Lets try moving from theory to computation by solving a problem. We discussed the basics of finite differences and saw that Central differences together with Zero boundary conditions yields a discrete matrix equation for the interior unknowns.

This discrete equation has the form $-\frac{K}{h^2} A\bar{u} = \bar{f}$

where $\bar{u} = [u_1, u_2 \dots u_{N-1}]$, $\bar{f} = [f_1, f_2 \cdots f_{N-1}]$

$$A = \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & 0 \\ & 1 & -2 & 1 & \\ & & \ddots & & \\ & 0 & & 1 & -2 \end{bmatrix}$$ and $K > 0$ is a constant.

Example problem:

- Consider $K=1$ and solve $-\frac{\partial^2}{\partial x^2} u = \sin(2x)$, $u(0) = u(\pi) = 0$

Key IDEA: We need to solve for the interior points using the discrete equation $-\frac{1}{h^2} A\bar{u} = \bar{f}$. To construct the mesh of $[0,\pi]$ we need to know how many internal points we will have.

To construct the matrix A we can use the included matlab function heatEqnTridiag(..) which exploits the built-in 'diag' command.

[See attached matlab code]

Question: What happens as you vary the number of internal points? What can you say, visually, about the error?

Question: As N gets larger what happens to h? What is the limit of the computed solution as $h \to 0$?

Experiment: try different right-hand sides. What happens?

Order of approximation:

We discussed different types of approximations to derivatives - forward, backwards and central. It was mentioned that some were "better" than others and that one way to measure "better" is called the order of the approximation. The higher the order, the better.

One way to make "order of approximation" arguments, especially for finite difference schemes, is via Taylor's Theorem; e.g. by employing a Taylor Series expansion.

Theorem (Taylor): Let $k \geq 1$ be an integer and let $f: \mathbb{R} \to \mathbb{R}$ be k-times differentiable at the point $a \in \mathbb{R}$. Then there exists a function $h_k: \mathbb{R} \to \mathbb{R}$ such that:

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(k)}(a)}{k!}(x-a)^k + h_k(x)(x-a)^k$$

and $\lim_{x \to a} h_k(x) = 0$

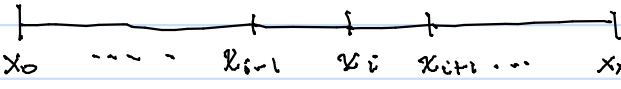
So Taylors theorem lets you rewrite f, near the point a, as a polynomial of degree k plus a remainder term. $f = P_k(x) + R(x)$

There are a few things you can verify in a textbook if you wish:

1) The polynomial $P_k(x)$ in $f = P_k(x) + R(x)$ is the "best fit" polynomial you can find to f near the point a.
2) $R(x)$ has the form $R(x) = h_k(x)(x-a)^k$. Introducing some extra assumptions on the smoothness of f allows the derivation of an explicit formula for $h_k(x)$.

Key IDEA: We can use Taylors theorem to determine the order, or efficacy, of a finite difference approximation.

How? Assume we have a mesh:



And we want to approximate $\frac{\partial f}{\partial x}$ at x_i. e.g: f_i' in our "discreet" notation

Suppose we use the forward difference approximation

$$\frac{\partial f}{\partial x}(x_i) \approx \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} = \frac{f_{i+1} - f_i}{h}$$

↑ expressed in our "discrete notation"

Let's assume that the function f is differentiable as many times as we want. Then by Taylor's theorem we have:

$$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''}{2!}(x_i)(x_{i+1} - x_i)^2 + \frac{f'''}{3!}(x_{i+1} - x_i)^3 + \cdots$$

"discrete" notation →

$$= f_i + h f_i' + \frac{h^2}{2!} f_i'' + \frac{h^3}{3!} f_i''' + \cdots$$

So that

$$\frac{1}{h}(f_{i+1} - f_i) = \frac{1}{h}\left(\left[f_i + h f_i' + \frac{h^2}{2!} f_i'' + \frac{h^3}{3!} f_i''' + \cdots \right] - f_i \right)$$

$$= \frac{1}{h}\left(h f_i' + \frac{h^2}{2!} f_i'' + \frac{h^3}{3!} f_i''' + \cdots \right)$$

$$= f_i' + \frac{h}{2!} f_i'' + \frac{h^2}{3!} f_i''' + \cdots$$

So we see that, in the best case where the function f is as differentiable as we like, Taylor's theorem tells us the forward difference at x_i gives us the derivative of f at x_i, f_i', plus a bunch of terms involving powers of the size of the mesh intervals h.

So we have

$$\left(\text{our numerical approximation}\right) = \left(\begin{matrix}\text{what we}\\ \text{want}\end{matrix}\right) + \left(\begin{matrix}\text{terms}\\ \text{involving } h\end{matrix}\right)$$

$$\frac{f_{i+1} - f_i}{h} = f_i' + (h f_i' + \cdots)$$

Defn: the "terms involving h", indicated above, is called the truncation error of the "numerical approximation" term.

Defn: We say that the order of a finite difference / numerical approximation is the lowest power of "h" that appears in the truncation error.

Conclusion: The forward difference method is a first order approximation of the derivative $\frac{\partial f}{\partial x}$ at x_i.

Theory versus Practice:

Theoretically speaking one can determine the order of any finite difference approximation using Taylor analysis. However doing this "in the real world", where you may have a numerical scheme for solving a high-dimension partial diff equation on a non-uniform mesh, is far from practical.

Typically the praxis of determing the order of a numerical scheme is computational and the process is referred to as a "grid convergence study".

Grid Convergence Studies to Determine Order of a Method:

- Assume you have a method of order h^α, where α is unknown, for solving some PDE of the form $Lu = f$ with associated boundary conditions.

- Suppose there exists some $\hat{f}$ for which the true solution $\hat{u}$ is known: e.g. $L\hat{u} = \hat{f}$

- Consider a mesh of uniform size h and let u_h denote the discrete solution, produced by your numerical method for solving "$Lu = f$", on this mesh.

- Then we can define an associated discreet error to u_h as:

$$e_h = \left(\sum_{i=0}^{N} |u(x_i) - u_h(x_i)|^2 \right)^{1/2}$$

Key idea: Assume that the numerical method is order α where α is to be determined. That is $e_h = Kh^\alpha$ where K is some constant. Then if we fix a value of h and compute e_h and $e_{h/2}$ it follows that

$$\frac{e_h}{e_{h/2}} = \frac{Kh^\alpha}{K(\frac{h}{2})^\alpha} = 2^\alpha$$

So that $\log_2\left(\frac{e_h}{e_{h/2}}\right) = \log_2(2^\alpha) = \alpha \log_2(2) = \alpha.$

(*) Keep in mind the standard "log" button on your calculator is log base 10. So you adapt to this by using the Change of base formula $\log_b(x) = \frac{\log_a(x)}{\log_a(b)}$ where $b = 2$ and $a = 10$

OR

instead of computing e_h and $e_{h/2}$ you compute e_h and $e_{h/10}$

So that $$\log_{10}\left(\frac{e_h}{e_{h/10}}\right) = \log_{10}\left(\frac{Kh^\alpha}{K(h/10)^\alpha}\right) = \log_{10}(10^\alpha) = \alpha$$

Note: the computed value of α will not necessarily match the theoretical value. There are many reasons for this. ONE reason can be that there are additional sources of error that have not been accounted for. However a very important observation is that how we measure the error can give different orders, α.

Key idea: The order of a numerical method can depend on the way that error is measured. This is not a "bad" thing (even though it sounds bad) at all - different error measurements can tell you different things about the approximations.

Remark: The error $e_h = \left(\sum_{i=0}^{N} |u(x_i) - u_h(x_i)|^2\right)^{1/2}$ is called "The L_2 error estimate"

Ex: Try estimating the order using the error estimate $e_h = \max_{i=0,1,\ldots,N} |u(x_i) - u_h(x_i)|$

This error measurement is called the "L_∞" or "supremum" error. What do you estimate for the order α if you use the error estimate $e_h = \sum_{i=0}^{N} |u(x_i) - u_h(x_i)|$? This is called the "$L_1$" error estimate.