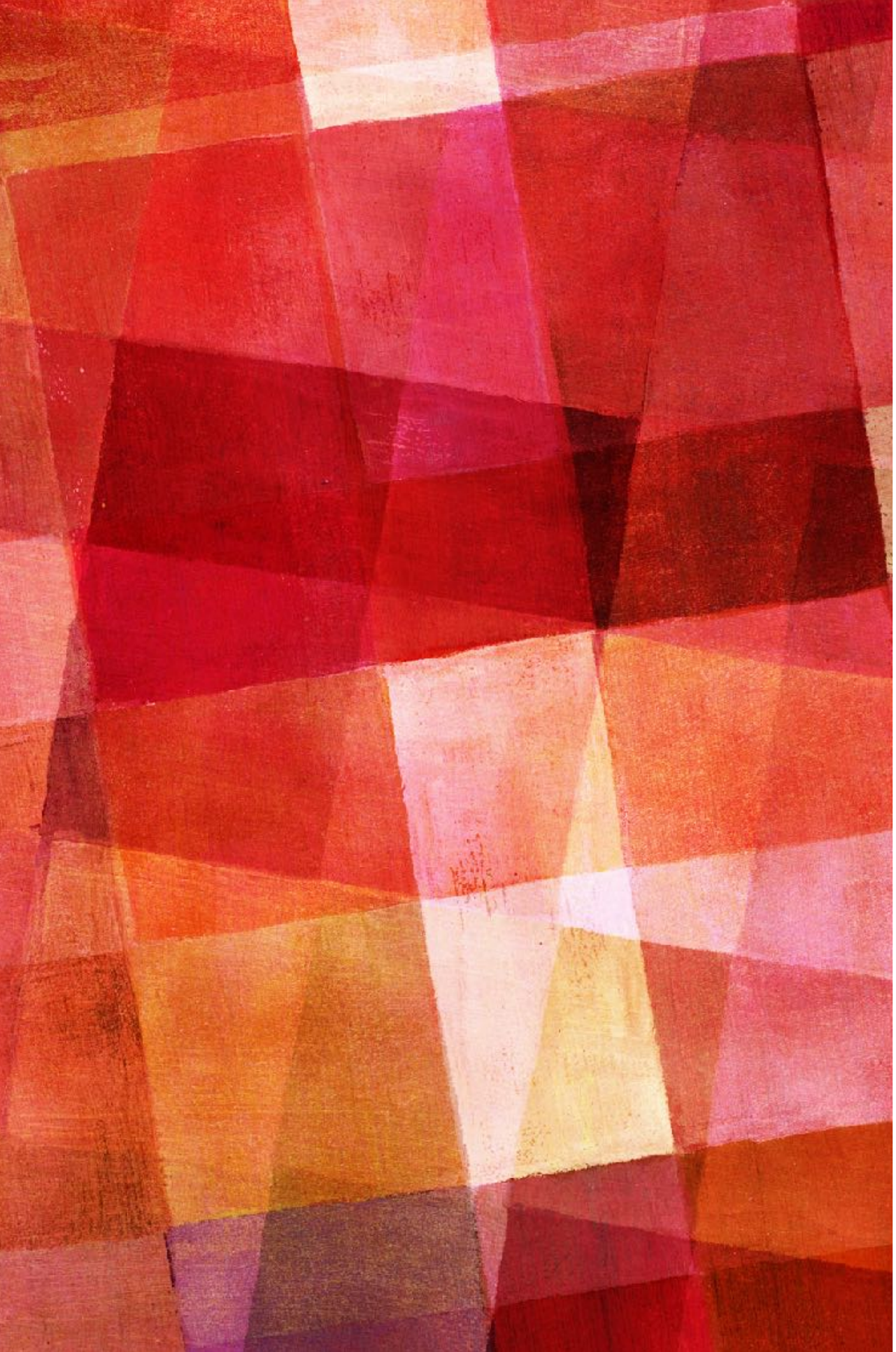
- A more complex example is to include charges in our area. Now we the Poisson equation:

$$\nabla^2\phi = -\frac{\rho}{\epsilon_0}$$

- In this case let the charges be squares, 20cm on a side with charge density $\pm 1 \mathrm{Cm}^{-2}$ and 0 voltage on the boundary.

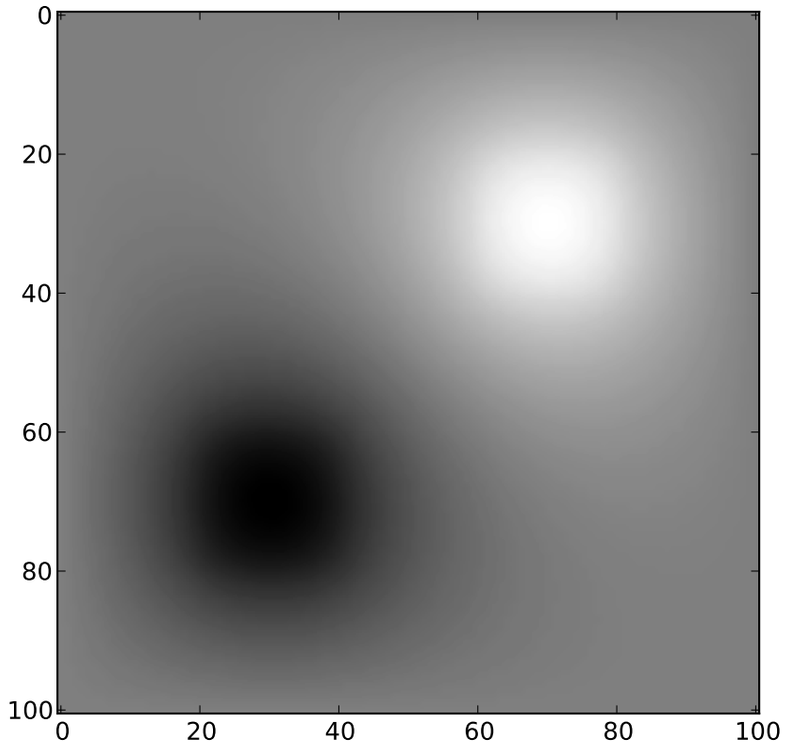
- Our finite difference formula now becomes:

$$\phi(x,y) = \frac{1}{4}[\phi(x+a,y) + \phi(x-a,y) + \phi(x,y+a) + \phi(x,y-a)] + \frac{a^2}{4\epsilon_0}\rho(x,y)$$



EXERCISE 9.1

- Modify a program to solve Poisson's equation for the system described in Example 9.2.
- Work in unites of $\varepsilon_0 = 1$ and have a target accuracy of 10^{-6} at each grid point.



FASTER METHODS FOR BOUNDARY VALUE PROBLEMS

- The Jacobi method gives good answers, but it can be very slow. For a 100×100 grid we are essentially solving 10,000 simultaneous equations with 10,000 unknowns.
- If we want greater accuracy or have a larger system this can quickly grow to a number of grid points that become impractical on most computers.
- Thus methods that improve the speed compared to the Jacobi method are very valuable. There are two we will look at:
  - Overrelaxation
  - Gauss-Seidel Method

OVERRELAXATION

- One way to try and make a relaxation approach work better is to try and overshoot each step. If we had a point whose initial value was 0.1 and then was evaluated to be 0.3 and whose correct value was 0.5, our convergence would go quicker if we set it to 0.4 instead of 0.3 on the next step.

- This is called overrelaxation. It works like this. Consider a set of values φ(x,y) on the grid points that return φ'(x,y) after one iteration. The new set can be written in terms of the old as

$$\phi'(x,y) = \phi(x,y) + \Delta\phi(x,y)$$

- where Δφ(x,y) is the change in φ at each step.

OVERRELAXATION

- Now we define a set of overrelaxed values φ_w by

$$\phi_w(x,y) = \phi(x,y) + (1+w)\Delta\phi(x,y)$$

- Where w>0. In other words we change the value of φ by a little more than we should by just the new evaluation. So in our algorithm we make the new evaluation by

$$\phi_w(x,y) = \frac{1+w}{4}[\phi(x+a,y) + \phi(x-a,y) + \phi(x,y+a) + \phi(x,y-a)] - w\phi(x,y)$$

- as long as we make a good choice for w, this will converge quicker than the normal Jacobi method. Of course if we choose w too big then we will overshoot the solution and instead this may take more iterations.

GAUSS-SEIDEL METHOD

- A second trick for speeding up the Jacobi method is to use the new values φ' instead of φ when possible.
- This has the advantage that we think φ' is more accurate then φ, so we should converge quicker.
- But usually more importantly, this means we never need the old values of φ', which means we can just use one array to store the values.
- We can combine Gauss-Seidel with overrelaxation and this is what is usually done, but for an unexpected reason.
- It turns out the simple Jacobi overrelaxation does not work, the problem becomes numerically unstable. Gauss-Stidel with overrelaxtion is stable. So it is the way to go.

GAUSS-SEIDEL METHOD

- The final issue we have is how to choose w. There is no best answer or method for finding a best answer.
- The best choice depends both on the specific equations being solved and the shape of the grid.
- w is usually determined by trial and error. It has been proven that $w<1$ in generally stable but $w>1$ is unstable.
- For the Jacobi problem on a square grid that we have looked at as an example it turns out the best value of w is around 0.9.
- This can give a factor of 10 speed up, which is considerable.

INITIAL VALUE PDES

- Now let's turn to initial value problems. In this case we are told the starting condition in our problem and our goal is to solve it at some later time.
- As an example let's consider the diffusion equation. We can consider just one spatial dimension for simplicity.

$$\frac{\partial \phi}{\partial t} = D \frac{\partial^2 \phi}{\partial x^2}$$

- Where D is a diffusion coefficient. When used to study heat flow this equation can also be called the heat equation.
- The variable φ(x,t) depends on both x and t. Like the Laplace equation we just solved it a partial differential equation with two independent variables.

INITIAL VALUE PDES

- One could imagine solving this problem in the same way, but making a 2D space-time grid and then solving for the values on the grid.
- However, this approach will not work. Unlike the boundary value problem we don't know the value of the points at a later time to use as a constraint.
- We have to use a different technique. The first one we will look at is the forward-time centered space method for solving partial differential equations, called FTCS for short.

FTCS METHOD

- We start by dividing the spatial dimensions into a grid of points. In this case with just one spatial dimension just a line of points. Let the spacing between point be a. Then the derivative can be written as:

$$\frac{\partial^2 \phi}{\partial x^2} = \frac{\phi(x+a,t) + \phi(x-a,t) - 2\phi(x,t)}{a^2}$$

- and there for we can write the diffusion equation as

$$\frac{d\phi}{dt} = \frac{D}{a^2}[\phi(x+a,t) + \phi(x-a,t) - 2\phi(x,t)]$$

- If we think of the values of φ at different points as separate variables then the above equation is just *simultaneous ordinary differential equations*. These can be solved by the methods we have discussed last section.

FTCS METHOD

- The catch is that there can be thousand or millions of equations depending on the spacing in our grid and the dimensionality of the problem.
- The most common method to use is Euler's method. While this might seem strange since we spent so much time on 4th order Runge-Kutta, the reason is because our equation is already an approximation. Solving it with more accuracy then it actually has doesn't make a lot of sense.
- The approximation of the second derivative has a second order error associated with it. Thus using anything higher than a first order scheme to solve it is probably unnecessary.

FTCS METHOD

- Applying Euler's method gives

$$\phi(x, t+h) = \phi(x,t) + h\frac{D}{a^2}[\phi(x+a,t) + \phi(x-a,t) - 2\phi(x,t)]$$

- So we start our gird using our initial condition to get all values of $\varphi(x,t_0)$. We then use the above equation to get $\varphi(x,t_0+h)$ which for each point depends on the value of the points above and below our point in question.
- Repeating in steps of h we can propagate our solution out to the value of t we desire. This is the same way we solved ODEs, but now we are also using the spatial information which accounts for the other partial derivative.

EXAMPLE 9.3 THE HEAT EQUATION

- As an example lets consider a container made of 1cm think stainless steel initially at a uniform temperature of 20C. The container is placed in a bath of cold water at 0C and filled with hot water at 50C.
- Our goal is to calculate the temperature profile of the steel as a function of distance x from the hot side to the cold side, as a function of time. For simplicity lets consider the container very large so we can treat this as a 1D problem.
- Thermal conduction is governed by the diffusion equation, the equation we just developed a numerical technique for solving. The *thermal diffusivity* of steel is D = 4.25E-6 m^2s^{-1}.

EXAMPLE PROGRAM HEAT.PY

```
# Constants
L = 0.01      # Thickness of steel in meters
D = 4.25e-6   # Thermal diffusivity
N = 100       # Number of divisions in grid
a = L/N       # Grid spacing
h = 1e-4      # Time-step
epsilon = h/1000

Tlo = 0.0     # Low temperature in Celcius
Tmid = 20.0   # Intermediate temperature in Celcius
Thi = 50.0    # Hi temperature in Celcius

t1 = 0.01
t2 = 0.1
t3 = 0.4
t4 = 1.0
t5 = 10.0
tend = t5 + epsilon
# Create arrays
T = empty(N+1,float)
T[0] = Thi
T[N] = Tlo
T[1:N] = Tmid
```

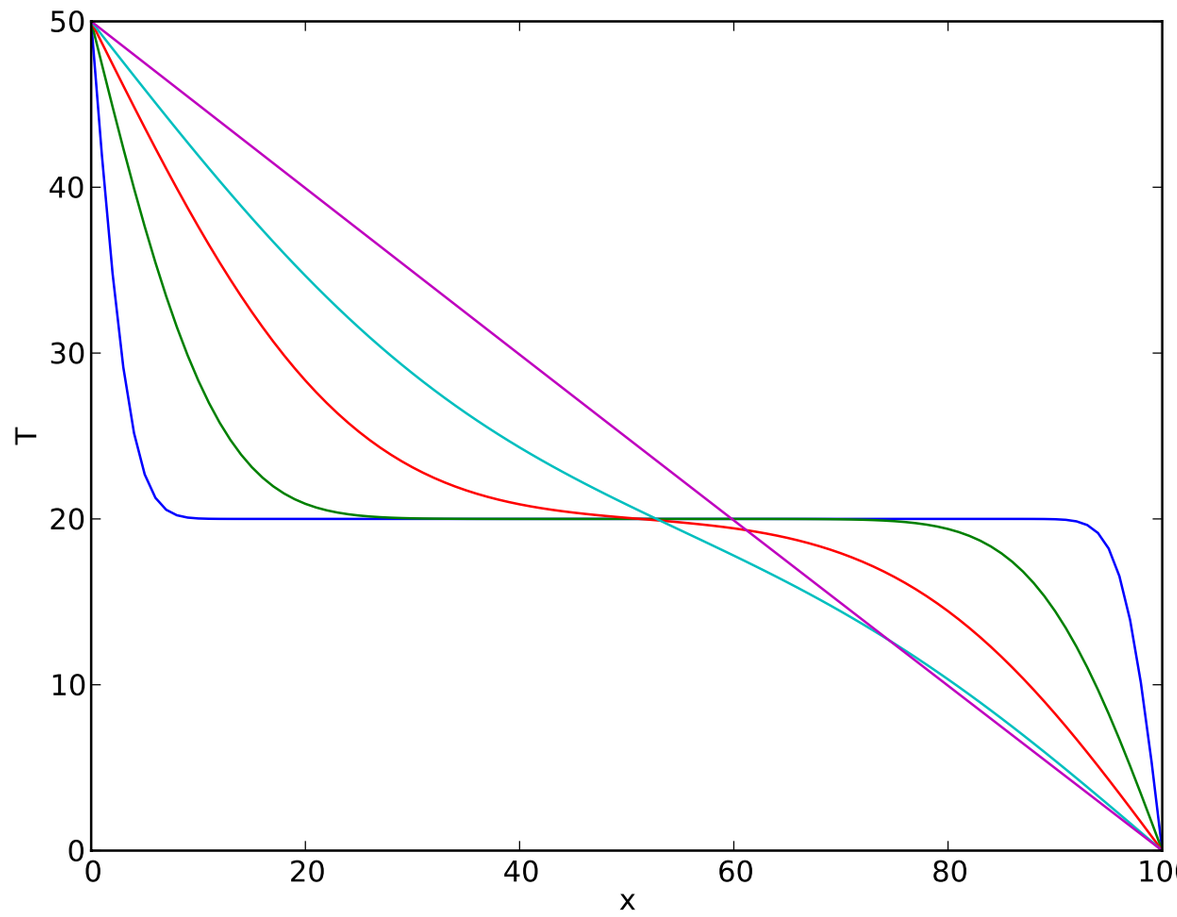
EXAMPLE PROGRAM HEAT.PY

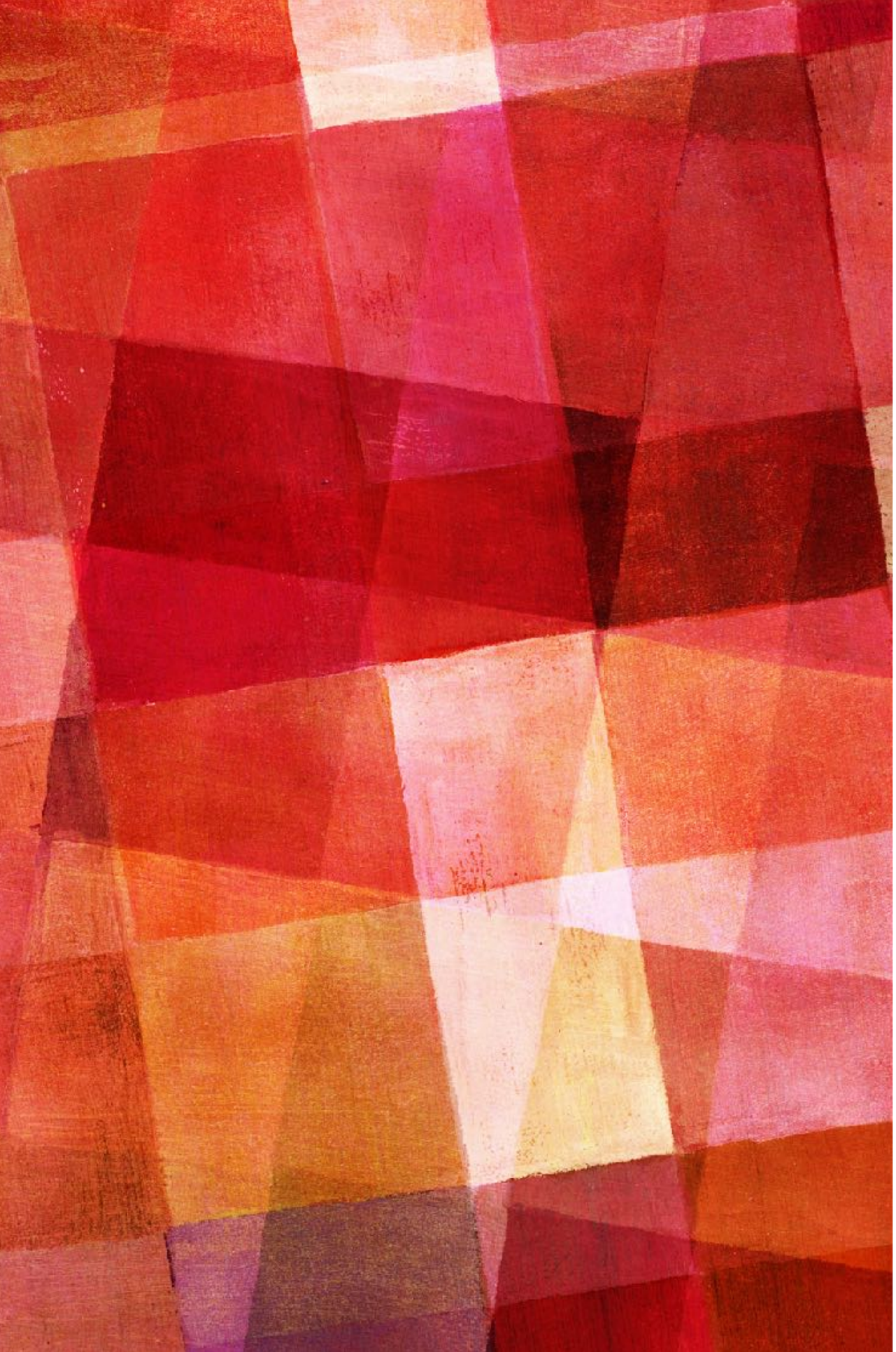
```
# Main loop
t = 0.0
c = h*D/(a*a)
while t<tend:

    # Calculate the new values of T
    for i in range(1,N):
        Tp[i] = T[i] + c*(T[i+1]+T[i-1]-2*T[i])
    T,Tp = Tp,T
    t += h

# Make plots at the given times
if abs(t-t1)<epsilon:
    plot(T)
if abs(t-t2)<epsilon:
    plot(T)
if abs(t-t3)<epsilon:
    plot(T)
if abs(t-t4)<epsilon:
    plot(T)
if abs(t-t5)<epsilon:
    plot(T)
```

- The output from that program is this graph.
- We see how the temperature profile evolves with time.
- At first the solution is close to our starting situation, most of the points have value Tmid.
- But by the last time plotted we have a linear temperature relationship.





EXERCISE 9.4

- A classic example of a diffusion problem with a time varying boundary condition is the diffusion of heat in the crust of the Earth, as surface temperature varies with the seasons. Suppose the mean daily temperature of a particular point on the surface varies as

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

- where τ=365days, A=10C and B=12C. At a depth of 20m below the surface almost all annual temperature variation is ironed out and the temperature is, to a good approximation, a constant 11C. We'll take the thermal diffusivity of the Earth's crust as $D=0.1\ m^2\ day^{-1}$.

- Write a program to calculate the temperature profile of the crust as a function of depth up to 20m and time up to 20 years. Start with a temperature of 10C for all x except the surface at 20m. Run the code for 9 years to allow it to settle down, then plot what you get in year 10 every 3 months.

NUMERICAL STABILITY

- The FTCS method works well for the diffusion equation, but there are other equations where it doesn't work as well.
- The wave equation is an important equation in physics, but one where FTCS performs badly. The wave equation can be written as

$$\frac{\partial^2\phi}{\partial t^2} - v^2\frac{\partial^2\phi}{\partial x^2} = 0 \qquad or \qquad \frac{\partial^2\phi}{\partial t^2} = v^2\frac{\partial^2\phi}{\partial x^2}$$

- using our finite difference method would give

$$\frac{d^2\phi}{dt^2} = \frac{v^2}{a^2}[\phi(x+a,t) + \phi(x-a,t) - 2\phi(x,t)]$$

- Now we can solve this by using our trick of turning a second order derivative into two first order derivatives

$$\frac{d\phi}{dt} = \psi(x,t) \qquad \frac{d\psi}{dt} = \frac{v^2}{a^2}[\phi(x+a,t) + \phi(x-a,t) - 2\phi(x,t)]$$

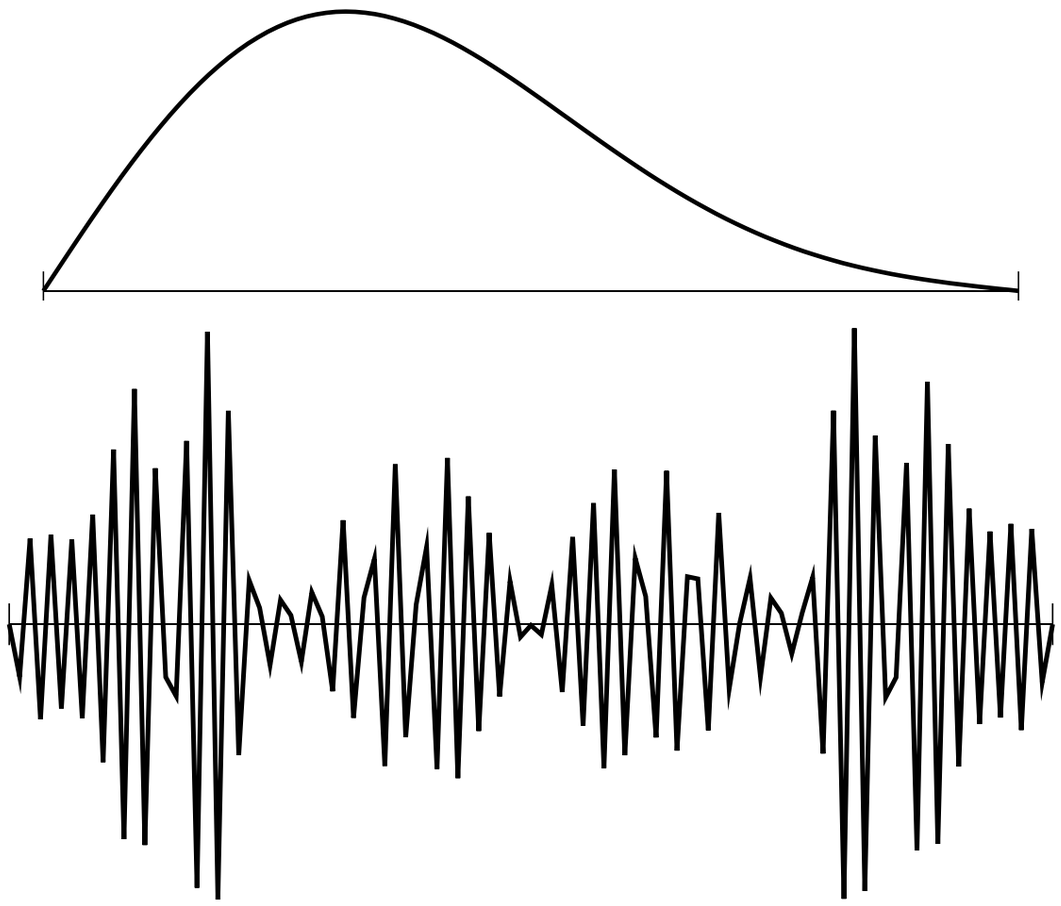
NUMERICAL STABILITY

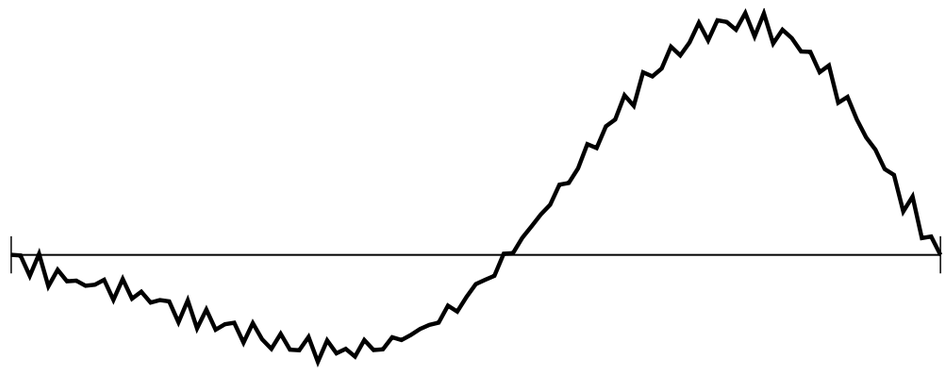
- Then applying Euler's method we get the FTCS equations

$$\phi(x, t+h) = \phi(x,t) + h\psi(x,t)$$

$$\psi(x, t+h) = \psi(x,t) + h\frac{v^2}{a^2}[\phi(x+a,t) + \phi(x-a,t) - 2\phi(x,t)]$$

- Now we can just iterate these equation forward in time





Solution for a vibrating string. Our solution starts out ok, but then evolves to something that doesn't seem physical at all.

VON NEUMANN STABILITY ANALYSIS

- What happened? Our solution seems to have become numerically unstable. If we continue to let this code run the errors will exceed the overflow value of the machine.
- Let's go back to our diffusion equation and check its stability to understand what is happening. Our FTCS equation is

$$\phi(x, t + h) = \phi(x, t) + h\frac{D}{a^2}[\phi(x + a, t) + \phi(x - a, t) - 2\phi(x, t)]$$

- An analysis due to von Neumann is to express φ as a Fourier series.

$$\phi(x, t) = \sum_k c_k(t)e^{ikx}$$

- We can now apply the FTCS equation to the Fourier series. Since the equation is linear we can advance each mode separately.

VON NEUMANN STABILITY ANALYSIS

- Plugging a single k value in gives

$$\phi_k(x, t+h) = c_k(t)e^{ikx} + h\frac{D}{a^2}c_k(t)[e^{ik(x+a)} + e^{ik(x-a)} - 2e^{ikx}]$$

$$= \left[1 + h\frac{D}{a^2}(e^{ika} + e^{-ika} - 2)\right]c_k(t)e^{ikx}$$

$$= \left[1 - h\frac{4D}{a^2}\sin^2\frac{1}{2}ka\right]c_k(t)e^{ikx}$$

- We see that each coefficient changes just based on k, not x or t. So we can write the coefficient at the next time step as

$$c_k(t+h) = \left[1 - h\frac{4D}{a^2}\sin^2\frac{1}{2}ka\right]c_k(t)$$

- Now we can see how a mode could be come unstable. If the factor in brackets is greater than 1 then that mode will grow each time step, becoming exponentially larger.

VON NEUMANN STABILITY ANALYSIS

$$c_k(t+h) = \left[1 - h\frac{4D}{a^2}\sin^2\frac{1}{2}ka\right]c_k(t)$$

- Luckily the term in brackets is always less than 1. One minus a positive number is always less than 1. However if the sin term becomes greater than 2 then we have the same problem.
- In this case the solution is stable as long as

$$h \leq \frac{a^2}{2D}$$

- Note this gives us a condition on our time step relative to spacing of our grid. If we don't choose steps small enough errors can grow exponentially and give completely unphysical answers.
- If h is small enough then all k values will decay except for k=0, which is what we expect for a diffusion equation, we should reach a solution that is uniform in space, without oscillations.

NUMERICAL STABILITY

- Now let us perform a similar analysis on the wave equation. Since we have two variables, φ and ψ, we create Fourier series for both of them with coefficients c_φ and c_ψ. From the same line of arguments we get

$$c_\phi(t+h) = c_\phi(t) - hc_\psi(t)$$

$$c_\psi(t+h) = c_\psi(t) - hc_\phi(t)\frac{4v^2}{a^2}\sin^2\frac{1}{2}ka$$

- we can express this as a vector $\mathbf{c}(t+h) = \mathbf{A}\mathbf{c}(t)$ where

$$\mathbf{A} = \begin{bmatrix} 1 & h \\ -h\frac{4v^2}{a^2}\sin^2\frac{1}{2}ka & 1 \end{bmatrix}$$

- so again each time step the values of c change by an operation that doesn't depend on x or t.

NUMERICAL STABILITY

- Now let us express **A** in terms of its eigenvectors, which we will call $\mathbf{v}_1$ and $\mathbf{v}_2$. Then there exist some values of α_1 and α_2 that give **c**=

  $\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2$. So we can write

$$\mathbf{c}(t+h) = \mathbf{A}(\alpha_1 v_1 + \alpha_2 v_2) = \alpha_1\lambda_1 v_1 + \alpha_2\lambda_2 v_2$$

$$\mathbf{c}(t+mh) = \alpha_1\lambda_1^m v_1 + \alpha_2\lambda_2^m v_2$$

- where λ are the corresponding eigenvalues for **A**. If either of them is greater than unity, they will grow and our result will be numerically unstable. Solving for the eigenvalue we get

$$|\lambda| = \sqrt{1 + \frac{4h^2v^2}{a^2}\sin^2\frac{1}{2}a^2}$$

- which we can see are always greater than 1. That is no matter what choice we make for h the numerical errors will grow exponentially. Thus one can only solve the wave equation with FTCS for short periods of time, before the errors grow large.

IMPLICIT METHOD

- Let us still consider the wave equation, but now let us substitute -h for h, that is we will take a step back instead of a step forward.

$$\phi(x, t-h) = \phi(x,t) - h\psi(x,t)$$

$$\psi(x, t-h) = \psi(x,t) - h\frac{v^2}{a^2}[\phi(x+a,t) + \phi(x-a,t) - 2\phi(x,t)]$$

- Let us now make a second substitution, let us change t to t+h. Then rearranging the terms we get

$$\phi(x, t+h) - h\psi(x,t+h) = \phi(x,t)$$

$$\psi(x, t+h) - h\frac{v^2}{a^2}[\phi(x+a,t+h) + \phi(x-a,t+h) - 2\phi(x,t+h)] = \psi(x,t)$$

- we now again have equations for φ and ψ at t+h, but they do not explicitly give us the values we have to solve the equations to find the values. Thus these equation are implicit.

IMPLICIT METHOD

- Now we can try repeating the Neumann stability analysis. Now we get $\mathbf{Bc}(t+h) = \mathbf{c}(t)$. $\mathbf{B}^{-1}$ is then

$$\mathbf{B}^{-1} = \frac{1}{1+h^2r^2}\begin{bmatrix} 1 & h \\ -hr^2 & 1 \end{bmatrix} \quad \textit{where} \qquad r = (2v/a)\sin\frac{1}{2}ka$$

- The eigenvalues in this case have the magnitude of

$$|\lambda| = \frac{1}{\sqrt{1+h^2r^2}}$$

- which we can see are always less than unity. The implicit method for the wave equation is *unconditionally stable*.

- However, we still have a problem. Our Fourier analysis tells us that each k value will decay away exponentially, which is not very wave like behavior. Our solution is stable but unphysical.

CRANK-NICOLSON METHOD

- The explicit FTCS method and the implicit method either grow or decay too much. What we want is a method somewhere in between.
- Such a method is called the Crank-Nicolson method. In this method we take the average of the two approaches.

$$\phi(x,t+h) - \frac{1}{2}h\psi(x,t+h) = \phi(x,t) + \frac{1}{2}h\psi(x,t)$$

$$\psi(x,t+h) - h\frac{v^2}{2a^2}[\phi(x+a,t+h) + \phi(x-a,t+h) - 2\phi(x,t+h)]$$
$$= \psi(x,t) + h\frac{v^2}{2a^2}[\phi(x+a,t) + \phi(x-a,t) - 2\phi(x,t)]$$

CRANK-NICOLSON METHOD

- In this case the matrix for stability analysis is just $\mathbf{B}^{-1}\mathbf{A}$ whose eigenvalues are

$$|\lambda| = \frac{\sqrt{(1 - h^2r^2 + 2ihr)(1 - h^2r^2 - 2ihr)}}{1 + h^2r^2} = 1$$

- The eigenvalues for this combination are exactly one. The Fourier coefficients neither grow or decay. This method is exactly on the boundary between the FTCS and implicit method. Waves won't grow or decay but propagate, exactly what we expect from the wave equation.

- While the Crank-Nicolson method is more complicated to write down, it is still relatively fast to solve. Each grid point still only depends on the values on each side, so as a matrix they are tridiagonal which we saw previously can be solved quickly by methods such as Gaussian elimination.

SPECTRAL METHOD

- The finite difference method is not the only approach to solving partial differential equations. There are many others that are widely used, like the finite element method. One method that can be extremely useful in certain situations is the spectral method or Fourier transform method.

- Let's consider again the wave equation for a wave on a fixed string of length L fixed at both ends so that $\varphi(0)=\varphi(L)=0$. Let's guess the solution can be expressed in the form

$$\phi_k(x,t) = \sin\frac{\pi k x}{L} e^{i\omega t}$$

- This automatically satisfies the boundary conditions and the wave equation as long as

$$\omega = \frac{\pi v k}{L}$$

SPECTRAL METHOD

- Now let's divide the string into N equal intervals bounded by N+1 grid points.

$$x_n = \frac{n}{N} L$$

- We have our guessed solution at any of these points by simply replacing x by x_n. We can state a general solution by combining these solutions in a linear combination.

$$\phi(x_n, t) = \frac{1}{N} \sum_{k=1}^{N-1} b_k \sin\left(\frac{\pi k n}{N}\right) \exp\left(i \frac{\pi v k t}{L}\right)$$

- at t=0 the exponential vanishes. We can divide the coefficients into real and imaginary parts, $b_k = \alpha_k + i\eta_k$.

$$\phi(x_n, 0) = \frac{1}{N} \sum_{k=1}^{N-1} \alpha_k \sin\left(\frac{\pi k n}{N}\right)$$

SPECTRAL METHOD

- which is just a Fourier series with coefficients a_k. Similarly the time derivative of the real part of the solution is

$$\frac{\partial \phi}{\partial t} = -\frac{\pi v}{L}\frac{1}{N}\sum_{k=1}^{N-1} k\eta_k \sin\left(\frac{\pi k n}{N}\right)$$

- which is also just a Fourier series with coefficients $k\eta_k$. So there is a choice of $k\eta_k$ that would match any initial condition on the derivative.
- The spectral method allows us to transform our PDE into a Fourier transform, which we know how to solve with FFT. The spectral method doesn't require time steps, we solve the function $\varphi(x,t)$ at all t. Thus even though it may take more computation than one time step using FTCS it will be much quicker than 10^6 time steps.

SPECTRAL METHOD

- The spectral method has some nice features, but also some limitations.
- It only works for problems with a simply shaped boundary condition. There is no straightforward way to adapt the method for more strangely shaped boundary conditions.
- Also, it is only applicable to linear differential equations because for nonlinear equations we can not add together the various terms in the Fourier series.
- The finite difference method has neither of these restrictions and such is more generally applicable, though one must check to see if it is numerically stable.