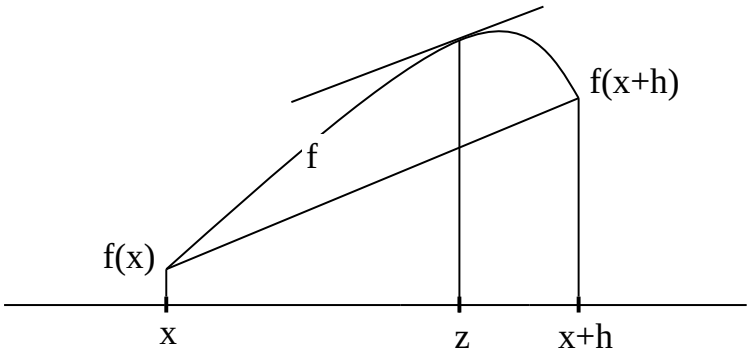
Taylor Series and Numerical Cancellation

The Mean Value Theorem

The **mean value theorem** (**MVT**): Let $f(x)$ be differentiable. For **some** z in $[x, x+h]$:

$$f'(z) = \frac{f(x+h) - f(x)}{h}.$$

This is *intuitively* clear from:



We can rewrite the MV formula as:

$$f(x+h) = f(x) + hf'(z).$$

A generalization if f is **twice** differentiable is

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2} f''(z),$$

for some z in $[x, x+h]$.

Taylor Series

Taylor's Theorem: Let f have continuous derivative of order $0, 1, \ldots, n+1$, then

$$f(x+h) = \sum_{k=0}^{n} \frac{f^{(k)}(x)}{k!} h^k + E_{n+1},$$

where the error (remainder) $E_{n+1} = \frac{f^{(n+1)}(z)}{(n+1)!} h^{n+1}$, $z \in [x, x+h]$.

Taylor series. If f has **infinitely** many derivatives,

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!} f''(x) + \frac{h^3}{3!} f'''(x) + \cdots,$$

where we assume $\lim_{n\to\infty} E_{n+1} = 0$.

Example.

$$\sin(x+h) = \sin(x) + h\sin'(x) + \frac{h^2}{2!}\sin''(x) + \frac{h^3}{3!}\sin'''(x) + \frac{h^4}{4!}\sin''''(x) + \cdots,$$

Letting $x = 0$, we get

$$\sin(h) = h - \frac{h^3}{3!} + \frac{h^5}{5!} - \cdots.$$

Numerical Approximation to $f'(x)$

Recall $f'(x)$ is the **slope** of the line tangent to the graph of f at $(x, f(x))$. If h is **small**, $f'(x)$ is a good approximation to the **slope** of the line through $(x, f(x))$ and $(x+h, f(x+h))$. We write

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}.$$

the **forward difference** approximation.

How **good** is this approximation? By the **Taylor theorem**:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2} f''(z).$$

Then

$$\frac{f(x+h) - f(x)}{h} - f'(x) = \frac{h}{2} f''(z).$$

The left hand side of the above equality is referred to as the **discretization error**: the difference between what we want and our approximation. We say the discretization error is $O(h)$.

Computing the Approximation to f'(x)

```
main()  /* diff1.c: approx. deriv. */
{int n; double x,h,approx,exact,error;
 x=1.0; h=1.0; n=0;
 printf("\n h exact approx error");
 while(n<20) {
  n++;
  h=h/10;      /* h=10^(-n) */
  approx=(sin(x+h)-sin(x))/h;/*app.deriv.*/
  exact=cos(x);   /*exact derivative */
  error=approx - exact; /*disczn error */
  printf("\n ... ,h,exact,apprx,err);
 }
}
```

We want to approximate the deriv of $f(x) = \sin(x)$ at $x = 1$. Knowing $\sin'(x) = \cos(x)$, we can compute the **exact** discretization error. The program uses **double precision**, and displays

$$(\sin(x+h) - \sin(x))/h,$$

with the **error**, for h from 0.1 to 10^{-20}.

Convergence of Approximation

```
h       exact        approx       error
e-03 5.403023e-01 5.398815e-01 -4.20825e-04
e-04 5.403023e-01 5.402602e-01 -4.20744e-05
```

```
e-05 5.403023e-01 5.402981e-01 -4.20736e-06
e-06 5.403023e-01 5.403019e-01 -4.20746e-07
e-07 5.403023e-01 5.403023e-01 -4.18276e-08
e-08 5.403023e-01 5.403023e-01 -2.96988e-09
e-09 5.403023e-01 5.403024e-01  5.25412e-08
e-10 5.403023e-01 5.403022e-01 -5.84810e-08
e-11 5.403023e-01 5.403011e-01 -1.16870e-06
e-12 5.403023e-01 5.403455e-01  4.32402e-05
e-13 5.403023e-01 5.395684e-01 -7.33915e-04
e-14 5.403023e-01 5.440093e-01  3.70697e-03
e-15 5.403023e-01 5.551115e-01  1.48092e-02
e-16 5.403023e-01 0.000000e+00 -5.40302e-01
```

Discretization error reduced by ~ 10 when h is reduced by 10, so the error **is** $O(h)$. But when h gets **too** small, the approximation starts to get **worse!**
Q: Why?

Explanation of Accuracy Loss

If $x = 1$, and $h < \epsilon/2 \approx 10^{-16}$, $x + h$ has the **same numerical value** as x, and so $f(x + h)$ and $f(x)$ **cancel** to give **0**: the answer has **no digits of precision**.

When h is a **little** bigger than $\epsilon/2$, the values **partially cancel**. For example, suppose that the first 10 digits of $f(x + h)$ and $f(x)$ are the same. Then, even though $\sin(x + h)$ and $\sin(x)$ are **accurate to 16 digits**, the **difference** has only **6 accurate digits**. This phenomenon is called **numerical cancellation** (catastrophic cancellation, or cancellation).

In summary, using h **too big** means a big **discretization** error, while using h **too small** means a big **cancellation** error. For the function $f(x) = \sin(x)$, for example, at $x = 1$, the best choice of h is about 10^{-8}, or $\sim \sqrt{\epsilon}$.

Numerical Cancellation

Numerical cancellation occurs when one number is subtracted from another number that is nearly equal to it. It is one of the main causes for deterioration in accuracy in computations. Here we give an analysis to show why it can cause severe loss of accuracy.

In a computation, usually operands have some errors. Instead of the correct values x and y, the computer works with two perturbed floating point numbers:

$$\hat{x} = x(1 + \delta_1), \qquad \hat{y} = y(1 + \delta_2),$$

where the relative errors δ_1 and δ_2 may be due to previous computations, physical experiments and/or rounding. Suppose we want to compute $x - y$. But we can only compute $\hat{x} - \hat{y}$. The computed value of $\hat{x} - \hat{y}$ is $(\hat{x} - \hat{y})(1 + \delta_3)$ with $|\delta_3| < \epsilon$ for any rounding mode. Is it a good approximation to $x - y$? Let us look at the relative error:

$$\begin{aligned}\left|\frac{(\hat{x} - \hat{y})(1 + \delta) - (x - y)}{x - y}\right| &= \left|\frac{x(1 + \delta_1)(1 + \delta_3) - y(1 + \delta_2)(1 + \delta_3) - (x - y)}{x - y}\right| \\ &= \left|\delta_3 + \frac{x}{x - y}\delta_1 + \frac{x}{x-y}\delta_1\delta_3 + \frac{y}{x-y}\delta_2 + \frac{y}{x-y}\delta_2\delta_3\right|.\end{aligned}$$

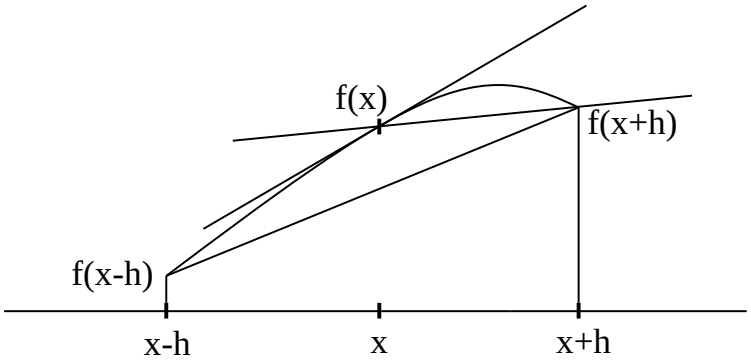
This suggests that when $|x-y| \ll |x|, |y|$, it is very likely that the relative error is very large even if $|\delta_1|$ and $|\delta_2|$ are very small. Thus, in numerical computing, we would like to avoid numerical cancellation if possible.

Example. The quadratic equation $ax^2 + bx + c = 0$ has two roots:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}, \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}.$$

If $|b| \gg |a|, |c|$, will there be any difficulty using these expressions to compute x_1 and x_2 accurately? If so, what is it and why, and how to avoid the problem? (Don't consider overflow or underflow.)

More Accurate Numerical Differentiation



As h **decreases**, the line through $(x-h, f(x-h))$ and $(x+h, f(x+h))$ gives a **better approximation** to the **slope of the tangent** to f at x than the line through $(x, f(x))$ and $(x+h, f(x+h))$. This observation leads to the approximation:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h},$$

the **central difference** formula. In the following we analyze the approximation error.

By the **Taylor theorem**, we have

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(z_1),$$

where z_1 is between x and $x+h$. Similarly,

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f'''(z_2),$$

where z_2 is between x and $x-h$.

Subtracting the 2nd from the 1st:

$$\frac{f(x+h) - f(x-h)}{2h} = f'(x) + \frac{h^2}{12}(f'''(z_1) + f'''(z_2))$$

So the **discretization error** is

$$\frac{h^2}{12}(f'''(z_1) + f'''(z_2)),$$

which is $O(h^2)$ instead of $O(h)$. When h is small enough, $O(h^2)$ will be (much) smaller than $O(h)$.