Previously considered:

Diffusion equation $$\frac{\partial u(x,t)}{\partial t} = D\frac{\partial^2 u(x,t)}{\partial x^2}$$

1st order wave equation (or advection equation) $$\frac{\partial u(x,t)}{\partial t} = c\frac{\partial u(x,t)}{\partial x}$$

We have seen that these equations often require different methods. E.g., **FTCS** is stable for the diffusion equation (if only for rather small time steps), but unconditionally unstable for the advection equation. Conversely, **leapfrog** is conditionally stable for the advection equation, but unstable for the diffusion equation.

So, what do we do if these two equations are combined in one?

Advection-diffusion (or convection-diffusion) equation

$$\frac{\partial u(x,t)}{\partial t} = D\frac{\partial^2 u(x,t)}{\partial x^2} + c\frac{\partial u(x,t)}{\partial x}$$

Concentration of particles that undergo Brownian motion (diffusion) in a fluid moving with velocity -*c*.

$$\frac{\partial u(x,t)}{\partial t} = D\frac{\partial^2 u(x,t)}{\partial x^2} + c\frac{\partial u(x,t)}{\partial x}$$

Also, Navier-Stokes equation for fluid motion has a similar form:

$$\rho\left(\frac{\partial \vec{v}}{\partial t} + (\vec{v}\cdot\nabla)\vec{v}\right) = -\nabla p + \eta\nabla^2\vec{v}$$

We have already looked at the stationary form of this equation when discussing boundary value methods for ODEs.

Let's see when it's possible to use FTCS.

$$\frac{u_m^{n+1} - u_m^n}{\tau} = D\frac{u_{m+1}^n - 2u_m^n + u_{m-1}^n}{h^2} + c\frac{u_{m+1}^n - u_{m-1}^n}{2h}$$

$$u_m^{n+1} = u_m^n + \frac{D\tau}{h^2}\left(u_{m+1}^n - 2u_m^n + u_{m-1}^n\right) + \frac{c\tau}{2h}\left(u_{m+1}^n - u_{m-1}^n\right)$$

$$u_m^n = e^{ikhm - \lambda\tau n} \qquad e^{-\lambda\tau} = 1 + \frac{2D\tau}{h^2}\left(\cos(kh) - 1\right) + i\frac{c\tau}{h}\sin(kh)$$

Stability condition: $$\left|1 + \frac{2D\tau}{h^2}\left(\cos(kh) - 1\right) + i\frac{c\tau}{h}\sin(kh)\right| \le 1$$

$$\left|1+\frac{2D\tau}{h^2}\left(\cos(kh)-1\right)+i\frac{c\tau}{h}\sin(kh)\right|\le 1$$

$$\left[1-\frac{4D\tau}{h^2}\sin^2(kh/2)\right]^2+\frac{4c^2\tau^2}{h^2}\sin^2(kh/2)\cos^2(kh/2)\le 1$$

$$-\frac{8D\tau}{h^2}\sin^2(kh/2)+\frac{16D^2\tau^2}{h^4}\sin^2(kh/2)[1-\cos^2(kh/2)]+\frac{4c^2\tau^2}{h^2}\sin^2(kh/2)\cos^2(kh/2)\le 0$$

$$F(k)=\left(\frac{16D^2\tau^2}{h^4}-\frac{8D\tau}{h^2}\right)+\left(\frac{4c^2\tau^2}{h^2}-\frac{16D^2\tau^2}{h^4}\right)\cos^2(kh/2)\le 0$$

Good news: whenever $D\neq 0$, this is negative for small enough τ.

$k=0$: $\frac{16D^2\tau^2}{h^4}-\frac{8D\tau}{h^2}\le 0 \;\Rightarrow\; \tau\le\frac{h^2}{2D}$. Same condition as for pure diffusion

For $\frac{|c|h}{2D}<1$, the second bracket is negative, and the above condition will be sufficient to ensure that F(k) ≤ 0 for all *k*. However, for $\frac{|c|h}{2D}>1$, this is not the case and the already strict condition on τ will become even stricter.

Very often in physics, when we are dealing with a problem where some objects undergo both random motion (described by the diffusion coeff. *D*) and directed motion (described by speed |*c*|) and there is also some length *L* in the problem, it is convenient to introduce a dimensionless number

$\mathrm{Pe}=\frac{|c|L}{D}$ called the Péclet number that characterizes relative importance

of advection and diffusion. Here our length is the mesh step.

$\mathrm{Pe}=\frac{|c|h}{2D}$ $\quad \mathrm{Pe}<1$ – no additional restrictions on the time step.

$\mathrm{Pe}>1$ – there are additional restrictions.

$$u_m^{n+1}=u_m^n+\frac{D\tau}{h^2}\left(u_{m+1}^n-2u_m^n+u_{m-1}^n\right)+\frac{c\tau}{2h}\left(u_{m+1}^n-u_{m-1}^n\right)$$

$$u_m^{n+1}=\left(\frac{D\tau}{h^2}+\frac{c\tau}{2h}\right)u_{m+1}^n+\left(1-2\frac{D\tau}{h^2}\right)u_m^n+\left(\frac{D\tau}{h^2}-\frac{c\tau}{2h}\right)u_{m-1}^n$$

We want the coefficients to be non-negative, otherwise, *u* may become negative, which is unphysical, if it represents, e.g., particle concentration.

$1-2\frac{D\tau}{h^2}\geq 0 \;\Rightarrow\; \tau\leq\frac{h^2}{2D}$ – basic stability condition

$\frac{|c|\tau}{2h}\leq\frac{D\tau}{h^2} \;\Rightarrow\; \mathrm{Pe}\leq 1$

$$\frac{\partial u(x,t)}{\partial t} = D\frac{\partial^2 u(x,t)}{\partial x^2} + c\frac{\partial u(x,t)}{\partial x} \qquad \mathrm{Pe} = \frac{|c|h}{2D}$$

Of course, can always reduce h to keep Pe≤1, but sometimes too restrictive.

For Pe > 1, have to use an upwind scheme. E.g., forward difference for c>0.

$$\frac{u_m^{n+1} - u_m^n}{\tau} = D\frac{u_{m+1}^n - 2u_m^n + u_{m-1}^n}{h^2} + c\frac{u_{m+1}^n - u_m^n}{h}$$

$$u_m^{n+1} = u_m^n + \frac{D\tau}{h^2}\left(u_{m+1}^n - 2u_m^n + u_{m-1}^n\right) + \frac{c\tau}{h}\left(u_{m+1}^n - u_m^n\right)$$

$$u_m^{n+1} = \left(\frac{D\tau}{h^2} + \frac{c\tau}{h}\right)u_{m+1}^n + \left(1 - 2\frac{D\tau}{h^2} - \frac{c\tau}{h}\right)u_m^n + \frac{D\tau}{h^2}u_{m-1}^n$$

Now $\tau \le \dfrac{1}{2D/h^2 + c/h} = \dfrac{h^2/(2D)}{1+\mathrm{Pe}}$ is sufficient for coefficients to be non-negative.

$$u_m^n = e^{ikhm - \lambda\tau n} \qquad e^{-\lambda\tau} = 1 + \frac{2D\tau}{h^2}\left(\cos(kh) - 1\right) + \frac{c\tau}{h}\left(e^{ikh} - 1\right)$$

$$\left|1 + \frac{2D\tau}{h^2}\left(\cos(kh) - 1\right) + \frac{c\tau}{h}\left(e^{ikh} - 1\right)\right| \le 1$$

$$\left|1+\frac{2D\tau}{h^2}\left(\cos(kh)-1\right)+\frac{c\tau}{h}\left(e^{ikh}-1\right)\right|\le 1$$

$$\left[1-\left(\frac{4D\tau}{h^2}+\frac{2c\tau}{h}\right)\sin^2(kh/2)\right]^2+\frac{4c^2\tau^2}{h^2}\sin^2(kh/2)\cos^2(kh/2)\le 1$$

$$F(k)=\left[\left(\frac{4D\tau}{h^2}+\frac{2c\tau}{h}\right)^2-2\left(\frac{4D\tau}{h^2}+\frac{2c\tau}{h}\right)\right]+\left[\frac{4c^2\tau^2}{h^2}-\left(\frac{4D\tau}{h^2}+\frac{2c\tau}{h}\right)^2\right]\cos^2(kh/2)\le 0$$

$$\left(\frac{4D\tau}{h^2}+\frac{2c\tau}{h}\right)^2-2\left(\frac{4D\tau}{h^2}+\frac{2c\tau}{h}\right)\le 0 \;\Rightarrow\; \tau\le\frac{1}{2D/h^2+c/h}=\frac{h^2/(2D)}{1+\mathrm{Pe}}$$

$$\frac{4c^2\tau^2}{h^2}-2\left(\frac{4D\tau}{h^2}+\frac{2c\tau}{h}\right)\le 0 \;\Rightarrow\; \tau\le\frac{2D}{c^2}+\frac{h}{c}=(2D/c^2)(1+\mathrm{Pe})$$ - less stringent

Disadvantage: 1st order space discretization.

Operator splitting

Consider $\frac{\partial u}{\partial t} = (\hat{A} + \hat{B})u$, where $\hat{A}$ and $\hat{B}$ are differential operators, e.g.,

$\hat{A} = D\frac{\partial^2}{\partial x^2}$ and $\hat{B} = c\frac{\partial}{\partial x}$. Assume they do not depend on *t* explicitly.

$$u(x,\tau) = e^{(\hat{A}+\hat{B})\tau} u(x,0)$$

$$e^{(\hat{A}+\hat{B})\tau} = 1 + \tau(\hat{A}+\hat{B}) + \frac{\tau^2}{2}(\hat{A}+\hat{B})^2 + O(\tau^3)$$

$$= 1 + \tau(\hat{A}+\hat{B}) + \frac{\tau^2}{2}(\hat{A}^2 + \hat{B}^2 + \hat{A}\hat{B} + \hat{B}\hat{A}) + O(\tau^3)$$

Note $e^{(\hat{A}+\hat{B})\tau}$ is in general not the same as $e^{\hat{B}\tau} e^{\hat{A}\tau}$. And indeed,

$$e^{\hat{B}\tau} e^{\hat{A}\tau} = \left[1 + \tau\hat{B} + \frac{\tau^2}{2}\hat{B}^2 + O(\tau^3)\right]\left[1 + \tau\hat{A} + \frac{\tau^2}{2}\hat{A}^2 + O(\tau^3)\right]$$

$$= 1 + \tau(\hat{A}+\hat{B}) + \frac{\tau^2}{2}(\hat{A}^2 + \hat{B}^2 + 2\hat{B}\hat{A}) + O(\tau^3)$$

Although in our case, it is the same, as $\hat{A}$ and $\hat{B}$ commute.

$$e^{(\hat{A}+\hat{B})\tau} = 1+\tau(\hat{A}+\hat{B})+\frac{\tau^2}{2}(\hat{A}+\hat{B})^2+O(\tau^3)$$

$$=1+\tau(\hat{A}+\hat{B})+\frac{\tau^2}{2}(\hat{A}^2+\hat{B}^2+\hat{A}\hat{B}+\hat{B}\hat{A})+O(\tau^3)$$

Even if $\hat{A}$ and $\hat{B}$ do not commute, we can use the following trick:

$$e^{\hat{A}\tau/2}e^{\hat{B}\tau}e^{\hat{A}\tau/2} = \left[1+\frac{\tau}{2}\hat{A}+\frac{\tau^2}{8}\hat{A}^2+O(\tau^3)\right]\left[1+\tau\hat{B}+\frac{\tau^2}{2}\hat{B}^2+O(\tau^3)\right]\left[1+\frac{\tau}{2}\hat{A}+\frac{\tau^2}{8}\hat{A}^2+O(\tau^3)\right]$$

$$=1+\tau(\hat{A}+\hat{B})+\frac{\tau^2}{2}\left(\frac{1}{4}\hat{A}^2+\hat{B}^2+\frac{1}{4}\hat{A}^2+\hat{A}\hat{B}+\hat{B}\hat{A}+\frac{1}{2}\hat{A}^2\right)+O(\tau^3)$$

$$=1+\tau(\hat{A}+\hat{B})+\frac{\tau^2}{2}\left(\hat{A}^2+\hat{B}^2+\hat{A}\hat{B}+\hat{B}\hat{A}\right)+O(\tau^3)$$

Accurate to 2nd order.

$$\frac{\partial u}{\partial t}=(\hat{A}+\hat{B})u \qquad \text{Auxiliary problems} \qquad \frac{\partial u}{\partial t}=\hat{A}u \text{ and } \frac{\partial u}{\partial t}=\hat{B}u$$

Suppose we have good, stable 2nd order schemes for the auxiliary problems.

They can be viewed as matrices $\boldsymbol{A}(\tau)$ and $\boldsymbol{B}(\tau)$ acting on the vector corresponding to time step *n* and giving the vector of values at time step *n*+1 for the corresponding equation. These matrices approximate the evolution operators $e^{\tau\hat{A}}$ and $e^{\tau\hat{B}}$, respectively. Then to obtain the matrix $\boldsymbol{C}(\tau)$ approximating $e^{\tau(\hat{A}+\hat{B})}$, we use

$$e^{\tau(\hat{A}+\hat{B})}\approx e^{\tau\hat{A}/2}e^{\tau\hat{B}}e^{\tau\hat{A}/2} \;\Rightarrow\; \boldsymbol{C}(\tau)\approx \boldsymbol{A}(\tau/2)\boldsymbol{B}(\tau)\boldsymbol{A}(\tau/2)$$

That is, advance by 1/2 step pretending that there is only the first term on the right-hand side (ignoring the second term). Then advance by a full step using only the second term. Then advance by 1/2 step using only the first term. In fact, we can rewrite

$$\boldsymbol{C}(n\tau)=\boldsymbol{A}(\tau/2)\boldsymbol{B}(\tau)\boldsymbol{A}(\tau/2)\boldsymbol{A}(\tau/2)\boldsymbol{B}(\tau)\boldsymbol{A}(\tau/2)\ldots\boldsymbol{A}(\tau/2)\boldsymbol{B}(\tau)\boldsymbol{A}(\tau/2)$$

$$¿\,\boldsymbol{A}(\tau/2)\boldsymbol{B}(\tau)\boldsymbol{A}(\tau)\boldsymbol{B}(\tau)\boldsymbol{A}(\tau)\ldots\boldsymbol{A}(\tau)\boldsymbol{B}(\tau)\boldsymbol{A}(\tau/2)$$

Finite element method for PDEs

Considered FEM for $y''(x)+qy(x)=f(x),\ y(a)=0,\ y(b)=0$

I will follow that derivation to skip similar steps and modify it for

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

Had residual $R(x)=y''(x)+qy(x)-f(x)=0$ I will omit f(x)

Now: $R(x)=D\,y''(x)-\dot{y}(x)=0$

Had: $\int_a^b w(x)\,R(x)\,dx=\int_a^b w(x)[y''(x)+qy(x)]\,dx=0$

$$\int_a^b [-w'(x)\,y'(x)+qw(x)\,y(x)]\,dx=0$$

Now: $\int_a^b [-Dw'(x)\,y'(x)-w(x)\,\dot{y}(x)]\,dx=0$ Weak form.

$$\int_a^b [-w'(x)\,y'(x)+qw(x)\,y(x)]\,dx=0$$

$$\int_a^b [-Dw'(x)\,y'(x)-w(x)\,\dot{y}(x)]\,dx=0$$

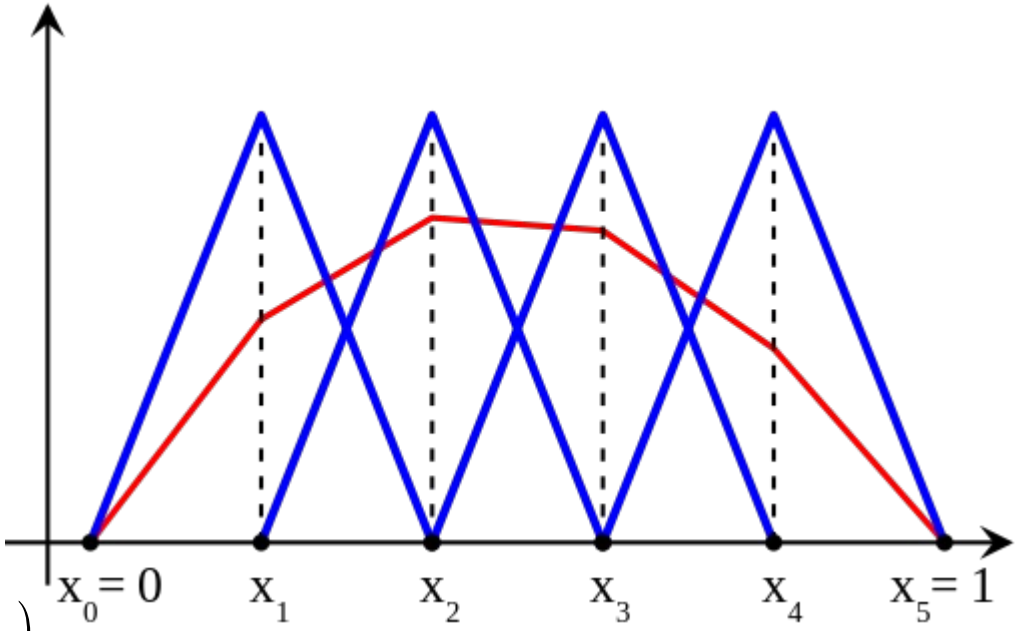
$$y(x)\approx\sum_{j=1}^{J} c_j\phi_j(x);\; w(x)\approx\sum_{j=1}^{J} d_j\phi_j(x)$$

Had:
$$\int_a^b\left[-\phi'_i(x)\sum_{j=1}^{J} c_j\phi'_j(x)+q\,\phi_i(x)\sum_{j=1}^{J} c_j\phi_j(x)\right]=0$$

Now:
$$\int_a^b\left[-D\,\phi'_i(x)\sum_{j=1}^{J} c_j(t)\phi'_j(x)-\phi_i(x)\sum_{j=1}^{J}\dot{c}_j(t)\phi_j(x)\right]=0$$

$$\int_a^b \left[-D\,\phi'_i(x) \sum_{j=1}^{J} c_j(t)\,\phi'_j(x) - \phi_i(x) \sum_{j=1}^{J} \dot{c}_j(t)\,\phi_j(x) \right] = 0$$

$$\phi_i = \begin{cases} \dfrac{x - x_{i-1}}{x_i - x_{i-1}}, & x_{i-1} < x < x_i, \\ \dfrac{x - x_{i+1}}{x_i - x_{i+1}}, & x_i < x < x_{i+1}, \\ 0 \text{ otherwise} \end{cases}$$



$$\int_a^b \phi_i(x)\,\phi_i(x)\,dx = \frac{1}{3}(x_{i+1} - x_{i-1})$$

$$\int_a^b \phi_i(x)\,\phi_{i+1}(x)\,dx = \frac{x_{i+1} - x_i}{6}$$

$$\int_a^b \phi'_i(x)\,\phi'_i(x)\,dx = \frac{1}{x_i - x_{i-1}} + \frac{1}{x_{i+1} - x_i}$$

$$\int_a^b \phi'_i(x)\,\phi'_{i+1}(x)\,dx = -\frac{1}{x_{i+1} - x_i}$$

For equidistant nodes:

$$\int_a^b \phi_i(x)\,\phi_i(x)\,dx = \frac{2h}{3}$$

$$\int_a^b \phi_i(x)\,\phi_{i+1}(x)\,dx = \frac{h}{6}$$

$$\int_a^b \phi'_i(x)\,\phi'_i(x)\,dx = \frac{2}{h}$$

$$\int_a^b \phi'_i(x)\,\phi'_{i+1}(x)\,dx = -\frac{1}{h}$$

$$\int_a^b \left[-D\,\phi'_i(x) \sum_{j=1}^{J} c_j(t)\,\phi'_j(x) - \phi_i(x) \sum_{j=1}^{J} \dot{c}_j(t)\,\phi_j(x) \right] = 0$$

$$\int_a^b \phi_i(x)\,\phi_i(x)\,dx = \frac{2h}{3} \qquad \int_a^b \phi_i(x)\,\phi_{i+1}(x)\,dx = \frac{h}{6} \qquad \int_a^b \phi'_i(x)\,\phi'_i(x)\,dx = \frac{2}{h}$$

$$\int_a^b \phi'_i(x)\,\phi'_{i+1}(x)\,dx = -\frac{1}{h}$$

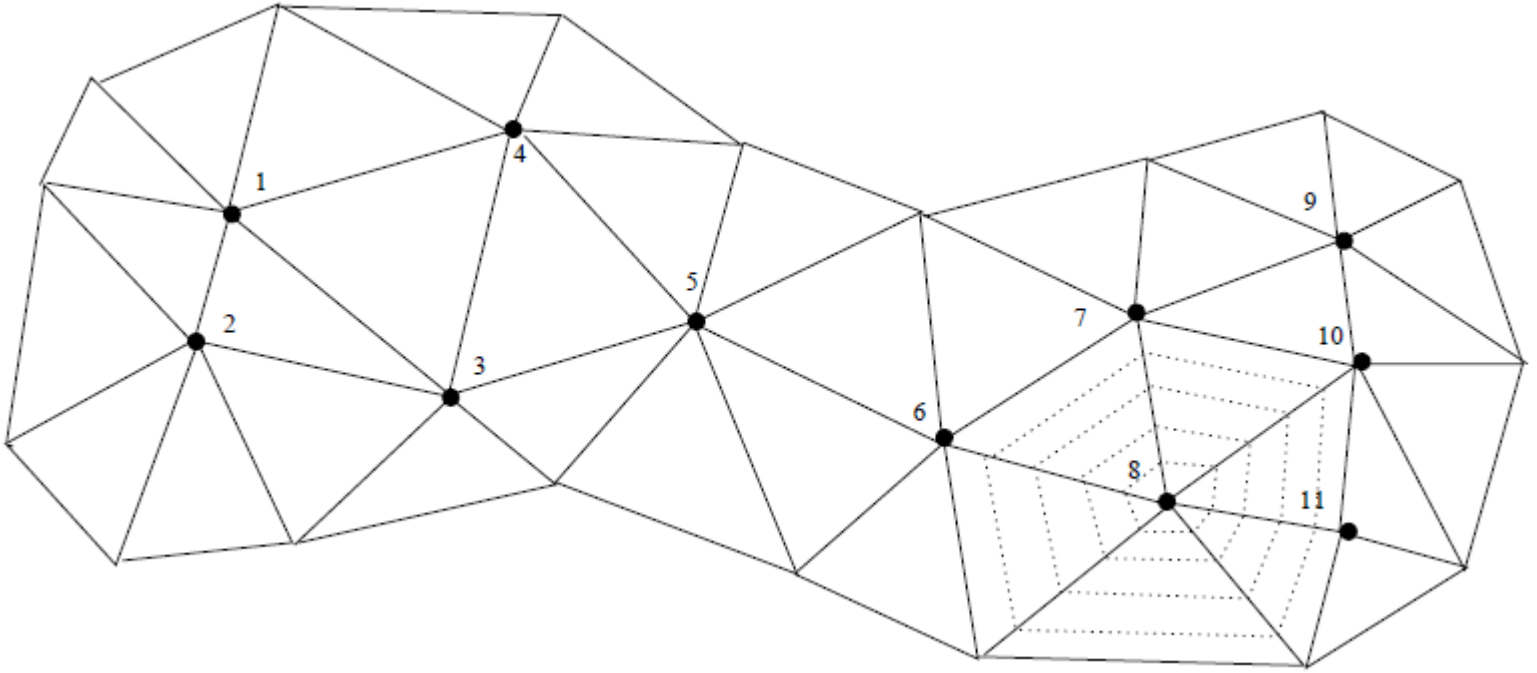
$$\frac{2h}{3}\dot{c}_i + \frac{h}{6}\dot{c}_{i-1} + \frac{h}{6}\dot{c}_{i+1} = D\left(-\frac{2}{h}c_i + \frac{1}{h}c_{i-1} + \frac{2}{h}c_{i+1} \right)$$

$$\frac{2}{3}\dot{c}_i + \frac{1}{6}\dot{c}_{i-1} + \frac{1}{6}\dot{c}_{i+1} = D\,\frac{c_{i-1} - 2c_i + c_{i+1}}{h^2}$$

When discretized in time, will be implicit even when it's simple Euler discretization. So might as well use trapezoidal (Crank-Nicolson).

FEM in 2D and 3D

Usually rectangular or triangular elements. Simplest functions are linear within each element and only nonzero at one of the corners.



Optimization of functions

Given a function $f(\vec{r})$, find $\vec{r}$ that minimizes it.

Local vs. global optimization. Global: find the point where the function reaches its lowest value over the whole domain where it is defined. Local: find any minimum.

We will consider local optimization **only**. Global optimization problems are **hard**. For a low-dimensional system where only a few minima are expected, use a local optimization algorithm many times, starting at different points. If, after a while, you stop finding new minima, you can hope that you have found all of them (or at least all that are low enough), and then the lowest of them is the global minimum.

This is impractical for high-dimensional systems (finding the ground state of an atomic cluster or a spin glass, or the native fold of a protein). Simulated annealing and genetic algorithms are general approaches, but they are stochastic, so we won't consider them.

If you need to find a **maximum**, just consider $-f(\vec{r})$.

At minima, $\nabla f(\vec{r})=0$. This is a system of equations (or just 1 equation in 1D), and we have already considered methods of solving such nonlinear equations. So why a separate consideration specifically for minima?

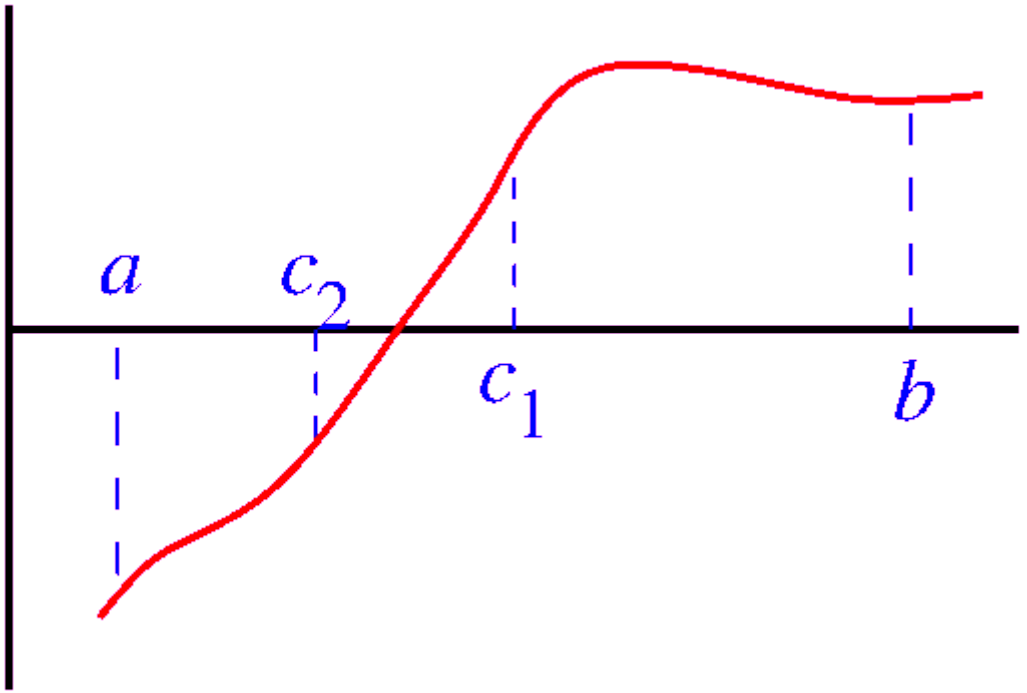
1. Not every solution of $\nabla f(\vec{r})=0$ is a minimum. In 1D, generically, there are only minima and maxima (although there can be points that are neither). But in higher dimensions, there are also saddles of various orders, and just by starting at an arbitrary point and solving $\nabla f(\vec{r})=0$, one is much more likely to find a saddle than a minimum.

In fact, even checking if what you have found is a minimum is not entirely trivial. You need to calculate the matrix of second derivatives (also known as the **Hessian** matrix) $H_{ij}=\frac{\partial^2 f}{\partial x_i x_j}$ and check if it is positive definite. Find the lowest eigenvalue, or do Cholesky decomposition (which should fail for non-positive-definite matrices).

2. We may not want to calculate the derivatives, or they may not even exist.

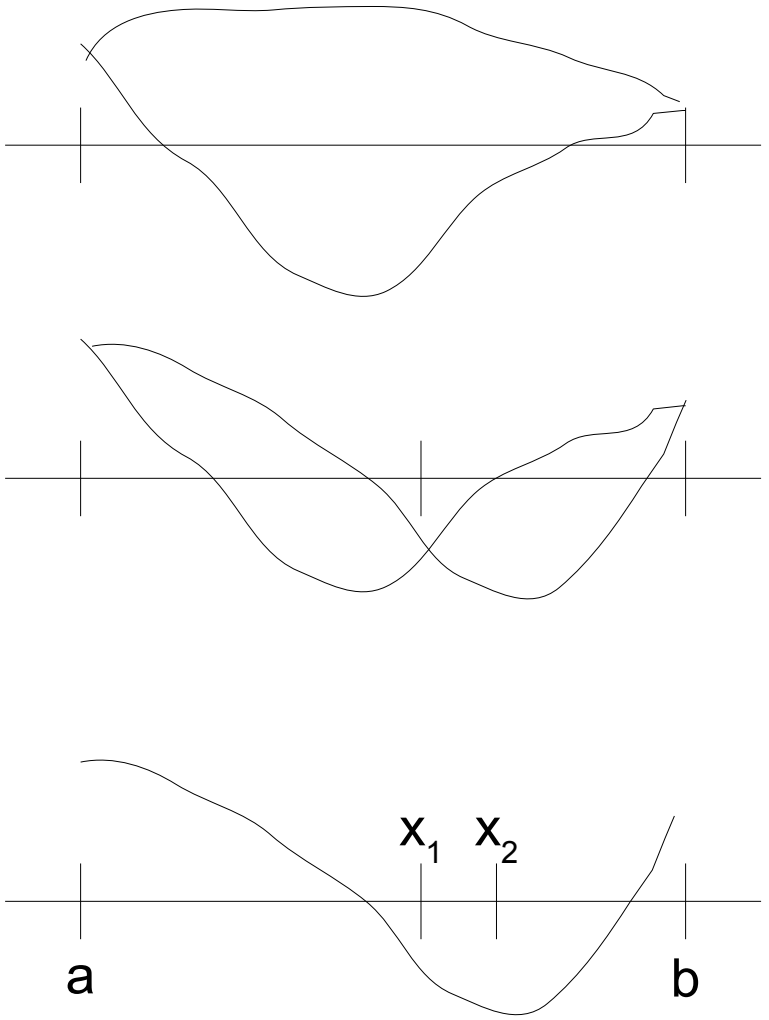
3. Finding a minimum is often easier than solving a general equation, especially in higher dimensions. For root-finding, we only have Newton-Raphson (plus some quasi-Newton methods). For optimization, there are many more.

Start with 1D. For root-finding, we had the bisection method where we took an interval that we assumed contained a single root and bisecting it and calculating the values of the function at bisecting points we narrowed down the interval at each step.



Is there something similar for minima?

First of all, how can we know that a given interval [a,b] contains a minimum? For root-finding, it was enough to check that f(a)f(b)<0. Obviously, for optimization just knowing f(a) and f(b) is not sufficient. We also need a point x_1 inside the interval. If $f(x_1) < f(a), f(b)$, then we know there is a minimum.



However, that still does not tell us if the minimum is between a and x_1 or between x_1 and b. So we need **one additional point** $x_2 > x_1$. If $f(x_1) < f(x_2)$, then the minimum is on $[a,x_2]$. If $f(x_2) < f(x_1)$, then the minimum is on $[x_1,b]$.

So then we can choose the interval and continue.

After choosing the new interval, we will already have one old point inside, so, just as for bisection, we only need to choose 1 additional point and calculate the value there. How do we choose the new point optimally?

Suppose we have $\frac{x_1-a}{b-a}=w<1/2 \;\Rightarrow\; \frac{b-x_1}{b-a}=1-w>1/2.$

If $\frac{x_2-x_1}{b-a}=z$, then the length of the new interval is either (1-*w*)(*b*-*a*), if x_1 is picked, or (*w*+*z*)(*b*-*a*), if x_2 is picked. If x_1 was previously placed optimally, then the optimal placement of x_2 is such that both intervals should be equal, i.e., $1-w=w+z \;\Rightarrow\; z=1-2w.$

At the next iteration, *w* and *z* should remain the same (or exchange places).

$$\frac{x_1-a}{x_2-a}=\frac{x_1-a}{b-a}\cdot b-a\,x_2-x_1+x_1-a=\frac{w}{z+w}=\frac{w}{1-w}=1-w$$

$$w^2-3w+1=0 \;\Rightarrow\; w=\frac{3-\sqrt{5}}{2}\approx 0.38197 \qquad 1-w=\frac{\sqrt{5}-1}{2}\approx 0.61803$$

$$1-w=\frac{\sqrt{5}-1}{2}\approx 0.61803$$ **Golden ratio.**

Golden section search. Just as bisection, very reliable (always converges), but slow (only linear), and actually slower than bisection, because the ratio of the intervals at subsequent steps is 1.61803 instead of 2.

When should we stop? Near the minimum, $f(x)\approx f(x_0)+\frac{1}{2}f''(x_0)(x-x_0)^2$

$$\frac{1}{2}f''(x_0)(x-x_0)^2<\epsilon|f(x_0)| \;\Rightarrow\; |x-x_0|<\sqrt{\epsilon}|x_0|\sqrt{\frac{2|f(x_0)|}{x_0^2 f''(x_0)}}$$

$|x-x_0|<\sim|x_0|\sqrt{\epsilon}$ Possible accuracy is much worse than machine precision.

An analog of the **false position** and **secant** methods. In these methods, approximated with a polynomial of the lowest degree that has a root, i.e., linear. Here, need to approximate with a polynomial of the lowest degree that has a minimum, i.e., a quadratic polynomial.

Successive parabolic interpolation.

Either use the same criterion as in golden section for choosing which point to discard (similar to false position; ensures bracketing), or always discard the oldest point (similar to secant; potentially faster convergence). In the latter case, superlinear convergence with exponent ≈1.324.

$$x = b - \frac{1}{2}\frac{(b-a)^2[f(b)-f(c)]-(b-c)^2[f(b)-f(a)]}{(b-a)[f(b)-f(c)]-(b-c)[f(b)-f(a)]}$$

Note the parabola may have a minimum outside the bracket or not have a minimum at all (but rather a maximum). Convergence not guaranteed. Generally, a good idea to combine with golden section using it when parabolic interpolation fails or its steps increase or do not decrease fast enough.

Newton's method

Application of Newton-Raphson to f'(x)=0.

For f(x) = 0 we had $x_{i+1} = x_i - f(x_i)/f'(x_i)$

So now $x_{i+1} = x_i - f'(x_i)/f''(x_i)$

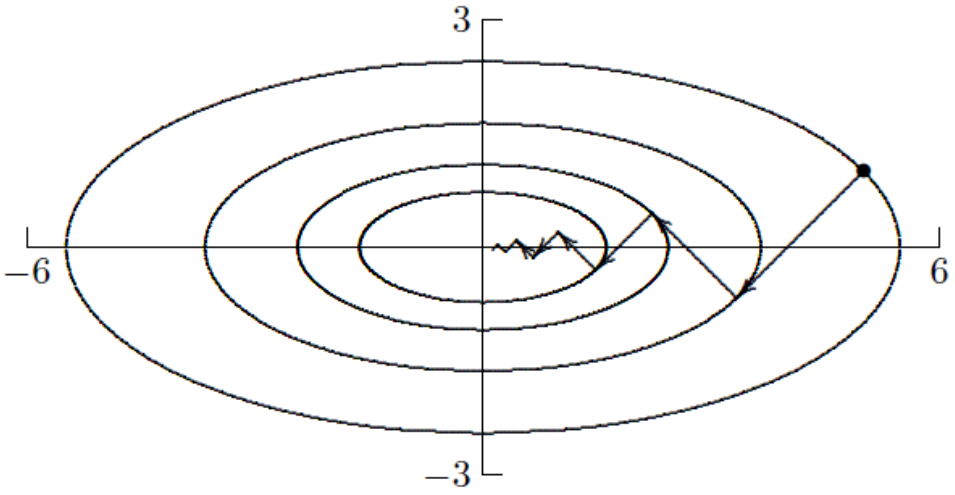
Can also be obtained by drawing a parabola at xi with the right first and second derivatives and finding its minimum.

Quadratic convergence, but unreliable and requires *f* ''.

Multidimensions

An approach that does not require derivatives: **Nelder-Mead simplex algorithm or amoeba algorithm**.

Form a simplex of n+1 points in n dimensions. At each step, the point with the largest value moves. Do reflection with respect to the centroid of the rest of the points. If better than the worst point, keep. If better than the best, move further. If worse, move in the opposite direction. If that does not help, shrink the simplex keeping the best point in place.



Heath, Scientific Computing: An Introductory Survey

Steepest descent

Calculate the gradient, move in the direction opposite to it. Do 1D minimization in this direction.

Problem: converges slowly for very asymmetric minima with both steep and flat directions.

Conjugate gradient

We considered conjugate gradient as an algorithm for solving linear systems

$$\boldsymbol{A}\,\vec{x}=\vec{b}\,,$$

but we derived it by considering a minimization problem for function

$$f\left(\vec{x}\right)=\frac{1}{2}\,\vec{x}\cdot\boldsymbol{A}\cdot\vec{x}-\vec{b}\cdot\vec{x}=\frac{1}{2}\,a_{ij}\,x_i\,x_j-b_i\,x_i$$

The algorithm works by building a sequence of conjugate directions

$\vec{p}^{\,(j)}\cdot\boldsymbol{A}\cdot\vec{p}^{\,(i)}=0$ and ensures that the residuals are orthogonal at each step:

$$\vec{r}^{\,(j)}\cdot\vec{r}^{\,(i)}=0$$

Can generalize the algorithm to arbitrary functions.

Start with initial guess $\vec{x}^{(0)}$.

$\vec{r}^{(0)} = \vec{b} - \boldsymbol{A}\vec{x}^{(0)}$.

$k = 0$

while $\|\vec{r}^{(k)}\| > \text{tolerance} \times \|\vec{b}\|$

if $k = 0$, $\vec{p}^{(0)} = \vec{r}^{(0)}$

else

$$\beta_k = \frac{\vec{r}^{(k)} \cdot \vec{r}^{(k)}}{\vec{r}^{(k-1)} \cdot \vec{r}^{(k-1)}}$$

$$\vec{p}^{(k)} = \vec{r}^{(k)} + \beta_k \vec{p}^{(k-1)}$$

end

$$\alpha_k = \frac{\vec{r}^{(k)} \cdot \vec{r}^{(k)}}{\vec{p}^{(k)} \cdot \boldsymbol{A} \cdot \vec{p}^{(k)}}$$

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} + \alpha_k \vec{p}^{(k)},$$

$$\vec{r}^{(k+1)} = \vec{r}^{(k)} - \alpha_k \boldsymbol{A}\vec{p}^{(k)},$$

$k = k + 1$

end

Note that the residual $\vec{r} = \vec{b} - \boldsymbol{A}\vec{x}$

is minus the gradient of the function.

If we follow this literally, we need to know ***A*** (the Hessian matrix). But since $\vec{r}^{(k)}$ is simply minus the gradient at step k, we can calculate it directly, by actually carrying out that step, i.e., doing line minimization in the direction of $\vec{p}^{(k-1)}$. On the other hand, calculation of $\vec{p}^{(k)}$ in terms of $\vec{r}^{(k)}$ and $\vec{p}^{(k-1)}$ does not involve ***A***.

This is the Fletcher-Reeves variant.

Polak-Ribiere: $\beta_k = \dfrac{\left(\vec{r}^{(k)} - \vec{r}^{(k-1)}\right) \cdot \vec{r}^{(k)}}{\vec{r}^{(k-1)} \cdot \vec{r}^{(k-1)}}$

Equivalent when the residuals are perfectly orthogonal.

Newton method can be used in multidimensions in principle, but will require a Hessian. For roots, we had

$$\vec{x}_{n+1} = \vec{x}_n - \boldsymbol{J}^{-1}\vec{f} \qquad \boldsymbol{J} = \left(\frac{\partial f_i}{\partial x_j}\right)$$

$$f_i \to \frac{\partial f}{\partial x_i} \qquad \boldsymbol{J} = \left(\frac{\partial^2 f}{\partial x_i \partial x_j}\right) = \boldsymbol{H}$$

$$\vec{x}_{n+1} = \vec{x}_n - \boldsymbol{H}^{-1}\nabla f$$

Can calculate $\boldsymbol{H}^{-1}\nabla f$ by solving the linear system iteratively. Do not need to converge. E.g., conjugate gradient iterations.

Quasi-Newton methods estimate the Hessian instead of calculating it accurately, updating the estimate at every step.

Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm

Solve $\boldsymbol{H}_k \vec{s}_k = -\nabla f(\vec{x}_k)$

$$\vec{x}_{k+1} = \vec{x}_k + \vec{s}_k$$

$$\vec{y}_k = \nabla f(\vec{x}_{k+1}) - \nabla f(\vec{x}_k)$$

$$\boldsymbol{H}_{k+1} = \boldsymbol{H}_k + (\vec{y}_k \otimes \vec{y}_k)/(\vec{y}_k \cdot \vec{s}_k) - (\boldsymbol{H}_k \cdot \vec{s}_k) \otimes (\boldsymbol{H}_k \cdot \vec{s}_k)/(\vec{s}_k \cdot \boldsymbol{H}_k \cdot \vec{s}_k)$$