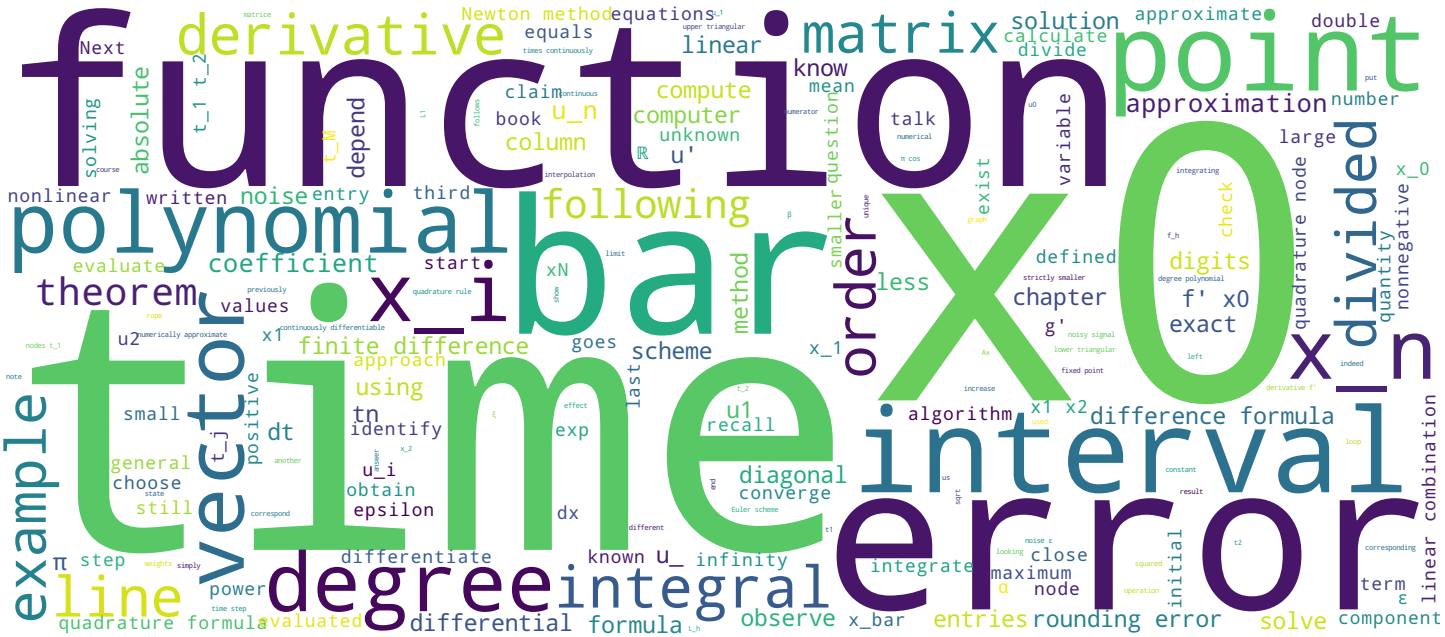
Dérivées numériques d'ordre 1 – Erreur d'arrondis

Introduction à l'analyse numérique

Prof. Marco Picasso



Video

EPFL

Dérivée d'ordre 1 ; erreur d'arrondis

$u(x) = 0 + \varepsilon \sin k\pi x$

$u'(x) = \varepsilon k\pi \cos k\pi x$

$\int u(x) = -\frac{\varepsilon}{k\pi} \cos k\pi x$

double c = 1.0/3.0

$\dot{u}(t) = f(t)$ $\quad u(t) = u(0) + \int_0^t f(s)ds$

Now we'll talk about the round-off error. In general, differentiating a noisy signal increases this noise an integrating a noisy signal lowers the noise. Why is that? For example, take a function u(x) which is the zero function, with for some reason a little noise ε*sin(k*π*x) with k large. If you calculate the amplitude of the noise it's ε on this function Now if we differentiate, we get u'(x) = ε*k*π*cos(k*π*x) So the noise of the derivative is ε*k*π so the noise increased since k is large. On the other hand, if we integrate, a primitive of u(x) is - ε/(k*π)*cos(k*π*x). This time the noise ε was divided by k*π k is large so the noise got smaller. In general, when solving a differential equation so u'(t)-=f(t) then we do not differentiate, we integrate u(t) is u(0) plus the integral from 0 to t of f(s)ds. Luckily when we solve a differential equation or a partial differential equation we integrate something, so the noise is lowered. So that does not bring problems. On the other side, when differentiating something, a numerical signal, we increase the noise. How can we formalize these lines? So declare a double, c, equal to a third.

Notes

Summary

0m 04s

Dérivée d'ordre 1 ; erreur d'arrondis

$u(x) = 0 + \varepsilon \sin k\pi x$ $\quad \dot{u}(t) = f(t)$ $\quad u(t) = u(0) + \int_0^t f(s)\,ds$

$u'(x) = \varepsilon k\pi \cos k\pi x$

$\int u(x) = -\frac{\varepsilon}{k\pi} \cos k\pi x$

double $c = 1.0/3.0$; $\quad c = 1/3$ $\quad \bar{c} = 0,\underbrace{333\ldots333}_{16 \text{ chiffres}}$ $\quad |c - \bar{c}| = \frac{1}{3} 10^{-16} = |c|\, 10^{-N}$ N chiffres sign.

erreur $\frac{f(x_0+h) - f(x_0)}{h}$?

erreur d'arrondis en évaluer $f(x_0)$: $\simeq |f(x_0)|\, 10^{-N}$

$f(x_0+h)$: $\simeq |f(x_0+h)|\, 10^{-N} \simeq |f(x_0)|\, 10^{-N}$

$\frac{f(x_0+h) - f(x_0)}{h} \simeq 2 \frac{|f(x_0)|\, 10^{-N}}{h}$

Si $h \gg 2|f(x_0)|\, 10^{-N}$ pas erreur d'arrondi

in a program written for example in C, a third is the exact value of c and its approximation on a computer with N digits, say c bar, with 16 digits, this is the case when using a double, c bar is 0.333 and so on there are here 16 3. On a computer with 16 digits, the error |c-c bar| will approximately be (1/3) * 10^-16 that is to say, the exact value of c, a third, times 10^-N, given a computer with N digits. So now let's check what is the error, I mean the rounding error, when using a finite difference formula? For example when using (f(x0 + h)- f(x0))/h to approximate the derivative f'(x0). This rounding error when evaluating f(x0) is approximatively f(x0) in absolute value times 10^-N where N is the number of digits, usually 16. If I want to evaluate f(x0+h) the error will be roughly f(x0+h)*10^-N but h is supposed small, so it is close to f(x0)*10^-N. So the rounding error to evaluate the finite difference formula (f(x0 + h)- f(x0))/h is more or less 2*f(x0)*10^-N / h. Observe that when h gets close to the value of the numerator, here f(x0)*10^-N. First, if h is a lot greater than 2*f(x0)*10^-N we do not observe rounding errors which is the general case since recall that N is 16 so 2*f(x0)*10^-16 is in very small in general. But if h is close to the order of this quantity, then we'll observe the effect of rounding errors.

Notes

Summary

2m 02s