chngcntr

9 Alg. de Horner: Polynomial evaluation

Consider

$$p(x) = \sum_{i=0}^{n} a_i x^i$$

We wish to compute $p(k)$ for a given $k \in \mathbb{R}$ minimizing the number of operations. Directly computing $a_0 + a_1 k_1 + \ldots$ leads to n sums. The ith term requires computing k^i, which means i product operations, for a totall of $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ products. The total number of operations is then

$$\Theta = n + n(n+1)/2$$

The associated complexity is $O(n^2)$.

Horner's method consists of re-writing $p(x)$ so that the number of products is reduced. One writes

$$p(x) = a_0 + xb_0$$

where $b_{n-1} = a_n$ and for $0 \leq i < n-1$:

$$b_{i-1} = a_i + xb_i$$

Let $p(x) = 3 + 5x - 4x^2 + 0x^3 + 6x^4$, giving $n = 4$. Then $b_3 = 6$ and

$$b_2 = a_3 + xb_3 = 6x, \qquad b_1 = a_2 + xb_2 = -4 + x(6x),$$
$$b_0 = a_1 + xb_1 = 5 + x(-4 + x(6x))$$

This finally gives

$$p(x) = 3 + xb_0 = 3 + x(5 + x(-4 + x(6x)))$$

Here, one must perform n sums again but only n products. Thus, there are $\Theta = n + n = 2n$ operations, giving a complexity of $O(n)$ (in the operation space). See the algorithm below:

$$
\begin{aligned}
&\textbf{input } n; a_i, i = 0, \ldots, n; x \\
&b_{n-1} \leftarrow a_n \\
&\textbf{for } i = n - 2 \textbf{ to } i = 0 \\
&\qquad b_i = a_{i+1} + x * b_{i+1} \\
&\textbf{od} \\
&y \leftarrow a_0 + x * b_0 \\
&\textbf{return } y
\end{aligned}
$$

It is easy to see in this code that the **for** loop performs $n - 1$ iterations, in each of which a single sum and a single product are computed. The nth sum and nth product are performed in the computation of y, the final result.

A more polished version includes the last computatoin (the one in the assignment of y) within the loop and makes no use of indexes:

$$
\begin{aligned}
&\textbf{input } n; a_i, i = 0, \ldots, n; x \\
&b \leftarrow a_n \\
&\textbf{for } i = n - 2 \textbf{ to } i = -1 \\
&\qquad b = a_{i+1} + x * b \\
&\textbf{od} \\
&\textbf{return } b
\end{aligned}
$$

In Python,

```
def horner(coefs, x):
  n = len(coefs)-1
  b = coefs[n]

  for i in reversed(range(-1, n-1)):
    b = coefs[i+1] + x*b

  return b
```

It is trivial to adapt the code so that it returns the coefficients $b_0, \ldots, b_{n-1}$ and not the final result, if needed.

10 Error

Let $r, \overline{r}$ be two real numbers s.t. the latter is an approximation of the first. We define the **error** of the approximation to be $r - \hat{r}$, and

$$\Delta r = |r - \overline{r}|, \qquad \delta r = \frac{\Delta r}{|r|}$$

With r unknown the strategy is to work with a known bound of r.

11 Non-linear equations

The general problem is to find members of the set $\mathcal{R}_f$ of roots of $f \in \mathbb{R} \to \mathbb{R}$. The numerical strategy is to iteratively approximate some $r \in \mathcal{R}_f$ until some pre-established threshold in the error of approximation is met.

More formally, the numerical strategy produces a sequence $\{x_k\}_{k\in\mathbb{N}}$ which satisfies

- $\lim_{k\to\infty} \{x_k\} = r$ for some $r \in \mathcal{R}_f$
- Either $e(x_k) < e(x_{k-1})$ or, more strongly, $\lim_{k\to\infty} e(x_k) = 0$, where $e(x_k)$ is some appropriate measure of the error of approximation.

11.1 Bisection

A very simple procedure: if a root exists in $[a, b]$, it iteratively shrinks $[a, b]$ in halves (keeping the halves which contain the root) until the interval is of sufficiently small length.

Theorem 1 (Intermediate value)**.** If f is continuous in $[a, b]$ and $f(a)f(b) < 0$, then $\exists r \in \mathcal{R}_f$ s.t. $r \in [a, b]$.

Assume f is continuous. A root exists in $[a, b]$ if $f(a)f(b) < 0$ (**Theorem 1**). If that is the case, the midpoint $(a + b)/2$ is taken as the approximation x_0. It is also trivial to observe that x_0 is *at most* at a distance of $(b - a)/2$ from the real root, so $e_0 = |x_0 - r| \leq (b - a)/2$.

If $f(x_0) = 0$ the procedure must end because a root was found. Otherwise, sufficies to find which half of the interval contains a root computing $f(a)f(c)$ and, if needed, $f(c)f(b)$.

The iterations may stop after reaching a maximum number of steps, when $|f(c)|$ is sufficiently close to zero, or when the error bound $|e_k| \leq (b_k - a_k)/2$ (where $[a_k, b_k]$ is the interval of this iteration) is sufficiently small.

(!) The algorithm not always converges. Take $f(x) = 1/x$. Clearly, it has no root. Yet setting $a = -1, b = 1$ in the initial iteration falsely passes the test. (The problem obviously is that f is not continuous in $[-1, 1]$.) If one sets

```
Input : a, b, δ, M, f
Output : Tupla de la forma: (r, cota de error)
f_a ← f(a)
f_b ← f(b)

if f_a * f_b > 0
    return ?
fi

for i = 1 to i = M do
    c ← a + (b − a)/2
    f_c ← f(c)
    if f_c = 0 then
        return (c, 0)
    fi
    ϵ = (b − a)/2
    if ϵ < δ then
        break
    fi
    if f_a * f_c < 0 then
        b ← c
        f_b = f(b)
    else
        a ← c
        f_a = f(a)
    fi
od
return (c, ϵ)
```

```
def bisection(f : callable, a : float, b : float, delta : float, M : int):

  s, e = f(a), f(b) # function values at (s)tart, (e)nd of interval

  if s*e > 0:
    raise ValueError("Interval [a, b] contains no root.")

  for i in range(M):

    c = a + (b-a)/2
    m = f(c) # value of f at (m)idpoint

    if m == 0:
      return c, 0

    e = (b-a)/2
    if e < delta:
      return c, e

    if s*m < 0:
      b = c
      e = f(b)
    else:
      a = c
      s = f(a)

  return c, e
```

Theorem 2. If $\{[a_i, b_i]\}_{i=0}^{\infty}$ are the intervals generated by the bisection method on iterations $i = 0, 1, \ldots$, then:

1. $\lim_{n\to\infty} a_n = \lim_{n\to\infty} b_n$ is a member of $\mathcal{R}_f$.
2. If $c_n = \frac{1}{2}(a_n + b_n), r = \lim_{n\to\infty} c_n$, then $|r - c_n| \leq \frac{1}{2^{n+1}}(b_0 - a_0)$

Proof. **(1)** It is clear that $a_i \leq a_{i+1}$ and $b_i \geq b_{i+1}$, since the interval on each iteration shrinks in one direction.

$\therefore$ a_n, b_n are monotonous.

But clearly a_n is bounded by b_0 and b_n is bounded by a_0.

$\therefore$ a_n, b_n are monotonous and bounded.

$\therefore$ Their limits exist.

It is also clear that the interval shrinks to half its size on each iteration:

$$b_n - a_n = \frac{1}{2}(b_{n-1} - a_{n-1}), \qquad n \geq 1 \tag{1}$$

By recurrence on (1),

$$b_n - a_n = \frac{1}{2^n}(b_0 - a_0), \qquad n \geq 0 \tag{2}$$

Then

$$\lim_{n\to\infty} a_n - \lim_{n\to\infty} b_n = \lim_{n\to\infty} (a_n - b_n) = \lim_{n\to\infty} \frac{1}{2^n}(b_0 - a_0) = 0 \tag{3}$$

$\therefore$ $\lim_{n\to\infty} a_n = \lim_{n\to\infty} b_n$.

Since the limit of a_n, b_n exists and f is by assumption continuous, the composition limit theorem applies and:

$$\begin{aligned}
&\lim_{n\to\infty} (f(a_n) \cdot f(b_n)) \\
&= \lim_{n\to\infty} f(a_n) \cdot \lim_{n\to\infty} f(b_n) && \{\text{Product of limits}\} \\
&= f\left(\lim_{n\to\infty} a_n\right) \cdot f\left(\lim_{n\to\infty} b_n\right) && \{\text{Composition limit theorem}\} \\
&= [f(r)]^2 && \left\{r = \lim_{n\to\infty} a_n\right\}
\end{aligned} \tag{4}$$

The invariant of the algorithm is $f(a_n)f(b_n) < 0$. But due to the last result,

$$\lim_{n\to\infty} f(a_n)f(b_n) \leq 0 \iff [f(r)]^2 \leq 0 \iff f(r) = 0$$

$\therefore$ $r = \lim_{n\to\infty} a_n = \lim_{n\to\infty} b_n$ is a root.

(2) Follows directly from result (2)

$$
\begin{aligned}
|r - c_n| &= \left| r - \frac{1}{2}(b_n - a_n) \right| \\
&\leq \left| \frac{1}{2}(b_n - a_n) \right| \\
&= \left| \frac{1}{2^{n+1}}(b_0 - a_0) \right| && \{\text{Result (2)}\}
\end{aligned}
$$

11.2 Newton's method

Taylor: repasito. El desarrollo de una f suficientemente diferenciable alrededor de un punto r espa

$$f(x) = f(r) + f'(r)(x-r) + \frac{f''(r)}{2!}(x-r)^2 + \ldots + \frac{f^{(n)}(r)}{n!}(x-r)^n + R_n(x)$$

donde $R_n(x)$ es el resto.

Usualmente, queremos tomar $r = x + h$, donde x es una aproximación de r y h el error de aproximación. Entonces es provechoso expandir $f(r)$ alreededor de su estimación x:

$$f(r) = f(x+h) = f(x) + f'(x)h + \frac{f''(x)}{2!}h^2 + \ldots + \frac{f^{(n)}(x)}{n!}h^n + R_n(h)$$

Esto es **recontra** útil porque nos dice cuánto se diferencia $f(r)$ de nuestra aproximación $f(x)$ (pues expresa $f(r)$ como $f(x)$ más algo).

Usualmente r, h son desconocidos pero h puede acotarse.

El resto R_n del teorema puede expresarse como sigue:

$$f(r) = f(x+h) = f(x) + f'(x)h + \frac{f''(x)}{2!}h^2 + \ldots + \frac{f^{(n)}(x)}{n!}h^n + \frac{f^{(n+1)}(\zeta)}{(k+1)!}h^{n+1}$$

para algún $\zeta \in (x, h)$. Esta forma de expresar el error de aproximación con el polinomio de Taylor se usará mucho.

Assume $r \in \mathcal{R}_f$ and $r = x + h$, with x an approximation of r and h its error. Assume f'' exists and is continuous in some I around x s.t. $r \in I$. What we explained on Taylor expansions around a point gives:

$$0 = f(r) = f(x+h) = f(x) + f'(x)h + O(h^2)$$

If x is sufficiently close to r, h is small and h^2 even smaller, so that $O(h^2)$ is unconsiderable:

$$0 \approx f(x) + hf'(x)$$

Therefore,

$$h \approx -\frac{f(x)}{f'(x)} \tag{1}$$

From this follows that $r = x + h$ is approximated by

$$r \approx x - \frac{f(x)}{f'(x)}$$

Since the approximation in (5) truncated the terms of $O(h^2)$ complexity, this new approximation is closer to r than x originally was. In other words, $x - f(x)/f'(x)$ is a better approximation to r than x itself.

Thus, if x_0 is an original approximation, we can define

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{2}$$

to produce a sequence of approximations. This is the fundamental idea of Newton's method.

Input: x_0, M, δ, ϵ;
$v \leftarrow f(x_0)$
if $|v| < \epsilon$ **then return** x_0 **fi**
for $k = 1$ **to** $k = M$ **do**
    $x_1 \leftarrow x_0 - \dfrac{v}{f'(x_0)}$
    $v \leftarrow f(x_1)$
    if $|x_1 - x_0| < \delta \vee v < \epsilon$ **then**
        return x_1
    fi
    $x_0 \leftarrow x_1$
od
return x_0

The predicate $|x_1 - x_0| < \delta$ checks whether our algorithm is adjusting x in a negligible degree. If that is the case, we should stop.

Theorem 3. If f'' continuous around $r \in \mathcal{R}_f$ and $f'(r) \neq 0$, then there is some $\delta > 0$ s.t. if $|r - x_0| \leq \delta$, then:

- $|r - x_n| \leq \delta$ for all $n \geq 1$.
- $\{x_n\}$ converges to r
- The convergence is quadratic, i.e. there is a constant $c(\delta)$ and a natural N s.t. $|r - x_{n+1}| \leq c\,|r - x_n|^2$ for all $n \geq N$.

Proof. Let $e_n = r - x_n$ be the error in the nth approximation. Assume f'' is continuous and $f(r) = 0$, $f'(r) \neq 0$. Then

$$\begin{aligned} e_{n+1} &= r - x_{n+1} \\ &= r - \left(x_n - \frac{f(x_n)}{f'(x_n)} \right) \\ &= r - x_n + \frac{f(x_n)}{f'(x_n)} \\ &= \frac{e_n f'(x_n) + f(x_n)}{f'(x_n)} \end{aligned} \tag{3}$$

Thus, the error at any given iteration is a function of the error at the previous iteration. Now consider the expansion of $f(r)$ as

$$f(r) = f(x_n - e_n) = f(x_n) + e_n f'(x_n) + \frac{e_n^2 f''(\zeta_n)}{2} \tag{4}$$

for ζ_n between x_n and r. This equation gives

$$e_n f'(x_n) + f(x_n) = -\frac{1}{2} f''(\zeta_n) e_n^2 \tag{5}$$

The expression in (5) is the numerator in (3), whereby we obtain via substitution:

$$e_{n+1} = -\frac{1}{2} \frac{f''(\zeta_n) e_n^2}{f'(x_n)} \tag{6}$$

Equation (6) ensures that the error scales quadratically. Now we wish to bound the error expression in (6). To bound e_{n+1}, we take $\delta > 0$ to define a neighbourhood of length δ around r. For any x in this neighbourhood, (6) reaches its maximum when the numerator is maximized and the denominator is minimized:

$$c(\delta) = \frac{1}{2} \frac{\max_{|x-r| \leq \delta} |f''(x)|}{\min_{|x-r| \leq \delta |f'(x)|}}$$

In other words, $c(\delta)$ is the maximum value which e_{n+1} can take if ζ_n, x_n are assumed to belong to the neighbourhood. Now we make two assumptions:

1. x_0 belongs to the neighbourhood, i.e. $|x_0 - r| \leq \delta$
2. δ is sufficiently small so that $\varrho := \delta c(\delta) < 1$.

Note that, since ζ_0 is between x_0 and r, assumption (1) ensures that ζ_0 is also in the neighbourhood, i.e. $|r - \zeta_0| \leq \delta$. Then we have:

$$|e_0| = \frac{1}{2} \, |f''(\zeta_0) / f'(x_0)| \leq c(\delta)$$

Then:

$$
\begin{aligned}
|x_1 - r| &= |e_1| \\
&= \left| e_0^2 \cdot \frac{1}{2} f''(\zeta_0)/f'(x_0) \right| \\
&\le |e_0^2| c(\delta) && \left\{ \frac{1}{2} f''(\zeta_0)/f'(x_0) \le c(\delta) \right\} \\
&\le |e_0| \delta c(\delta) && \{ |e_0| \le \delta \} \\
&= |e_0|\, \varrho && \{ \varrho = \delta c(\delta) \} \\
&< |e_0| && \{ \varrho < 1 \} \\
&\le \delta
\end{aligned}
$$

$\therefore$ $|e_1| < |e_0| \le \delta$, which means the error decreases. This argument may be repeated inductively, giving:

$$
\begin{aligned}
&|e_1| \le \varrho\, |e_0| \\
&|e_2| \le \varrho\, |e_1| \le \varrho^2\, |e_0| \\
&|e_3| \le \varrho\, |e_2| \le \varrho^3\, |e_0| \\
&\vdots
\end{aligned}
$$

In general, $|e_n| \le \varrho^n\, |e_0|$. And since $0 \le \varrho < 1$, we have $\varrho^n \to 0$ when $n \to \infty$, entailing that $|e_n| \to 0$ when $n \to \infty$.

Theorem 4. If f'' is continuous in $\mathbb{R}$, and if f is increasing, convex, and has a root, then said root is unique and Newton's method converges to it from any starting point.

Recall that f is convex if $f''(x) > 0$ for all x. Graphically, it is convex if the line connecting two arbitrary points of f lies above the curve of f between those two points.

11.3 Secant method

In Netwon's method,

$$
x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}
$$

The function of interest is f. We cannot escape computing $f(x_n)$, but it would be desirable to avoid the computation of $f'(x_n)$, which may potentially be expensive. Since

$$
f'(x) = \lim_{h \to x} \frac{f(x) - f(h)}{x - h}
$$

it is natural to suggest

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} \tag{1}$$

Graphically, this means we are not using the line tangent to the point $(x_n, f(x_n))$ but the line secant to the points $(x_n, f(x_n)$ and $(x_{n-1}, f(x_{n-1}))$. The point x_{n+1} is then the value of x where this secant line has a root.

11.4 Fixed point iteration

The key observation is this: if $r \in \mathcal{R}_f$, then $g(x) = x - kf(x)$ has r as fixed point, for any $k \in \mathbb{R}$. Inversely, if g has a fixed point in r, then $r \in \mathcal{R}_f$.

Theorem 5. (1) Let $g \in C[a,b]$ and assume $g(x) \in [a,b]$ for all $x \in [a,b]$. Then there is a fixed point of g in $[a,b]$.

(2) If, on top of previous conditions, g is differentiable in (a,b) and there is some $k < 1$ s.t. $|g'(x)| \leq k$ for all $x \in (a,b)$, then the fixed point referred in (1) is unique.

Theorem 6 (Mean value theorem)**.** Let $f : [a,b] \to \mathbb{R}$ continuous and differentiable on (a,b) with $a < b$. Then there is some $c \in (a,b)$ s.t.

$$f'(c) = \frac{f(b) - f(a)}{b - a}$$

The interpretation is simple: consider the line secant to f on a, b. The theorem ensures that there is some point c s.t. the line tangent to c is parallelt to said secant (equal slopes).

Proof. (1) If a or b are fixed points the proof is done so assume otherwise. Since $g(x) \in [a,b]$, we have $g(a) > a$ and $g(b) < b$.

Take $\varphi(x) = g(x) - x$, which is continuous and defined in $[a,b]$. Then

$$\varphi(a) = g(a) - a > 0, \qquad \varphi(b) = g(b) - b < 0$$

Then $\varphi(a)\varphi(b) < 0$. Then, by the intermediate value theorem, φ has a root in (a,b). In otherwords, there is at least one p s.t.

$$\varphi(p) = g(p) - p = 0$$

$\therefore\ g(p) = p$ is a fixed point of g.

(2) Assume two distinct fixed points p, q exist in $[a,b]$. The mean value theorem ensures the existence of some ζ between p, q (and thus in $[a,b]$) s.t.t

$$g'(\zeta) = \frac{g(a) - g(b)}{a - b} \iff g'(\zeta)(a - b) = g(a) - g(b) \tag{1}$$

By hypothesis, $|g'(x)| \leq k < 1$. Since p, q are assumed to be fixed points, equation (1) gives:

$$\begin{aligned} |p - q| &= |g(p) - g(q)| \\ &= |g'(\zeta)|\,|p - q| \\ &\leq k\,|p - q| < |p - q| \end{aligned}$$

But this is absurd. The contradiction arises from assuming p, q to be distinct. Therefore, the fixed point is unique.

The fixed point algorithm begins with an approximation p_0. Then,

$$p_n = g(p_{n-1})$$

If g continuous and the sequence converges, then it converges to a fixed point, since:

$$p := \lim_{n\to\infty} p_n = \lim_{n\to\infty} g(p_{n-1}) = g\left(\lim_{n\to\infty} p_{n-1}\right) = g(p)$$

Input: p, M, δ
$p_{\text{previous}} = p$
for $i = 1$ **to** $i = M$ **do**
  $p \leftarrow g(p)$
  if $\left|p - p_{\text{previous}}\right| < \delta$ **then**
    return p
  fi
  $p_{\text{previous}} = p$
od
return p

Theorem 7. Let $g \in C[a, b]$ be a self-map of $[a, b]$ differentiable in (a, b). Assume there is a constant $0 < k < 1$ s.t. $|g'(x)| \leq k$ for all $x \in (a, b)$.

For all $p_0 \in [a, b]$, the sequence $p_n = g(p_{n-1})$ converges to the unique f ixed point p in (a, b).

Proof. The mean value theorem ensures that

$$\begin{aligned} |p_n - p| &= |g(p_{n-1}) - g(p)| \\ &= |g'(\zeta_n)||(p_{n-1} - p)| \\ &\leq k\,|p_{n-1} - p| \end{aligned}$$

with $\zeta_n \in (a, b)$. More succintly, with $e_n := p_n - p$,

$$|e_n| \leq k\,|e_{n-1}| \leq k\,|e_{n-2}| \leq \ldots \leq k\,|e_0|$$

By recurrence,

$$|e_n| \leq k^n\,|e_0| \tag{2}$$

Since $0 < k < 1$, $k^n \to 0$ when $n \to \infty$, which entails $|e_n| \to 0$ whene $n \to \infty$.