Preliminaries

Anne Kværnø

Oct 19, 2018

Corresponding Python code: `preliminaries.py`.

1 Introduction

In your mathematical courses so far, you have learned how to solve different mathematical problems, like linear and nonlinear equations and differential equations. You have learned how to differentiate and to integrate functions. Unfortunately, only simplified models from "real life" applications can be treated by these techniques, for more complex and realistic problems the best we can aim for is some kind of an approximate solution, found on a computer by some clever numerical algorithms. In this part of the course, our concern is how to develope, analyse, implement and test a selection of such algorithms.

In this note, we will present some mathematical results that will be used frequently, as well as some definitions and concepts. Most of it should be known from previous courses, in particular from Mathematics 1 and 3.

Whenever some theoretical error analysis has been established, it should be verified numerically. This is done based on a test problem with a known exact solution, so the error in our numerical experiment can be evaluated, and the theory numerically verified. Examples of such verifications will be demonstrated.

2 Vector spaces and norms

Real vector space.

A *real vector space* is a set V together with the operations $+$ (addition) and $\cdot$ (multiplication with a scalar) which for all $x, y, z \in V$ and $\alpha, \beta \in \mathbb{R}$ satisfy the following conditions:

1) $x + y \in V$
2) $x + y = y + x$
3) $x + (y + z) = (x + y) + z$
4) There exist a $0 \in V$ such that $x + 0 = x$
5) For all $x \in V$ there is a $-x \in V$ such that $x + (-x) = 0$
6) $\alpha x \in V$
7) $\alpha(\beta x) = (\alpha\beta)x$
8) $1x = x$
9) $\alpha(x + y) = \alpha x + \alpha y$
10) $(\alpha + \beta)x = \alpha x + \beta x$

Examples: The following vector spaces will be used thoughout the course:

- $\mathbb{R}^m$ is the set of all real vectors with m components.
- $\mathbb{R}^{m\times n}$ is the set of all $m \times n$ real matrices.
- $C^m[a, b]$ is the set of all functions with continuous first m derivatives on the interval $[a, b]$. It is common to use $C[a, b]$ rather than $C^0[a, b]$ for all continuous functions.

- $\mathbb{P}_n$ is the set of all polynomials of degree n or less.

Notice that $C^n[a,b] \subset C^m[a,b]$ if $n > m$. Further, $\mathbb{P}_n \subset C^\infty[\mathbb{R}]$.

Norms. The norm $\|\cdot\|$ of an element x in a vector space V is essentially a measure of the size of the element. The norm obeys the following rules:

Norm $\|\cdot\|$.

For all $x, y \in V$ and $\alpha \in \mathbb{R}$ the following holds

$$\begin{aligned} \|x\| &\geq 0 && \text{and} \quad \|x\| = 0 \quad \Leftrightarrow \quad x \equiv 0 \\ \|\alpha x\| &= |\alpha|\,\|x\|, \\ \|x+y\| &\leq \|x\| + \|y\| && \text{(Triangle inequality).} \end{aligned}$$

Norms for $C[a,b]$ and $\mathbb{R}^m$:

- For $f \in C[a,b]$:

  either $\|f\|_\infty = \max_{x\in[a,b]} |f(x)|$.

  or $\|f\|_2 = \sqrt{\int_a^b f(x)^2 dx}$.

- For $\mathbf{x} \in \mathbb{R}^m$, we will use:

  either $\|\mathbf{x}\|_\infty = \max_{i=1}^m |x_i|$,

  or $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^m x_i^2}$.

Note that we will use bold symbols for vectors $\mathbf{x} \in \mathbb{R}^m$.

As is demonstrated in these examples, the norm to which we refer is usually marked with a subscript. We will always use the absolute value as the norm of a real number, thus $\|x\| = |x|$ whenever $x \in \mathbb{R}$.

Example 1: Let $\mathbf{x} = [1, -6, 3, -1, 5]^T \in \mathbb{R}^5$. Then

$$\begin{aligned} \|\mathbf{x}\|_2 &= \sqrt{1+36+9+1+25} = 8.4853, \\ \|\mathbf{x}\|_\infty &= \max\{1,6,3,1,5\} = 6. \end{aligned}$$

Norms of vectors in $\mathbb{R}^m$ are implemented in Python.

See `norm_of_a_vector()` in `preliminaries.py`.

Example 2: Let $f(x) = \sin(x)$ on $[0, 2\pi]$, so $f \in C^\infty[0, 2\pi]$. In this case

$$\begin{aligned} \|f\|_2 &= \sqrt{\int_0^{2\pi} \sin^2(x) dx} = \sqrt{\pi} = 1.7725 \\ \|f\|_\infty &= \max_{x\in[0,2\pi]} |\sin(x)| = 1. \end{aligned}$$

Norms on function spaces require some sort of numerical approximations. In this case, the interval $[0, 2\pi]$ has been divided into $N = 1000$ uniform subintervals, so $x_i = (2\pi i)/N$, $i = 0, 1, \ldots, N$. The norm $\|f\|_\infty$ is approximated by $\max_i |f(x_i)|$, and the integral required for $\|f\|_2$ is computed by the trapezoidal rule.

See `norm_of_a_function` in `preliminaries.py`.

3 Convergence and errors

Convergence of a sequence.

Let $\{x_k\}_{k=0}^{\infty}$ be an infinite sequence of real numbers. The sequence converges to x, if, for any $\varepsilon > 0$ there exist a positive integer $N(\varepsilon)$ such that $|x_k - x| < \varepsilon$ whenever $k > N(\varepsilon)$.

Common notations:

$$\lim_{k\to\infty} x_k = x \qquad \text{or} \qquad x_k \to x \text{ as } k \to \infty.$$

Example 3: It is known that the sequence $x_k = (1 + 1/k)^k \to e = 2.7182\ldots$ as $k \to \infty$. The sequence is monotone, so $|x_{k+1} - e| < |x_k - e|$. The following program demonstrates the concept of convergence for this sequence: Given an ε, the positive integer $N(\epsilon)$ is returned.

See `convergence_sequence` in `preliminaries.py`. Clearly, this code can only be used for monotone series.

Warning: The sequence converges very slowly, so given a very small `epsilon` will result in N too large. This can be compensated by making `Nmax` larger, and increment k in larger steps.

3.1 Convergence of an iterative process

Let X be the exact solution of a problem, and X_k a numerical approximation achieved by some iterative process $X_{k+1} = G(X_k)$. In this case the iterations converge towards X if

$$\lim_{k\to\infty} \|X - X_k\| = 0$$

Let $e_k = \|X - X_k\|$ measure the error. In practice, you have to choose an appropriate norm, which depends on the problem and what you might be interesting in measuring. The order of convergence is p if there exist a positive constant M such that

$$e_{k+1} \leq M e_k^p$$

Notation: The case $p = 1$ is called *linear* convergence, $p = 2$ is called *quadratic* convergence and $p = 3$ *cubic* convergence.

Numerical verification of the order. To verify the order, we make the assumptions that $e_{k+1} = C_k e_k^p$, and that the C_k do not change much from one iteration to the next one. This assumption is usually reasonable when the error becomes small. The order p can then be computed numerically by the following procedure: Take the expressions for the error for two subsequent iterations, assuming that $C_{k+1} \approx C_k \approx C$. Then divide the expression by each other to get rid of the unknown constant C, take the logarithm on both sides and solve for the order p.

$$\begin{array}{rcl} e_{k+2} & \approx & C e_{k+1}^p \\ e_{k+1} & \approx & C e_k^p \end{array} \quad \Rightarrow \quad \frac{e_{k+2}}{e_{k+1}} \approx \left(\frac{e_{k+1}}{e_k}\right)^p \quad \Rightarrow \quad \log\left(\frac{e_{k+2}}{e_{k+1}}\right) \approx p \log\left(\frac{e_{k+1}}{e_k}\right) \quad \Rightarrow \quad p \approx \frac{\log\left(e_{k+2}/e_{k+1}\right)}{\log\left(e_{k+1}/e_k\right)}$$

We are usually not so interested in the constant C, but given p and the error for two iterations, this can easily be approximated.

Example 4: Newton's method applied to the equation $f(x) = 0$ is given by

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}.$$

Let r be a solution of the equation. It can be proved that the error $r - x_k$ satisfies

$$r - x_{k+1} = -\frac{f''(\xi_k)}{2f'(x_k)}(r - x_k)^2, \qquad \text{where } \xi_k \text{ a real number between } x_k \text{ and } r.$$

Since the error in this case is a real number, its norm is $e_k = |r - x_k|$ and

$$e_{k+1} = C_k e_k^2 \quad \text{where} \quad C_k = \frac{|f''(\xi_k)|}{2|f'(x_k)|}.$$

Notice that $C_k \to |f''(r)|/(2|f'(r)|$ as $x_k \to r$.

The constant M is given by the upper bound of C_k. Or more precisely: let $I_\delta = [x - \delta, x + \delta]$ be some interval around the solution r. Assume there exists constants L and K such that $|f'(x)| \leq L$ and $|f''(x)| \geq K$ for all $x \in I_\delta$. Then $M = K/(2L)$ and

$$e_{k+1} \leq M e_k^2.$$

The convergence is quadratic, and the iterations converge for all starting values x_0 chosen such that

$$M e_0 < 1,$$

often described as "sufficiently close to the solution".

Let us now verify the theoretical result by applying Newtons method to the problem $x^2 - a = 0$ for some $a > 0$. The iterations become

$$x_{k+1} = x_k - \frac{x_k^2 - a}{2x_k} = \frac{x_k^2 + a}{2x_k}, \qquad k = 0, 1, 2, \ldots .$$

with the exact solution $r = \sqrt{a}$. From the discussion above, we expect $C \approx |f''(r)/(2f'(r)|$ with $r = \sqrt{a}$, which in our case becomes $C \approx 1/(2\sqrt{a})$. Use the following code to see if the theoretical considerations hold in practice:

See `ooc_iterations()` in `preliminaries.py`.

3.2 Convergence of h-dependent approximations

Let X be the exact solution, and $X(h)$ some numerical solution depending on a parameter h, and let $e(h)$ be the norm of the error, so $e(h) = \|X - X(h)\|$. The numerical approximation $X(h)$ converges to X if $e(h) \to 0$ as $h \to 0$. The order of the approximation is p if there exists a positive constant M such that

$$e(h) \leq M h^p$$

The Big $\mathcal{O}$-notation: A function $f(x) = \mathcal{O}(g(x))$ as $x \to a$ if and only there exist positive numbers δ and M such that

$$|f(x)| \leq M|g(x)| \qquad \text{when} \qquad 0 < |x - a| < \delta.$$

Let $a = 0$ and $f(h) = e(h)$, thus the error of an approximation of order p can be written as

$$E(h) = \mathcal{O}(h^p).$$

This is often used when we are not directly interested in any expression for the constant M, we only need to know it exists.

Numerical verification. The following is based on the assumption that $e(h) \approx Ch^p$ for some unknown constant C. This assumption is usually reasonable for sufficiently small h.

Choose a test problem for which the exact solution is known and compute the error for a sequence of smaller h's, for instance $h_k = H/2^k$, $k = 0, 1, 2, \ldots$. The procedure is then quite similar to what was done for iterative processes.

$$\begin{array}{rcl} e(h_{k+1}) & \approx & Ch_{k+1}^p \\ e(h_k) & \approx & Ch_k^p \end{array} \qquad \Rightarrow \qquad \frac{e(h_{k+1})}{e(h_k)} \approx \left(\frac{h_{k+1}}{h_k}\right)^p \qquad \Rightarrow \qquad p \approx \frac{\log\left(e(h_{k+1})/e(h_k)\right)}{\log\left(h_{k+1}/h_k\right)}$$

Since

$$e(h) \approx Ch^p \qquad \Rightarrow \qquad \log e(h) \approx \log C + p \log h$$

a plot of $e(h)$ as a function of h using a logarithmic scale on both axes will be a straight line with slope p. Such a plot is referred to as an *error plot* or a *convergence plot.*

Some terminology. Let X be the exact solution of some problem, and $X(h)$ the numerical approximation of that problem. The following concepts are of interest:

- The error $E(h)$: $E(h) = X - X(h)$. This is something which obviously is only known if the exact solution is known (which it will be in our test problems, but not in real life problems). Still, most error analysis will start trying to find an expression for this error, but it will typically contain some higher order derivatives evaluated in some unknown point.
- The error bound: Typically of the form $\|X - X(h)\| \leq Mh^p$. If M is known, this can be used to decide how small h has to be to guarantee that the error is below some tolerance.
- Error estimate $\mathcal{E} \approx E$. This is an approximation to the error, and something that can be computed, and included in practical codes. How to compute those will be described for each problem we will discuss later in the course.

Example 5: Consider the trapzoidal rule for numerical integration. It is known that

$$\int_a^b f(x)dx = T(h) + E(h)$$

where $T(h)$ is the numerical approximation given by

$$T(h) = h\left(\frac{1}{2}f(x_0) + \sum_{i=0}^{n} f(x_i) + \frac{1}{2}f(x_n)\right), \qquad x_n = a + ih, \quad h = \frac{b-a}{n},$$

and the error $E(h)$ is known to be

$$E(h) = -\frac{b-a}{12}f''(\xi)h^2, \qquad \xi \in (a,b).$$

Assume there exist an M such that $|f''(x)| \leq M$ for all $x \in (a,b)$. Let $e(h) = |E(h)|$ (notice that $E(h)$ is a scalar) so

$$e(h) \leq Mh^2.$$

So the error of the trapezoidal rule is of order 2, and $E(h) = \mathcal{O}(h^2)$.

Use this to verify the order of the trapezoidal rule, as given above. As test example, choose

$$\int_0^\pi \sin(x)dx = 2.$$

In this case, we expect the order to be p. The constant $C = |f''(\xi)|\pi/12$ for some unknown $\xi \in [0,\pi]$. Thus $0 < C < \pi/12 = 0.2617\ldots$, but we can not be more precise. The upper bound for the error is $e(h) \leq \pi/12\, h^2$.

The following code can be used to confirm the result. It also returns an approximation to C, so we can at least check if it is within the expected bound. See `ooc_h()` in `preliminaries.py`.

4 Taylor-expansions

Given a function $f \in C^\infty[a,b]$. Choose a point x and an increment h such that $x, x+h \in [a,b]$. The Taylor expansion of f around x is then given by

$$f(x+h) = \sum_{k=0}^{\infty} \frac{h^k}{k!} f^{(k)}(x).$$

The function is called analytic in x if the series converges for sufficiently small values of $|h|$. In numerics, we will usually work with the trunctated series, also called the Taylor polynomial:

$$f(x+h) = \sum_{k=0}^{m} \frac{h^k}{k!} f^{(k)}(x) + R_{m+1}(x)$$

where the remainder term is given by

$$R_{m+1}(x) = \frac{h^{m+1}}{(m+1)!} f^{(m+1)}(\xi)$$

where ξ is some unknown point between x and $x+h$. The truncated series only require $f \in C^{m+1}$ near x. The truncated expansion will often simply be written as

$$f(x+h) = \sum_{k=0}^{m} \frac{h^k}{k!} f^{(k)}(x) + \mathcal{O}(h^{m+1}).$$

5 Some other useful results

Result 1: Let $f \in C[a,b]$ and let u be some number between $f(a)$ and $f(b)$, then there exist at least one $\xi \in (a,b)$ such that $f(\xi) = u$.

Result 2: Let $f \in C[a,b]$. Given k nodes $x_i \in [a,b]$ and k positive weights $w_i > 0$, $i = 1, \ldots, k$. Then there exists at least one $\xi \in (a,b)$ such that

$$\sum_{i=1}^{k} w_i f(x_i) = f(\eta) \sum_{i=1}^{k} w_i. \tag{1}$$

Result 3: Let $f \in C^1[a,b]$. Then there exists at least one $\xi \in (a,b)$ such that

$$f'(\xi) = \frac{f(b) - f(a)}{b-a} \tag{2}$$

Result 4: (Rolle's theorem) Let $f \in C^1[a,b]$ and $f(a) = f(b) = 0$. Then there exists at least one $\xi \in (a,b)$ such that $f'(\xi) = 0$.