Contents

Chapter 1

System of nonlinear equations

"Approximate infinite dim problem by finite dim problem"

In thish chapter, we consider carious iterative methods for solving the root of the system of equations

$$\mathbf{f}(\mathbf{x}) = 0,$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^n$ is a vector valued function.

1.1 Operations with digital computer

Floating-point numbers

A common way of expressing real number is decimal system. For example, 325.7 is expressed as

$$3 \times 10^2 + 2 \times 10^1 + 5 + 7 \times 10^{-1}.$$

In most computers, numbers are represented using binary system as follows:

$$(11.0101)_2 = 1 \cdot 2 + 1 + 1 \cdot 2^{-2} + 1 \cdot 2^{-4}.$$

To convert decimal number into binary expression, we let

$$(0.3)_{10} = (.a_1 a_2 a_3 \cdots)_2.$$

Multiplying by 2

$$0.6 = (a_1.a_2 a_3 \cdots)_2, \quad a_i = 0, 1.$$

Thus $a_1 = 0$. Repeating, we obtain $a_2 = 1$, $a_3 = 0$. round up, round off, chop.

Normalized scientific notation

$$\begin{aligned} 875.33432 &= .87533432 \times 10^3 \\ 0.00067 &= .67 \times 10^{-3}. \end{aligned}$$

In decimal number system the normalized notation is $x = \pm r \times 10^n$, $1/10 \leq r < 1$. In binary we write $x = \pm q \times 2^m$, $\frac{1}{2} \leq q < 1$. Here, q is called the mantissa and m is the exponent.

Memory in computer K32

Assume a 32-bit machine called K32. Any word(number) is allocated a 32 bits, where one bit is for the $\pm$ sign, seven for exponent, 23 for mantissa. Any real number has the form $x = \pm q \times 2^m$,$(1 \leq q < 2)$. The normalized binary expression is $q = (1.f)_2$. Here the first digit is always 1 because $1 \leq q < 2$.

$$\begin{aligned} 0.67 &= \frac{a_1}{2} + \frac{a_2}{2^2} + \frac{a_3}{2^3} + \cdots \\ 1.34 &= a_1 + \frac{a_2}{2} + \frac{a_3}{2^2} + \cdots \Rightarrow a_1 = 1, \\ 0.68 &= a_2 + \frac{a_3}{2} + \frac{a_4}{2^2} + \cdots \Rightarrow a_2 = 0, \\ 1.36 &= a_3 + \frac{a_4}{2} + \frac{a_5}{2^2} + \cdots \Rightarrow a_3 = 1. \end{aligned}$$

Thus $0.67 = (1.010\cdots)_2$. But if $0.43 = \frac{a_1}{2} + \frac{a_2}{2^2} + \frac{a_3}{2^3} + \cdots$, then

$$\begin{aligned} 0.86 &= a_1 + \frac{a_2}{2} + \frac{a_3}{2^2} + \cdots \Rightarrow a_1 = 0 \\ 1.72 &= a_2 + \frac{a_3}{2} + \frac{a_4}{2^2} + \cdots \Rightarrow a_2 = 1 \\ 1.44 &= a_3 + \frac{a_4}{2} + \frac{a_5}{2^2} + \cdots \Rightarrow a_3 = 1 \end{aligned}$$

Thus $0.43 = (0.1101\cdots)_2$ in normalized form is

$$(1.101\cdots)_2 \times 2^{-1}.$$

One does not record the first digit, always use next 23 digits to record.

$s =$ (sign)	$e =$ exponent 8 bit	$f =$ normalized mantissa 23 bits

A word in computer K32

Here, $0 < e < (11111111)_2 = 2^8 - 1 = 255$, so $-126 \leq m \leq 127$. If

$$x = (-1)^s q \times 2^m,$$

with $q = (1.f)_2$, f is the 23 bit floating part and $m = e - 127$. (Here $0 < e < 255$) Consider $x = -1024.125 = -(2^{10} + 2^{-3}) = (-1)^1(10000000000.001)_2$. $s = 1, q = 1.0000000000001, m = 10$.

$$s = 1, \quad e = m + 127 = 137 = (10001001)_2, \quad f = (0.0000000000001)_2.$$

- The largest and the smallest number K32 can deal with is between $10^{-38} \sim 10^{38}$
- The Precision is $2^{-23} \approx 1.2 \times 10^{-7}$.
- Integer case: Use all 31 bit(except the sign) So we can represent the numbers between $-(2^{31} - 1)$ and $(2^{31} - 1) \approx 2.1 \times 10^9$.

Overflow, underflow

If a number is too big ($m \geq 127$) machine will say 'overflow' or NaN(Not a number). The subsequent computation is meaningless. if the number is too small in size ($m \leq -127$) it says 'underflow' and the number is treated as zero.

Machine ϵ

A positive number that is negligible compared to the unity is called the machine epsilon, denoted by m_ϵ. It is the smallest number satisfying $1.0 + m_\epsilon \neq 1.0$ and can be computed by the following:

input $s \leftarrow 1$
for $k = 1, 2, \cdots, 2000$ do
  $s \leftarrow 0.5s$
  $t \leftarrow s + 1.0$
  if $(t \leq 1.0)$ then
    $s \leftarrow 2.0s$

```
      output k − 1, s
    endif
endfor
```

The value m_ϵ in K32 is roughly 10^{-7} for single precision. The m_ϵ bounds the *relative error* in representing a number:

$$\left| \frac{fl(x) - x}{x} \right| \leq m_\epsilon.$$

Note : one should not confuse the machine m_ϵ with the smallest number a machine can represent, which is close to 2^{-127}. If you change the above code without assigning the value to t, i.e., change the 5-th line to *if* $(s + 1 \leq 1.0)$ *then*, you get different output. (s value stored in the registry-CPU)

1.1.1 Absolute, relative error, Loss of significance

Loss of significance

$$\begin{aligned} x &= 0.123456789 \\ y &= 0.123455586 \\ x - y &= 0.000001203 \end{aligned}$$

If this calculation were performed in a decimal computer with 6 mantissas, then

$$\begin{aligned} fl(x) &= 0.123457 \\ fl(y) &= 0.123456 \\ fl(x) - fl(y) &= 0.000001 \end{aligned}$$

Relative error is large:

$$\frac{x - y - [fl(x) - fl(y)]}{x - y} = \frac{0.000001203 - 0.000001}{0.000001203} = \frac{0.000000303}{0.000001203} = 25\%$$

The zero is just place holders.

Subtraction of close numbers

In evaluating

$$\sqrt{x^2+1}-1$$

it is easy to lose significant digits if x is small. This phenomena is called **subtractive cancelation**. **representing small number as a difference of two large number is a bad idea**.

Example 1.1.1. (1) The avoid the subtractive cancelation in the evaluation of $\sqrt{x^2+1}-1$, we change it to $x^2/\sqrt{x^2+1}+1$. On the hand held calculator, (having 9 significant digits) with $x = 0.00001$

$$\sqrt{x^2+1}-1 = 0.5\times 10^{-8}, \text{ while } x^2/\sqrt{x^2+1}+1 = 0.4\times 10^{-8}$$

a relative error is

$$\frac{0.1\times 10^{-8}}{0.4\times 10^{-8}} \approx 25\%$$

(2) Assume a computer with 6 digits accuracy:

$$\begin{aligned} \text{True value } x = 0.0000024 \text{ stored as} &\quad \doteq 0.000002 \\ \text{True value } y = 0.0000005 \text{ stored as} &\quad \doteq 0.000001 \\ \text{True value } x-y = 0.0000019 \text{ computed as} &\quad \doteq 0.000001 \end{aligned}$$

The rel error is

$$\frac{x-y-[fl(x)-fl(y)]}{x-y} = \frac{0.0000019-0.000001}{0.0000019} = \frac{0.0000009}{0.0000019} = 47\%$$

(3) As another typical example consider

$$e^{-x}-(1-x+\frac{x^2}{2}) = -\frac{x^3}{3!}+\frac{x^4}{4!}-\frac{x^5}{5!}+\cdots \tag{1.1}$$

The true value of $e^{-4.5}-(1-4.5+\frac{4.5^2}{2})$ is close to -6.61389100. But computing it using the series expnsion $-\frac{4.5^3}{3!}+\frac{4.5^4}{4!}-\frac{4.5^5}{5!}+\cdots$ could be

terrible!

$$\begin{aligned} -\frac{4.5^3}{3!}+\frac{4.5^4}{4!} &= -15.1875+17.0859375=1.8984375 \\ -\frac{4.5^3}{3!}+\frac{4.5^4}{4!}-\frac{4.5^5}{5!} &= -15.1875+17.0859375-15.37734375=-13.47880 \\ -\frac{4.5^3}{3!}+\frac{4.5^4}{4!}-\frac{4.5^5}{5!}+\frac{4.5^6}{6!} &= -15.1875+17.0859375-15.37734375 \\ & \quad +11.5330078=-1.945792 \end{aligned}$$

To avoid subtractive cancelation use a nested multiplication.

$$\begin{aligned} e^{-x}-(1-x+\frac{x^2}{2}) &= -\frac{x^3}{3!}(1-\frac{x}{4}(1-\frac{x}{5}(1-\frac{x}{6})))+\cdots \qquad (1.2) \\ &= -15.1875(1-\frac{4.5}{4}(1-\frac{4.5}{5}(1-(1-\frac{4.5}{6}))))=-4.746 \\ &\doteq -\frac{x^3}{3!}(1-\frac{x}{4}(1-\frac{x}{5}(1-\frac{x}{6}(1-\frac{x}{7})))) \\ &= -15.1875(1-\frac{4.5}{4}(1-\frac{4.5}{5}(1-\frac{4.5}{6}(1-\frac{4.5}{7}))))=-8.7125 \end{aligned}$$

Exercise 1.1.2. (1) Write a computer code (use single precision if possible) to carry out the evaluations in (1.1) and (1.2) . For (1.1, mimick Taylor expansion; save the result of each terms like $\frac{x^3}{3!}$ as $temp=\frac{x^3}{3!}$ then add. For example,

input $temp=\frac{x^2}{2!},\ sum=0$
for $k=1,2,\cdots,2000$ do
  $temp2=-temp*x/(k+2)$
  $sum=sum+temp2,\ temp=temp2$
  output k, sum
endfor

Other wise, all the computations are carried out in the CPU and you do not see much difference.

Compare the difference in the values. How many terms are needed to get accurate value in each case ?

(2) Write a code (as given above) to compute the machine ϵ in your machine and execute to find it. Do eith single precision or Double precision

(specify).

(3) What if you change the 5-th line of the code to *if* $(s + 1 \leq 1.0)$ *then* ?

1.2 Roots of a nonlinear equation

1.2.1 Iterative method-scalar equation

We first consider one dimensional case.

Given $f(x)$ find all $\alpha \in [a, b]$ such that $f(\alpha) = 0$.

Case 1) $f(x)$ can be computed exactly for all $x \in [a, b]$.

Case 2) $f(x)$ is known only by data.

Definition 1.2.1. α is a zero of multiplicity m of $f(x)$. If

$$m = \text{lub}\{k \mid \lim_{x \to \alpha} \frac{|f(x)|}{|x - \alpha|^k} < \infty\}.$$

The equation $f(x) = 0$ can be written in a variety of forms:

$$\begin{aligned} & x = f(x) + x = g(x) \\ \text{or} \quad & x = x - \phi(x)f(x), \quad \phi(x) \neq 0 \quad \text{for } x \in [a, b] \\ \text{or} \quad & x = x - F(f(x)), \quad F(0) = 0, \quad F(y) \neq 0 \quad \text{for } y \neq 0. \end{aligned}$$

These leads to a so called 'Picard iteration method'.

Picard iteration method

(1) Start with some x_0.

(2) Put $x_{k+1} = g(x_k)$.

Theorem 1.2.2. *[Convergence of Picard iteration method] Suppose the following condition hold.*

(1) $|g(x) - g(x')| \leq \lambda |x - x'|$, $\forall x, x' \in [x_0 - \rho, x_0 + \rho] = I$

(2) $0 \leq \lambda < 1$

(3) $|x_0 - g(x_0)| \leq (1 - \lambda)\rho$.

Then

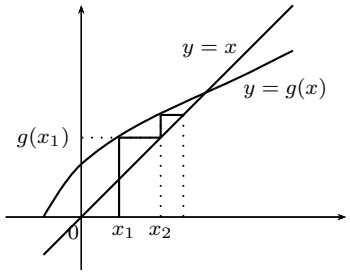


Figure 1.1: Picard's method

(1) the sequence $\{x_k\}$ *generated by* $x_{k+1} = g(x_k)$*, stays in* I

(2) there exists an α *satisfying* $\alpha = g(\alpha)$ *and* $x_k \to \alpha$

(3) α *is unique.*

If g *satisfies* (1), (2) *of the above condition, we say it is a contraction map.*

Proof. 1. $|x_0 - x_1| = |x_0 - g(x_0)| \le (1-\lambda)\rho \Rightarrow x_1 \in I$

$$\begin{aligned} |x_{k+1} - x_k| &= |g(x_k) - g(x_{k-1})| \le \lambda |x_k - x_{k-1}| \le \cdots \le \lambda^k |x_1 - x_0| \\ |x_{k+1} - x_0| &\le |x_{k+1} - x_k| + |x_k - x_{k-1}| + \cdots + |x_1 - x_0| \\ &\le (\lambda^k + \lambda^{k-1} + \cdots + \lambda + 1)(1-\lambda)\rho = (1 - \lambda^{k+1})\rho. \end{aligned}$$

Thus $x_{k+1} \in I$.

2.

$$\begin{aligned} |x_n - x_{n+p}| &\le |x_n - x_{n+1}| + |x_{n+1} - x_{n+2}| + \cdots + |x_{n+p-1} - x_{n+p}| \\ &\le \lambda^n(1-\lambda)\rho + \lambda^{n+1}(1-\lambda)\rho + \cdots + \lambda^{n+p-1}(1-\lambda)\rho \\ &= \lambda^n(1-\lambda)\rho(1 + \lambda + \cdots + \lambda^{p-1}) = \lambda^n \rho \cdot (1 - \lambda^p) < \varepsilon \end{aligned}$$

for all sufficiently large n and p. Hence $\{x_n\}$ is a Cauchy sequence in $\mathbb{R}$ and thus converges. $g(\alpha) = \alpha$ holds by continuity.

3. Assume $g(\alpha) = \alpha$, $g(\beta) = \beta$. Then

$$|\alpha - \beta| = |g(\alpha) - g(\beta)| \le \lambda |\alpha - \beta| < |\alpha - \beta|, \text{ since } \lambda < 1.$$

This is a contradiction unless $\alpha = \beta$. □

If ρ is large in Theorem 1.2.2, then condition (3) is easily satisfied. But if ρ is small then condition (3) is not easily satisfied. Instead condition (1) $\lambda = \max |g'(x)|$ is easily satisfied.

Remark 1.2.3. In particular, if $g(x) \in C^1(I)$ and $|g'(x)| \leq \lambda < 1$, then the conditions (1) and (2) are satisfied and the convergence is guaranteed as long as x_0 is close to $g(x_0)$.

We now present a slightly different version.

Theorem 1.2.4. *Assume*

(1) $g(\alpha) = \alpha$ *i.e.,* α *is a solution of the problem*

(2) $|g(x) - g(x')| \leq \lambda |x - x'|$ *for* $x \in I = [\alpha - \rho, \alpha + \rho]$

(3) $0 \leq \lambda < 1$.

Then, for any $x_0 \in I$, $\{x_k\}$ *stays in* I *and* $\{x_k\} \to \alpha$.

Proof. $|x_1 - \alpha| = |g(x_0) - g(\alpha)| \leq \lambda |x_0 - \alpha| < \rho$ since $x_0 \in [\alpha - \rho, \alpha + \rho]$. Hence $x_1 \in I$ and by induction,

$$\begin{aligned} |x_k - \alpha| &= |g(x_{k-1}) - g(\alpha)| \leq \lambda |x_{k-1} - \alpha| \\ &= \lambda |g(x_{k-2}) - g(\alpha)| \leq \lambda^2 |x_{k-2} - \alpha| \leq \cdots \leq \lambda^k |x_0 - \alpha| \leq \lambda^k \cdot \rho < \rho. \end{aligned}$$

Thus $\{x_k\}$ stays in I and furthermore, $\{x_k\}$ converges to α. □

Order of Convergence

Given a numerical algorithm, the number μ satisfying

$$|x_k - \alpha| \leq M |x_{k-1} - \alpha|^\mu$$

for some fixed number $M \geq 0$ is called the *order of convergence*. Hence Picard method is a order one method.

Assume $\mu = 1$ then

$$|x_k - \alpha| \leq M |x_{k-1} - \alpha| \leq M^2 |x_{k-2} - \alpha| \leq \cdots \leq M^k |x_0 - \alpha|.$$

Thus M must satisfy $0 \leq M < 1$ to assure the convergence. (For any initial guess) The magnitude of M is crucial in linear order method.

When $\mu > 1$, multiply a constant C,

$$C|x_k - \alpha| \leq CM|x_{k-1} - \alpha|^{\mu}.$$

Choose $C = M^{\frac{1}{\mu-1}}$. Then $CM = C^{\mu}$ and if we define $e_k = C|x_k - \alpha|$, we see

$$C|x_k - \alpha| \leq C^{\mu}|x_{k-1} - \alpha|^{\mu} \Rightarrow e_k \leq e_{k-1}^{\mu} \leq (e_{k-2}^{\mu})^{\mu} \leq \cdots \leq e_0^{\mu^{1+2+\cdots+k}}.$$

So we require the initial guess to be close enough to assure the convergence. Here

$$\mu \approx \frac{\log e_k}{\log e_{k-1}} = \frac{\log C + \log e_k}{\log C + \log e_{k-1}} \approx \frac{\log |x_k - \alpha|}{\log |x_{k-1} - \alpha|} \tag{1.3}$$

assuming $|x_{k-1} - \alpha| << 1$. If $\mu = 1$ we say, it is linear and if $\mu = 2$, it is quadratic.

In practice it is hard to compute this number since we do not know C. Instead it can be computed using three term relations (See later sections).

1.2.2 Error propagation

In actual computation, it may not be possible to evaluate $g(x)$ exactly due to rounding error or $g(x)$ may be given as a numerical table such as numerical solution of differential equations. Thus, the actual iteration scheme may be represented as

$$x_{k+1} = g(x_k) + \delta_k,$$

where $|\delta_k| < \delta$ for some known bound δ.

Theorem 1.2.5. *Suppose*

(1) $\alpha = g(\alpha)$

(2) $|g(x) - g(\alpha)| \leq \lambda|x - \alpha|$, $0 \leq \lambda < 1$, $x \in [\alpha - \rho_0, \alpha + \rho_0]$

(3) $|x_0 - \alpha| \leq \rho_0$ *where* $0 < \rho_0 < \rho - \dfrac{\delta}{1-\lambda}$

(4) $x_{k+1} = g(x_k) + \delta_k$, $|\delta_k| < \delta$.

Then, $|\alpha - x_k| \leq \rho$ *and*

$$|\alpha - x_k| \leq \frac{\delta}{1-\lambda} + \lambda^k(\rho_0 - \frac{\delta}{1-\lambda}).$$

Method of bisection

Assume we have two points a, b such that $f(a)f(b) < 0$. Put

$$r_0 \;=\; a, \quad s_0 = b$$

and for $n = 0,\ 1,\ 2,\ \ldots,$ do the following:

(1) Set $z_n = \frac{r_n+s_n}{2}$ and compute $f(z_n)$.

(2) With $p = f(r_n) \cdot f(z_n)$ do

  (a) If $p = 0$ (or $|p| <$ pre-specified TOL), then stop and accept $\alpha = z_n$

  (b) If $p < 0$ then set $r_{n+1} = r_n$ and $s_{n+1} = z_n$

  (c) If $p > 0$ then set $r_{n+1} = z_n$ and $s_{n+1} = s_n$.

Note that $r_n \le \alpha \le s_n$. Convergence analysis: Since $z_{n+1} = \frac{1}{2}(r_{n+1} + s_{n+1})$, we see

$$|r_{n+1} - s_{n+1}| \;=\; \begin{cases} |r_n - z_n| = \left|r_n - \frac{r_n+s_n}{2}\right| = \frac{|r_n-s_n|}{2} \\ |z_n - s_n| = \left|\frac{r_n+s_n}{2} - s_n\right| = \frac{|r_n-s_n|}{2}. \end{cases} \tag{1.4}$$

Hence

$$|r_{n+1} - s_{n+1}| = \frac{|r_n - s_n|}{2} \le \cdots \le \frac{|r_0 - s_0|}{2^{n+1}} \to 0.$$

Since $r_{n+1} \le \alpha \le s_{n+1}$, $\{r_n\}$ converges to α. Hence so does $\{z_n\}$.

$$\begin{aligned} |z_n - \alpha| \;&=\; \left|\frac{r_n + s_n}{2} - \alpha\right| = \left|\frac{r_n - \alpha}{2} + \frac{s_n - \alpha}{2}\right| (*) \\ &\le\; \left|\frac{r_n - \alpha}{2}\right| + \left|\frac{s_n - \alpha}{2}\right| \\ &=\; \frac{s_n - \alpha}{2} + \frac{\alpha - r_n}{2} = \frac{s_n - r_n}{2} \\ &\le\; \frac{|r_0 - s_0|}{2^{n+1}} \to 0. \end{aligned}$$

Considering the sign of terms in (*), we have

$$\begin{aligned} |z_n - \alpha| \;&=\; \left|\frac{r_n + s_n}{2} - \alpha\right| = \left|\frac{r_n - \alpha}{2} + \frac{s_n - \alpha}{2}\right| \\ &\le\; \max\left(\left|\frac{r_n - \alpha}{2}\right| + \left|\frac{s_n - \alpha}{2}\right|\right). \end{aligned}$$

This is a first order (linear) method.

1.2.3 Pseudo code for bisection method

input r, s and f with $f(r)f(s) < 0$
for $k = 1, 2, \cdots, 2000$ do
  set $z = (r + s)/2$ and compute $p = f(z)$
  If $|p| < tol$ then set z as solution and break;
  If $p < 0$ then set $r = z$ (* do not set s to save time *)
  else $s = z$
  endif
  output $k - 1, z$
end

Method of false position-Regular Falsi

Assume $f(a)f(b) < 0$. Use a linear approximation using first two data $(s_n, f(s_n))$ and $(r_n, f(r_n))$ to obtain a new prediction. Let

$$z_n = s_n - \frac{(s_n - r_n)f(s_n)}{f(s_n) - f(r_n)} = \frac{r_n f(s_n) - s_n f(r_n)}{f(s_n) - f(r_n)}$$

and proceed as in bisection method.

Secant method

Secant method is (almost) the same as the method of false position but we do no check if $f(a)f(b) < 0$, so it saves some computation. Thus given two initial guesses z_0, z_1, we let

$$z_{n+1} = \frac{z_{n-1}f(z_n) - z_n f(z_{n-1})}{f(z_n) - f(z_{n-1})} = z_n - \frac{f(z_n)(z_n - z_{n-1})}{f(z_n) - f(z_{n-1})}, \quad n = 1, 2, \cdots . \quad (1.5)$$

Convergence behavior will be investigated later.

A special case of Picard iteration

Assume $g \in C^2$ and $g'(\alpha) = 0$ in the Picard iteration $x_{k+1} = g(x_k)$. In this case we can show the convergence is quadratic(same as Newton's method):

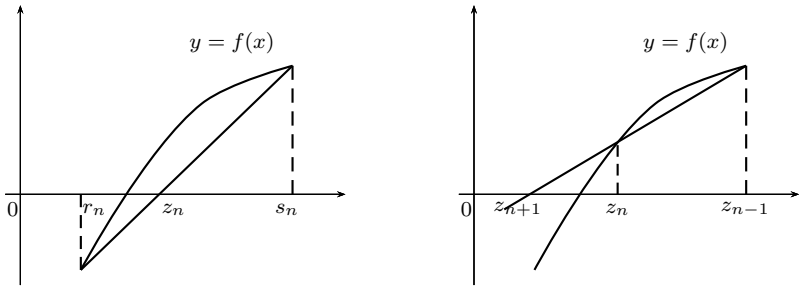


Figure 1.2: Method of false position and secant method

Since

$$\begin{aligned} g(x) &= g(\alpha) + g'(\alpha)(x-\alpha) + \frac{g''(\xi)}{2}(x-\alpha)^2, \qquad \xi \in (x,\alpha) \\ &= g(\alpha) + \frac{g''(\xi)}{2}(x-\alpha)^2, \end{aligned}$$

we have from $x_{k+1} = g(x_k)$,

$$|x_{k+1} - \alpha| = |g(x_k) - \alpha| = |g(x_k) - g(\alpha)| \leq \left|\frac{g''(\xi)}{2}\right| |x_k - \alpha|^2. \quad (1.6)$$

Therefore $e_{k+1} \leq M e_k^2$ where $M = \max \frac{|g''(x)|}{2}$. (Quadratic convergence): Multiply by M and set $\bar{e}_{k+1} = M e_k$, we have

$$\bar{e}_{k+1} \leq (\bar{e}_k)^2 \leq (\bar{e}_{k-1}^2)^2 \leq \cdots \leq_0^{1+2+\cdots+2^k} = (\bar{e}_0)^{2^{k+1}}.$$

If $|\bar{e}_0| = M|x_0 - \alpha| < 1$, it converges quickly.

Newton's method

Let us now derive the Newton's method for solving $f(x) = 0$. To consider a more general scheme, we put

$$g(x) = x - \phi(x) f(x).$$

This is a consistent scheme if $\phi(x) \neq 0$ near the solution. We see that

$$g'(x) = 1 - \phi'(x) f(x) - \phi(x) f'(x).$$

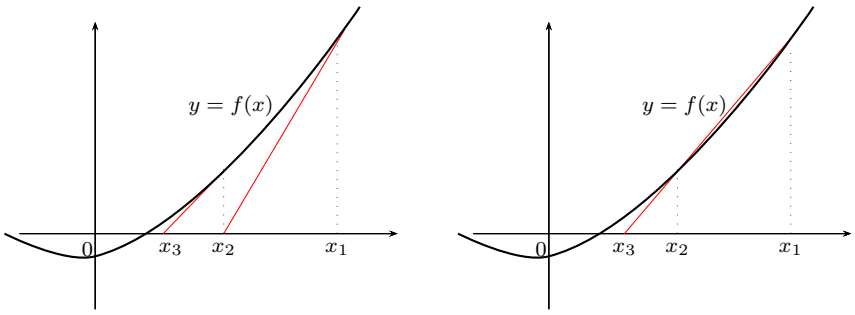


Figure 1.3: Newton's method and secant method

We see from the previous discussion that we get a second order convergence if $g'(\alpha) = 0$. Hence we choose ϕ so that

$$g'(\alpha) = 1 - \phi'(\alpha)f(\alpha) - \phi(\alpha)f'(\alpha) = 1 - \phi(\alpha)f'(\alpha) = 0,$$

i.e, if $\phi(x) \equiv \frac{1}{f'(x)}$, the scheme becomes $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$ and it was shown to be second order, provided $f'(\alpha) \neq 0$.

Defect of Newton's method: If the value is given by table or algorithm, (i.e., as a solution of D.E.) how to find $f'(x)$?

Convergence of Newton's method

Suppose $f \in C^2$ in a neighborhood of $\bar{x}$, where $f(\bar{x}) = 0$ and $f'(\bar{x}) \neq 0$. The Newton's method is:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \qquad f(\bar{x}) = 0.$$

Let $e_n = x_n - \bar{x}$ be the error. Use the Taylor expansion near x_n,

$$\begin{aligned} 0 &= f(\bar{x}) = f(x_n) + f'(x_n)(\bar{x} - x_n) + \frac{f''(c)(\bar{x} - x_n)^2}{2}, \quad c \in [x_n, \bar{x}] \\ e_{n+1} &= e_n - \frac{f(x_n)}{f'(x_n)} = \frac{e_n f'(x_n) - f(x_n)}{f'(x_n)} = \frac{f''(c)e_n^2}{2f'(x_n)}. \end{aligned}$$

Let $M = \sup_{[a,b]} |f''(x)|$ and $m = \inf_{[a,b]} |f'(x)|$. Then by continuity $m > 0$.

$$|e_{n+1}| \leq \frac{M}{2m}|e_n^2| = \frac{2m}{M} \cdot \left(\frac{M}{2m}|e_n|\right)^2.$$

Define $E_{n+1} = \frac{M}{2m}|e_{n+1}|$, then we see

$$E_{n+1} \leq E_n^2 \leq \cdots \leq E_0^{2^{n+1}}.$$

Hence the Newton's method converges if $E_0 = \frac{M}{2m}|e_0| < 1$.

Case of multiple root

Suppose $f(\bar{x}) = f'(\bar{x}) = 0$, $f'(x) \neq 0$ except $x_n = \bar{x}$ near $\bar{x}$, $f''(\bar{x}) \neq 0$. Then we have by Taylor series

$$\begin{aligned} e_{n+1} &= \frac{f''(c)e_n^2}{2f'(x_n)} \\ &= \frac{f''(c)e_n^2}{2f'(x_n) - 2f'(\bar{x})} \\ &= \frac{f''(c)e_n^2}{2(x_n - \bar{x})f''(\xi)} = \frac{f''(c)}{2f''(\xi)}e_n. \end{aligned}$$

Thus only 1st order convergence, i.e, $|e_{n+1}| \approx \frac{1}{2}|e_n|$ like bisection. But Newton method is better because bisection method cannot be used for a double root.

Theorem 1.2.6 (Mean value theorem for integral)**.** *Let f be continuous and g be integrable which does not change sign on $[a, b]$. Then there exists a point $c \in [a, b]$ satisfying*

$$\int_a^b f(x)g(x)\,dx = f(c)\int_a^b g(x)\,dx.$$

General multiple root

Now $\bar{x}$ is a root of $f(x) = 0$ of multiplicity p, i.e, $f(\bar{x}) = \cdots f^{(p-1)}(\bar{x}) = 0$ and $f^{(p)}(\bar{x}) \neq 0$. By the integral form of Taylor's formula applied to f and f' respectively, we have

$$f(x) = f(\bar{x}) + f'(\bar{x})(x-\bar{x}) + \cdots + \frac{f^{(p-1)}(\bar{x})}{(p-1)!}(x-\bar{x})^{p-1} + \int_{\bar{x}}^{x} \frac{(x-t)^{p-1}}{(p-1)!} f^{(p)}(t)\,dt \tag{1.7}$$

$$f'(x) = f'(\bar{x}) + f''(\bar{x})(x-\bar{x}) + \cdots + \frac{f^{(p-1)}(\bar{x})}{(p-2)!}(x-\bar{x})^{p-2} + \int_{\bar{x}}^{x} \frac{(x-t)^{p-2}}{(p-2)!} f^{(p)}(t)\, dt \tag{1.8}$$

On the other hand, since $f(\bar{x}) = \cdots f^{(p-1)}(\bar{x}) = 0$, we see (1.7), (1.8) becomes

$$f(x) = \int_{\bar{x}}^{x} \frac{(x-t)^{p-1}}{(p-1)!} f^{(p)}(t)\, dt \tag{1.9}$$

$$f'(x) = \int_{\bar{x}}^{x} \frac{(x-t)^{p-2}}{(p-2)!} f^{(p)}(t)\, dt. \tag{1.10}$$

Multiply the second equation by $(x - \bar{x})$ and subtract first equation from it, we have

$$(x-\bar{x})f'(x) - f(x) = \int_{\bar{x}}^{x} \frac{f^{(p)}(t)}{(p-1)!}(x-t)^{p-2}[(p-1)(x-\bar{x}) - (x-t)]\, dt. \tag{1.11}$$

Note that the quantity in bracket does not change sign. Hence by integral mean value theorem, there is a point c between x and $\bar{x}$ satisfying

$$\begin{aligned} &= \frac{f^{(p)}(c)}{(p-1)!} \int_{\bar{x}}^{x} (x-t)^{p-2}[(p-1)(x-\bar{x}) - (x-t)]\, dt \\ &= \frac{f^{(p)}(c)}{(p-1)!} \left(\frac{1-p}{p}\right) (x-\bar{x})^{p}. \end{aligned} \tag{1.12}$$

Hence we have

$$\begin{aligned} e_{n+1} &= e_n - \frac{f(x_n)}{f'(x_n)} = \frac{e_n f'(x_n) - f(x_n)}{f'(x_n)} = \frac{f^{(p)}(c) e_n^p \cdot \frac{1-p}{p}}{(p-1)!} \Big/ \frac{e_n^{p-1} \cdot f^{(p)}(\xi)}{(p-1)!} \\ &= \frac{1-p}{p} \cdot \frac{f^{(p)}(c)}{f^{(p)}(\xi)} \cdot e_n \approx \frac{1-p}{p} e_n. \end{aligned}$$

Since $\frac{p-1}{p} < 1$ we have first order convergence. However, the convergence rate deteriorate as p grows.

Improvement of convergence

Assume we know the multiplicity of a root in advance. Consider the following modification of Newton's method for the case of p-th multiple root.

$$x_{n+1} = x_n - p\frac{f(x_n)}{f'(x_n)}. \tag{1.13}$$

Proceeding as in (1.7), (1.8) with $p+1$ replacing p, we have

$$f(x) = \frac{f^{(p)}(\bar{x})}{p!}(x-\bar{x})^p + \int_{\bar{x}}^{x} \frac{(x-t)^p}{p!} f^{(p+1)}(t)\, dt \tag{1.14}$$

$$f'(x) = \frac{f^{(p)}(\bar{x})}{(p-1)!}(x-\bar{x})^{p-1} + \int_{\bar{x}}^{x} \frac{(x-t)^{p-1}}{(p-1)!} f^{(p+1)}(t)\, dt \tag{1.15}$$

Hence

$$(x-\bar{x})f'(x) - pf(x) = \int_{\bar{x}}^{x} \frac{f^{(p+1)}(t)}{(p-1)!}(x-t)^{p-1}[(x-\bar{x})-(x-t)]\, dt. \tag{1.16}$$

Again by integral mean value theorem, there exists a point c between $\bar{x}$ and x such that

$$\begin{aligned}(x-\bar{x})f'(x) - pf(x) &= \frac{f^{(p+1)}(c)}{(p-1)!}\int_{\bar{x}}^{x}(x-t)^{p-1}[(x-\bar{x})-(x-t)]\, dt \\ &= \frac{f^{(p+1)}(c)}{(p-1)!}\frac{(x-\bar{x})^{p+1}}{p(p+1)}.\end{aligned}$$

Hence using (1.10), we see

$$\begin{aligned}e_{n+1} &= e_n - p\frac{f(x_n)}{f'(x_n)} = \frac{e_n f'(x_n) - pf(x_n)}{f'(x_n)} \\ &= \frac{e_n^2}{p(p+1)}\frac{f^{(p+1)}(c)}{f^{(p)}(\xi)}.\end{aligned}$$

If $p = 1$ this reduces to original Newton's method. Note that this scheme may diverge for a problem with simple root. One has to know the multiplicity in advance to use this version. But after a few iteration one can find p (since it is an integer).

Nonconvergence of Newton's method

There are certain instances when the Newton's method does not converge. (It may diverge or fall into some periodic behavior)

1.2.4 Variants of Newton's Method

A few **difficulties** with the Newton's method

(1) It may not be possible to find a formula for f'.

(2) Even if the formula for f' is available, it may be too expansive to compute.

(3) Newton's method may fail to converge.

Finite Differences

We may replace $f'(x_n)$ by

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

for some h.

Quasi Newton Method

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

(1) One choice is $s_n = \frac{f(x_n)-f(x_{n-1})}{x_n - x_{n-1}}$(secant method)

(2) another choice is to use **constant slope** $s_n = c$.

Damping

Failure of Newton's method may happen for example, when we overshoot the value: Try Newton's method for $f(x) = 1/x - 10$ with $x_0 = 10$. We will see it fails to converge.

Thus we use damping:

$$x_{n+1} = x_n - \mu_n \frac{f(x_n)}{f'(x_n)} \tag{1.17}$$

for $0 < \mu_n < 1$ and $\mu_n \to 1$ as $n \to \infty$.

Mixing Methods

Start with bisection (guaranteed) and after a few iteration **switch to Newton's method**(fast).

Convergence of Secant method

One of the drawback of Newton's method is that it requires to evaluate the derivative which is not only costly but sometimes not available. So we suggest a variant of Newton's method which does not require the evaluation of derivative.

We replace the derivative $f'(x_n)$ in the Newton method by the backward finite difference

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

which is available during the iterations(Need two initial values, however.). Thus the **secant method** is given in the form

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

Convergence proof:

$$\begin{aligned}
x_{n+1} - \bar{x} &= x_n - \bar{x} - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})} \\
&= \frac{(x_n - \bar{x})(f(x_n) - f(x_{n-1})) - f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})} \\
&= \frac{(x_n - \bar{x})}{f(x_n) - f(x_{n-1})} \left[f(x_n) - f(x_{n-1}) - \frac{f(x_n) - f(\bar{x})}{x_n - \bar{x}} (x_n - x_{n-1}) \right] \\
&= \frac{(x_n - \bar{x})(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})} \left[\frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} - \frac{f(x_n) - f(\bar{x})}{x_n - \bar{x}} \right] \\
&= (x_n - \bar{x})(x_{n-1} - \bar{x}) \frac{f''(\eta_n)}{2f'(\xi_n)}, \qquad \eta_n, \xi_n \in (x_{n-1}, \bar{x}).
\end{aligned}$$

In the last equality, we used MVT and second divided difference formula: For a C^2 function f we have (Later or if $|x_n - \bar{x}| \leq |x_{n-1} - \bar{x}|$)

$$\frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} - \frac{f(x_n) - f(\bar{x})}{x_n - \bar{x}} = \frac{f''(\eta_n)}{2} (x_{n-1} - \bar{x}). \tag{1.18}$$

Thus we have

$$|e_{n+1}| \leq |e_n| \cdot |e_{n-1}| \cdot M,$$

where $M = \frac{\max}{2} \left| \frac{f''(\eta_n)}{f'(\xi_n)} \right|$. Put $q_n = M|x_n - \bar{x}|$. Then

$$q_{n+1} \leq q_n \cdot q_{n-1}. \tag{1.19}$$

We now solve the difference equation $q_{n+1} = q_n \cdot q_{n-1}$ to find out the convergence order. Let $z_n = \log q_n$. Then we obtain a Fibonacci sequence

$$z_{n+1} \quad = \quad z_n + z_{n-1}$$

whose solution can be obtained in the form: $z_n = c_1\lambda_1^n + c_2\lambda_2^n$ where λ_i is the sol. of $\lambda^2 - \lambda - 1 = 0$. $\lambda_1 = (1+\sqrt{5})/2$ (larger one), $\lambda_2 = (1-\sqrt{5})/2$. One can find the values c_1, c_2 by the initial condition. Thus

$$q_n \quad = \quad e^{z_n} = e^{c_1\lambda_1^n + c_2\lambda_2^n}.$$

For large n, we have $q_{n+1} \sim q_n^{\lambda_1}$. Thus $Me_{n+1} \leq (Me_n)^{\lambda_1}$and the sequence $\{q_n\}$ converges provided $|Me_1| < 1$.

Remark 1.2.7. 1. If $\bar{x}$ is a double root, $M = \frac{\max}{2}\left|\frac{f''(\eta_n)}{f'(\xi_n)}\right|$ may be ∞. Look at $y = x^2$. In this case, we have no convergence.

2. $q_{n+1} \sim q_n^{\lambda_1}$, $\lambda_1 = \frac{1+\sqrt{5}}{2} \doteq 1.618$. This suggest that secant method may be better than bisection (linear) or function iteration.

Computing the order of convergence

Given some iterative method to find a zero of $f(\bar{x}) = 0$. Assuming we know the exact solution(or a close approximation of it), we can estimate the order of convergence. Let $e_{n+1} = |x_{n+1} - \bar{x}|$. Assume $e_{n+1} \approx Me_n^{\mu}$, M constant. Then $e_n \approx Me_{n-1}^{\mu}$. Then dividing we obtain

$$\frac{e_{n+1}}{e_n} \approx \left(\frac{e_n}{e_{n-1}}\right)^{\mu}. \tag{1.20}$$

Thus we have

$$\mu \approx \log\left(\frac{e_{n+1}}{e_n}\right) \Big/ \log\left(\frac{e_n}{e_{n-1}}\right). \tag{1.21}$$

One can also estimate M.

Definition 1.2.8. Let x_n be a sequence. Then the first *forward difference* is

$$\Delta x_n = x_{n+1} - x_n \tag{1.22}$$

Successive forward differences are defined recursively, as follows: The k-th

forward difference is given in terms of the $k-1$ st forward difference as

$$\Delta^k x_n = \Delta(\Delta^{k-1} x_n) \tag{1.23}$$

Aitken's δ^2 method

Aitken's method (George Aitken, 1895-1967) is based on the following observation. For any linearly convergent method we have

$$\lim \left| \frac{x_{n+1} - \bar{x}}{x_n - \bar{x}} \right| = \lambda > 0. \tag{1.24}$$

Thus for sufficiently large n, we might expect that

$$\frac{x_{n+1} - \bar{x}}{x_n - \bar{x}} \approx \frac{x_{n+2} - \bar{x}}{x_{n+1} - \bar{x}} \approx \lambda > 0. \tag{1.25}$$

Solving for $\bar{x}$ we obtain

$$\bar{x} = x_n + \frac{x_{n+2}x_n - x_{n+1}^2 - x_n \Delta^2 x_n}{\Delta^2 x_n}. \tag{1.26}$$

Expanding the numerator, we see

$$\bar{x} = x_n - \frac{(\Delta x_n)^2}{\Delta^2 x_n}. \tag{1.27}$$

This motivates us to define a new sequence (while computing x_n) by

$$(Ax)_n = x_n - \frac{(\Delta x_n)^2}{\Delta^2 x_n}. \tag{1.28}$$

Steffensen's method

Based on the Aitkin's method (applied to the fixed point equation $x = f(x)$ using $x_{n+1} = f(x_n)$) we obtain Steffensen's method. It is defined by

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

Note the similarity with Aitken's method. The x_{n+1} on the left side is like $(Ax)_n$ in Aitken's method.

One can show that if $f \in C^2$ and $f'(\bar{x}) \neq 1$ then

$$e_{n+1} = \frac{f''(\bar{x})f'(\bar{x})}{2(f'(\bar{x})-1)}e_n^2 + O(e_n^3).$$

Use Taylor expansion ($f(\bar{x}) = \bar{x}$)

$$\begin{aligned} f(x_n) &= f(\bar{x}+e_n) = e_n f'(\bar{x}) + \frac{1}{2}e_n^2 f''(\bar{x}) + O(e_n^3) \equiv \bar{x} + g(e_n) \\ (f(x_n)-x_n)^2 &= (g(e_n)-e_n)^2 = e_n^2(f'(\bar{x})-1+\frac{1}{2}e_n f''(\bar{x}))^2 \\ &= e_n^2(f'(\bar{x})-1)^2 + e_n^3 f''(\bar{x})(f'(\bar{x})-1) + O(e_n^4). \end{aligned}$$

Continuing

$$\begin{aligned} f(f(x_n)) &= f(\bar{x}+g(e_n)) = f(\bar{x}) + g(e_n)f'(\bar{x}) + \frac{1}{2}g(e_n)^2 f''(\bar{x}) + O(e_n^3) \\ y_n &\equiv \bar{x} + g(e_n)[f'(\bar{x}) + \frac{1}{2}g(e_n)f''(\bar{x})] + O(e_n^3) \\ &= \bar{x} + e_n(f'(\bar{x}) + \frac{1}{2}e_n f''(\bar{x}))[f'(\bar{x}) + \frac{1}{2}g(e_n)f''(\bar{x})] + O(e_n^3) \\ &= \bar{x} + e_n(f'(\bar{x}) + \frac{1}{2}e_n f''(\bar{x}))[f'(\bar{x}) + \frac{1}{2}e_n(f'(\bar{x}) + \frac{1}{2}e_n f''(\bar{x}))f''(\bar{x})] \\ &= \bar{x} + e_n(f'(\bar{x}))^2 + \frac{1}{2}e_n^2 f'(\bar{x})f''(\bar{x})(1+f'(\bar{x})) + O(e_n^3). \end{aligned}$$

$$f(f(x_n)) - 2f(x_n) + x_n = e_n(f'(\bar{x})-1)^2 + \frac{1}{2}e_n^2 f''(\bar{x})(f'(\bar{x})+2)(f'(\bar{x})-1) + O(e_n^3).$$

Since $e_{n+1} - e_n = x_{n+1} - x_n$

$$\begin{aligned} e_{n+1} &= e_n - \frac{[f(x_n)-x_n]^2}{f(f(x_n)) - 2f(x_n) + x_n} = \frac{f''(\bar{x})f'(\bar{x})e_n^2 + O(e_n^3)}{2(f'(\bar{x})-1)(1+O(e_n))} \\ &= \frac{f''(\bar{x})f'(\bar{x})}{2(f'(\bar{x})-1)}e_n^2 + O(e_n^3). \end{aligned}$$

Remark 1.2.9. Equivalently, Steffensen's method to find the solution of $f(x) = 0$ ($f(x) - x = 0$ above) has the following form: Given x_0, find

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

for $n = 0, 1, 2, 3, ...,$ where the slope function $g(x_n)$ is a composite of the

original function given by the following formula:

$$g(x_n) = \frac{f(x_n + f(x_n)) - f(x_n)}{f(x_n)}.$$

The function g is the average slope of the function f between the last sequence point $(x, y) = (x_n, f(x_n))$ and the auxiliary point $(x, y) = (x_n + h, f(x_n + h))$, with the step $h = f(x_n)$.

Advantages and drawbacks

The main advantage of Steffensen's method is that it has quadratic convergence like Newton's method.

The price for the quick convergence is the double function evaluation(secant method needs only one function evaluation): both $f(x_n)$ and $f(x_n + h)$ must be calculated, which might be time-consuming if f is a complicated function.

Exercise 1.2.10. (1) Use Taylor expansion to prove the divided difference formula (1.18).

(2) Let $f_1(x) = (x-1)(x^7 + 6x^6 + 3x^2 - 3)$ and $f_2(x) = 2x^7 - x^6 - 3.5x^4 + 2$.

(a) Use Newton's method to find the zero of f_1 near 0.757628... (another one is near $-0.779945...$). Also find the zero of f_2 between -1 and 0.5.

(b) Find the root of f_2 between -1 and 0.5.

(c) Repeat the problem with Steffenson's method.

(Start with several initial values near the zero and see what happens). In each case, find the order of convergence (use $|x_n - \bar{x}|$, not $|x_n - x_{n-1}|$) and compare with theoretical results. Also, use graphical tools to draw the graph and plot some intermediate values.

(3) Find first ten roots of the equation $\tan x = x$.
Set

$$g(t) = \frac{f(t) - f(x_n)}{(t - x_n)}$$

and use MVT for g

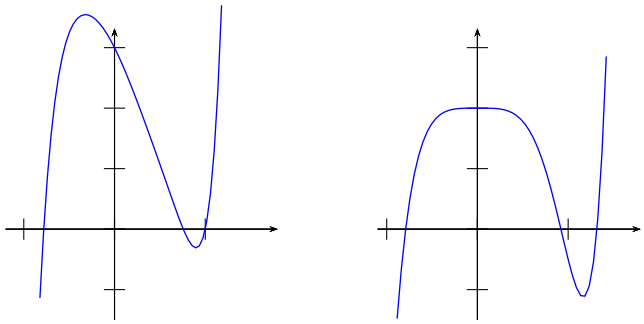


Figure 1.4: $(x-1)(x^7+6x^6+3x^2-3)$ and $2x^7-x^6-3.5x^4+2$

1.2.5 Higher degree interpolation and inverse interpolation

Quadratic interpolation-Muller's method

Recall the method of using a linear interpolation to find a root of function (e.g., secant method or regular-falsi method): Given a bracket $[x_0, x_1]$, we interpolate f by a linear function and let the zero of the linear function be the approximate root of f.

Can we use higher degree polynomial to generate a sequence of iterates? Given data $(x_k, f(x_k)) = (x_k, y_k), k = 1, 2, \cdots$, one can construct a polynomial p s.t. $y_k = p(x_k), k = 1, 2, \cdots$. But solving $p(\bar{x}) = 0$ is difficult! Muller used a quadratic interpolation. This is a direct extension of secant method to solve $f(\alpha) = 0$. It is of high order and works for complex roots when the starting value is a complex number. But it may give us a complex number even when we are expecting real solution.

Now we describe Muller's method: Given $f(x)$, and x_0, x_1, x_2 near α. Construct a second degree polynomial p such that $p(x_0) = f(x_0)$, $p(x_1) = f(x_1)$, $p(x_2) = f(x_2)$. Let $p(x) = a(x-x_2)^2 + b(x-x_2) + c$. Then

$$\begin{aligned} a(x_0-x_2)^2 + b(x_0-x_2) + c &= f(x_0) \\ a(x_1-x_2)^2 + b(x_1-x_2) + c &= f(x_1) \\ a(x_2-x_2)^2 + b(x_2-x_2) + c &= f(x_2) \end{aligned}$$

With $\delta_0 = x_0 - x_2$, $\delta_1 = x_1 - x_2$, $r_0 = f(x_0) - f(x_2)$, $r_1 = f(x_1) - f(x_2)$, one

obtains

$$a = \frac{\delta_1 r_0 - \delta_0 r_1}{\delta_0 \delta_1 (\delta_0 - \delta_1)}, \qquad b = \frac{\delta_0^2 r_1 - \delta_1^2 r_0}{\delta_0 \delta_1 (\delta_0 - \delta_1)}, \qquad c = f(x_2).$$

Thus

$$x - x_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

We take the one with smaller modulus.(Reason:Assume three consecutive points lie on a smooth convex curve, it is reasonable to expect smaller value is a good approximation.)

Remark 1.2.11. (1) Müller's method can find complex root also.

(2) Newton's method also finds complex root if started with complex numbers.

(3) To find the root of a polynomial, it is suggested to use the *companion matrix*. Let

$$C = \begin{bmatrix} 0 & 1 & 0 & \\ & \ddots & \ddots & 0 \\ & & \ddots & 1 \\ -a_n & -a_{n-1} & \dots & -a_1 \end{bmatrix}$$

We can show that

$$|C - \lambda I| = 0$$

if and only if $p(\lambda) = \lambda^n + a_1 \lambda^{n-1} + \cdots + a_{n-1}\lambda + a_n = 0$.

(4) Order of convergence for Müller's method. $e_{n+1} \doteq e_n^{1.839}$.

(5) Müller's method may not converge for triple root. In this case use perturbation.

Inverse interpolation

An alternative is to use the inverse interpolation: Given data $(x_k, f(x_k)) = (x_k, y_k), k = 1, 2, \cdots$, where y_k are distinct, we construct a polynomial $p(y)$ such that $x_k = p(y_k)$. Set $p(0)$ as the next iterate.

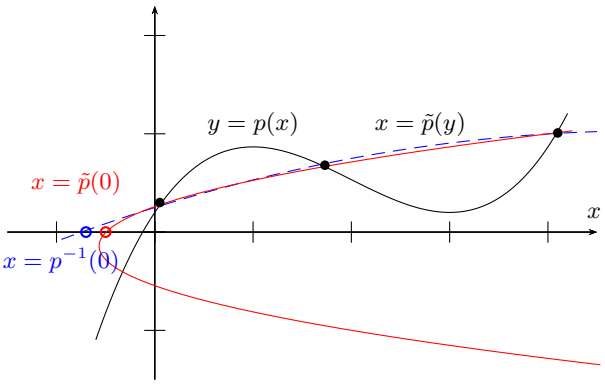


Figure 1.5: quadratic(blue) and inverse quadratic(red) interpolation

Example 1.2.12 (inverse quadratic interpolation)**.** Assume we have data $(x_1, f(x_1)), (x_2, f(x_2)), (x_3, f(x_3))$. The quadratic polynomial $p(y)$ s.t. $p(y_k) = x_k, k = 1, 2, 3$ is

$$p(y) = x_1 \frac{(y-y_2)(y-y_3)}{(y_1-y_2)(y_1-y_3)} + x_2 \frac{(y-y_1)(y-y_3)}{(y_2-y_1)(y_2-y_3)} + x_3 \frac{(y-y_1)(y-y_2)}{(y_3-y_1)(y_3-y_2)}$$

New approximation is

$$\begin{aligned} p(0) &= x_1 \frac{y_2 y_3}{(y_1-y_2)(y_1-y_3)} + x_2 \frac{y_1 y_3}{(y_2-y_1)(y_2-y_3)} + x_3 \frac{y_1 y_2}{(y_3-y_1)(y_3-y_2)} \\ &= \frac{x_1 y_2 y_3 (y_3-y_2) + x_2 y_3 y_1 (y_1-y_3) + x_3 y_1 y_2 (y_2-y_1)}{(y_1-y_2)(y_2-y_3)(y_3-y_1)} \end{aligned}$$

This is similar to Muller's but no need to worry complex roots, or no need to choose which one of the roots. The convergence order is 1.839.

Example 1.2.13. Consider a quadratic interpolating poly with data:

$$(1, 1), (2, 1.5), (3, 0.3)$$

Lagrange interpolation $p_n(x) = \sum_{i=0}^{n} f(x_i) \prod_{j \neq i} \frac{(x-x_j)}{(x_i-x_j)}$ gives

$$p(x) = 1 \frac{(x-2)(x-3)}{(-1)(-2)} + 1.5 \frac{(x-1)(x-3)}{1(-1)} + 0.3 \frac{(x-1)(x-2)}{2}.$$

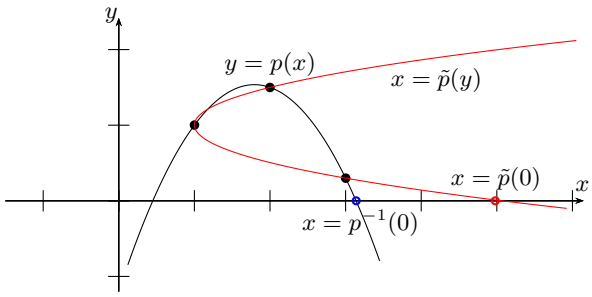


Figure 1.6: quadratic/inverse quadratic interp. Example 1.2.13

But its inverse quadratic interpolation $x = \tilde{p}(y)$ is

$$\tilde{p}(y) = 1\frac{(y-1.5)(y-.3)}{(-.5)(.7)} + 2\frac{(y-1)(y-.3)}{.5(1.2)} + 3\frac{(y-1)(y-1.5)}{(-.7)(-1.2)}.$$

Thus

$$p(x) = \frac{1}{2}(-1.7x^2 + 6.1x - 2.4), \quad \tilde{p}(y) = 2.047619y^2 - 8.1190524y + 5.0714298.$$

Two roots of $p(x)$ are 3.318 and 0.4498 while $\tilde{p}(0) = 5.0714298$..

1.2.6 Safeguarded Methods

We may consider a hybrid method between fast method and safe method. For example combine (fast) Newton's Method with the (safe) bisection method. Start with a bracket $[x_0, x_1]$. If each Newton's step gives new iterate x_{new}, if $x_{new} \in [x_0, x_1]$ use either $[x_0, x_{new}]$ or $[x_{new}, x_1]$ (whichever is a bracket). Otherwise throw away x_{new} and use bisection.

1.2.7 Nested multiplication

First note on the efficient polynomial evaluation. Use recursive evaluation as follows:(*Nested multiplication* or *synthetic division*) Fore example,

$$\begin{aligned} p(x) &= x^4 - 5x^3 + 7x^2 - 12x + 13 \\ &= (((x-5)x + 7)x - 12)x + 13. \end{aligned}$$

Comput. costs:

$$4 + 4 + 3 + 2 + 4 = 17, \quad vs \;\; 7.$$

More generally,

$$\begin{aligned} p(x) &= a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0 \\ &= (\cdots(((a_n x + a_{n-1})x + a_{n-2})x + a_{n-3})x \cdots + a_1)x + a_0. \end{aligned} \tag{1.29}$$

In this way, we not only save time but to avoid underflow(imagine what will happen if x is very small and n is large). High power terms like x^n may be neglected if computed individually even if they have some significance if added together.

Horner's Method of Synthetic Division

For a polynomial $p_n(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0$ we can write it as

$$p_n(x) = (x - x_0)q(x) + p(x_0),$$

where $q(x) = b_{n-1}x^{n-1} + \cdots + b_0$. We expand it.

$$\begin{aligned} & (x - x_0)(b_{n-1}x^{n-1} + \cdots + b_0) + p(x_0) \\ = \;& b_{n-1}x^n + (b_{n-1} - x_0 b_{n-2})x^{n-1} + \cdots + (b_0 - x_0 b_1)x + p(x_0) \end{aligned}$$

Comparing this with $p_n(x)$, we see

$$\begin{aligned} b_{n-1} &= a_n \\ b_{n-2} &= a_{n-1} + x_0 b_{n-1} \\ &\vdots \\ b_0 &= a_1 + x_0 b_1 \\ p(x_0) &= a_0 + x_0 b_0. \end{aligned}$$

This is nothing but (1.29). Exer. Derive this and compare the comput. complexity.

Input n, $a_i (0 \le i \le n)$, x_0
$b_{n-1} \leftarrow a_n$

for $k = n-1, n-2, \cdots, 0$ do
  $b_{k-1} \leftarrow a_k + x_0 b_k$
end
output $(b_i, -1 \le i \le n-1)$

Note that $b_{-1} = p(x_0)$.

If this is done by pencil we have

	a_n	a_{n-1}	$\cdots$	a_0
x_0		$x_0 b_{n-1}$	$\cdots$	$x_0 b_0$
$---$	$---$	$-----$	$---$	$---$
$+$	b_{n-1}	b_{n-2}	$\cdots$	b_{-1}

Remark 1.2.14. If you do not need a_k after the work, you can overwrite a_k to save memory. See complete Horner's algorithm.

Example 1.2.15. Use Horner's algorithm for $p(x) = x^4 - 4x^3 + 7x^2 - 5x - 2$ at $x_0 = 3$. Then

$$p(x) = (x-3)(x^3 - x^2 + 4x + 7) + 19.$$

Repeated Horner's method

If we use Horner's method at a fixed value c repeatedly, we can expansion of the poly. and find the derivatives. If we have

$$p(x) = c_n(x - x_0)^n + c_{n-1}(x - x_0)^{n-1} + \cdots c_1(x - x_0) + c_0,$$

we have $f'(x_0) = c_1$ and $f^{(k)}(x_0)/k! = c_k$. To find all the coefficients c_k, we will use Horner's algorithm repeatedly, called a Complete Horner's Algorithm:

input $n, a_i (0 \le i \le n), x_0$
for $k = 0, \cdots, n-1$ do
  for $j = n-1, n-2, \cdots, k$ do
    $a_j \leftarrow a_j + x_0 a_{j+1}$
  end
end
output $a_i (0 \le i \le n)$

Here we overwrite a_j to save memory.

Remark 1.2.16. Horner's method gives an efficient numerical way of computing the higher order derivatives of a polynomial: $p^{(k)}(x_0) = k!c_k$ which may be hard to find by simple repetition of divided difference. But there is a systematic way of computing divided difference. (Later)

Consider Newton's method again:

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

A partial Horner's method can be used to compute $f(x_0)$ and $f'(x_0)$. Thus to find a root of a polynomial by Newton's method, we do not need to provide the derivative. First we define repeated (partial) Horner's method :
Horner (x_0, α, β)

input $n, a_i (0 \le i \le n), x_0$
$\alpha \leftarrow a_n$
$\beta \leftarrow 0$
for $k = n-1, n-2, \cdots, 0$ do
  $\beta \leftarrow \alpha + x_0\beta$
  $\alpha \leftarrow a_k + x_0\alpha$
end
output α, β

Here $\alpha = p(x_0)$ $\beta = p'(x_0)$. Here to order of computation of α, β has to be observed. So this can be combined to give Newton's method:

Newton's Methods for polynomials

input $n, a_i (0 \le i \le n), x_0, M, \epsilon$
for $j = 1$ to M do
  Horner $(n, (a_i : 0 \le i \le n), x_0, \alpha, \beta)$
  $x_1 \leftarrow x_0 - \alpha/\beta$
  output α, β, x_1
  if $|x_1 - x_0| < \epsilon$, then stop
  $x_0 \leftarrow x_1$
enddo

1.2.8 Root of a polynomial

If we are interested in finding one of the zeros of a polynomial and if we know approximate value, then Picard method or Newton method can be used. However, if one is interested in finding many(or all of) roots, above methods are not effective.

Companion matrix

There is a way to find all the zeros of a polynomial using a matrix. A **companion matrix** for a polynomial $p(x) = x^n + a_1 x^{n-1} + \cdots + a_n$ defined as

$$C = \begin{bmatrix} 0 & 1 & 0 & \\ & \ddots & \ddots & 0 \\ & & \ddots & 1 \\ -a_n & -a_{n-1} & \dots & -a_1 \end{bmatrix}$$

Lemma 1.2.17. *We have*

$$p(x) = \det(xI - C) = \det \begin{bmatrix} x & -1 & 0 & \\ & \ddots & \ddots & 0 \\ & & \ddots & -1 \\ a_n & a_{n-1} & \dots & x + a_1 \end{bmatrix}$$

and hence the roots of the polynomial $p(x)$ are the eigenvalues of C.

Proof. Let $p_n(x) := \det(xI - C)$. Use the first row to expand to see

$$\begin{aligned}
p_n(x) &= \begin{vmatrix} x & -1 & \dots & & 0 \\ 0 & x & -1 & \dots & 0 \\ 0 & 0 & x & -1 & 0 \\ \vdots & & & & -1 \\ a_n & a_{n-1} & \dots & a_2 & x+a_1 \end{vmatrix} (n \times n) \\
&= xp_{n-1}(x) - (-1)\begin{vmatrix} 0 & -1 & \dots & & 0 \\ 0 & x & -1 & \dots & 0 \\ 0 & 0 & x & -1 & 0 \\ \vdots & & & & -1 \\ a_n & a_{n-2} & \dots & a_2 & x+a_1 \end{vmatrix} (n-1) \times (n-1) \\
&= xp_{n-1}(x) + (-1)^n a_n \begin{vmatrix} -1 & \dots & & & 0 \\ x & -1 & \dots & & 0 \\ 0 & x & -1 & 0 & 0 \\ \dots & \dots & x & -1 & 0 \\ 0 & & 0 & x & -1 \end{vmatrix} (n-2) \times (n-2) \\
&= xp_{n-1}(x) + a_n \\
&= x(xp_{n-2}(x) + a_{n-1}) + a_n \\
&= x(x(\cdots(xp_1(x) + a_2) + \cdots + a_{n-2}) + a_{n-1}) + a_n,
\end{aligned}$$

with $p_1(x) = x + a_1$. Then this is nothing but the $p(x)$ written in the nested multiplication form. Hence we have $p_n(x) = p(x)$. □

Bairstow' Method for complex roots

It is possible that a polynomial of real coefficients can have complex roots. To find complex roots of such case with a real arithmetic, the method of Bairstow is useful.

Theorem 1.2.18. *We let*

$$p(z) = q(z)(z^2 - uz - v) + r(z),$$

where

$$\begin{aligned} q(z) &= b_n z^{n-2} + b_{n-1} z^{n-3} + \cdots + b_3 z + b_2 \quad (1.30) \\ r(z) &= b_1(z-u) + b_0. \quad (1.31) \end{aligned}$$

Then b_k are given by

$$b_k = a_k + u b_{k+1} + v b_{k+2}, \quad (k = n, n-1, \cdots, 0)$$

where $b_{n+1} = b_{n+2} = 0$.

Proof. Use the method of undetermined coefficients for $p(z) = q(z)(z^2 - uz - v) + r(z)$. From

$$\sum_{k=0}^{n} a_k z^k = \left(\sum_{l=2}^{n} b_l z^{l-2} \right) (z^2 - uz - v) + b_1(z-u) + b_0$$

$$\sum_{k=0}^{n} a_k z^k = \left(\sum_{k=0}^{n-2} b_{k+2} z^k \right) (z^2 - uz - v) + b_1(z-u) + b_0,$$

we see

$$\begin{aligned} a_k &= b_k - u b_{k+1} - v b_{k+2} \quad (0 \le k \le n-2) \quad (1.32) \\ a_{n-1} &= b_{n-1} - u b_n \\ a_n &= b_n. \end{aligned}$$

□

If p has the factor $(z^2 - uz - v)$, then we have

$$\begin{aligned} b_0(u,v) &= 0 \\ b_1(u,v) &= 0. \end{aligned}$$

Here b_0, b_1 are (implicit) functions of u, v. We will apply Newton's Methods here to find u and v from which we can find the root of $p_n(x) = 0$ by the formula.

Let $c_k = \frac{\partial b_k}{\partial u}, d_k = \frac{\partial b_{k-1}}{\partial v}$. We will need c_0, d_1, c_1 and d_2. For this purpose, we have to compute all $c_k's$ and $d_k's$. Note that a_k is independent of u, v. Taking

the derivatives of (1.32) w.r.t u and v

$$\begin{aligned} 0 = \frac{\partial a_k}{\partial u} &= c_k - b_{k+1} - u\frac{\partial b_{k+1}}{\partial u} - v\frac{\partial b_{k+2}}{\partial u} \\ 0 = \frac{\partial a_k}{\partial v} &= d_{k+1} - b_{k+2} - u\frac{\partial b_{k+1}}{\partial v} - v\frac{\partial b_{k+2}}{\partial v} \end{aligned}$$

we obtain

$$\begin{aligned} c_k &= b_{k+1} + uc_{k+1} + vc_{k+2}, \quad (c_{n+1} = c_n = 0) \\ d_k &= b_{k+1} + ud_{k+1} + vd_{k+2}. \end{aligned}$$

From this we see that $c_k = d_k$ and we will use just one of them. We want to compute $\delta u, \delta v$ from

$$\begin{bmatrix} \frac{\partial b_0}{\partial u} & \frac{\partial b_0}{\partial v} \\ \frac{\partial b_1}{\partial u} & \frac{\partial b_1}{\partial v} \end{bmatrix} \begin{bmatrix} \delta u \\ \delta v \end{bmatrix} = - \begin{bmatrix} b_0(u,v) \\ b_1(u,v) \end{bmatrix}$$

In other words,

$$\begin{bmatrix} c_0 & d_1 \\ c_1 & d_2 \end{bmatrix} \begin{bmatrix} \delta u \\ \delta v \end{bmatrix} = \begin{bmatrix} c_0 & c_1 \\ c_1 & c_2 \end{bmatrix} \begin{bmatrix} \delta u \\ \delta v \end{bmatrix} = - \begin{bmatrix} b_0(u,v) \\ b_1(u,v) \end{bmatrix}.$$

So with $J = c_0c_2 - c_1^2$, we have

$$\delta u = (c_1b_1 - c_2b_0)/J, \quad \delta v = (c_1b_0 - c_0b_1)/J.$$

Algorithm-Bairstow

input $n, a_i (0 \le i \le n), u, v, it$
$b_n \leftarrow a_n$
$c_n \leftarrow 0$
$c_{n-1} \leftarrow a_{n-1}$
for $j = 1, 2, \cdot, n-1, n$ do
    $b_{n-1} \leftarrow a_{n-1} + ub_n$
    for $k = n-2, n-3, \cdots, 0$ do
        $b_k \leftarrow a_k + ub_{k+1} + vb_{k+2}$
        $c_k \leftarrow b_{k+1} + uc_{k+1} + vc_{k+2}$
    end
    $J \leftarrow c_0c_2 - c_1^2$

$u \leftarrow u + (c_1 b_1 - c_2 b_0)/J$
$v \leftarrow v + (c_1 b_0 - c_0 b_1)/J$
output j, u, v, b_0, b_1
end

1.3 Systems of nonlinear equations

Let $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$. Seek a point $\bar{\mathbf{x}}$ such that $\mathbf{f}(\bar{\mathbf{x}}) = 0 \in \mathbb{R}^n$, where

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(x_1, \ldots, x_n) \\ \vdots \\ f_n(x_1, \ldots, x_n) \end{bmatrix}.$$

Example 1.3.1. Consider a complex equation $e^{2z} + z^2 + 2 = 0$. Solving

$$e^{2x}(\cos 2y + i \sin 2y) + x^2 - y^2 + i2xy + 2 = 0,$$

we get

$$\begin{cases} e^{2x} \cos 2y + x^2 - y^2 + 2 = 0 \\ e^{2x} \sin 2y + 2xy = 0. \end{cases}$$

A Norm $\| \cdot \|$ on $\mathbb{R}^n \to \mathbb{R}$ is a function satisfying

(1) $\|x\| \geq 0$ and $\|x\| = 0$ iff $x = 0$

(2) $\|\alpha x\| = |\alpha| \|x\|$

(3) $\|x + y\| \leq \|x\| + \|y\|$.

Example 1.3.2. (1) $\|x\|_2 = \sqrt{\sum x_i^2}$

(2) $\|x\|_\infty = \max |x_i|$

(3) $\|x\|_1 = \sum |x_i|$.

Taylor expansion in several variables

$$\begin{aligned} f(x,y) &= f(x_0,y_0) + (x-x_0)\frac{\partial f}{\partial x}(x_0,y_0) + (y-y_0)\frac{\partial f}{\partial y}(x_0,y_0) \\ &\quad + \frac{1}{2}\left((x-x_0)^2\frac{\partial^2 f}{\partial x^2} + 2(x-x_0)(y-y_0)\frac{\partial^2 f}{\partial x \partial y} + (y-y_0)^2\frac{\partial^2 f}{\partial y^2}\right) + \cdots \\ &= f(x_0,y_0) + \left[\tfrac{\partial f}{\partial x}, \tfrac{\partial f}{\partial y}\right]\begin{bmatrix} x-x_0 \\ y-y_0 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} x-x_0 \\ y-y_0 \end{bmatrix}^T \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}\begin{bmatrix} x-x_0 \\ y-y_0 \end{bmatrix} + \cdots \\ &= f(\mathbf{x}_0) + Df(\mathbf{x}_0)\cdot(\mathbf{x}-\mathbf{x}_0) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_0)^T \cdot D^2 f(\mathbf{x}_0)\cdot(\mathbf{x}-\mathbf{x}_0) + \cdots \end{aligned}$$

Here $D^2 f = (\frac{\partial f}{\partial x_i \partial x_j})_{ij}$ is called the Hessian of f.

1.3.1 Picard iteration to solve $\mathbf{f}(\mathbf{x}) = 0$

Assume $\mathbf{f} \in C^2$. Put $\mathbf{x} = \mathbf{x} - A(\mathbf{x})\mathbf{f}(\mathbf{x}) :\equiv \mathbf{g}(\mathbf{x})$, where A is an $n \times n$ nonsingular matrix. Define a Picard iteration

$$\mathbf{x}^{(k+1)} = \mathbf{g}(\mathbf{x}^{(k)}).$$

Theorem 1.3.3. *Suppose the following hold.*

(1) $\mathbf{g}(\boldsymbol{\alpha}) = \boldsymbol{\alpha}$

(2) $\mathcal{B} = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x} - \boldsymbol{\alpha}\|_\infty \le \rho\}$

(3) $\left|\frac{\partial g_i}{\partial x_j}(\mathbf{x})\right| \le \frac{\lambda}{n}$, *for all* i, j *or* $\sum_j \left|\frac{\partial g_i}{\partial x_j}(\mathbf{x})\right| \le \lambda, \quad \forall\, \mathbf{x} \in \mathcal{B}$

(4) $0 < \lambda < 1$.

Then for any $\mathbf{x}^{(0)} \in \mathcal{B}$, *the sequence generated by* $\mathbf{x}^{(k+1)} = \mathbf{g}(\mathbf{x}^{(k)})$ *satisfies*

(1) the sequence $\mathbf{x}^{(k)}$ *remains in* $\mathcal{B}$

(2) $\{\mathbf{x}^{(k)}\} \to \boldsymbol{\alpha}$

(3) $\boldsymbol{\alpha}$ *is unique.*

Proof. For any $\mathbf{x}$, $\mathbf{y} \in \mathcal{B}$, we have

$$g_i(\mathbf{x}) = g_i(\mathbf{y}) + \sum_j \frac{\partial g_i}{\partial x_j}(\xi_{ij})(x_j - y_j).$$

Hence

$$|g_i(\mathbf{x}) - g_i(\mathbf{y})| \le \sum_j \left|\frac{\partial g_i}{\partial x_j}\right| |x_j - y_j| \le \lambda \|\mathbf{x} - \mathbf{y}\|_\infty.$$

Thus $\|\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{y})\|_\infty \le \lambda \|\mathbf{x} - \mathbf{y}\|_\infty$ and $\mathbf{g}$ is a contraction map. Hence

$$\|\mathbf{x}^{(k)} - \boldsymbol{\alpha}\|_\infty \le \lambda \|\mathbf{x}^{(k-1)} - \boldsymbol{\alpha}\|_\infty \le \cdots \le \lambda^k \|\mathbf{x}^{(0)} - \boldsymbol{\alpha}\|_\infty < \lambda^k \rho < \rho.$$

Thus, $\mathbf{x}^{(k)} \in \mathcal{B}$ for all k and $\mathbf{x}^{(k)} \to \boldsymbol{\alpha}$. □

Now we derive a second order scheme: Let

$$\mathbf{g}(\mathbf{x}) = \mathbf{g}(\boldsymbol{\alpha}) + D\mathbf{g}(\boldsymbol{\alpha}) \cdot (\mathbf{x} - \boldsymbol{\alpha}) + \frac{1}{2}(\mathbf{x} - \boldsymbol{\alpha}) \cdot D^2\mathbf{g}(\bar{\xi})(\mathbf{x} - \boldsymbol{\alpha}).$$

Assume $Dg(\boldsymbol{\alpha}) = 0$, and we expect that

$$\|\mathbf{x}^{(k+1)} - \boldsymbol{\alpha}\| \le K \|\mathbf{x}^{(k)} - \boldsymbol{\alpha}\|^2 \qquad \text{(2nd order convergence)}$$

We know that there is ξ_i such that

$$g_i(\mathbf{x}) - g_i(\boldsymbol{\alpha}) = \frac{1}{2} \sum_{j,k} \frac{\partial^2 g_i}{\partial x_j \partial x_k}(\xi_i)(x_j - \alpha_j)(x_k - \alpha_k).$$

Suppose there is a number M such that

$$\max_{i,j,k} \left|\frac{\partial^2 g_i}{\partial x_j \partial x_k}\right| \le \frac{2M}{n^2}$$

then

$$\|\mathbf{x}^{(k)} - \boldsymbol{\alpha}\|_\infty = \|\mathbf{g}(\mathbf{x}^{(k-1)}) - \mathbf{g}(\boldsymbol{\alpha})\|_\infty \le \frac{1}{2} n^2 \cdot \frac{2M}{n^2} \|\mathbf{x}^{(k-1)} - \boldsymbol{\alpha}\|_\infty^2 = M \|\mathbf{x}^{(k-1)} - \boldsymbol{\alpha}\|_\infty^2.$$

Thus we obtain a second order convergence.

Question is how to choose A so that $D\mathbf{g}(\boldsymbol{\alpha}) = 0$? Note the derivative of $A(\mathbf{x})$ is a $n \times n \times n$ matrix. Writing componentwise, we have

$$g_i(\mathbf{x}) = x_i - \mathrm{Row}_i(A(\mathbf{x}))\mathbf{f}(\mathbf{x}) = x_i - \sum_k a_{ik}(\mathbf{x}) f_k(\mathbf{x}).$$

Take derivative w.r.t x_j and set to 0.

$$0 = \frac{\partial g_i}{\partial x_j}(\boldsymbol{\alpha}) = \delta_{ij} - \sum_k (a_{ik}(\boldsymbol{\alpha})\frac{\partial f_k}{\partial x_j}(\boldsymbol{\alpha}) + \frac{\partial a_{ik}}{\partial x_j} f_k(\boldsymbol{\alpha})) = \delta_{ij} - \sum_k a_{ik}(\boldsymbol{\alpha})\frac{\partial f_k}{\partial x_j}(\boldsymbol{\alpha}).$$

In vector form

$$0 = D\mathbf{g}(\boldsymbol{\alpha}) = I - A(\boldsymbol{\alpha})D\mathbf{f}(\boldsymbol{\alpha}).$$

Thus, if we put $A(\mathbf{x}) = (D\mathbf{f}(\mathbf{x}))^{-1}$ then we obtain $\mathbf{g}(x) = \mathbf{x} - (D\mathbf{f}(\mathbf{x}))^{-1}\mathbf{f}(\mathbf{x})$, This is nothing but Newton's method:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - (D\mathbf{f}(\mathbf{x}^{(k)}))^{-1}\mathbf{f}(\mathbf{x}^{(k)}).$$

Theorem 1.3.4. *Let $\mathcal{B} \equiv \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x} - \mathbf{x}^{(0)}\| \le \rho\}$ where $\|\cdot\|$ is any norm, and assume*

(1) $\|\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{y})\| \le \lambda\|\mathbf{x} - \mathbf{y}\|$ *on* $\mathcal{B}$

(2) $0 < \lambda < 1$

(3) $\|\mathbf{g}(\mathbf{x}^{(0)}) - \mathbf{x}^{(0)}\| \le (1 - \lambda)\rho$

holds. Then the sequence generated by $\mathbf{x}^{(k+1)} = \mathbf{g}(\mathbf{x}^{(k)})$ *satisfies*

(1) $\mathbf{x}^{(k)} \in \mathcal{B}$, $\forall k$ *and*

(2) $\exists$ *unique* α *such that* $\lim_k \mathbf{x}^{(k)} = \alpha$ *and* $\mathbf{g}(\alpha) = \alpha$.

Proof. Skip. □

Computational cost of Newton's Method

(1) Evaluation of $\mathbf{f}(\mathbf{x}^{(k)}) \cdots$.

(2) Evaluation of $D\mathbf{f}(\mathbf{x}^{(k)}) \cdots$ may be by finite diff. quotient.

(3) Solution of the linear system $D\mathbf{f}(\mathbf{x}^{(k)})\mathbf{y} = \mathbf{f}(\mathbf{x}^{(k)})$... Most expansive.

1.4 Newton Type Method for Systems

Let $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ be a differentiable map. We consider solving the roots of the equation:

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}.$$

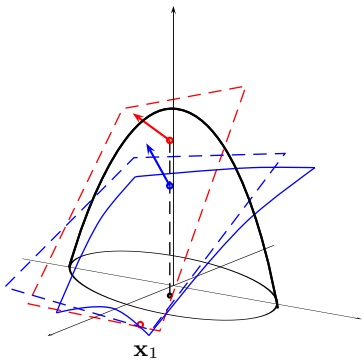


Figure 1.7: Newton's Method- intersection of two tangent plane

Taylor expansion: with $R(\mathbf{x})$ a remainder term, we have

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x}^{(0)}) + D\mathbf{f}(\mathbf{x}^{(0)})(\mathbf{x} - \mathbf{x}^{(0)}) + \mathbf{r}(\mathbf{x}).$$

We replace $\mathbf{f}(\mathbf{x})$ by $\mathbf{f}(\mathbf{x}^{(0)}) + D\mathbf{f}(\mathbf{x}^{(0)})(\mathbf{x} - \mathbf{x}^{(0)})$ and setting

$$\mathbf{f}(\mathbf{x}^{(0)}) + D\mathbf{f}(\mathbf{x}^{(0)})(\mathbf{x} - \mathbf{x}^{(0)}) \approx 0,$$

we obtain an approximate solution. Thus

$$\begin{aligned} \mathbf{x} &\approx \mathbf{x}^{(0)} - [D\mathbf{f}(\mathbf{x}^{(0)})]^{-1}\mathbf{f}(\mathbf{x}^{(0)}) \\ \text{Define } \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} - [D\mathbf{f}(\mathbf{x}^{(k)})]^{-1}\mathbf{f}(\mathbf{x}^{(k)}). \end{aligned}$$

Remark 1.4.1. To compute $[DF(\mathbf{x}_0)]^{-1}\mathbf{f}(\mathbf{x}_0)$, it is not desirable to form the inverse $[D\mathbf{f}(\mathbf{x}_0)]^{-1}$. Instead we solve

$$DF(\mathbf{x}^{(k)})\Delta\mathbf{x}^{(k)} = -\mathbf{f}(\mathbf{x}^{(k)})$$

and set $\mathbf{x}_{k+1} = \mathbf{x}^{(k)} + \Delta\mathbf{x}^{(k)}$. If $DF(\mathbf{x}^{(k)})$ is (close to) singular we expect a poor convergence. We stop if the relative error is under tolerance.

$$\rho^{(k+1)} = \frac{\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|}{\|\mathbf{x}^{(k+1)}\|} < TOL.$$

1.4.1 Broyden's Method

The Newton's method reads:

$$\mathbf{x}^{(k+1)} \approx \mathbf{x}^{(k)} - [D\mathbf{f}(\mathbf{x}^{(k)})]^{-1}\mathbf{f}(\mathbf{x}^{(k)}). \tag{1.33}$$

When n is large, the cost of computing $[D\mathbf{f}(\mathbf{x}^{(k)}]^{-1}\mathbf{f}(\mathbf{x}^{(k)})$ is high. How to reduce the cost?

Quasi-Newton's Method

Replace (1.33) by

$$\mathbf{x}^{(k+1)} \approx \mathbf{x}^{(k)} - A_k\mathbf{f}(\mathbf{x}^{(k)}),$$

where A_k is an approximation of $[D\mathbf{f}(\mathbf{x}^{(k)})]^{-1}$. Typically $A_k \to [D\mathbf{f}(\mathbf{x}^{(k)})]^{-1}$ as $k \to \infty$. Usually the convergence may be slower but the cost of calculating $A_k\mathbf{f}(\mathbf{x}^{(k)})$ is cheaper so that the over all cost is cheap.

One of such method is the Broyden's Method, a secant like method.

Broyden's Method

Recall when $n = 1$ the secant method replaces $f'(x_k)$ by the linear approximation of derivative:

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}. \tag{1.34}$$

This cannot be applied to higher dimensional. However, observe that it is equivalent to

$$f'(x_k)(x_k - x_{k-1}) \approx f(x_k) - f(x_{k-1}). \tag{1.35}$$

We shall derive a matrix which approximate $D\mathbf{f}(\mathbf{x}_k)$. We require

$$A_k(\mathbf{x}_k - \mathbf{x}_{k-1}) = \mathbf{f}(\mathbf{x}_k) - \mathbf{f}(\mathbf{x}_{k-1}). \tag{1.36}$$

Unfortunately this requirement does not define a unique matrix A_k. To define it we shall use A_{k-1} (assuming it is available from previous step) and require A_k satisfies (1.36) only along $\mathbf{x}_k - \mathbf{x}_{k-1}$ direction, while in the direction orthogonal to $\mathbf{x}_k - \mathbf{x}_{k-1}$, A_k has same effect as A_{k-1}, i.e.,

$$A_k\mathbf{z} = A_{k-1}\mathbf{z}, \tag{1.37}$$

for all $\mathbf{z}$ orthogonal to $\mathbf{x}_k - \mathbf{x}_{k-1}$. It can be obtained as a rank one modification of A_{k-1}. It can be verified that the matrix

$$A_k = A_{k-1} + \frac{\mathbf{f}(\mathbf{x}_k) - \mathbf{f}(\mathbf{x}_{k-1}) - A_{k-1}(\mathbf{x}_k - \mathbf{x}_{k-1})}{\|\mathbf{x}_k - \mathbf{x}_{k-1}\|^2}(\mathbf{x}_k - \mathbf{x}_{k-1})^T \qquad (1.38)$$

satisfies both (1.36) and (1.37). This is summarized as Broyden's Method:

	Broyden's Method
1.	Set : $A_0 = D\mathbf{f}(\mathbf{x}_0)$.
2.	Compute $\mathbf{x}_1 = \mathbf{x}_0 - D\mathbf{f}(\mathbf{x}_0)^{-1}\mathbf{f}(\mathbf{x}_0)$
3.	Begin loop : $k = 1$
4.	Compute $\mathbf{x}_{k+1} = \mathbf{x}_k - A_k^{-1}\mathbf{f}(\mathbf{x}_k)$ using (1.38)
5.	Repeat unless certain stopping criteria is met

This method is converges super linearly but takes much less time to converge. It is advised to reset A_k to $D\mathbf{f}(\mathbf{x}_k)$ periodically.

The formula (1.38) is given in the form

$$A_k = A_{k-1} + \mathbf{u}\mathbf{v}^T, \qquad (1.39)$$

where

$$\mathbf{u} = \frac{\mathbf{f}(\mathbf{x}_k) - \mathbf{f}(\mathbf{x}_{k-1}) - A_{k-1}(\mathbf{x}_k - \mathbf{x}_{k-1})}{\|\mathbf{x}_k - \mathbf{x}_{k-1}\|^2}, \quad \mathbf{v} = (\mathbf{x}_k - \mathbf{x}_{k-1})^T.$$

The product of the form $\mathbf{u}\mathbf{v}^T := \mathbf{u} \otimes \mathbf{v}$ is sometimes called **outer product** and

$$\text{rank } \mathbf{u}\mathbf{v}^T = 1.$$

The formation of A_k by the formula (1.39) is called **rank-one update(modification)**. In this case, the computation $A_k^{-1}\mathbf{x}$ is easily carried out using the action $A_{k-1}^{-1}\mathbf{a}$ for some vector $\mathbf{a}$. We can implement the Broyden's method using the Sherman-Morrison formula(1950). [24] H. A. van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG

Sherman-Morrison formula

Lemma 1.4.2 (Sherman-Morrison formula)**.** *If A is nonsingular and $\mathbf{v}^T A^{-1}\mathbf{u}+1 \neq 0$, then $A+\mathbf{u}\mathbf{v}^T$ is nonsingular and*

$$(A+\mathbf{u}\mathbf{v}^T)^{-1} = A^{-1} - \frac{A^{-1}\mathbf{u}\mathbf{v}^T A^{-1}}{1+\mathbf{v}^T A^{-1}\mathbf{u}}. \tag{1.40}$$

Exercise 1.4.3. (1) Derive the formula (1.38) using the rank one update (modification) of A_{k-1}.

(2) Verify Sherman-Morrison formula.

(3) Compute $D\mathbf{f}(\mathbf{x})$ when $F(\mathbf{x}) = A\mathbf{x}+\mathbf{c}$ for an $m \times n$ matrix A and $\mathbf{x} \in \mathbb{R}^n, \mathbf{c} \in \mathbb{R}^m$.

(4) Determine what Newton's method amounts to if the problem is linear.

(5) Explain how you would exploit the Sherman-Morrison formula to Newton's method.

(6) Prove theorem 1.3.4.

1.4.2 Jacobi or Gauss-Seidel type of iteration

First consider a linear equation $A\mathbf{x} = \mathbf{b}$ and consider a splitting the matrix $A = D-L-U$, where D is a diagonal matrix and L, U are strictly lower (resp. upper) triangular matrix. This leads to

$$\begin{aligned} D\mathbf{x} &= (L+U)\mathbf{x}+\mathbf{b} \\ \mathbf{x} &= D^{-1}(L+U)\mathbf{x}^{(k)} + D^{-1}\mathbf{b}. \end{aligned}$$

Hence we define

$$\begin{aligned} \mathbf{x}^{(k+1)} &= D^{-1}(L+U)\mathbf{x}^{(k)} + D^{-1}\mathbf{b} \qquad \text{Jacobi} \\ &= G\mathbf{x}^{(k)} + \mathbf{b}_1. \end{aligned}$$

Another possibility is to consider a splitting of the form

$$\begin{aligned} (D-L)\mathbf{x} &= U\mathbf{x}+\mathbf{b} \\ \mathbf{x}^{(k+1)} &= (D-L)^{-1}U\mathbf{x}^{(k)} + (D-L)^{-1}\mathbf{b} \qquad \text{Gauss-Seidel} \\ &= G\mathbf{x}^{(k)} + \mathbf{b}_2. \end{aligned}$$

These scheme will be shown to converges if $\rho(G) < 1$, where

$$G = D^{-1}(L+U), \text{ or } (D-L)^{-1}U.$$

Recall $\rho(G) \leq \|G\|$ for any norm. Hence they will converge, in particular, if $\|G\| < 1$ for some norm.

Jacobi type

Consider the *Jacobi type* iteration for a system of nonlinear equation choose $\mathbf{x}^{(0)} = (x_1^{(0)}, \ldots, x_n^{(0)})$. For $k = 0, 1, 2, \ldots$ calculate $\mathbf{x}^{(k+1)}$ from the relation

$$\begin{cases} f_1(x_1^{(k+1)}, x_2^{(k)}, \ldots, x_n^{(k)}) = 0 \\ f_2(x_1^{(k)}, x_2^{(k+1)}, x_3^{(k)}, \ldots, x_n^{(k)}) = 0 \\ \quad \cdots \\ f_n(x_1^{(k)}, \ldots, x_{n-1}^{(k)}, x_n^{(k+1)}) = 0. \end{cases}$$

If we solved(express) explicitly, we may write

$$\begin{cases} x_1^{(k+1)} = g_1(x_2^{(k)}, \ldots, x_n^{(k)}) \\ x_2^{(k+1)} = g_2(x_1^{(k)}, x_3^{(k)}, \ldots, x_n^{(k)}) \\ \quad \cdots \\ x_n^{(k+1)} = g_n(x_1^{(k)}, \ldots, x_{n-1}^{(k)}). \end{cases}$$

In practice, solving for each $x_i^{(k+1)}$, $i = 1, 2, \cdots, n$ one can use root finding scheme developed earlier, i.e., bisection, secant or Newton's method again. To use Newton's method, one can proceed as follows:

Treat x_1 as a function of $x_2, \ldots, x_n$ and take the total derivative of $f_1(x_1, \ldots, x_n) = 0$ w.r.t. x_2, we get

$$0 = \frac{df_1}{dx_2} = \frac{\partial f_1}{\partial x_1} \cdot \frac{\partial x_1}{\partial x_2} + \frac{\partial f_1}{\partial x_2} \quad \therefore \ \frac{\partial x_1}{\partial x_2} = \frac{\partial g_1}{\partial x_2} = -\frac{\partial f_1/\partial x_2}{\partial f_1/\partial x_1}.$$

Set $x_1^{(k+1)} = x_1^{(k)} - \frac{\partial f_1/\partial x_2}{\partial f_1/\partial x_1}$. In general, Treat x_i as a function of $x_1, x_2, \ldots, x_n$ except x_i and take the derivative of $f_i(x_1, \ldots, x_n) = 0$ w.r.t x_j, we get

$$\frac{\partial f_i}{\partial x_i}\frac{\partial x_i}{\partial x_j} + \frac{\partial f_i}{\partial x_j} = 0, \quad (j \neq i) \Rightarrow \frac{\partial x_i}{\partial x_j} = -\frac{\partial f_i}{\partial x_j} \Big/ \frac{\partial f_i}{\partial x_i}.$$

For $k = 1, 2, \cdots$,
For $i = 1, 2, \cdots, n$ do:

$$x_i^{(k+1)} = x_i^{(k)} - \frac{\partial f_i}{\partial x_j} / \frac{\partial f_i}{\partial x_i}(x_i^{(k)}).$$

$$DG = \begin{pmatrix} 0 & \times & \times \\ \times & \ddots & \times \\ \times & \times & 0 \end{pmatrix}.$$

This scheme will converge if $\|DG\| \leq \rho < 1$.

Gauss-Seidel type

We solve for $x_1^{(k+1)}, x_1^{(k+2)}, \cdots$ from the equations given in the following order:

$$\begin{cases} f_1(x_1^{(k+1)}, x_2^{(k)}, \ldots, x_n^{(k)}) = 0 \\ f_2(x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k)}) = 0 \\ \qquad\qquad \cdots \\ f_n(x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k+1)}) = 0. \end{cases}$$

SOR(Successive over relaxation) type

This is a combination between Jacobi and Gauss-Seidel type: For $k = 0, 1, \cdots$ do: put

$$x_i^{(k+1)} = (1 - \omega)x_i^{(k)} + \omega x_i^{(k+\frac{1}{2})},$$

where $x_i^{k+\frac{1}{2}}$ for $i = 1, 2 \cdots, n$ is computed as follows:

$$\begin{cases} f_1(x_1^{(k+\frac{1}{2})}, x_2^{(k)}, \ldots, x_{n-1}^{(k)}, x_n^{(k)}) = 0 \\ f_2(x_1^{(k+1)}, x_2^{(k+\frac{1}{2})}, x_3^{(k)}, \ldots, x_n^{(k)}) = 0 \\ \qquad\qquad \cdots \\ f_n(x_1^{(k+1)}, \ldots, x_{n-1}^{(k+1)} \ldots, x_n^{(k+\frac{1}{2})}) = 0. \end{cases}$$

When $\omega = 0$, SOR is equal to Jacobi and when $\omega = 1$, SOR is equal to Gauss-Seidel method. However, the value $x_i^{k+\frac{1}{2}}$ is different from the value computed in Gauss-Seidel method even though they look similar.(At each step $i = 1, 2, \cdots, n$, we solve similar i-th equation but with updated value by

$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \omega x_i^{(k+\frac{1}{2})})$.

Chapter 2

Approximation and interpolation

2.1 Piecewise linear approximation

Given $\{(x_i, f(x_i)\}_{i=0}^m$ we want to approximate $f(x)$ for $x \notin \{x_0, \ldots, x_m\}$ by some piecewise linear approximation. Assume

(1) f is continuous.

(2) $x_{k+1} - x_k$ are small for $k = 0, \ldots, m-1$.

(3) $x \in [x_0, x_m]$

For $x \in (x_k, x_{k+1})$ define

$$\begin{aligned} L(x) &\equiv f(x_k) + \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k}(x - x_k) \\ &= \frac{x_{k+1} - x}{x_{k+1} - x_k} f(x_k) + \frac{x - x_k}{x_{k+1} - x_k} f(x_{k+1}) \\ &= w_0(x) f(x_k) + w_1(x) f(x_{k+1}), \quad w_0 + w_1 = 1,\ 0 \le w_0, w_1 \le 1 \end{aligned}$$

Then $\lim_{m\to\infty} L(x) = f(x)$ if mesh $\to 0$ uniformly and $f \in C^0[a, b]$.

Proof.

$$\begin{aligned} f(x) - L(x) &= f(x) - w_0(x)f(x_k) - w_1(x)f(x_{k+1}) \\ &= w_0[f(x) - f(x_k)] + w_1[f(x) - f(x_{k+1})] \\ |f(x) - L(x)| &\leq |w_0|\,|f(x) - f(x_k)| + |w_1|\,|f(x) - f(x_{k+1})] \\ &\leq \max\{|f(x) - f(x_k)|, |f(x) - f(x_{k+1})|\} \end{aligned}$$

□

Moreover, if $f(x) \in C^2[a, b]$, we have the following result.

Theorem 2.1.1. *Assume $f(x) \in C^2[a, b]$ and $x \in (a, b)$ is given. If $L(x) = f(a) + \dfrac{f(b) - f(a)}{b - a}(x - a)$ then $\exists\ c(x) \in (a, b)$ such that*

$$f(x) - L(x) = \frac{(x - a)(b - x)}{2} f''(c(x))$$

and moreover, if $|f''(x)| \leq M_2$, then $|f(x) - L(x)| \leq \frac{(b-a)^2}{8} M_2$.

Proof. Use the Taylor expansion for $f(a)$ and $f(b)$ at x to get

$$f(a) = f(x) + f'(x)(a - x) + \frac{(a - x)^2}{2} f''(c_1) \tag{2.1}$$

$$f(b) = f(x) + f'(x)(b - x) + \frac{(b - x)^2}{2} f''(c_2),\ a < c_1 < x < c_2 < b. \tag{2.2}$$

Hence

$$f(b) - f(a) = (b - a)f'(x) + \frac{(b - x)^2}{2} f''(c_2) - \frac{(a - x)^2}{2} f''(c_1). \tag{2.3}$$

Substitute (2.1)-(2.3) into $L(x)$ to see

$$\begin{aligned} L(x) &= f(x) + f'(x)(a-x) + \frac{(a-x)^2}{2} f''(c_1) \\ &\quad + f'(x) + \frac{(b-x)^2 f''(c_2) - (a-x)^2 f''(c_1)}{2(b-a)}(x-a) \\ &= f(x) + \left[\frac{(a-x)^2}{2} - \frac{(a-x)^2}{2(b-a)}(x-a) \right] f''(c_1) + \frac{(b-x)^2(x-a)}{2(b-a)} f''(c_2) \\ &= f(x) + \frac{(a-x)^2(b-x)}{2(b-a)} f''(c_1) + \frac{(b-x)^2(x-a)}{2(b-a)} f''(c_2) \\ &= f(x) + \alpha_1(x) f''(c_1) + \alpha_2(x) f''(c_2), \quad 0 < \alpha_1(x), \alpha_2(x). \end{aligned}$$

We want to estimate

$$L(x) - f(x) = \alpha_1(x) f''(c_1) + \alpha_2(x) f''(c_2).$$

To do so, put $\psi(\xi) = \alpha_1(x) f''(c_1) + \alpha_2(x) f''(c_2) - (\alpha_1 + \alpha_2) f''(\xi)$. Note that we treat x as constant here. Since $\psi(c_1)\psi(c_2) = -\alpha_1\alpha_2(f''(c_2) - f''(c_1))^2 < 0$ there is a $c(x) \in (c_1, c_2)$ such that $\psi(c(x)) = 0$. Thus

$$\begin{aligned} L(x) - f(x) &= (\alpha_1 + \alpha_2) f''(c(x)) = \frac{(a-x)(b-x)}{2(b-a)}(a - x - b + x) f''(c(x)) \\ &= \frac{(x-a)(b-x)}{2} f''(c(x)). \end{aligned}$$

Now to derive the error bound, we observe $\max_{[a,b]} |x-a||x-b| = \frac{(b-a)^2}{4}$. □

2.2 Basic problem of approximation theory

Suppose V is a finite dimensional subspace of $C[a,b]$. Given $f \in C^n[a,b]$, find $p \in V$ so that p is as close to f as possible. Usually $V = P_n\{\text{polynomial of degree } \leq n\}$. We need some kind of norm to measure a function in V.

Example 2.2.1. The following is a norm if $V = P_n$.

$$\|f(x)\| = \sum_{i=0}^{n} |f(x_i)|.$$

Question

(1) Does there exist a function $p \in V$ such that $\|f - p\|$ is minimum?

(2) Unique?

(3) How to find it?

(4) How close is V to C^k?

(5) What basis for V is convenient?

Example 2.2.2. Let $f \in C^n[a,b]$ and $V = P_n$. Let $x_0 \in [a,b]$ be a fixed point. If we define a new norm $\|g\|_t = \sum_{k=0}^{n} |g^{(k)}(x_0)|$ then, the truncated Taylor series at x_0 is the best approximation w.r.t $\|\cdot\|_t$, i.e, if $p(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k$ then $\|f - p\|_t = 0$.

Let $f \in C^0[a,b]$ and let $K = \{x_0, x_1, \ldots, x_n\} \subset [a,b]$. Define $V \equiv PL(a,b\,;\,K)$: the set of continuous functions which is piecewise linear on each subintervals (x_i, x_{i+1}). Then $\dim V = n+1$ and basis functions are

$$T_i(x) = \begin{cases} \dfrac{x - x_{i-1}}{x_i - x_{i-1}} & \text{on } [x_{i-1}, x_i] \\ \dfrac{x - x_{i+1}}{x_i - x_{i+1}} & \text{on } [x_i, x_{i+1}] \\ 0 & \text{elsewhere} \end{cases}$$

In this case $\exists$ unique $p \in V$ such that $\|f - p\|_t = 0$.

2.3 Approximation by polynomial interpolation

Definition 2.3.1. Interpolation of a given function f defined on an interval $[a,b]$ by a polynomial p: Given a set of specified points $\{(x_i, f(x_i)\}_{i=0}^{n}$ with $\{x_i\} \subset [a,b]$, the polynomial p of degree n satisfying

$$p(x_i) = f(x_i), \quad i = 1, \cdots, n$$

is called an **polynomial interpolation**. Nonpolynomial interpolation can be defined, but rarely used.

Here we shall use the semi-norm to measure the error:

$$\|f\| \equiv \sum_{i=0}^{n} |f(x_i)|.$$

Theorem 2.3.2. *If $V \equiv P_n(x)$, polynomial of degree n, then for given f, $\exists!$ $p \in V$ such that $\|f - p\| = 0$.*

Proof. We want to find a polynomial of the form $p = a_0 + a_1 x + \cdots + a_n x^n$ such that $p(x_i) = f(x_i)$, for $i = 0, \ldots, n$, i.e,

$$\begin{aligned} p(x_1) &= a_0 + a_1 x_1 + \cdots + a_n x_1^n = f(x_1) \\ &= \cdots \\ p(x_n) &= a_0 + a_1 x_n + \cdots + a_n x_n^n = f(x_n) \end{aligned}$$

In matrix form,

$$\begin{bmatrix} 1 & x_0 & \cdots & x_0^n \\ 1 & x_1 & \cdots & x_1^n \\ & & \cdots & \\ 1 & x_n & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f(x_0) \\ \vdots \\ f(x_n) \end{bmatrix} \tag{2.4}$$

This equation involves a Van der Monde matrix whose determinant is $\prod_{i>j}(x_i - x_j)$. Thus we see a unique solution exists as long as the interpolation points are distinct. □

A basis for $V = P_n$

A naive basis is $\{1, x, , \ldots, x^n\}$. Is it convenient? No!

(1) To compute the coefficients in (2.4), one has to solve a linear system.

(2) Moreover, the Van der Monde matrix is extremely ill-conditioned!

Example 2.3.3. Consider

$$\begin{aligned} x \qquad\quad -y &= 1+\epsilon \\ (1+10^{-9})x \quad -y &= 1 \end{aligned} \tag{2.5}$$

Without ϵ, the exact solution is $x = 0, y = -1$, while with ϵ perturbation, $x = -10^9\epsilon$, $y = -10^9\epsilon - 1 - \epsilon$!

Lagrange basis

Construct a polynomial basis $w_i(x)$ such that $w_i(x_j) = \delta_{ij}, \quad i, j = 0, \cdots, n$. Start with the polynomial

$$w_i(x) = k \prod_{j \neq i}^{n} (x - x_j)$$

so that $w_i(x_j) = \delta_{ij},\ j = 0, \cdots, n$ is easily satisfied. We require

$$w_i(x_i) \quad = \quad k \prod_{j \neq i} (x_i - x_j) = 1$$

Then we obtain $\frac{1}{k} = \prod_{j \neq i}(x_i - x_j)$ and hence

$$w_i(x) = \prod_{j \neq i} \frac{(x - x_j)}{(x_i - x_j)}.$$

These are the **Lagrange interpolating polynomials**. Now using these, one can readily construction a polynomial interpolation.

Proposition 2.3.4. *Let $f \in C[a, b]$ and let p_n be the unique element of $P_n(x)$ such that $f(x_i) = p_n(x_i)$, $i = 0, 1, \ldots, n$. Then*

$$p_n(x) = \sum_{i=0}^{n} f(x_i) \prod_{j \neq i} \frac{(x - x_j)}{(x_i - x_j)}.$$

Question

(1) How big is $\|f(x) - p_n(x)\|_\infty$?

(2) What happens if nodes are close?

(3) $\lim_{n \to \infty} p_n(x) =$?

(4) Is Lagrange best?

Error Estimate

Theorem 2.3.5. *Let $f(x) \in C^{n+1}[a, b]$. If $a = x_0, x_1, \ldots, x_n = b$ are $n + 1$ distinct points and $p_n(x)$ is the Lagrange (any) interpolating polynomial, then there exists a function $\xi(x) \in (a, b)$ such that*

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i).$$

Proof. Let $d(x) = \prod_{i=0}^{n}(x - x_i)$ and define

$$\Phi(x) \equiv \frac{f(x) - p_n(x)}{d(x)} \equiv \frac{R(x)}{d(x)}. \tag{2.6}$$

First we claim $\Phi(x)$ has a removable singularity at x_j. In fact we see by L'Hopital's rule

$$\lim_{x\to x_j}\Phi(x)=\frac{R'(x_j)}{\prod_{i\neq j}(x_j-x_i)}<\infty.$$

Thus $\Phi(x)$ is regarded as continuous.

In fact $\Phi\in C^{(n+1)}[a,b]$. Let x be fixed(not a variable!) different from x_i. Then the function defined by

$$\Omega(z)=f(z)-p_n(z)-d(z)\Phi(x)\in C^{(n+1)}[a,b]$$

vanishes(as a function of z) at $x_i, i=0,\cdots,n$ ($n+1$ nodal points). But it also vanishes at x by (2.6). Thus $\Omega(z)$ vanishes at $n+2$ distinct points in (a,b). Thus by Rolle's Theorem,

$$\begin{aligned}
\Omega'(z) &\qquad \text{has } n+1 \text{ distinct zeros in } (a,b)\\
\Omega''(z) &\qquad \text{has } n \text{ distinct zeros in } (a,b)\\
&\cdots\\
\Omega^{(n+1)}(z) &\qquad \text{has at least one zero in } (a,b).
\end{aligned}$$

We denote the last zero by $\xi(x)$. Thus

$$\Omega^{(n+1)}(\xi(x))=0=f^{(n+1)}(\xi)-(n+1)!\,\Phi(x).$$

Hence

$$\begin{aligned}
\Phi(x)\equiv\frac{f(x)-p_n(x)}{d(x)}&=\frac{f^{(n+1)}(\xi)}{(n+1)!}\\
\therefore\qquad f(x)-p_n(x)&=\frac{f^{(n+1)}(\xi(x))}{(n+1)!}\prod_{i=0}^{n}(x-x_i).
\end{aligned}$$

□

We now estimate $\prod_{i=0}^{n}(x-x_i)$ for the uniformly spaced case($x_i-x_{i-1}=h$) as follows: First one can easily see that the maximum value is assumed at

either of the extreme intervals (x_0, x_1) or (x_{n-1}, x_n). Thus

$$\begin{aligned}\max\left|\prod_{i=0}^{n}(x-x_i)\right| &\leq \max_{(x_0,x_1)}|(x-x_0)(x-x_1)| \max_{(x_0,x_1)}|(x-x_2)(x-x_3)|\cdots(x-x_n)| \\ &\leq \frac{h^2}{4}\cdot(2h)\cdots(hn) = \frac{h^{n+1}}{4}n!.\end{aligned}$$

Thus $\|f(x)-p(x)\|_\infty \leq \frac{h^{n+1}}{4(n+1)}\max|f^{(n+1)}(x)|$.

Another way.(Kincaid) Assume $x_j < x < x_{j+1}$. Then

$$\begin{aligned}\left|\prod_{i=0}^{n}(x-x_i)\right| &\leq \max_{(x_j,x_{j+1})}|(x-x_j)(x-x_{j+1})|\cdot|\prod_{i=0}^{j-1}(x-x_i)|\cdot|\prod_{i=j+1}^{n}(x-x_i)| \\ &\leq \frac{h^2}{4}\prod_{i=0}^{j-1}(x_{j+1}-x_i)\cdot\prod_{i=j+1}^{n}(x_i-x_{j+1}) \\ &\leq \frac{h^2}{4}\prod_{i=0}^{j-1}(j-i+1)h\cdot\prod_{i=j+1}^{n}(i-j)h \\ &= \frac{h^{n+1}}{4}n!.\end{aligned}$$

Now we consider the second and third question. Polynomial of degree n has the tendency of $n-1$ oscillation(has $n-1$ local extreme points). Thus, if the interval is fixed and n gets higher(as a result the interpolating points will get close together), it may oscillate unnecessarily. Indeed the following example by Runge shows it.

Example 2.3.6 (Runge)**.**

$$f(x) = \frac{1}{1+x^2} \qquad \text{on} \quad [-a, b]$$

Given $\{(x_i, f(x_i)) \mid i = 0, 1, \ldots, n\}$, let

$$h = \frac{b+a}{n}, \qquad x_k = x_0 + kh, \quad k = 0, \ldots, n.$$

Interpolate $f(x)$ by polynomial of degree n.

$$p_n(x_k) = f(x_k), \qquad k = 0, 1, \ldots, n.$$

Runge showed $\lim_{n\to\infty}\|f(x)-p_n(x)\|_\infty = \infty$.

Thus Runge's example shows higher degree polynomial is not always good for interpolation. This suggests us to use lower degree polynomial on each subinterval.

Exercise 2.3.7. (1) Write down a computer code to find a Newton form of interpolating polynomial(call NIP)

(a) use it to find Newton inter. poly $p_k\,(k = 4, 5, 6)$ for $f(x) = x^8 - x^7 + 5x^4 - 23x^2 - x$ on $[0, 2]$

$$(x_i, f(x_i))_{i=0}^{k},\ x_i = \frac{2}{k}.$$

Draw the graph of $f(x)$ and p_k on the same plane to compare.

(b) Repeat the same for $f(x) = \sin x$ on $[0, \frac{\pi i}{2}]$ with

$$(x_i, f(x_i))_{i=0}^{8},\ x_i = \frac{\pi i}{16}.$$

Draw the graph of $f(x)$ and p_k to compare.

(2) Find poly. interpolation of Runge example (with uniformly spaced points) for $a = b = 5$ with $n = 5, 10, 20, 40$ (polynomial of degree n). Draw the graph and see what happens as n grows.

(3) Is there anyway to improve if you are allowed to change the location of points ?

Theorem 2.3.8 (Weierstrass approximation theorem)**.** *The set of all polynomials (usually denoted by $P[a, b]$) is uniformly dense in $C[a, b]$.*

Bernstein polynomial

We have a concrete construction of poly. approximation satisfying Weierstrass theorem by Bernstein. Assume $I = [0, 1]$. Let

$$\beta_{n,j}(x) \equiv \binom{n}{j} x^j (1 - x)^{n-j} \in P_n(x)$$

and set

$$B_n(f\,;\,x) \equiv \sum_{j=0}^{n} f(x_j)\beta_{n,j}(x), \qquad x_j = j/n.$$

Let $\beta_{n,j}(x) \equiv \binom{n}{j} x^j (1-x)^{n-j} \in P_n(x)$, on $[0,1]$.

$$B_n(f\,;\,x) \equiv \sum_{j=0}^{n} f(x_j)\beta_{n,j}(x), \qquad x_j = j/n.$$

Then we have

$$\sum_{j=0}^{n} \beta_{n,j}(x) = 1 \tag{2.7}$$

$$\sum_{j=0}^{n} \frac{j}{n}\beta_{n,j}(x) = x \tag{2.8}$$

$$\sum_{j=0}^{n} \frac{j^2}{n^2}\beta_{n,j}(x) = (1-\frac{1}{n})x^2 + \frac{x}{n} \tag{2.9}$$

Proof. (1) Expand $(x+1-x)^n$.

(2) We have $(a+b)^n = \sum_{j=0}^{n} \binom{n}{j} a^j b^{n-j}$. Differentiate $(a+b)^n$ w.r.t a, to obtain

$$n(a+b)^{n-1} = \sum_{j=1}^{n} \binom{n}{j} j a^{j-1} b^{n-j}.$$

Multiply by a/n,

$$a(a+b)^{n-1} = \sum_{j=1}^{n} \frac{j}{n} \binom{n}{j} a^j b^{n-j}.$$

Put $a = x$, $b = 1-x$,

(3) Differentiate twice $(a+b)^n$ w.r.t a, to obtain

$$n(n-1)(a+b)^{n-2} = \sum_{j=2}^{n} \binom{n}{j} j(j-1) a^{j-2} b^{n-j}.$$

Multiply by a^2/n^2, we get

$$\frac{a^2 n(n-1)(a+b)^{n-2}}{n^2} = \sum_{j=2} c\frac{j(j-1)}{n^2} \binom{n}{j} a^j b^{n-j}.$$

Put $a = x$, $b = 1 - x$,

$$\begin{aligned} x^2(1-\frac{1}{n}) &= \sum_{j=2}^{n} \frac{j^2}{n^2} \binom{n}{j} x^j (1-x)^{n-j} - \sum_{j=2}^{n} \frac{j}{n^2} \binom{n}{j} x^j (1-x)^{n-j} \\ &= \sum_{j=0}^{n} \frac{j^2}{n^2} \beta_{n,j}(x) - \frac{1}{n} \sum_{j=0}^{n} \frac{j}{n} \beta_{n,j}(x) \\ &= \sum_{j=0}^{n} \frac{j^2}{n^2} \beta_{n,j}(x) - \frac{x}{n} \quad \text{by (2).} \end{aligned}$$

(4) By induction, we can show $\{\beta_{n,j}(x)\}$ form a basis for $P_n(x)$.

(5) For $x \in [0,1]$, $\beta_{n,j} \geq 0$. □

Definition 2.3.9. $\omega(f,\delta) = \text{lub } |f(x) - f(x')|$, $|x - x'| < \delta$, x, $x' \in [0,1]$ is called the modules of continuity of f.

Theorem 2.3.10. *If* $f(x) \in C[0,1]$, $\|f(x) - B_n(f\,;x)\|_\infty \leq \frac{9}{4}\omega(f, n^{-\frac{1}{2}})$.

Remark 2.3.11. This is not an interpolating approximation in general.

Proof. Fix x. Then

$$f(x) - B_n(f\,;x) = \sum_{j=0}^{n} [f(x) - f(x_j)]\beta_{n,j}(x) = s_1(x) + s_2(x),$$

where

$$s_1(x) = \sum_{j \in I} [f(x) - f(x_j)]\beta_{n,j}(x), \qquad I = \{j : |x - x_j| < \delta\}$$

so that

$$|s_1(x)| \leq \sum_{j \in I} |f(x) - f(x_j)|\beta_{n,j}(x) \leq \omega(f,\delta) \sum_{j \in I} \beta_{n,j}(x) \leq \omega(f,\delta).$$

Now

$$|s_2(x)| \leq \sum_{j \notin I} |f(x) - f(x_j)|\beta_{n,j}(x).$$

Insert $\{\xi_i\}_{i=1}^{p}$ uniformly between x and x_j where $p \equiv [\frac{x - x_j}{\delta}]$, so that $\frac{x - x_j}{p+1} < \delta$. Then

$$f(x) - f(x_j) = [f(x) - f(\xi_1)] + [f(\xi_1) - f(\xi_2)] + \cdots + [f(\xi_p) - f(x_j)]$$

and

$$\begin{aligned}
|s_2(x)| &\leq \sum_{j\notin I}(p+1)\omega(f,\delta)\beta_{n,j}(x) \leq \sum_{j\notin I}(1+\frac{|x-x_j|}{\delta})\omega(f,\delta)\beta_{n,j}(x) \\
&\leq \omega(f,\delta)[1+\frac{1}{\delta^2}\sum_{j=0}^{n}(x^2-2xx_j+x_j^2)\beta_{n,j}(x)](x_j=\frac{j}{n}) \\
&= \omega(f,\delta)[1+\frac{1}{\delta^2}(x^2-2x^2+(1-\frac{1}{n})x^2+\frac{1}{n}x)] \\
&= \omega(f,\delta)[1+\frac{x(1-x)}{\delta^2 n}] \leq \omega(f,\delta)(1+\frac{1}{4\delta^2 n}).
\end{aligned}$$

Now choosing $\delta = 1/\sqrt{n}$ we have

$$|f(x)-B_n(f\,;\,x)| \leq |s_1(x)|+|s_2(x)| \leq \omega(f,\delta)(2+\frac{1}{4\delta^2 n}) = \frac{9}{4}\omega(f\,;\,n^{-\frac{1}{2}}).$$

□

Corollary 2.3.12. (Weierstrass Approximation Theorem) *If $f \in C[0,1]$ and $\varepsilon > 0$ is given, then there exists an integer $n(\varepsilon)$ and $p_n(x) \in P_n(x)$ such that for all $n \geq n_0$, it holds that*

$$\|f(x)-p_n(x)\|_\infty < \varepsilon.$$

Caution: This $p_n(x)$ is not an interpolating polynomial.

Newton's form of interpolating polynomial

We now describe an efficient way of calculating the coefficients of an interpolating polynomial. We know the Lagrangian form of interpolation is

$$p_n(x) = \sum_{i=0}^{n} f(x_i)\prod_{j\neq i}^{j=n}\frac{x-x_j}{x_i-x_j}.$$

This may be useful to analyze, but it is not efficient for computation. We now present a Newton form of interpolating polynomial. The idea is to compute $p_n(x)$ by induction using the se

$$\{1,(x-x_0),(x-x_0)(x-x_1),\ldots,\prod_{i=0}^{n-1}(x-x_i)\}$$

as another basis for $P_n(x)$. Since $p_n(x)$ is one degree higher than $p_{n-1}(x)$, one can set

$$p_n(x) = p_{n-1}(x) + q_n(x),$$

with $q_n(x_j) = 0$, $j = 0, 1, \ldots, n-1$. Then $p_n(x_i) = p_{n-1}(x_i)$ for $i = 0, \cdots, n-1$ and we hope $p_n(x_n) = f(x_n)$. Thus $q_n(x) = a_n \prod_{i=0}^{n-1}(x - x_i)$ and

$$\begin{aligned} p_n(x) &= p_{n-1}(x) + a_n \prod_{i=0}^{n-1}(x - x_i) \\ &= p_{n-2}(x) + a_{n-1} \prod_{i=0}^{n-2}(x - x_i) + a_n \prod_{i=0}^{n-1}(x - x_i) \\ &\quad \cdots \\ &= a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n \prod_{i=0}^{n-1}(x - x_i). \end{aligned}$$

If we denote $f[x_0, \ldots, x_k]$ for a_k, then

$$p_n(x) = \sum_{k=0}^{n} a_k \prod_{i=0}^{k-1}(x - x_i) \equiv \sum_{k=0}^{n} f[x_0, \ldots, x_k] \prod_{i=0}^{k-1}(x - x_i). \tag{2.10}$$

This is called the **Newton's form of interpolation** and $f[x_0, \ldots, x_k]$ are called the **divided difference**.

`https://en.wikipedia.org/wiki/Newton_polynomial`

For $k+1$ data points we construct the Newton basis as With $n_k(x) = \prod_{j=0}^{k-1}(x - x_j)$, $k = 0, \cdots, n$

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & x_1 - x_0 & 0 & \cdots & 0 \\ 1 & x_2 - x_0 & \prod_{j=0}^{1}(x_2 - x_j) & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ 1 & x_k - x_0 & \cdots & \cdots & \prod_{j=0}^{k-1}(x_k - x_j) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \\ \vdots \\ \\ a_k \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \\ \vdots \\ \\ y_k \end{bmatrix}$$

System is written as

$$\sum_{j=0}^{k} a_j n_j(x_k) = y_k, \ k = 0, \cdots, n.$$

Solving we get

$$\begin{array}{lllll} a_0 & & & & = y_0 \\ a_0 & +a_1(x_1-x_0) & & & = y_1 \\ a_0 & +a_1(x_2-x_0) & +a_2(x_2-x_0)(x_2-x_1) & & = y_2 \\ . & \cdots & \cdots & & . \\ a_0 & +a_1(x_k-x_0) & +a_2(x_k-x_0)(x_2-x_1) & \cdots \quad +a_k\prod_{j=0}^{k-1}(x_k-x_j) & = y_k \end{array}$$

Computing the divided difference

Example 2.3.13. (1) $f[x_i] = f(x_i)$.

(2) $f[x_0, x_1] = \frac{f[x_1]-f[x_0]}{x_1-x_0} = \frac{f(x_1)-f(x_0)}{x_1-x_0}$.

(3) $f[x_0, x_1, x_2] = \frac{f[x_1,x_2]-f[x_0,x_1]}{x_2-x_0}$.

(4) $p_2(x) = f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1)$.

3.

$$y_2 - y_0 - a_1(x_2 - x_0) = f[x_2] - f[x_0] - f[x_0, x_1](x_2 - x_0)$$

Hence

$$f[x_0, x_1, x_2] = \frac{y_2 - y_0 - f[x_0, x_1](x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)} = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}$$

Now we show how to compute $f[x_0, \ldots, x_k]$ efficiently. We write $p_n(x)$ in two ways(*by reverse ordering starting from* x_n),

$$a_0 + a_1(x - x_0) + \cdots + a_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}) = \sum_{k=0}^{n} a_k \prod_{i=0}^{k-1}(x - x_i)$$

$$b_0 + b_1(x - x_n) + \cdots + b_n(x - x_n)(x - x_{n-1})\cdots(x - x_1) = \sum_{k=0}^{n} b_k \prod_{i=n}^{n-k+1}(x - x_i),$$

where $b_n = f[x_n, \ldots, x_0]$ by definition. Comparing the highest order term, we get

$$a_n = f[x_0, \ldots, x_n] = b_n = f[x_n, \ldots, x_0].$$

Reordering the points, we also see that

$$b_n := f[x_n, \ldots, x_0] = f[x_{i(0)}, \ldots, x_{i(n)}],$$

for any permutation $i(k)$ of numbers $\{0, 1, \cdots, n\}$. Hence the **divided difference** is independent of the order of its arguments x_i.

Subtract two expressions for p_n to get

$$0 = a_n[(x - x_0) - (x - x_n)](x - x_1)\cdots(x - x_{n-1}) + (a_{n-1} - b_{n-1})x^{n-1} + \cdots$$

Comparing the coefficients of x^{n-1}, we see

$$a_n(x_n - x_0) + (a_{n-1} - b_{n-1}) = 0.$$

Since $b_{n-1} = f[x_n, \ldots, x_1] = f[x_1, \ldots, x_n]$ and $a_{n-1} = f[x_0, \ldots, x_{n-1}]$($\because$ b_{n-1} is defined using n points $x_n, \cdots x_1$ and a_{n-1} is defined using n points $x_0, \cdots x_{n-1}$), we see

$$a_n = f[x_0, \ldots, x_n] = \frac{b_{n-1} - a_{n-1}}{x_n - x_0} = \frac{f[x_1, \ldots, x_n] - f[x_0, \ldots, x_{n-1}]}{x_n - x_0}.$$

Thus Newton's formula is useful when a new point of interpolation is added to the existing interpolation.

Example 2.3.14. Suppose we are given $x_0, \ldots, x_n$ and p_n. If we have one more point x_{n+1}. Then p_{n+1} is constructed by adding one more term to p_n:

$$p_n(x) = p_{n-1}(x) + f[x_0, \ldots, x_{n-1}, x_n] \prod_{i=0}^{n-1}(x - x_i). \tag{2.11}$$

Error of Newton's form of interpolating polynomial

We recall Lagrangian polynomial satisfies the error form (Theorem 2.3.5)

$$f(x) \;=\; \sum_{i=0}^{n} f(x_i) \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} + \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n}(x - x_i). \tag{2.12}$$

The Newton form of polynomial for the points $x_0, \cdots, x_n$ is

$$p_n(x) = \sum_{k=0}^{n} f[x_0, \ldots, x_k] \prod_{i=0}^{k-1}(x - x_i).$$

Since the Newton form of interpolating polynomial is just another form of Lagrange interpolation, they are essentially the same polynomials. Hence the error term of the Newton interpolation is the same as that in (2.12). However,

we can derive the error form directly from the construction:

Theorem 2.3.15. *We have*

$$f(x) = \sum_{k=0}^{n} f[x_0, \ldots, x_n] \prod_{i=0}^{k-1}(x - x_i) + f[x_0, \ldots, x_n, x] \prod_{i=0}^{n}(x - x_i). \quad (2.13)$$

Proof. If we treat x as an added point for interpolation(i.e. we consider an interpolation with data $\{x_0, x_1, \cdots, x_n, x\}$), the new interpolation polynomial (with variable ξ) is

$$p_{n+1}(\xi) = p_n(\xi) + f[x_0, \ldots, x_n, x] \prod_{i=0}^{n}(\xi - x_i). \quad (2.14)$$

Since $p_{n+1}(\xi)$ interpolates at x also, we see that

$$f(x) = p_{n+1}(x) = \sum_{k=0}^{n} a_k \prod_{i=0}^{k-1}(x - x_i) + f[x_0, \ldots, x_n, x] \prod_{i=0}^{n}(x - x_i). \quad (2.15)$$

□

Corollary 2.3.16. *There exist a point $\xi \in [x_0, x_1, \ldots, x_n]$ such that the following holds.*

$$f[x_0, x_1, \ldots, x_n, x] = \frac{f^{(n+1)}(\xi)}{(n+1)!}. \quad (2.16)$$

By the same way there exist a point $\xi \in [x_0, x_1, \ldots, x_n]$ such that

$$f[x_0, x_1, \ldots, x_n] = \frac{f^{(n)}(\xi)}{n!}. \quad (2.17)$$

Roughly speaking, the divided difference is similar to Taylor coefficients. Also note that this formula holds when some of x_i coincide each other.

Proof. By comparing (2.15) with (2.12), we see the result. □

2.3.1 Hermite Interpolation

We would like to construct a polynomial p of certain degree that not only interpolate the function values, but also its derivatives up to certain order.

Let us construct a polynomial p such that

$$\begin{aligned} p(x_i) &= f(x_i), \qquad i = 0, \ldots, n \\ p^{(j)}(x_i) &= f^{(j)}(x_i), \quad j = 1, 2, \ldots, \gamma_i, \quad i = 0, \ldots, n. \end{aligned} \tag{2.18}$$

Simply these conditions can be put

$$p^{(j)}(x_i) = f^{(j)}(x_i), \quad j = 0, 1, \ldots, \gamma_i, \quad i = 0, \ldots, n. \tag{2.19}$$

The number of conditions are $\sum_{i=0}^{n}(\gamma_i + 1)$.

Lemma 2.3.17. *There exists a unique polynomial $p(x)$ of degree $\leq m = n + \sum_{i=0}^{n} \gamma_i$ satisfying above conditions.*

Proof. First let us assume $f(x_i) = f^{(j)}(x_i) = 0$, $\forall\ i,\ j = 1, 2, \ldots, \gamma_i$. Let

$$p(x) = \sum_{k=0}^{m} a_k (x - \alpha)^k.$$

Then $p(x)$ has a zero of multiplicity $\gamma_i + 1$ at x_i, thus $p(x)$ has $\sum_{i=0}^{n}(\gamma_i + 1) = m + 1$ zeros, and as a polynomial of degree m, $p(x) \equiv 0$ and therefore, the homogeneous linear system in the unknowns $\{a_i\}_{i=0}^{m}$ formed by (2.18) has only trivial solution, which in turn, implies that a unique solution exists for the nonhomogeneous system. □

Naive Construction

Let us assume $n = 1$ and $\gamma_0 = \gamma_1 = 1$. $m = 1 + \sum_{i=0}^{1} \gamma_i = 3$.

(1) Method of undetermined "coefficients." With $h = x_1 - x_0$ we let $p(x) = \sum_{k=0}^{3} a_k (x - x_0)^{3-k}$, $\quad h = x_1 - x_0$ and ask it satisfies

$$\begin{aligned} p(x_0) &= a_3 \\ p(x_1) &= a_0 h^3 + a_1 h^2 + a_2 h + a_3 \\ p'(x_0) &= a_2 \\ p'(x_1) &= 3a_0 h^2 + 2a_1 h + a_2. \end{aligned}$$

Solving for a_k, $k = 0, \ldots, 3$, we get $p(x)$.

(2) Method of undetermined "weight"

We let $p(x) = \sum_{i=0}^{n} \sum_{j=0}^{\gamma_i} w_{i,j}(x) f^{(j)}(x_i)$ and seek to determine $w_{i_j}(x)$ so that above relation is exact for $(x - x_0)^j$, $j = 0, \ldots, 3$.

Construction by Lagrangian form

For simplicity, we consider only the case when $\gamma_i = 1$, $i = 0, \ldots, n$.

We seek a polynomial of degree $2n + 1$ such that

$$p(x_i) = f(x_i), \qquad p'(x_i) = f'(x_i), \qquad i = 0, \ldots, n.$$

A nice candidate for the basis is

$$\{\varphi_j(x), \psi_j(x),\ j = 0, 1, \ldots, n\}.$$

where for all $i, j = 0, \ldots, n$

$$\begin{array}{llll} \varphi_j(x_i) & = \delta_{ij}, & \varphi_j'(x_i) & = 0 \\ \psi_j(x_i) & = 0, & \psi_j'(x_i) & = \delta_{ij}. \end{array}$$

Then we see that

$$p(x) = \sum_{j=0}^{n} [f(x_j)\varphi_j(x) + f'(x_j)\psi_j(x).]$$

Construction of $\varphi_j(x), \psi_j(x)$ can be carried out as follows: We compute $\psi_j(x)$ first. Since $\psi_j(x)$ has double zeros at all x_i except j, and has a simple zero at x_j, we see that ψ_j is of the form

$$\psi_j(x) \quad = \quad C_0(x - x_j) \prod_{i \neq j} (x - x_i)^2.$$

Furthermore, it must satisfy $\psi_j'(x_j) = 1$. Hence

$$\psi_j'(x_j) \quad = \quad C_0 \prod_{i \neq j} (x_j - x_i)^2 = 1.$$

Thus $C_0 = 1/\prod_{i \neq j} (x_j - x_i)^2$ and hence

$$\psi_j(x) = (x - x_j)[\ell_j(x)]^2,$$

where $\ell_j(x) = \prod_{i \neq j} \frac{x - x_i}{x_j - x_i}$.

Now we construct $\varphi_j(x)$. Since $\varphi_j(x)$ has double zeros at all x_i except j, and has a simple zero at $\varphi_j(x_j) = 1$, we see $\varphi_j(x)$ has double factor of $x - x_i$ except $i = j$. Hence by considering the degree, one can set $\varphi_j(x) =$

$(a(x-x_j)+b)[\ell_j(x)]^2$. Since

$$\varphi_j(x_j) = b = 1 \text{ and } \varphi_j'(x_j) = a + 2b\ell_j'(x_j) = 0,$$

we see

$$\varphi_j(x) \quad = \quad [\ell_j(x)]^2[1 - 2\ell_j'(x_j)(x - x_j)].$$

Now we need to compute $\ell_j'(x_j)$. Taking the logarithmic derivative of $\ell_j(x) = \prod_{i\neq j} \frac{x-x_i}{x_j-x_i}$, we have

$$\frac{\ell_j'(x)}{\ell_j(x)} = \sum_{i\neq j} \frac{1}{x - x_i}.$$

Evaluating at x_j we have

$$\ell_j'(x_j) = \sum_{i\neq j} \frac{1}{x_j - x_i} = \sum_{j\neq i} \frac{1}{x_j - x_i}.$$

Thus

$$1 - 2\ell_j'(x)(x - x_j) = 1 - 2\sum_{i\neq j} \frac{x - x_j}{x_j - x_i}.$$

Hence

$$\varphi_j(x) = [\ell_j(x)]^2 \left(1 - 2\sum_{i\neq j} \frac{x - x_j}{x_j - x_i}\right).$$

Exercise 2.3.18. (1) Write down L.I.P for 4 point a, b, c, d and let $b \to a$ and $c \to d$.

Construction by Newton form

The the divided difference is not defined if any of two points are equal. What happens to the divided difference if two points converge to one point? We see

$$\lim_{x_1\to x_0} f[x_0, x_1] = \lim_{x_1\to x_0} \frac{f(x_1) - f(x_0)}{x_1 - x_0} = f'(x_0).$$

Thus, it is legitimate to define

$$f[x_0, x_0] = f'(x_0).$$

For higher order, we see from the error formula (Corollary 2.16)

$$f[x_0, x_1, \ldots, x_n, x] = \frac{f^{(n+1)}(\xi)}{(n+1)!}, \quad \xi \in [x_0, x_1, \ldots, x_n].$$

If some of the points, say $x_1, \ldots, x_i \to x_0$, then

$$\begin{aligned} f[\underbrace{x_0, x_0, \ldots, x_0}_{i+1}, \cdots, x_n, x] &= \lim_{x_i \to x_0} f[x_0, x_1, \ldots, x_i, \cdots, x_n, x] \\ &= \lim_{x_i \to x_0} \frac{1}{i!} \frac{\partial f^{(i)}}{\partial x_0^i} [\xi, x_{i+1}, \cdots, x_n, x]. \end{aligned}$$

Thus the divided difference is defined for any multiple argument. For example, if all the points $x_1, \ldots, x_n, x \to x_0$, then we define

$$f[x_0, x_0, \ldots, x_0] = \frac{f^{(n+1)}(x_0)}{(n+1)!}.$$

From the interpolation polynomial

$$f(x_0)+f[x_0,x_1](x-x_0)+f[x_0,x_1,x_2](x-x_0)(x-x_1)+\cdots+f[x_0,\cdots,x_n]\prod_{i=0}^{n-1}(x-x_i)$$

one can show that a Hermite interpolation can be obtained from the Newton interpolation by simply repeating the interpolation points as many times as the multiplicity. Note that we can easily construct Newton's form of Hermite interpolation for any order of derivatives γ_i, $i = 0, \ldots, n$.

Example 2.3.19. (1). If $n = 2$ and $\gamma_0 = 1, \gamma_1, \gamma_2 = 0$, then use $\{x_0, x_0, x_1, x_2\}$ as the interpolation points to find $p(x)$ as

$$f(x_0)+f[x_0,x_0](x-x_0)+f[x_0,x_0,x_1](x-x_0)(x-x_0)+f[x_0,x_0,x_1,x_2](x-x_0)^2(x-x_1)$$

Then $p(x)$ satisfies $p(x_i) = f(x_i), i = 0, 1, 2$ and $p'(x_0) = f'(x_0)$.

(2). If $n = 2$ and $\gamma_0 = \gamma_1 = 1, \gamma_2 = 0$, then use $\{x_0, x_0, x_1, x_1, x_2\}$ as the interpolation points to find $p(x)$ equals

$$\begin{aligned} & f(x_0) + f[x_0, x_0](x - x_0) + f[x_0, x_0, x_1](x - x_0)(x - x_0) \\ + \quad & f[x_0, x_0, x_1, x_1](x - x_0)^2(x - x_1) + f[x_0, x_0, x_1, x_1, x_2](x - x_0)^2(x - x_1)^2. \end{aligned}$$

Then one can check $p(x)$ satisfies

$$p(x_i) = f(x_i), i = 0, 1, 2, \text{ and } p'(x_0) = f'(x_0), p'(x_1) = f'(x_1).$$

Verification.

$$\begin{aligned} p'(x_1) &= f[x_0, x_0] + 2f[x_0, x_0, x_1](x_1 - x_0) + f[x_0, x_0, x_1, x_1](x_1 - x_0)^2 \\ &= f'(x_0) + 2f[x_0, x_0, x_1](x_1 - x_0) + (f[x_0, x_1, x_1] - f[x_0, x_0, x_1])(x_1 - x_0) \\ &= f'(x_0) + f[x_0, x_0, x_1](x_1 - x_0) - f[x_0, x_1, x_1](x_1 - x_0) \\ &= f'(x_0) + (f[x_1, x_0] - f[x_0, x_0]) - (f[x_0, x_1] - f[x_1, x_1]) = f'(x_1). \end{aligned}$$

In general, one can conjecture the following theorem.

Theorem 2.3.20. *Suppose x_0 is repeated $\gamma_0 + 1$-times in the Newton's interpolation, $x_0, \cdots, x_0, x_1, x_2, \cdots, x_n$. Then the polynomial defined by these data*

$$\begin{aligned} p(x) = f[x_0] + f[x_0, x_0](x - x_0) + \cdots + f[x_0, \ldots, x_0](x - x_0)^{\gamma_0} \\ + \sum_{k=1}^{n} f[x_0, \ldots, x_0, x_1, \cdots, x_k](x - x_0)^{\gamma_0+1} \prod_{j=1}^{k-1}(x - x_j). \end{aligned} \tag{2.20}$$

It satisfy

$$p^{(j)}(x_0) = f^{(j)}(x_0), \ j = 0, \cdots, \gamma_0 \text{ and } p(x_k) = f(x_k), \quad k = 1, \cdots, n. \tag{2.21}$$

(Just treat as if all x_0 are distinct, then set them equal.) Similarly, the polynomial constructed from the data $x_0, \cdots, x_0, x_1, \cdots, x_1, x_2, \cdots, x_n$ (Here x_1 is repeated $\gamma_1 + 1$-times) is

$$\begin{aligned} p(x) = f[x_0] + f[x_0, x_0](x - x_0) + \cdots + f[x_0, \ldots, x_0](x - x_0)^{\gamma_0} \\ + f[x_0, \ldots, x_0, x_1](x - x_0)^{\gamma_0}(x - x_1) + \cdots \\ + f[x_0, \ldots, x_0, x_1, \ldots, x_1](x - x_0)^{\gamma_0}(x - x_1)^{\gamma_1} \\ + \sum_{k=2}^{n} f[x_0, \ldots, x_0, x_1, \ldots, x_1, x_2, \cdots, x_k](x - x_0)^{\gamma_0+1}(x - x_1)^{\gamma_1+1} \prod_{j=2}^{k-1}(x - x_j). \end{aligned}$$

It satisfies

$$p^{(j)}(x_0) = f^{(j)}(x_0),\ j = 0, \cdots, \gamma_0, \quad p^{(j)}(x_1) = f^{(j)}(x_1),\ j = 0, \cdots, \gamma_1. \tag{2.22}$$

Proof. Assume $k = 1$ and Let $h = x_1 - x_0$.

$$p(x_1) = f[x_0] + f[x_0, x_0]h + \cdots + f[\underbrace{x_0, \ldots, x_0}_{\gamma_0+1}]h^{\gamma_0} + f[\underbrace{x_0, \ldots, x_0}_{\gamma_0+1}, x_1]h^{\gamma_0+1}. \tag{2.23}$$

$$\begin{aligned}
& p(x_1) \\
= \ & f[x_0] + f[x_0, x_0]h + \cdots + f[\underbrace{x_0, \ldots, x_0}_{\gamma_0+1}]h^{\gamma_0} + \frac{f[x_0, \ldots, x_0, x_1] - f[x_0, \ldots, x_0]}{h}h^{\gamma_0+1} \\
= \ & f[x_0] + f[x_0, x_0]h + \cdots + f[x_0, \ldots, x_0]h^{\gamma_0} + (f[x_0, \ldots, x_0, x_1] - f[x_0, \ldots, x_0])h^{\gamma_0} \\
= \ & f[x_0] + f[x_0, x_0]h + \cdots + f[\underbrace{x_0, \ldots, x_0}_{\gamma_0}]h^{\gamma_0-1} + f[\underbrace{x_0, \ldots, x_0}_{\gamma_0}, x_1]h^{\gamma_0}
\end{aligned}$$

Continuing,

$$\begin{aligned}
= \ & f[x_0] + f[x_0, x_0]h + f[x_0, x_0, x_0]h^2 + f[x_0, x_0, x_0, x_1]h^3 \\
= \ & f[x_0] + f[x_0, x_0]h + f[x_0, x_0, x_0]h^2 + \frac{f[x_0, x_0, x_1] - f[x_0, x_0, x_0]}{h}h^3 \\
= \ & f[x_0] + f[x_0, x_0]h + f[x_0, x_0, x_1]h^2 \\
= \ & f[x_0] + f[x_0, x_0]h + (f[x_0, x_1] - f[x_0, x_0])h \\
= \ & f[x_0] + f[x_0, x_1]h = f[x_0] + f[x_1] - f[x_0] = f[x_1].
\end{aligned}$$

What if you have more than one point ? □

Example 2.3.21. Suppose we are given the data

$$f(x_0), f(x_1), f^{(j)}(x_0), j = 1, 2 \text{ and } f'(x_1).$$

Then by filling in the following table, one can find the corresponding Hermite interpolation.

$$\begin{aligned}
& p(x) = f[x_0] + f[x_0, x_0](x - x_0) + f[x_0, x_0, x_0](x - x_0)^2 + f[x_0, x_0, x_0, x_1](x - x_0)^3 \\
& + f[x_0, x_0, x_0, x_1, x_1](x - x_0)^3(x - x_1)
\end{aligned}$$

x_0	$f[x_0]$	$f[x_0,x_0]$	$f[x_0,x_0,x_0]$	$f[x_0,x_0,x_0,x_1]$	$f[x_0,x_0,x_0,x_1,x_1]$
x_0	$f[x_0]$	$f[x_0,x_0]$	$f[x_0,x_0,x_1]$	$f[x_0,x_0,x_1,x_1]$	
x_0	$f[x_0]$	$f[x_0,x_1]$	$f[x_0,x_1,x_1]$		
x_1	$f[x_1]$	$f[x_1,x_1]$			
x_1	$f[x_1]$				

$$x_0,x_0,x_0,x_1,x_2, f(x_0)=1, f'(x_0)=2, f''(x_0)=2, f(x_1)=3, f(x_2)=4$$

So the Hermite interpolation is

x_i	$f(x_i)$	$f[x_0,x_0]$	$f[x_0,x_0,x_0]$	$f[x_0,x_0,x_0,x_1]$	$f[x_0,x_0,x_0,x_1,x_2]$
0	1	2	2/2!	-1/1	3/4/2
0	1	2	0	-1/2/2	
0	1	2	-1/2		
1	3	1			
2	4				

Table 2.1: Newton's interpol.

$$p(x)=1+2x+x^2-x^3+\frac{3}{8}x^2(x-1)$$

The following holds. (See Issacson, p.254)

Corollary 2.3.22. *Suppose $f^{(m)}$ is continuous in $[a,b]$, x,y,z are distinct points in $[a,b]$ and $0\le p,q,r\le m$. Then*

$$f[\underbrace{x,\dots,x}_{p+1},\underbrace{y,\dots,y}_{q+1},\underbrace{z,\dots,z}_{r+1}]=\frac{1}{p!q!r!}\frac{\partial^p}{\partial x^p}\frac{\partial^q}{\partial y^q}\frac{\partial^r}{\partial z^r}f[x,y,z]. \tag{2.24}$$

Error of Hermite interpolation

Now what happens to the error term of Newton's interpolating polynomial if $x_i\to x_{i+1}$ for some i ? For simplicity, assume $\gamma_i=1$ for all i so that the interpolating nodes are all repeated once: $x_0,x_0,\cdots,x_n,x_n$. This corresponds to the case $x_0,x_1,\cdots,x_{2n+1}$ with $x_{2i+1}\to x_i$, $i=0,\cdots,n$. Then we see the error term from (2.15) becomes

$$f[x_0,x_0,\cdots,x_n,x_n,x]\prod_{i=0}^{n}(x-x_i)^2=\frac{f^{(2n+2)}(\xi)}{(2n+2)!}\prod_{i=0}^{n}(x-x_i)^2. \tag{2.25}$$

2.4 Polynomial of best approximation

Definition 2.4.1. Let

$$d_n(f) \equiv \min_{p_n(x)\in P_n(x)} \|f(x) - p_n(x)\|_\infty.$$

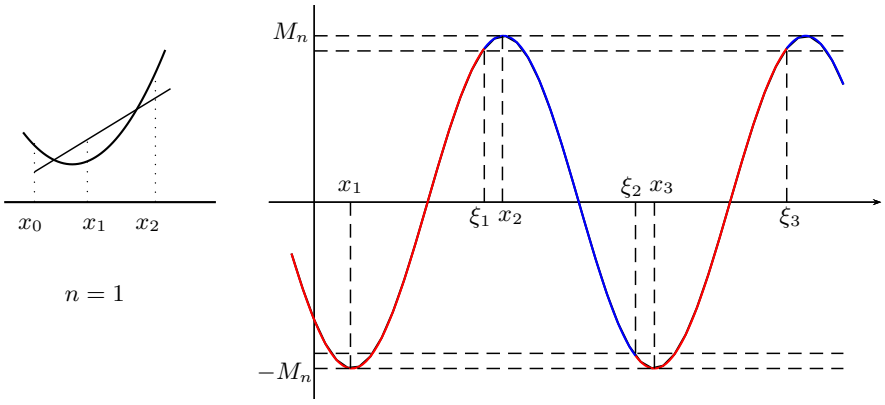
A polynomial which attains this minimum is called the **polynomial of best approximation**(P.B.A) to $f(x)$ on $[a, b]$.

The question is can we find it ? How ? First, we study some of its property.

Theorem 2.4.2 (De La Valeé Poussin)**.** *If there exists a polynomial $p_n(x)$ which oscillates about $f(x)$ $n+2$-times, i.e., $f(x_j) - p_n(x_j) = (-1)^j e_j$, $j = 0, 1, \ldots, n+1$, where all e_j are of one sign, then $d_n(f) \geq \min_j |e_j|$.*

Proof. Suppose there exists a polynomial $q_n(x)$ such that $\|f(x) - q_n(x)\|_\infty < \min_j |e_j|$. Then $q_n(x_j) - p_n(x_j)$ have the same sign as $f(x_j) - p_n(x_j)$ because $q_n(x) - p_n(x) = f(x) - p_n(x) - [f(x) - q_n(x)]$. Hence, $q_n(x) - p_n(x)$ has $n+1$ zeros, but as a polynomial of degree n, it must be identically zero. So $p_n(x) \equiv q_n(x)$ and hence $\|f(x) - p_n(x)\|_\infty < \min_j |e_j|$. This is a contradiction. □

Theorem 2.4.3 (Chebyshev)**.** *$p_n(x)$ is the polynomial of best approximation* (P.B.A.) *to $f(x)$ in $[a, b]$ if and only if $f(x) - p_n(x)$ takes the value $\pm\|f - p_n\|_\infty$ with alternating signs at least $(n+2)$ times in $[a, b]$. Moreover, $p_n(x)$ is unique.*



Proof. ⇐ Let $\{x_j\}_{j=0}^{n+1}$ be the points at which the maximum deviation is attained with alternating sign. By D.V. Poussin's theorem, $d_n(f) \geq \|f(x) - p_n(x)\|_\infty \equiv M_n$. Hence $p_n(x)$ is a P.B.A.

($\Rightarrow$) Conversely, suppose $p_n(x)$ is a polynomial of degree n such that $f(x)-p_n(x)$ attains $\pm\|f(x)-p_n(x)\|_\infty$ with alternating signs $k(\leq n+1)$ times only. Then without loss of generality, we may assume that for some x_i with $a \leq x_1 < x_2 < \cdots < x_k \leq b$, it holds that

$$f(x_j) - p_n(x_j) = (-1)^j \|f(x) - p_n(x)\|_\infty = (-1)^j M_n, \qquad j = 1, 2, \ldots, k.$$

Then there exist $\xi_1, \ldots, \xi_{k-1}$ separating x_i's such that in the odd interval

$$f(x) - p_n(x) < M_n - \varepsilon \quad \text{for} \quad x \in [a, \xi_1] \cup [\xi_2, \xi_3] \cup \cdots = I_1,$$

while in the even interval

$$-M_n + \varepsilon < f(x) - p_n(x) \quad \text{for} \quad x \in [\xi_1, \xi_2] \cup [\xi_3, \xi_4] \cup \cdots = I_2.$$

Let $r(x) = \prod_{i=1}^{k-1}(x-\xi_i)$(its degree is $\leq n$). Then with $g(x) = \pm r(x)/2\|r(x)\|_\infty$, and $q_n(x) = p_n(x) + \varepsilon g(x)$, (signs are chosen so that $g(x) < 0$ on I_1, $g(x) > 0$ on I_2) Since $|f(x) - p_n(x)| \leq M_n$,

$$-M_n < -M_n - \varepsilon g(x) \leq f(x) - q_n(x) \leq M_n - \varepsilon(g(x)+1) \leq M_n - \varepsilon/2 < M_n$$

on I_1 and

$$-M_n < -M_n + \varepsilon - \varepsilon g(x) < f(x) - q_n(x) = f(x) - p_n(x) - \varepsilon g(x) < M_n - \varepsilon g(x) < M_n$$

on I_2. Hence $\|f - q_n\|_\infty < M_n$ and this means $q_n(x)$ is a better approximation. Hence $p_n(x)$ cannot be a P.B.A. Thus if $p_n(x)$ is a P.B.A., it must oscillates at least $n+2$-times.

Uniqueness: If $p_n(x)$, $q_n(x)$ are two P.B.A.'s, then

$$d_n(f) \leq \|f(x) - \frac{1}{2}(p_n(x) + q_n(x))\|_\infty \leq \frac{1}{2}\|f - p_n\|_\infty + \frac{1}{2}\|f - q_n\|_\infty = d_n(f).$$

Thus $\frac{1}{2}(p_n(x) + q_n(x))$ is also a P.B.A. So equality must hold. By the result of theorem, there exist $n+2$ points where maximum is attained, i.e.,

$$|f(x_i) - p_n(x_i)| \quad = \quad \frac{1}{2}|f(x_i) - p_n(x_i)| + \frac{1}{2}|f(x_i) - q_n(x_i)|, \quad i = 0, 1, \ldots, n+1.$$

and the terms must have the same sign. Thus $f(x_i) - p_n(x_i) = f(x_i) - q_n(x_i)$,

$i = 0, 1, \ldots, n+1$, and since p_n, q_n are polynomials of degree n, $p_n(x) \equiv q_n(x)$. □

Observation: The PBA p_n must interpolate f at least $n+1$ distinct points.

2.4.1 Chebyshev polynomials

We have seen that the P.B.A. must oscillate $n+2$-times about $f(x)$ and hence interpolates $f(x)$ at $n+1$ distinct points, but we do not know the points and value $\|f - p_n\|_\infty$. But when $f \in C^{(n+1)}[a, b]$, we have the error formula

$$e_n(x) = f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i). \tag{2.26}$$

Our goal is to minimize this error term. However, it is hard to estimate f^{n+1}. Hence, instead of minimizing $\|e_n(x)\|_\infty$, we focus on minimizing $\prod_{i=0}^{n}(x - x_i)$. This will be a good approximation as long as $f^{(n+1)}(\xi)$ is not too large. If it is constant(i.e., f is a polynomial of degree $n+1$), the error will be minimized if we choose the points $x_0, \cdots, x_n$ so that $\prod_{i=0}^{n}(x - x_i)$ has smallest maximum in $[a, b]$. Hence we will indeed have the PBA.

Thus we may consider the following:

Find the best approximation $p_n(x) \in P_n$ to the function $f(x) = x^{n+1}$.

In this case the error $f(x) - p_n(x)$ will be of the form: $C_n \prod_{i=0}^{n}(x - x_i)$,. ($C_n = \frac{f(x)^{(n+1)}}{(n+1)!} = 1$.) To minimize $\prod_{i=0}^{n}(x - x_i)$, we have to enforce $n+2$ times uniform oscillations by Chebyshev(Chebysheff) Theorem. This suggest a polynomial whose behavior is similar to a *trigonometric function*. Assume $[a, b] = [-1, 1]$. Note that $\cos(n+1)\theta$ oscillates $n+2$ times between -1 and 1 in $[0, \pi]$, i.e.,

$$\cos(n+1)\theta_j = (-1)^j, \quad \text{for } \theta_j = \frac{j\pi}{n+1},\ j = 0, \cdots, n+1.$$

We let $\theta = \cos^{-1} x$ and consider $\cos(n+1)\cos^{-1} x$, i.e.,

$$T_{n+1}(x) = A_{n+1} \cos[(n+1)\cos^{-1} x], \quad x \in [-1, 1].$$

Recall the addition formula for cosine function:

$$\cos(n+1)\theta + \cos(n-1)\theta = 2\cos\theta\cos n\theta.$$

If we let $t_n(x) = \cos[n \cos^{-1} x]$, we obtain a recurrence relation

$$t_{n+1}(x) + t_{n-1}(x) = 2xt_n(x), \qquad n = 1, 2, \ldots$$

From this we get $t_0 = 1, \quad t_1(x) = x, \quad t_2(x) = 2xt_1 - t_0 = 2x^2 - 1$, and

$$t_n(x) = 2^{n-1}x^n + \cdots .$$

So if we choose $A_{n+1} = 2^{-n}$, then $T_{n+1}(x)$ is a monic polynomial and its maximum oscillation is 2^{-n}. $T_{n+1}(x)$ is called **'Chebyshev' polynomial of the first kind**.

$$T_{n+1}(x) = 2^{-n} \cos[(n+1) \cos^{-1} x] = (x - x_0)(x - x_1) \cdots (x - x_n). \quad (2.27)$$

Since $T_{n+1}(x)$ is the error term of P.B.A(up to a constant), we have

$$f(x) - p_n(x) = \prod_{i=0}^{n} (x - x_i) = T_{n+1}(x).$$

The polynomial $p_n(x)$ interpolates x^{n+1} at $x_i (i = 0, \cdots, n)$, the zeros of $T_{n+1}(x)$, i.e., the points where

$$\cos[(n+1) \cos^{-1} x_i] = 0 \text{ or } (n+1) \cos^{-1} x_i = \frac{(2i+1)\pi}{2}.$$

Such points are called **Chebyshev points** and we use these points as interpolation points for general f. The points $\frac{(2k+1)\pi}{2n+2}, k = 0, \ldots, n$ are evenly distributed on $[-1, 1]$ and the Chebyshev points on $[-1, 1]$ are the cosine of these points,

$$x_k = \cos\left(\frac{(2k+1)\pi}{2n+2}\right), \quad k = 0, \ldots, n.$$

We use the zeros of Chebyshev polynomial of degree $n+1$ to interpolate with a polynomial of degree n.

Theorem 2.4.4. *If $f \in C^{n+1}[-1, 1]$ then the PBA using these interpolation points satisfies*

$$\|f(x) - p_n(x)\|_\infty \leq \frac{1}{2^n} \max_{[-1,1]} \left| \frac{f^{(n+1)}(\xi)}{(n+1)!} \right|. \quad (2.28)$$

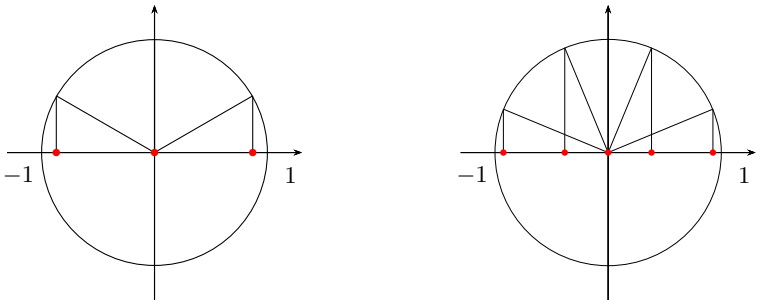


Figure 2.1: Chebyshev points on $[-1,1]$, $n=3$ and $n=5$

In particular, if $f(x)=x^{n+1}$, *then*

$$f(x)-p_n(x)=\prod_{i=0}^{n}(x-x_i)$$

and $\|f(x)-p_n(x)\|_\infty = 2^{-n}$.

If the interval is $[a,b]$ and $f(y):[a,b]\to\mathbb{R}$, then change of variable gives the following interpolating points for f:

$$y_k=\frac{1}{2}[(b-a)x_k+(a+b)],\quad k=0,\dots,n.$$

The maximum of $\prod_{j=0}^{n}(y-y_j)$ is

$$\begin{aligned}\max_{[a,b]}\prod_{j=0}^{n}(y-y_j) &= \left|\frac{b-a}{2}\right|^{n+1}\max_{[-1,1]}\prod_{j=0}^{n}(x-x_j)\\ &= \frac{1}{2^n}\left|\frac{b-a}{2}\right|^{n+1}.\end{aligned}$$

Exercise 2.4.5. (1) Solve Runge example again with Chebyshev points for $n=5,10,20,30$ (Use polynomial of degree n- i.e., use roots of Chebysheff polynomial $T_{n+1}(\xi)=0$). Check $\|f(x)-p_n(x)\|_\infty$ by taking sufficiently many points between interpolation points. Also draw graphs for $f(x), p_n(x)$ and $f'(x), p_n'(x)$.

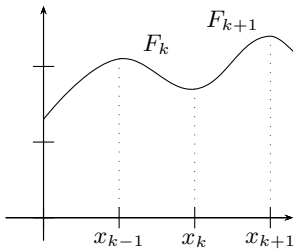


Figure 2.2: Cubic Spline on $[x_{k-1}, x_k]$ and $[x_k, x_{k+1}]$

2.5 Cubic splines

Although Chebyshev points are good for P.B.A. they are sometimes inconvenient since the points are not uniform. Also the error of the derivative is large. Runge's example suggests us to use a lower degree polynomial on each subinterval. Let $y = f \in C^2[a,b]$, $x_0 = a < x_1, \ldots, < x_n = b$. We wish to construct an interpolation $s(x) \in C^2[a,b]$, which is piecewise cubic polynomial on $I_k = [x_{k-1}, x_k]$, $k = 1, \cdots, n$. Let $F_k(x) \equiv s(x)|_{I_k}$. Then each $F_k(x)$ is a cubic polynomial on I_k and

(1) $F_k(x_{k-1}) = y_{k-1}$ and $F_k(x_k) = y_k$ for $k = 1, \cdots, n$,

(2) $F_k'(x_k^-) = F_{k+1}'(x_k^+)$ for $k = 1, \cdots, n-1$, (interior derivative)

(3) $F_k''(x_k^-) = F_{k+1}''(x_k^+)$ for $k = 1, \cdots, n-1$. (interior 2nd derivative)

Condition (1) automatically enforces interpolation at x_k for $k = 0, \cdots, n$ and continuity x_k for $k = 1, \cdots, n-1$. Since $s(x)$ is a cubic polynomial in each interval $I_k = [x_{k-1}, x_k]$, $k = 1, \cdots, n$, we have $4n$ unknowns. Now we count the number of conditions:

- $2n$ from (1) and
- $2(n-1)$ interior derivative continuity from (2) and (3),

total of $4n-2$ conditions. We need to impose two more conditions.

Since $F_k''(x)$ is linear we may write

$$F_k''(x) = d_{k-1}(\frac{x_k - x}{h_k}) + d_k(\frac{x - x_{k-1}}{h_k}), \tag{2.29}$$

for some constants d_k and $h_k = x_k - x_{k-1}$. Note that F_k is constructed to satisfy $F_k''(x_k^-) = F_{k+1}''(x_k^+)$ for $k = 1, \cdots, n-1$.

$$F_k'(x) = -d_{k-1}\frac{(x_k - x)^2}{2h_k} + d_k\frac{(x - x_{k-1})^2}{2h_k} + c_{1,k} \tag{2.30}$$

$$F_k(x) = d_{k-1}\frac{(x_k - x)^3}{6h_k} + d_k\frac{(x - x_{k-1})^3}{6h_k} + c_{2,k}(x - x_{k-1}) + c_{3,k}(x_k - x) \tag{2.31}$$

$F_k(x)$ must interpolate $f(x)$ at x_{k-1} and x_k. Hence

$$F_k(x_{k-1}) = d_{k-1}\frac{h_k^2}{6} + c_{3,k}h_k = y_{k-1}, \quad F_k(x_k) = d_k\frac{h_k^2}{6} + c_{2,k}h_k = y_k.$$

Thus $c_{2,k} = \frac{y_k}{h_k} - \frac{d_k h_k}{6}$, $c_{3,k} = \frac{y_{k-1}}{h_k} - \frac{d_{k-1}h_k}{6}$ and $c_{1,k} = c_{2,k} - c_{3,k}$. Now compute the derivative at x_k: We see

$$\begin{aligned} F_k'(x_k^-) &= \frac{d_k h_k}{2} + (\frac{y_k}{h_k} - \frac{d_k h_k}{6}) - (\frac{y_{k-1}}{h_k} - \frac{d_{k-1}h_k}{6}) \\ &= \frac{d_k h_k}{3} + \frac{d_{k-1}h_k}{6} + (\frac{y_k - y_{k-1}}{h_k}). \end{aligned} \tag{2.32}$$

On the other hand, we consider $F_{k+1}'(x_k^+)$. Replacing k by $k+1$ in (2.30) and evaluating at x_k, we get

$$\begin{aligned} F_{k+1}'(x_k^+) &= -\frac{d_k h_{k+1}}{2} + (\frac{y_{k+1}}{h_{k+1}} - \frac{d_{k+1}h_{k+1}}{6}) - (\frac{y_k}{h_{k+1}} - \frac{d_k h_{k+1}}{6}) \\ &= -\frac{d_k h_{k+1}}{3} - \frac{d_{k+1}h_{k+1}}{6} + (\frac{y_{k+1} - y_k}{h_{k+1}}). \end{aligned} \tag{2.33}$$

The continuity of derivative: Equating (2.32) with (2.33) and arranging in terms of unknowns d_k, $k = 0, 1 \cdots, n$

$$\frac{h_k}{6}d_{k-1} + \frac{h_k + h_{k+1}}{3}d_k + \frac{h_{k+1}}{6}d_{k+1} = \frac{y_{k+1} - y_k}{h_{k+1}} - \frac{y_k - y_{k-1}}{h_k}, \quad 1 \le k \le n-1.$$

(Note the rhs is difference between (approx. of) $y_{k+1}'(x_k)$ and $y_k'(x_k)$.) We see two more conditions are needed to guarantee the existence of the solution. A common choice is to let $d_0 = d_n = 0$. In this case, we obtain a tridiagonal system

$$A\mathbf{x} = \mathbf{k}, \tag{2.34}$$

where $\mathbf{x} = (d_1, \cdots, d_{n-1})$. A typical row of A looks like

$$[0, \ldots, \frac{h_i}{6}, \frac{h_i + h_{i+1}}{3}, \frac{h_{i+1}}{6}, \ldots, 0].$$

Hence A is diagonally dominant and thus nonsingular. The spline obtained in this way is called the **natural spline**. To find it, one has to solve the system of linear equations (2.34). Fortunately, the system is tridiagonal, which is easy to solve.

Solution of a tridiagonal system by LU-decomposition.

$$\begin{bmatrix} b_1 & c_1 & \ldots & 0 \\ a_1 & b_2 & c_2 & \vdots \\ \vdots & \ddots & \ddots & c_{n-1} \\ 0 & \ldots & a_{n-1} & b_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ \ell_1 & 1 & 0 & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & \ell_{n-1} & 1 \end{bmatrix} \begin{bmatrix} d_1 & c_1 & \ldots & 0 \\ 0 & d_2 & c_2 & \vdots \\ \vdots & \ddots & \ddots & c_{n-1} \\ 0 & \ldots & 0 & d_n \end{bmatrix}$$

LU decomposition costs $2(n-1)$ multiplication $n-1$ addition.

$$\circ \begin{cases} d_1 = b_1 \\ \text{For } i = 1, \ldots, n-1 \\ \quad \ell_i = a_i / d_i \\ \quad d_{i+1} = b_{i+1} - \ell_i c_i. \end{cases}$$

Forward substitution $L\mathbf{y} = \mathbf{k}$ costs $n-1$ mult. $n-1$ add.

$$\circ \begin{cases} y_1 = k_1 \\ \text{For } i = 2, \ldots, n \\ \quad y_i = k_i - \ell_{i-1} y_{i-1}. \end{cases}$$

Back substitution $U\mathbf{x} = \mathbf{y}$ costs $2(n-1)+1$ mult. $n-1$ add.

$$\circ \begin{cases} x_n = y_n / d_n \\ \text{For } i = n-1, \ldots, 1 \\ \quad x_i = (y_i - c_i y_{i+1}) / d_i. \end{cases}$$

It requires $(5n-4)$ multiplication and $3n-3$ addition.

Theorem 2.5.1. (Holliday) *Among all C^2-function which interpolate f at $\{x_i\}_{i=0}^n$, the natural cubic spline has smallest curvature.*

Proof. Let $g(x)$ be any C^2-interpolant. Then we have

$$\begin{aligned} 0 &\leq \int_a^b [g''(t) - s''(t)]^2\, dt \\ &= \int_a^b [g''(t)]^2\, dt - 2\int_a^b [g''(t) - s''(t)]s''(t)\, dt - \int_a^b [s''(t)]^2\, dt. \end{aligned}$$

We will show the second term is zero, which completes the proof,

$$\begin{aligned} &\int_a^b [g''(t) - s''(t)]s''(t)\, dt \\ =\ &\sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} [g''(t) - s''(t)]s''(t)\, dt \\ =\ &\sum_{i=0}^{n-1} \left[(g'(t) - s'(t))s''(t)|_{x_i}^{x_{i+1}} - \int_{x_i}^{x_{i+1}} (g'(t) - s'(t))s'''\, dt \right] \\ =\ &\sum_{i=0}^{n-1} [(g'(t) - s'(t))s''(t) - (g(t) - s(t))s'''(t)|_{x_i}^{x_{i+1}}]. \end{aligned}$$

By telescoping series, the first term equals $(g'(x_n) - s'(x_n))s''(x_n) - (g'(x_0) - s'(x_0))s''(x_0) = 0$ by the assumption $s''(x_0) = s''(x_n) = 0$. Since s''' is constant on each interval and $g(x_i) = s(x_i)$, the second one is also zero. □

Remark 2.5.2. There are other choices of d_0 and d_n. Suppose the values $f'(x_0)$, $f'(x_n)$ are known. Then we obtain so called a **clamped spline.**

Theorem 2.5.3. *Let $f \in C^2[a,b]$ and s be the natural cubic spline with node $\{x_i\}_{i=0}^n$, then with $h = \max_k h_k$*

$$\begin{aligned} \|f - s\|_\infty &\leq h^{3/2} \left(\int_a^b |f''|^2 \right)^{1/2} \\ \|f' - s'\|_\infty &\leq h^{1/2} \left(\int_a^b |f''|^2 \right)^{1/2}. \end{aligned}$$

Thus cubic spline is also good to approximate f'.

Proof. We prove the second estimate first. Fix $x \in [x_{i-1}, x_i] = I_i$. By Rolle's

theorem, there exists $\tau \in I_i$ such that $f'(\tau) - s'(\tau) = 0$. Then we have

$$\int_\tau^x [f''(t) - s''(t)]\, dt \;=\; f'(t) - s'(t)|_\tau^x = f'(x) - s'(x)$$

and by the proof of Holliday's Theorem,

$$\begin{aligned} |f'(x) - s'(x)| &= |\int_\tau^x (f'' - s'')\, dt| \le \left(\int_\tau^x |f'' - s''|^2 dt\right)^{1/2} (\int_\tau^x dt)^{1/2} \\ &\le h^{1/2} \left(\int_a^b |f'' - s''|^2 dt\right)^{1/2} = h^{1/2} \left(\int_a^b (|f''|^2 - |s''|^2) dt\right)^{1/2} \\ &\le h^{1/2} \left(\int_a^b |f''|^2 dt\right)^{1/2}. \end{aligned}$$

Now for the first estimate we see, for $x \in [x_i, x_{i+1}]$

$$\begin{aligned} |f(x) - s(x)| &= \left|\int_{x_i}^x (f'(t) - s'(t))\, dt\right| \le \int_{x_i}^x |f'(t) - s'(t)|\, dt \\ &\le h\|f' - s'\|_\infty = h^{3/2} \left(\int_a^b |f''|^2\right)^{1/2}. \end{aligned}$$

□

Exercise 2.5.4. (1) Write down the explicit entry of A and $\mathbf{k}$ in (2.34).

(2) (Clamped spline) Derive the equation $A\mathbf{x} = \mathbf{k}$ of size $(n+1) \times (n+1)$ with data $s'(x_0) = f'(x_0)$, $s'(x_n) = f'(x_n)$ instead of $d_0 = d_n = 0$ in (2.34). Write down the entry of A and $\mathbf{k}$ explicitly.

(3) (Computer) Construct a spline approximation to $f(x) = \frac{1}{1+x^2}$ on $[-5, 5]$ with $n = 10, 20, 40, 80$ with equally spaced subintervals. Let $E_h = \|s_h(x) - f(x)\|_\infty$ and $E'_h = \|s'_h(x) - f'(x)\|_\infty$ and estimate E_h and E'_h (compute these norms by choosing many (say, 10) points in each subinterval).

Fill out the table below by computing the ratio:$=\log_2 \frac{E_{2h}}{E_h}$. Draw graphs and discuss the results.

(4) (Computer) Construct a spline approximation to and compute $\|s(x) - f(x)\|_\infty$ and $\|s'(x) - f'(x)\|_\infty$ for $f(x) = e^{0.8x}$ on $[-3, 3]$.

	E_h	ratio	E'_h	ratio
$n = 10$		*		*
$n = 20$				
$n = 40$				
$n = 80$				

Table 2.2: Error of Cubic spline

2.6 The B-Splines

We begin with the knots one the real line :

$$\cdots < t_{-2} < t_{-1} < t_0 < t_1 < t_2 < \cdots$$

2.6.1 The B-Splines of degree k

$$B_i^0(x) = \begin{cases} 1 & \text{if } t_i \le x < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

Now higher order splines are constructed as follows: $(k \ge 1)$

$$B_i^k(x) = \left(\frac{x - t_i}{t_{i+k} - t_i}\right) B_i^{k-1}(x) + \left(\frac{t_{i+k+1} - x}{t_{i+k+1} - t_{i+1}}\right) B_{i+1}^{k-1}(x) \quad (2.35)$$

$$\equiv V_i^k B_i^{k-1}(x) + \left(1 - V_{i+1}^k\right) B_{i+1}^{k-1}(x). \quad (2.36)$$

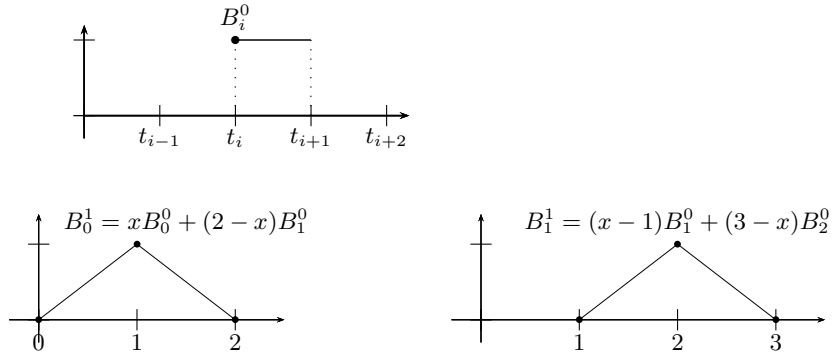


Figure 2.3: The B-Spline B_i^0 and B_i^1

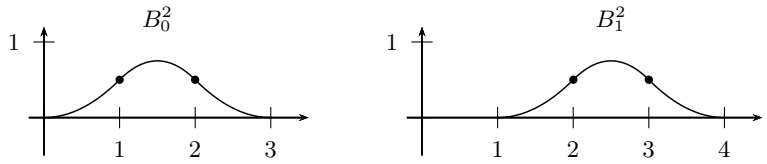


Figure 2.4: The B-Spline B_0^2 and B_1^2

Assume $t_0 = 0, t_1 = 1, t_2 = 2, \cdots$. Then

$$B_0^2(x) = \frac{x}{2}B_0^1 + (1 - \frac{x-1}{2})B_1^1 \tag{2.37}$$

$$= \begin{cases} \frac{x}{2} \cdot x & \text{on } 0 \le x < 1 \\ \frac{x}{2}(2-x) + \frac{3-x}{2}(x-1) = -x^2 + 3x - 1.5 & \text{on } 1 \le x < 2 \\ \frac{(3-x)^2}{2} & \text{on } 2 \le x < 3 \end{cases}$$

$$B_1^2(x) = \frac{x}{2}B_1^1 + \frac{3-x}{2}B_2^1 = B_0^2(x-1) \tag{2.38}$$

$$= \begin{cases} \frac{(x-1)^2}{2} & \text{on } 1 \le x < 2 \\ -(x-1)^2 + 3(x-1) - 1.5 & \text{on } 2 \le x < 3 \\ \frac{(4-x)^2}{2} & \text{on } 3 \le x < 4 \end{cases} \tag{2.39}$$

$$B_0^3(x) = \frac{x}{3}B_0^2 + (1 - \frac{x-1}{3})B_1^2 = \frac{x}{3}B_0^2 + (\frac{4-x}{3})B_1^2 \tag{2.40}$$

$$= \begin{cases} \frac{x}{3} \cdot \frac{x^2}{2} & \text{on } 0 \le x < 1 \\ \frac{x}{3} \cdot (-x^2 + 3x - 1.5) + \frac{4-x}{3}\frac{(x-1)^2}{2} & \text{on } 1 \le x < 2 \\ \frac{x}{3} \cdot \frac{(3-x)^2}{2} + \frac{4-x}{3}(-x^2 + 5x - 5.5) & \text{on } 2 \le x < 3 \\ \frac{4-x}{3} \cdot \frac{(4-x)^2}{2} & \text{on } 3 \le x < 4 \end{cases} \tag{2.41}$$

Properties of B-Splines

(1) For $k \ge 1$, $B_i^k(x) \ne 0$ if and only if $x \in (t_i, t_{i+k+1})$. For $k = 0$, $B_i^0(x) \ne 0$ if and only if $x \in [t_i, t_{i+1})$.

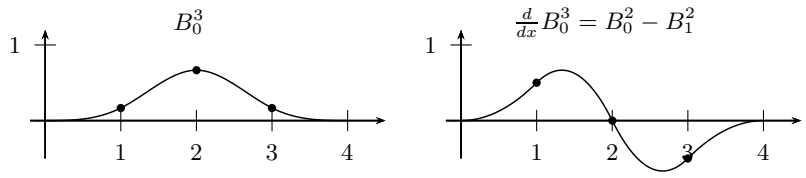


Figure 2.5: The B-Spline B_0^3 and $\frac{d}{dx}B_0^3$

(2) $\sum_{i=-\infty}^{\infty} B_i^k \equiv 1$

(3)

$$\sum_{i=-\infty}^{\infty} c_i B_i^k = \sum_{i=-\infty}^{\infty} \left[c_i V_i^k + c_{i-1}(1 - V_i^k) \right] B_i^{k-1}$$

Derivatives of B-Splines

Lemma 2.6.1. *For $k \geq 2$*

$$\frac{d}{dx}B_i^k(x) = \left(\frac{k}{t_{i+k} - t_i} \right) B_i^{k-1}(x) - \left(\frac{k}{t_{i+k+1} - t_{i+1}} \right) B_{i+1}^{k-1}(x). \tag{2.42}$$

For $k = 1$ the equation holds for all x except $x = t_i, t_{i+1}, t_{i+2}$.

Lemma 2.6.2. *For $k \geq 1$ the B-splines B_i^k belongs to $C^{k-1}(\mathbb{R})$.*

2.7 B-Splines: Applications

We want to see the relation between B-Splines and the spline function introduced before. where we considered functions that are $C^{k-1}(\mathbb{R})$ and piecewise polynomials of degree $\leq k$ on each of the intervals $[t_0, t_1], [t_1, t_2], \cdots, [t_{n-1}, t_n]$. The set of all such functions will be denoted by S_n^k (We assume the knot are given from the beginning.) We view the functions in S_n^k defined on $[t_0, t_n]$.

Basis for S_n^k

Theorem 2.7.1. *The following set is a basis for S_n^k.*

$$\{B_i^k(x)|_{[t_0,t_n]} : -k \leq i \leq n-1\} \tag{2.43}$$

Hence the dimension of S_n^k is $k+n$.

Proof. This follows since, for $k \geq 1$, $B_i^k(x) \neq 0$ if and only if $x \in (t_i, t_{i+k+1})$. $B_{-k}^k(x) \neq 0$ and $B_{n-1}^k(x) \neq 0$. □

Relation with cubic Splines

Let $k = 3$. Then the functions $\{B_i^3, i = -3, -2, \cdots, n-1\}$ form a basis for cubic spline on $[0, n]$.

$$S_3(x) = \sum_{j=-3}^{n-1} c_j B_j^3(x). \tag{2.44}$$

We know that we need 4 conditions to uniquely specify a cubic, and in $[t_0, t_n]$ there are n intervals, so altogether a total of $4n$ conditions are needed. The continuity conditions are automatically satisfied in the $n-1$ interior points, (thats $3(n-1)$ conditions) The other requirement is that S_3 must match the given points, i.e. $S_3(t_k) = f(t_k)$, for $k = 0, \cdots, n$ $(n+1$ conditions). So there are two unspecified conditions, and like before we can take $S_3''(t_0) = S_3''(t_n) = 0$.

B-Splines as Interpolations

Spline functions can be used for interpolations at points *other than the knots.* Let $x_1 < x_2 < \cdots < x_n$ be a given set of **nodes** and $y_1, y_2, \cdots, y_n$ are values we want to interpolate. We would like to find an interpolation function of the form $\sum_{j=1}^n c_j B_j^k$, i.e,

$$\sum_{j=1}^{n} c_j B_j^k(x_i) = y_i \tag{2.45}$$

A condition for this system to be uniquely solved is given by Schoenberg-Whitney.

Theorem 2.7.2 (Schoenberg-Whitney Theorem)**.** *For the matrix $A_{ij} = B_j^k(x_i)$ to be nonsingular it is nec. and suff. that there be no 0 element on its diagonal. In other words, $B_i^k(x_i) \neq 0$.*

Corollary 2.7.3. *Assume $k \geq 1$. Since $B_i^k(x) \neq 0$ if and only if $x \in (t_i, t_{i+k+1})$, we must have $t_i < x_i < t_{i+k+1}$ for the interpolation equation (2.45) have a solution.*

This solution is not unique (incase the interpolation points are knots). Suppose $x_i = t_i$ for all i.

$$\sum_{j=-\infty}^{\infty} c_j B_j^k(t_i) = y_i, \quad (1 \leq i \leq n) \tag{2.46}$$

For $k = 0, 1$ this system has unique solution. However, for $k = 2$, this system has n equation in $n + 1$ unknowns. For $k = 3$, this system has n equation in $n + 2$ unknowns. (2.45) becomes

$$c_{i-3}B_{i-3}^3(t_i) + c_{i-2}B_{i-2}^3(t_i) + c_{i-1}B_{i-1}^3(t_i) = y_i, \quad (1 \leq i \leq n) \tag{2.47}$$

A useful condition is to impose $S''(t_0) = S''(t_n) = 0$ (Natural spline).

Noninterpolatory approximation

This procedure is introduced by Schoenberg(1967). Given f we define a spline function Sf by

$$Sf = \sum_{i=-\infty}^{\infty} f(x_i)B_i^k(x), \quad x_i = \frac{1}{k}(t_{i+1} + \cdots + t_{i+k}) \tag{2.48}$$

Assume $k \geq 2$ (2.48) does not interpolate f.

Our task is to investigate whether a continuous function f can be approximated by splines. We hold k while increasing the number of knots.

We introduce a special spline function that approximates f

$$g(x) = \sum_{i=-\infty}^{\infty} f(t_{i+2})B_i^k(x). \tag{2.49}$$

With this we have

Theorem 2.7.4 (Schoenberg-Whitney Theorem)**.**

$$\max_{t_0,t_n} |f(x) - g(x)| \leq k\omega(f, \delta), \tag{2.50}$$

where $\omega(f, \delta)$ the the modulus of continuity:

$$\omega(f, \delta) = \max_{|s-t|\leq\delta} |f(s) - f(t)|$$

and $\delta = \max_{-k\le i\le n+1} |t_i - t_{i-1}|$.

As a corollary to Wierstrass approx. theorem, we conclude that spline approx. (2.49) can be arb. close to f if $\delta \to 0$.

2.8 Bézier Curves

We introduce some basic idea of computer aided geometric design. Consider a mapping $f : D \to \mathbb{R}^m, m = 2, 3$ where D is an interval in $\mathbb{R}$. In computer graphics it is essential that the geometric objects can be visualized and manipulated effectively. For simplicity we assume $D = [0, 1]$. In gnenral, we use the mapping $(x - a)/(b - a)$.

Definition 2.8.1. For $n \in \mathbb{Z}^+$ we denote by P_n^m be the linear space of polynomials

$$p(x) = \sum_{k=0}^{n} a_k x^k, \ x \in \mathbb{R},$$

where $a_0, \cdots, a_n \in \mathbb{R}^m$.

We observe

$$1 \equiv \sum_{k=0}^{n} \binom{n}{k} x^k (1-x)^{n-k}$$

and set

$$B_k^n(x) := \binom{n}{k} x^k (1-x)^{n-k}, \qquad k = 0, \cdots, n.$$

Theorem 2.8.2. *Then we have*

(1) $B_k^n(x) \ge 0$

(2) $B_k^n(x) = B_{n-k}^n(1-x)$

(3) $B_0^n(x) = (1-x)B_0^{n-1}(x), \ B_n^n(x) = xB_{n-1}^{n-1}(x)$

(4) $x = 0$ *is a zero of* $B_k^n(x)$ *of order* k *and* $x = 1$ *is a zero of order* $n - k$.

(5) Each polynomial $B_k^n(x)$ *assume its maximum at* $x = k/n$.

(6) They satisfy

$$B_k^n(x) = xB_{k-1}^{n-1}(x) + (1-x)B_k^{n-1}(x)$$

for $k = 1, 2, \cdots, n$.

(7) The polynomials $B_0^n(x), \cdots, B_n^n(x)$ form a basis for P_n.

Proof. (5) follows from

$$\frac{d}{dx}B_k^n(x) = \binom{n}{k} x^{k-1}(1-x)^{n-k-1}(k-nx), \qquad k = 0, \cdots, n.$$

For (6) observe

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}.$$

□

Define

$$B_k^n(x; a, b) \equiv B_k^n\left(\frac{x-a}{b-a}\right).$$

Definition 2.8.3. The polynomial

$$p(x) = \sum_{k=0}^{n} B_k^n(x; a, b), \ x \in \mathbb{R}$$

is called the **Bézier polynomials** and the graph is called **Bézier curves** The coefficients $\mathbf{b}_0, \cdots, \mathbf{b}_n \in \mathbb{R}^m$ are called the **control points**.

The convex hull of points $\mathbf{b}_k, k = 0, \cdots, n$ is defined by

$$con\{\mathbf{b}_0, \cdots, \mathbf{b}_n\} := \left\{ \sum_{k=0}^{n} \alpha_k \mathbf{b}_k : \alpha_k \geq 0, \sum_{k=0}^{n} \alpha_k = 1 \right\}$$

Proposition 2.8.4. *(1) Note that $\mathbf{p}(a) = \mathbf{b}_0$, $\mathbf{p}(b) = \mathbf{b}_n$. i.e, the end points of Bézier polynomial and the Bézier curve coincide.*

(2) The Bézier curve is contained in the convex hull of Bézier curve points:

(3) $\mathbf{p}'(0) = n(\mathbf{b}_1 - \mathbf{b}_0)$, $\mathbf{p}'(1) = n(\mathbf{b}_n - \mathbf{b}_{n-1})$ i.e, the Bézier curve has the same tangent lines as the Bézier polygon.

Example 2.8.5 (Bézier curve of order 1 and 2)**.** (1) For $k = 1$ the Bézier curve is the line segment joining two control points, i.e., since $B_0^1 = x$, $B_1^1 = (1-x)$, we have

$$\mathbf{p}(x) = \mathbf{b}_0 x + \mathbf{b}_1(1-x).$$

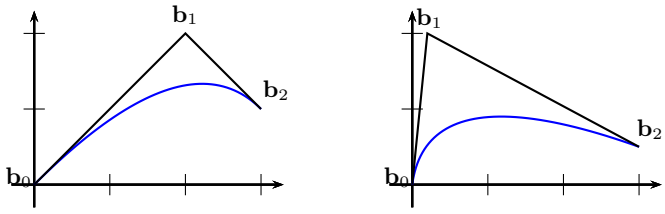


Figure 2.6: Bézier curves of degree 2 and polygons

$$B_0^2 = x^2,\ B_1^2 = 2x(1-x),\ B_2^2 = (1-x)^2$$

(2) On $[0,3]$, $B_0^2 = x^2/9,\ B_1^2 = 2x(3-x)/9,\ B_2^2 = (3-x)^2/9$. With control points $\mathbf{b}_0 = (0,0)$, $\mathbf{b}_1 = (2,2)$, and $\mathbf{b}_2 = (3,1)$, we have

$$\mathbf{p}(x) = (0,0)x^2/9 + (2,2)2x(3-x)/9 + (3,1)(3-x)^2/9 \tag{2.51}$$

Or componentwise

$$p_1(x) = 4x(3-x)/9 + (3-x)^2/3 = (3-x)(4x+9-3x)/9 = (3-x)(x+9)/9$$
$$p_2(x) = 4x(3-x)/9 + (3-x)^2/9 = (3-x)(4x+3-x)/9 = (3-x)(x+1)/3.$$

(3) With control points $\mathbf{b}_0 = (0,0)$, $\mathbf{b}_1 = (0.2,2)$, and $\mathbf{b}_2 = (3,0.5)$, we have

$$\mathbf{p}(x) = (0,0)x^2/9 + (0.2,2)2x(3-x)/9 + (3,0.5)(3-x)^2/9. \tag{2.52}$$

Componentwise

$$p_1(x) = 0.4x(3-x)/9 + (3-x)^2/3 = (3-x)(9-2.6x)/9$$
$$p_2(x) = 4x(3-x)/9 + 0.5(3-x)^2/9 = (3-x)(2.5x+1.5)/9.$$

Example 2.8.6 (Bézier curve of order 3)**.** On $[0,4]$

$$B_0^3(3,7) = x^3/64,\ B_1^3 = 3x^2(4-x)/64,\ B_2^3 = 3x(4-x)^2/64,\ B_3^3 = (4-x)^3/64.$$

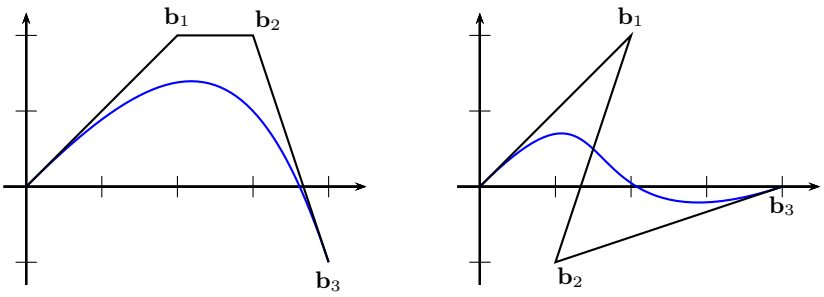


Figure 2.7: Bézier splines of order 3

With control points $\mathbf{b}_0 = (0,0)$, $\mathbf{b}_1 = (2,2)$, $\mathbf{b}_2 = (3,2)$, and $\mathbf{b}_3 = (4,-1)$ we have

$$\mathbf{p}(x) = (2,2)3x^2(4-x)/64 + (3,2)3x(4-x)^2/64 + (4,-1)(4-x)^3/64. \quad (2.53)$$

Or componentwise

$$p_1(x) = 6x^2(4-x)/64 + 9x(4-x)^2/64 + 4(4-x)^3/64 = (x^2+4x+64)(4-x)/64$$
$$p_2(x) = 6x^2(4-x)/64 + 6x(4-x)^2/64 - (4-x)^3/64 = (-x^2+32x-16)(4-x)/64.$$

With control points $\mathbf{b}_0 = (0,0)$, $\mathbf{b}_1 = (2,2)$, $\mathbf{b}_2 = (1,-1)$, and $\mathbf{b}_3 = (4,0)$ we have

$$\mathbf{p}(x) = (2,2)3x^2(4-x)/64 + (1,-1)3x(4-x)^2/64 + (4,0)(4-x)^3/64. \quad (2.54)$$

Or componentwise

$$p_1(x) = 6x^2(4-x)/64 + 3x(4-x)^2/64 + 4(4-x)^3/64 = (7x^2-20x+64)(4-x)/16$$
$$p_2(x) = 6x^2(4-x)/64 - 3x(4-x)^2/64 = (9x^2-12x)(4-x)/64.$$

2.8.1 Bézier spline

Bézier spline is a Composite Bézier curve. For example, on $[0,3]$ same as above, and on $[3,7]$, we take

$$B_0^2(3,7) = (x-3)^2/16,\ B_1^2 = 2(x-3)(7-x)/16,\ B_2^2 = (7-x)^2/16.$$

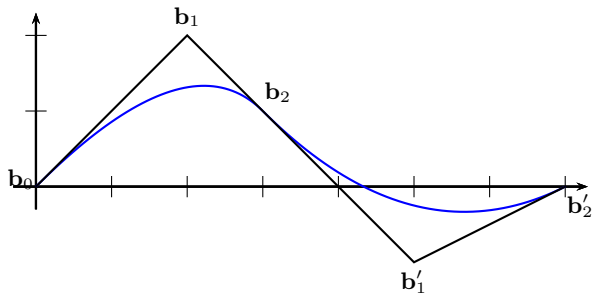


Figure 2.8: Bézier splines

With control points $b_0' = (3,1)$, $b_1' = (5,-1)$, and $b_2' = (7,0)$, we have

$$\mathbf{p}(x) = (3,1)(x-3)^2/16 + (5,-1)2(x-3)(7-x)/16 + (7,0)(7-x)^2/16 \quad (2.55)$$

Or if $\mathbf{p}(x) = (p_1(x), p_2(x))$, componentwise

$$\begin{aligned} p_1(x) &= 3(x-3)^2/16 + 10(x-3)(7-x)/16 + 7(7-x)^2/16 \\ &= (-16x + 160)/16 \\ p_2(x) &= (x-3)^2/16 - 2(x-3)(7-x)/16 = (x-3)(3x-17)/16. \end{aligned}$$

Chapter 3

Numerical integration

In this chapter, we study how to approximate the definite integral of some smooth functions. An idea is to use an interpolating polynomial $F(x)$ to evaluate the integral. Thus we have

$$\int_a^b f(x)\,dx \approx \int_a^b F(x)\,dx.$$

Our purpose is how to such an interpolation so that the error is minimized.

3.1 Quadrature based on equal intervals

We would like to design a quadrature for the following integral:

$$\int_a^b f(x)\,dx.$$

One point formula-rectangle rule: If $F(x)$ is a constant interpolation at x_0, then the above integral is approximated by $(b-a)f(x_0)$ called a **rectangle rule**. Using Taylor expansion, one can show that there exist ξ_0 and ξ_0 in $[a,b]$ such that the following hold.

$$\int_a^b f(x)\,dx = \begin{cases} (b-a)f(x_0) + \frac{(b-a)^2}{2}f'(\xi_0), \text{ if } x_0 \neq \frac{b+a}{2} \\ (b-a)f(x_0) + \frac{(b-a)^3}{24}f''(\xi_1), \text{ if } x_0 = \frac{b+a}{2}. \end{cases} \tag{3.1}$$

Thus one point formula is exact for constant polynomial(except mid point rule), so it is called a **0-th order method**.

Two point formula-trapezoidal rule:

$$\int_a^b f(x)\,dx = \frac{b-a}{2}(f(a)+f(b)) - \frac{(b-a)^3}{12}f''(\xi). \tag{3.2}$$

This formular can be derived by the error formula for linear interpolation:

$$f(x) - p_1(x) = \frac{f''(\xi)}{2}(x-a)(x-b).$$

This is exact for linear polynomials, hence it is a **first order method**.

Composite trapezoidal rule

If $[a,b]$ is partitioned into

$$a = x_0 < x_1 < \cdots < x_n = b,$$

we can use trapezoidal rule to each subinterval. Here the nodes are not necessarily uniformly spaced. Thus we obtain

$$\begin{aligned}\int_a^b f(x)\,dx &= \sum_{i=1}^{n}\int_{x_{i-1}}^{x_i} f(x)\,dx \\ &\doteq \frac{1}{2}\sum_{i=1}^{n}(x_i - x_{i-1})(f(x_i)+f(x_{i-1})).\end{aligned}$$

If the nodes are uniformly spaced with $h = (b-a)/n$, then the composite trapezoidal rule becomes

$$\int_a^b f(x)\,dx = \frac{h}{2}\left[f(a) + 2\sum_{i=1}^{n-1} f(a+ih) + f(b)\right].$$

The error in this case is

$$-\frac{(b-a)^2}{12}f''(\xi).$$

Three point formula-Simpson rule. Use Newton interpolation at three equally spaced points x_0, x_1, x_2. Derivation : Let

$$\int_{x_0}^{x_2} f(x)\,dx = Af(x_0) + Bf(x_1) + Cf(x_2), \tag{3.3}$$

where $f \in P_2$. For simplicity, we can assume $x_0 = -h, x_1 = 0$ and $x_2 = h$. Setting $f = 1, x, x^2$, we obtain

$$\begin{aligned} 2h &= A + B + C \\ 0 &= -Ah + Ch \\ \frac{2h^2}{3} &= Ah^2 + Bh^2. \end{aligned}$$

Solving for A, B and C, we obtain $A = C = \frac{h}{3}$ and $B = \frac{4}{3}h$. Hence we get the following Simpson's rule

$$\int_{x_0}^{x_2} f(x)\,dx \sim \frac{h}{3}(f(x_0) + 4f(x_1) + f(x_2)) \equiv s_1(f). \tag{3.4}$$

One can readily check that this is exact for a polynomial up to degree three. Hence this is a third order method, in fact one can show that

$$\int_{x_0}^{x_2} f(x)\,dx = \frac{h}{3}(f(x_0) + 4f(x_1) + f(x_2)) - \frac{h^5}{90}f^{(4)}(\xi_1), \quad \xi_1 \in [x_0, x_2]. \tag{3.5}$$

Composite Simpson's rule. If the domain is large, divide it by even number of intervals and applying Simpson's rule to a pair of subintervals, one can find a more accurate approximation. Let $x_i = a + ih, h = (b-a)/2n$. Then

$$\int_a^b f(x)\,dx = \int_{x_0}^{x_2} f(x)\,dx + \int_{x_2}^{x_4} f(x)\,dx + \cdots + \int_{x_{2n-2}}^{x_{2n}} f(x)\,dx.$$

If Simpson's rule is used for each integral, we obtain

$$\int_a^b f(x)\,dx \sim \frac{h}{3}\sum_{i=1}^{n}[f(x_{2i-2}) + 4f(x_{2i-1}) + f(x_{2i})].$$

To avoid repetitions it is rearranged as

$$\frac{h}{3}\left[f(x_0) + 2\sum_{i=2}^{n} f(x_{2i-2}) + 4\sum_{i=1}^{n} f(x_{2i-1}) + f(x_{2n})\right].$$

Five-eight rule. If we use the polynomial interpolation at the equally spaced points x_0, x_1, x_2 to approximate integral on $[x_0, x_1]$, we get five-eight rule. This is useful when we have information at a point x_2 outside the interval

$[a,b] = [x_0, x_1]$. Derive the following formula.

$$\int_{x_0}^{x_1} f(x)\,dx = \frac{h}{12}(5f(x_0)+8f(x_1)-f(x_2))+\frac{h^4}{24}f^{(3)}(\xi_2), \quad \xi_2 \in [x_0, x_2]. \tag{3.6}$$

3.1.1 General high order quadrature formula

Assume we have nodes $x_0, \cdots, x_n$ in $[a,b]$. We use the Lagrange interpolating polynomial to derive a quadrature. Consider

$$p_n(x) = \sum_{i=0}^{n} f(x_i)\ell_i(x), \tag{3.7}$$

where

$$\ell_i(x) = \prod_{j\neq i}^{n} \frac{(x-x_j)}{(x_i-x_j)}, \;\; 0 \le i \le n.$$

With some positive weight function $w(x)$, we let

$$\int_a^b f(x)w(x)\,dx \sim \int_a^b p_n(x)w(x)\,dx = \sum_{i=0}^{n} f(x_i)\int_a^b \ell_i(x)w(x)dx.$$

If the nodes are equally spaced, it is called the Newton-Cotes formula :

$$\int_a^b f(x)w(x)\,dx \sim \sum_{i=0}^{n} A_i f(x_i),$$

where

$$A_i = \int_a^b \ell_i(x)w(x)dx.$$

Table 3.1: Closed Newton Cotes formula

$\int_{x_0}^{x_1} f(x)\,dx =$	$\frac{h}{2}(f_0+f_1)$ (trapezoid)	$-\frac{h^3}{12}f^{(2)}(\xi)$
$\int_{x_0}^{x_2} f(x)\,dx =$	$\frac{h}{3}(f_0+4f_1+f_2)$ (Simpson)	$-\frac{h^5}{90}f^{(4)}(\xi)$
$\int_{x_0}^{x_3} f(x)\,dx =$	$\frac{3h}{8}(f_0+3f_1+3f_2+f_1)$	$-\frac{3h^5}{80}f^{(4)}(\xi)$
$\int_{x_0}^{x_4} f(x)\,dx =$	$\frac{2h}{45}(7f_0+32f_1+12f_2+32f_3+7f_4)$	$-\frac{8h^7}{945}f^{(6)}(\xi)$

Error of numerical integration

We consider **Newton-Cotes formula** (uniform interval). We assume the following situation: Let $a \le x_0 < x_1 < \cdots < x_n \le b$, $x_i - x_{i-1} = h$ and consider an approximation of $\int_a^b f(x)dx$ by some quadrature based on the data $(x_0, f(x_0), \cdots, (x_n, f(x_n))$. Use Taylor formula with remainder:

$$f(x) = f(x_0) + f'(x_0)(x-x_0) + \cdots + \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n + \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)^{n+1}$$

or Newton's interpolating polynomial

$$p_n(f) = f(x_0) + f[x_0, x_1](x-x_0) + \cdots + f[x_0, x_1, \cdots, x_n](x-x_0)\cdots(x-x_{n-1}).$$

By the error formula, we have

$$f(x) = p_n(f) + f[x_0, x_1, \cdots, x_n, x](x-x_0)\cdots(x-x_n).$$

Integrating the error term of Newton's formula we have

$$E_n(f) = \int_a^b f[x_0, x_1, \cdots, x_n, x] \prod_{i=0}^n (x-x_i)dx = \int_a^b \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^n (x-x_i)dx.$$

Since the term $\prod_{i=0}^n (x-x_i)$ does not change sign on each subinterval (x_s, x_{s+1}), we can use mean value theorem to see the above quantity is

$$\frac{f^{(n+1)}(\bar{\xi}_s)}{(n+1)!} \int_{x_s}^{x_{s+1}} \prod_{i=0}^n (x-x_i)dx = \frac{f^{(n+1)}(\bar{\xi}_s)}{(n+1)!} \times O(h^{n+2}),$$

for some $\bar{\xi}_s \in (x_s, x_{s+1})$.

A more precise error formula is follows (See Issacson-Keller). Let $\pi_n(x) = \prod_{i=0}^n (x-x_i)$ and $W_n(x) = \int_a^x \pi_n(x)\,dx$.

Lemma 3.1.1. *For n even we have*

(1) $W_n(a) = W_n(b) = 0$,

(2) $W_n(x) > 0$, $a < x < b$.

Proof. (1) $W_n(a)$ is trivial. To see $W_n(b) = 0$ holds, note that, for example, $\pi_n(x) = (x-x_0)(x-x_1)(x-x_2)$ is antisymmetric with respect to the mid point x_1. General case follows from this.

(2) Note $W_n(x_0) = W_n(x_2) = \cdots = W_n(x_n) = 0$. Thus it suffices to consider again the case $\pi_n(x) = (x - x_0)(x - x_1)(x - x_2)$. Direct computation shows $W_n(x) > 0$ for $a < x < b$. □

Lemma 3.1.2. *Let $x \in [x_0, x_n]$ be any point. There exists $\xi(x)$ such that*

$$\frac{d}{dx} f[x_0, \cdots, x_n, x] = \frac{f^{(n+2)}(\xi(x))}{(n+2)!}. \tag{3.8}$$

Proof.

$$\begin{aligned}
\frac{d}{dx} f[x_0, \cdots, x_n, x] &= \lim_{h \to 0} \frac{f[x_0, \cdots, x_n, x+h] - f[x_0, \cdots, x_n, x]}{h} \\
&= \lim_{h \to 0} \frac{f[x_0, \cdots, x_n, x+h] - f[x, x_0, \cdots, x_n]}{x + h - x} \\
&= \lim_{h \to 0} f[x_0, \cdots, x_n, x, x+h] \\
&= f[x_0, \cdots, x_n, x, x] = \frac{f^{(n+2)}(\xi(x))}{(n+2)!}
\end{aligned}$$

for some $\xi(x) \in [x_0, x_n]$ (Corollary 2.3.16). □

Theorem 3.1.3. *The error by Newton-Cotes formula is*

$$E_n(f) = \begin{cases} \dfrac{f^{(n+2)}(\bar{\xi})}{(n+2)!} \displaystyle\int_a^b x \prod_{i=0}^{n} (x - x_i) dx & \text{if } n \text{ is even,} \\ \dfrac{f^{(n+1)}(\bar{\xi})}{(n+1)!} \displaystyle\int_a^b \prod_{i=0}^{n} (x - x_i) dx & \text{if } n \text{ is odd.} \end{cases} \tag{3.9}$$

Thus an extra accuracy is obtained for n even.

Proof. Let n be even. Then from Newton's form of interpolating polynomial, we have

$$\begin{aligned}
E_n(f) &= \int_a^b \pi_n(x) f[x_0, \cdots, x_n, x]\, dx \\
&= W_n(x) f[x_0, \cdots, x_n, x]|_a^b - \int_a^b W_n(x) \frac{d}{dx} f[x_0, \cdots, x_n, x]\, dx \\
&= -\int_a^b W_n(x) \frac{d}{dx} f[x_0, \cdots, x_n, x]\, dx.
\end{aligned}$$

Hence

$$E_n(f) = -\int_a^b W_n(x) \frac{f^{(n+2)}(\xi_1)}{(n+2)!}\, dx.$$

Since $W_n(x) > 0$ by Lemma 3.1.1, we have by MVT for integral that

$$E_n(f) = -\frac{f^{(n+2)}(\eta)}{(n+2)!}\int_a^b W_n(x)\,dx,\ a < \eta < b.$$

Moreover, integration by parts shows

$$\begin{aligned}\int_a^b W_n(x)\,dx &= xW_n(x)|_a^b - \int_a^b x\frac{d}{dx}W_n(x)\,dx \\ &= -\int_a^b x\pi_n(x)\,dx > 0.\end{aligned}$$

We skip odd case. □

We see

$$E_n(f) = \begin{cases} O(h^{n+3}) & \text{if } n \text{ is even,} \\ O(h^{n+2}) & \text{if } n \text{ is odd.} \end{cases} \tag{3.10}$$

Now we present the error analysis of the special case of **Simpson's rule**. Let $n = 2$ and $x_1 - x_0 = x_2 - x_1 = h$. One can use Taylor's theorem to derive the error estimate, but we present a method based on Hermite interpolation which can be extended to higher degree. Let $\widetilde{F}_3(x)$ be a polynomial of degree 3 such that

$$\widetilde{F}_3(x_i) = f(x_i),\ i = 0, 1, 2 \text{ and } \widetilde{F}_3'(x_1) = f'(x_1).$$

Then by Hermite interpolation formula(Newton's interpolation with repeated nodes at x_1)

$$f(x) - \widetilde{F}_3(x) = (x - x_0)(x - x_1)^2(x - x_2)\frac{f^{(4)}(\xi(x))}{4!}.$$

Since Simpson's rule is exact for a polynomial of degree three, we have

$$E(f) = \int_{x_0}^{x_2}[f(x) - \widetilde{F}_3(x)]\,dx = \int_{x_0}^{x_2} f(x)\,dx - \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)].$$

On the other hand,

$$\begin{aligned} E(f) &= \int_{x_0}^{x_2} (x-x_0)(x-x_1)^2(x-x_2)\frac{f^4(\xi(x))}{4!}\,dx \\ &= \frac{f^{(4)}(\xi_1)}{4!}\int_{x_0}^{x_2}(x-x_0)(x-x_1)^2(x-x_2)\,dx \end{aligned}$$

by the integral mean value theorem. The last integral can be computed directly and we see $E(f) = -\frac{h^5}{90}f^{(4)}(\xi_1)$. Hence with $h = (b-a)/2$, we have

$$\int_a^b f(x)\,dx = s_1(f) - \frac{(b-a)^5}{2^5 \cdot 90}f^4(\xi).$$

Posteriori Estimate

Even though an error term of the form $O(h^k)$ suggests how big the error is, we still do not know how accurate the answer is. We introduce a method to estimate the error after the computation. We take Simpson's rule, for example. Assume $f^{(4)}(\xi) \doteq f^{(4)}$ is constant on a small interval $[a,b]$, we use Simpson's rule on each of the interval $[a,x_1],[x_1,b]$, where $x_1 = (a+b)/2$.

$$\int_a^b f(x)\,dx = \int_a^{\frac{a+b}{2}} f(x)\,dx + \int_{\frac{a+b}{2}}^b f(x)\,dx = s_2(f) - \frac{(b-a)^5}{2^5 \cdot 2^5 \cdot 90}f^{(4)} \times 2,$$

where $s_2(f)$ is the result of Simpson's rule applied on $[a,x_1]$ and $[x_1,b]$ resp.

$$\begin{aligned} s_2(f) - s_1(f) &= -\frac{(b-a)^5}{2^5 \cdot 90}[1 - \frac{1}{2^4}]f^{(4)} = -\frac{15}{16}\frac{(b-a)^5}{2^5 \cdot 90}f^{(4)}, \\ \frac{s_2(f) - s_1(f)}{15} &= -\frac{1}{16}\frac{(b-a)^5}{2^5 \cdot 90}f^{(4)} = \int_a^b f(x)\,dx - s_2(f). \end{aligned}$$

L.H.S is computable by machine, which is a good indication of error. If this error is undesirable, we can divide the interval so that $|s_2 - s_1| < \varepsilon$.

3.2 Gaussian quadrature-unequal intervals

Check that the quadrature

$$\int_{-1}^1 f(x)\,dx \doteq f(-\sqrt{\frac{1}{3}}) + f(\sqrt{\frac{1}{3}})$$

is exact up to degree 3. The degree of precision of the following quadrature is 5.

$$\int_{-1}^{1} f(x)\,dx \doteq \frac{5}{9}f(-\sqrt{\frac{3}{5}}) + \frac{8}{9}f(0) + \frac{5}{9}f(\sqrt{\frac{3}{5}}).$$

A three point formula is of the form:

$$\alpha_0 f(x_0) + \alpha_1 f(x_1) + \alpha_2 f(x_2).$$

More generally, we consider a quadrature with unspecified points:

$$\int_a^b f(x)w(x)dx \approx \sum_{i=0}^{n} A_i f(x_i), \tag{3.11}$$

where w is a fixed *positive* weight function. This formula is exact for polynomial of degree up to n if and only if(let $f(x) = \ell_i(x)$)

$$A_i = \int_a^b w(x) \prod_{j\neq i}^{n} \frac{x - x_j}{x_i - x_j} dx.$$

Note that there are no restrictions on the nodes. By placing nodes at proper places, we can obtain $2n + 1$ order of approximation.

Theorem 3.2.1. *Let q be a nonzero polynomial of degree $n + 1$ which is orthogonal to P_n(the set of all polynomials of degree $\leq n$) with respect to the weight w, i.e, we have*

$$\int_a^b q(x)p(x)w(x)dx = 0, \text{ for all } p(x) \in P_n.$$

If $x_0, x_1, \cdots, x_n$ are the zeros of q, then the quadrature formula (3.11) is exact for all $f \in P_{2n+1}$.

Proof. Let $f \in P_{2n+1}$. Dividing f by q, we obtain

$$f = qp + r \quad (\text{degree of } p, r < n + 1)$$

and we see $f(x_i) = r(x_i)$. Since q is orthogonal to p w.r.t w and the formula (3.11) is exact for polynomial of degree n, thus we have

$$\int_a^b fw\,dx = \int_a^b rw\,dx = \sum_{i=0}^{n} A_i r(x_i) = \sum_{i=0}^{n} A_i f(x_i).$$

□

Now the remaining task is how to find orthogonal polynomials and their zeros. Fortunately, a few cases are well-known: Let $[-1,1]$. Then with $w=1$, the Legendre polynomials are orthogonal on $[-1,1]$. i.e,

$$\int_{-1}^{1} p_n(x)p_m(x)\,dx = \delta_{nm}.$$

A few Legendre polynomials are:

$$\begin{aligned}
p_0(x) &= \frac{1}{\sqrt{2}}, \quad p_1(x) = \sqrt{\frac{3}{2}}x \\
p_2(x) &= \sqrt{\frac{5}{2}}\frac{1}{2}(3x^2-1), \quad p_3(x) = \sqrt{\frac{7}{2}}\frac{1}{2}(5x^3-3x) \\
p_4(x) &= \sqrt{\frac{9}{2}}\frac{1}{8}(35x^4-30x^2+3), \quad p_5(x) = \sqrt{\frac{11}{2}}\frac{1}{8}(63x^5-70x^3+15x) \\
p_6(x) &= \sqrt{\frac{13}{2}}\frac{1}{24}(7\cdot 33x^6-63\cdot 5x^4+35\cdot 3x^2-5) \\
&\ ,\ldots, \\
p_n(x) &= \left(n+\frac{1}{2}\right)^{1/2}\frac{1}{n!\,2^n}\frac{d^n}{dx^n}(x^2-1)^n := \left(n+\frac{1}{2}\right)^{1/2}\Phi_n(x).
\end{aligned}$$

Exercise 3.2.2 (Rodriguez). Let $\Phi_n(x) = \frac{1}{n!\,2^n}\frac{d^n}{dx^n}(x^2-1)^n$ and show that

(1) $\Phi_n(1)=1$, $\Phi_n(-1)=(-1)^n$.

(2) Φ_n is generated by the recursive formula

$$\Phi_{n+1} = \frac{2n+1}{n+1}x\Phi_n - \frac{n}{n+1}\Phi_{n-1},\ \Phi_0=1,\ \Phi_1=x.$$

Observe $\Phi_{n+1}(x) - ax\Phi_n(x) = \sum_{i=0}^{n}\alpha_i\Phi_i(x)$ and determine a, α_i.

(3) $\Phi_n(x)$ is a solution of Legendre differential equation

$$(1-x^2)y'' - 2xy' + n(n+1)y = 0.$$

(4) $\{\Phi_n\}$ are orthogonal.

(5) Find the weights $A_i, i=0,\cdot,n$ for $n=1,2,3,4$.

Orthogonality. For $m > n$,

$$\begin{aligned}
&\int_{-1}^{1} D^n[(x^2-1)^n]D^m[(x^2-1)^m]\,dx \\
= \;& D^n[(x^2-1)^n]D^{m-1}[(x^2-1)^m]|_{-1}^{1} - \int_{-1}^{1} D^{n+1}[(x^2-1)^n]D^{m-1}[(x^2-1)^m]\,dx \\
= \;& (-1)^2 \int_{-1}^{1} D^{n+2}D^{m-2}\,dx = \cdots \\
= \;& (-1)^n \int_{-1}^{1} D^{n+n}D^{m-n}\,dx = C_{2n}(-1)^n \int_{-1}^{1} D^{m-n}\,dx = 0.
\end{aligned}$$

Here C_{2n} is $2n$-th derivative of $(x^2-1)^n$ which is constant. The last integral vanishes as $D^{m-n-1}(x^2-1)^n\big|_{-1}^{1}$.

When $m = n$,

$$\begin{aligned}
\frac{1}{(2n)!}\int_{-1}^{1}\cdots &= \frac{(-1)^n}{(2n)!}\int_{-1}^{1} D^{2n}\cdot D^0 = (-1)^n\int_{-1}^{1}(x^2-1)^n\,dx \\
&= x(x^2-1)^n|_{-1}^{1} - n\int_{-1}^{1} 2x^2(x^2-1)^{n-1}\,dx \\
&= (-1)^2 2^2 n(n-1)\int_{-1}^{1}(x^2-1)^{n-2}\frac{x^4}{3}\,dx = \cdots = 2^n n!\,\frac{1}{1\cdot 3\cdot 5\cdots} \\
&= 2^n n!\frac{x^{2n+1}|_{-1}^{1}}{1\cdot 3\cdot 5\cdots(2n+1)} = \frac{2^{n+1}n!}{1\cdot 3\cdot 5\cdots(2n+1)}.
\end{aligned}$$

$$\therefore \qquad \int_{-1}^{1} D^n\cdot D^n = \frac{(2n)!\,2^{n+1}\cdot n!}{1\cdot 3\cdot 5\cdots(2n+1)} = \frac{(2^n\cdot n!)^2}{n+\frac{1}{2}}.$$

Normalizing, we obtain the **orthonormality** of $\Phi_n(x)$.

3.2.1 Error of Gaussian quadrature

Now we estimate the error of Gaussian quadrature.

Motive: Given $f \in C^{2n+2}[a,b]$, think of an interpolating poly p of degree $2n+1$ with interpolation points at $x_i, i = 0,\cdots,n$ for which the error form $E(\xi(x))$ is known (Hermite interpolation). Then we know

$$\int_a^b E(\xi(x))dx = \int_a^b (f(x)-p(x))w(x)dx.$$

Then we can evaluate $\int_a^b p(x)w(x)dx$ by Gaussian quadrature. A choice is the Hermite interpolation polynomial p of degree $2n+1$ with $\gamma = 1$ at $x_i, i =$

$0, \cdots, n$.

Lemma 3.2.3. *In Gaussian quadrature, the coefficients are positive and their sum is* $\int_a^b w(x)dx$. *In particular, if* $[a,b] = [-1,1]$ *and* $w \equiv 1$, *then* $\sum_{i=0}^n A_i = 2$.

Proof. Fix n and let q be a polynomial of degree $n+1$ which is orthogonal to P_n. The zeros of q are denoted by $x_0, \cdots, x_n$. Let $p(x) = q(x)/(x - x_j)$ for some j. Since $p^2(x)$ is of degree at most $2n$, Gaussian quadrature with $x_0, \cdots, x_n$ will be exact for $p^2(x)$. Hence

$$0 < \int_a^b p^2(x)w(x)dx = \sum_{i=0}^n A_i p^2(x_i) = A_j p^2(x_j)$$

so that $A_j > 0$. Now use Gaussian quadrature for $f(x) = 1$ to see

$$\int_a^b w(x)dx = \sum_{i=0}^n A_i.$$

□

Theorem 3.2.4. *For any* $f \in C^{2n+2}[a,b]$, *the error term* $E(f)$ *in the Gaussian quadrature*

$$\int_a^b f(x)w(x)dx = \sum_{i=0}^n A_i f(x_i) + E(f)$$

satisfies

$$E(f) = \frac{f^{(2n+2)}(\bar{\xi})}{(2n+2)!} \int_a^b q^2(x)w(x)dx,$$

for some $a < \bar{\xi} < b$ *and* $q(x) = \prod_{i=0}^n (x - x_i)$.

Proof. From (2.25) we see the Hermite interpolation polynomial p of degree at most $2n+1$ with $\gamma = 1$ at $x_i, i = 0, \cdots, n$ satisfies

$$f(x) - p(x) = \frac{f^{(2n+2)}(\xi(x))}{(2n+2)!} q^2(x).$$

Hence

$$\int_a^b (f(x) - p(x))w(x)\,dx = \int_a^b \frac{f^{(2n+2)}(\xi(x))}{(2n+2)!} q^2(x)w(x)\,dx.$$

Since the Gaussian quadrature is exact for polynomial of degree $2n+1$, we have

$$\int_a^b p(x)w(x)\,dx = \sum_{i=0}^{n} A_i p(x_i) = \sum_{i=0}^{n} A_i f(x_i).$$

Hence

$$\int_a^b f(x)w(x)\,dx - \sum_{i=0}^{n} A_i f(x_i) = \int_a^b \frac{f^{(2n+2)}(\xi(x))}{(2n+2)!} q^2(x)w(x)\,dx. \qquad (3.12)$$

Furthermore, the mean value theorem(by positiveness of q^2) implies that there exists some $\bar{\xi}$ such that the RHS of (3.12) equals

$$\frac{f^{(2n+2)}(\bar{\xi})}{(2n+2)!}\int_a^b q^2(x)w(x)\,dx.$$

If we let $k_n := \frac{(2n)!}{2^n(n!)^2}$ be the leading coefficients of $\Phi_n(x)$, then

$$q(x) = \frac{p_{n+1}(x)}{\sqrt{n+\frac{3}{2}}\,k_{n+1}}.$$

So

$$E_{n+1}(f) = \frac{f^{(2n+2)}(\bar{\xi})}{(2n+2)!}\int_a^b q^2(x)w(x)\,dx = \frac{2}{2n+3}\frac{f^{(2n+2)}(\bar{\xi})}{(2n+2)!k_{n+1}^2}.$$

□

References

(1) Philip J. Davis and P. Rabinowitz. Methods of Numerical Intergraion, 2nd ed, 1984. AP.

(2) Philip J. Davis, Interpolation and Approximation, 1963, Dover.

(3) Hildebrand, F.B. Introduction to Numerical analysis, 1956, McGraw-Hill.

(4) Carl-Erik Froberg, Numerical mathematics, 1985 Benjamin/Cummings Pub. Company inc. pp. 288–296.

(5) http:/mathworld.wolfram.com/Chebyshev-GaussQuadrature.html

(6) http:/mathworld.wolfram.com/LobattoQuadrature.html

(7) http:/mathworld.wolfram.com/RadauQuadrature.html

(8) Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables, Abramowitz, M. and Stegun.

3.3 More on Gauss type quadrature

Gauss Lobatto quadrature

Let ϕ_n be the Legendre polynomial of degree n on $[-1,1]$ and let $x_i, (i = 0, \cdots, n)$ be the zeros of $p(x) = \phi_{n+1}(x) + \lambda\phi_n(x) + \mu\phi_{n-1}(x)$. Here λ, μ are chosen so that $p(-1) = p(1) = 0$. Then we have $x_0 = -1$ and $x_n = 1$. Now consider the following quadrature

$$A_0 f(x_0) + A_n f(x_n) + \sum_{i=1}^{n-1} A_i f(x_i) \tag{3.13}$$

to approximate $\int_{-1}^{1} f(x)dx$.

Theorem 3.3.1. *The quadrature (3.13) is exact for polynomials degree up to* $2n-1$.

Hence the coefficients A_i and quadrature points x_i's are determined by letting $f = 1, x, \cdots, x^{2n-1}$.

Proof. First λ, μ are determined by the conditions $p(-1) = p(1) = 0$. Assume the formula is exact for P_n. Let f be in P_{2n-1}. Then dividing f by p, we can write $f = pq + r$ for some $q \in P_{n-2}$ and $r \in P_n$.

$$\begin{aligned}
\int_{-1}^{1} f(x)dx &= \int_{-1}^{1} pq\,dx + \int_{-1}^{1} r\,dx \\
&= 0 + \int_{-1}^{1} r\,dx \text{ (orthogonality)} \\
&= \sum_{i=0}^{n} A_i r(x_i) \text{ (since } r \text{ is degree } n) \\
&= \sum_{i=0}^{n} A_i f(x_i) \text{ (since } p(x_i) = 0 \text{ for } i = 0, \cdots, n).
\end{aligned}$$

□

Example 3.3.2. For $n = 3$, we note that

$$p = 35x^4 - 30x^2 + 3 + \lambda(5x^3 - 3x) + \mu(3x^2 - 1).$$

Using $p(\pm 1) = 0$, we get $\lambda = 0, \mu = -4$. Hence

$$p(x) = 35x^4 - 42x^2 + 7 = 7(5x^2 - 1)(x^2 - 1).$$

Thus $x_1 = -\frac{1}{\sqrt{5}}$, $x_2 = \frac{1}{\sqrt{5}}$.

When $n = 3$, there is a special method to find the formula. Assume the following symmetric formula:

$$A_0 f(-1) + A_1 f(-x_1) + A_1 f(x_1) + A_0 f(1).$$

By symmetry, it is exact for any odd degree polynomial. We can find A_0, A_1 and x_1 by imposing the condition that it is exact for $f = 1, x^2, x^4$.

$$\begin{aligned} A_0 + A_1 &= 1 \\ A_0 + A_1 x_1^2 &= \frac{1}{3} \\ A_0 + A_1 x_1^4 &= \frac{1}{5}. \end{aligned}$$

Solving we get $x_1 = -\frac{1}{\sqrt{5}}$. Thus, the Gauss-Lobatto formula in this case is

$$\frac{1}{6} f(-1) + \frac{5}{6} f(-\frac{1}{\sqrt{5}}) + \frac{5}{6} f(\frac{1}{\sqrt{5}}) + \frac{1}{6} f(1).$$

(For other general formula, See Beurling's book)

Assume the formula

$$A_0 f(-1) + A_n f(-x_1) + \sum_{i=1}^{n-1} A_i f(x_i)$$

is exact up to degree $2n - 1$, where the sum is up to $n/2$ for n even and up to

$(n+1)/2$ for n odd.

$$\begin{aligned} f=1: \quad & A_0 + A_n + \sum_{i=1}^{n-1} A_i = 2 \\ f=x: \quad & -A_0 + A_n + \sum_{i=1}^{n-1} A_i x_i = 0 \\ : \quad & \cdots \\ f=x^k: \quad & (-1)^k A_0 + A_n + \sum_{i=1}^{n-1} A_i x_i^k = \frac{2}{k+1} \\ & \cdots \end{aligned}$$

Weight and points of Gauss Lobatto quadrature

Let $\Phi_n(x)$ be the unnormalized Legendre polynomial of degree n and let $\phi_n(x)$ be its normalized(leading coefficient is 1) Legendre polynomial. Let $k_n = \frac{(2n)!}{2^n(n!)^2}$ be the leading coefficients of $\Phi_n(x)$. Then general Gauss Lobatto quadrature based on $n+1$ points for $[-1,1]$ is

$$\int_{-1}^{1} f(x)dx = \frac{2}{n(n+1)}(f(-1)+f(1)) + \sum_{i=1}^{n-1} A_i f(x_i),$$

where $x_i, i = 1, \cdots, n-1$ are zeros of $\phi_n'(x)$, the derivative of Legendre polynomial of degree n and

$$A_i = \frac{2}{n(n+1)k_n^2\phi_n^2(x_i)}.$$

The error is of the form

$$E = c_n f^{(2n)}(\xi)$$

which is exact up to degree $2n-1$ compared to the $2n+1$(Gauss quadrature); we have given up the freedom of location of points.

General Gauss Radau quadrature

Let $\Phi_n(x)$ be the unnormalized Legendre polynomial of degree n and let $\phi_n(x)$ be its normalized(leading coefficient is 1) Legendre polynomial. Let $k_n = \frac{(2n)!}{2^n(n!)^2}$ be the leading coefficients of $\Phi_n(x)$. Then the general Gauss Radau quadrature based on $n+1$ points(the point -1 plus n points in the open

interval) for $[-1,1]$ is

$$\int_{-1}^{1} f(x)dx = \frac{2}{(n+1)^2}f(-1) + \sum_{i=1}^{n} A_i f(x_i),$$

where $x_i, i = 1, \cdots, n$ are the zeros of

$$\frac{k_n\phi_n(x) + k_{n+1}\phi_{n+1}(x)}{x-1}$$

and

$$A_i = \frac{1-x_i}{(n+1)^2 k_n^2 \phi_n^2(x_i)}, i = 1, \cdots, n.$$

The error is of the form

$$E = d_n f^{(2n+1)}(\xi).$$

General Gauss Chebysheff quadrature

If we use $w(x) = \frac{1}{\sqrt{1-x^2}}$ for the Gauss quadrature, we obtain Chebysheff polynomial of first kind and consequently obtain a quadrature. The nodes x_i are the zeros of a polynomial orthogonal w.r.t $w(x) = \frac{1}{\sqrt{1-x^2}}$.

It turns out the Chebysheff polynomial satisfies

$$\int_{-1}^{1} T_m(x)T_n(x)\frac{1}{\sqrt{1-x^2}}\,dx = c_{mn}\delta_{mn} \tag{3.14}$$

and hence the zeros are

$$x_k = \cos\left(\frac{2k+1}{2(n+1)}\pi\right), \quad k = 0, \cdots, n.$$

The interpolating polynomial p satisfies

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_0)\cdots(x-x_n). \tag{3.15}$$

Thus Gauss Chebysheff quadrature with weight $w(x) = \frac{1}{\sqrt{1-x^2}}$ based on $n+1$ points for $[-1,1]$ is

$$\int_{-1}^{1} f(x)w(x)\,dx = \sum_{i=0}^{n} A_i f(x_i),$$

where $x_i, i = 0, \cdots, n$ are the zeros of $T_{n+1}(x)$ and it can be shown that

$$A_i = \int_{-1}^{1} \frac{\ell_i(x)}{\sqrt{1-x^2}}\, dx = \frac{\pi}{n+1}, \quad i = 0, \cdots, n. \tag{3.16}$$

Remark 3.3.3. If we use $w(x) = \sqrt{1-x^2}$ instead, we obtain Chebysheff polynomial of second kind and hence obtain Gauss Chebysheff quadrature of second kind which will not be discussed further.

$$x_k = \cos\left(\frac{k}{(n+2)}\pi\right), \quad w_i = \frac{\pi}{n+2}\sin^2\left(\frac{k}{(n+2)}\pi\right), \quad k = 0, \cdots, n.$$

Exercise 3.3.4. (1) Verify (3.14) and (3.16).

(2) Find quadrature points $x_0, \cdots, x_n$ and weight $A_0, \cdots, A_n$ for Gauss-Legendre quadrature, when $n = 1, 2, 3$

(3) Do the same for Gauss- Lobatto quadrature, when $n = 2, 3, 4$

Note $T_0 = 1, T_1(x) = x$ and for $n \geq 0$ we have

$$T_{n+1}(x) = 2^{-n}\cos[(n+1)\cos^{-1}x] = (x-x_0)(x-x_1)\cdots(x-x_n)$$

is the monic polynomial having $x_i, i = 0, \cdots, n$ as zeros. The Gauss Chebysheff quadrature using these points is exact for P_{2n+1}. Hence for $m \leq 2n+1$

$$\int_{-1}^{1} \frac{T_m(x)}{\sqrt{1-x^2}}\, dx = \sum_{i=0}^{n} A_i T_m(x_i).$$

Since

$$\sum_{i=0}^{n} A_i T_m(x_i) = \begin{cases} \pi & m = 0 \\ 0 & m = 1, 2, \cdots, n. \end{cases}$$

we see

$$\begin{aligned} m=0: &\quad \sum_{i=0}^{n} A_i = \pi \\ m=1: &\quad \sum_{i=0}^{n} A_i T_1(x_i) = 0 \\ m=2: &\quad \sum_{i=0}^{n} A_i T_2(x_i) = 0 \\ &\quad \cdots = 0 \\ m=n: &\quad \sum_{i=0}^{n} A_i T_n(x_i) = 0. \end{aligned}$$

If m is even, $x_i = -x_{n-i}$ and $x_{\frac{n}{2}} = 0$. If m is odd, $x_i = -x_{n-i}$

Lemma 3.3.5. *For* $0 \le m < n$

$$\sum_{i=0}^{n} T_{m+1}(x_i) = 0.$$

Proof. We see $x_i = -x_{n-i}$ and $x_{\frac{n}{2}} = 0$ if n is even. First assume m is even.

$$\begin{aligned} T_{m+1}(x_i) &= 2^{-m}\cos[(m+1)\cos^{-1}x_i] \\ &= 2^{-m}\cos[(m+1)\cos^{-1}(-x_{n-i})] \\ &= -2^{-m}\cos[(m+1)\cos^{-1}(x_{n-i})] \\ &= -T_{m+1}(x_{n-i}). \end{aligned}$$

So in this case we have the result. Next assume m is odd. We see by the same way as above,

$$T_{m+1}(x_i) = T_{m+1}(x_{n-i})$$

and $\cos[(m+1)\cdot 0] = \cos[(m+1)\pi] = 1$. ???

A clean proof is:

$$\begin{aligned}\sum_{i=0}^{n}\cos(2i+1)t &= \sum_{i=0}^{n}\frac{\cos(2i+1)t\sin t}{\sin t}\\ &= \frac{1}{2\sin t}\sum_{i=0}^{n}[\sin(2i+2)t-\sin(2i)t]\\ &= \frac{1}{2\sin t}\sin(2n+2)t\\ &= \frac{1}{2\sin t}\sin(k\pi)=0,\end{aligned}$$

when $t=t_k=\frac{k\pi}{2n+2}$ for $k=1,2,\cdots,n$. □

So $A_i=\frac{\pi}{n+1}$ are the solution by checking.

$$\begin{aligned}m=0: &\quad \sum_{i=0}^{n}A_i &&=\pi\\ m=k: &\quad \sum_{i=0}^{n}A_i\cos[k\frac{2i+1}{2(n+1)}\pi] &&=0\\ &\quad \cdots\\ m=n: &\quad \sum_{i=0}^{n}A_i\cos[n\frac{2i+1}{2(n+1)}\pi] &&=0.\end{aligned}$$

Another method, Try: Note $\int_{-1}^{1}\frac{T_k(x)}{\sqrt{1-x^2}}\,dx=\sum_{i=0}^{n}A_ix_i^k$ for $k=0,1,\cdots,n$,

$$\begin{aligned}f=1: &\quad \sum_{i=0}^{n}A_i=\int_{-1}^{1}\frac{1}{\sqrt{1-x^2}}\,dx=\pi\\ f=x: &\quad \sum_{i=0}^{n}A_ix_i=\int_{-1}^{1}\frac{x}{\sqrt{1-x^2}}\,dx=0\\ f=x^2: &\quad \sum_{i=0}^{n}A_ix_i^2=\int_{-1}^{1}\frac{x^2}{\sqrt{1-x^2}}\,dx=\frac{\pi}{2}\end{aligned}$$

$$\begin{aligned}f=x^k(\text{even}): &\quad \sum_{i=0}^{n}A_ix_i^k=\int_{-1}^{1}\frac{x^k}{\sqrt{1-x^2}}\,dx=2\frac{k-1}{k}\frac{k-3}{k-2}\cdots\frac{\pi}{2}\\ f=x^k(\text{odd}): &\quad \sum_{i=0}^{n}A_ix_i^k=\int_{-1}^{1}\frac{x^k}{\sqrt{1-x^2}}\,dx=0\end{aligned}$$

$$\int_0^1 \frac{x^{2n}}{\sqrt{1-x^2}}\,dx = \int_0^{\pi/2} \sin^{2n}\theta\,d\theta = \frac{2n-1}{2n}\frac{2n-3}{2n-2}\cdots\frac{\pi}{2}.$$

Chapter 4

Numerical solution of O.D.Es

4.1 Introduction

Consider a (scalar valued) initial value problem

$$\begin{cases} x'(t) &= f(t,x) \\ x(t_0) &= x_0. \end{cases} \tag{4.1}$$

Theorem 4.1.1 (Existence and uniqueness)**.** *Let f and $\frac{\partial f}{\partial x}$ be continuous on the region $R = [0,T] \times [a,b]$. Then there is some interval $[0,T_1]$ in which there exists a unique solution to (4.1).*

Higher order Equations

We may consider higher order equations in n unknowns. k-th order DE has the following implicit form:

$$\mathbf{f}(t, \mathbf{x}, \mathbf{x}', \cdots, \mathbf{x}^{(k)}) = 0$$

where $\mathbf{f} : \mathbb{R}^{rn+n+1} \to \mathbb{R}^n$ is a known function and $\mathbf{x}(t) \in \mathbb{R}^n$ is the unknown. If an ODE can be written as

$$\mathbf{x}^{(k)} = \mathbf{f}(t, \mathbf{x}, \mathbf{x}', \cdots, \mathbf{x}^{(k-1)})$$

then it is explicit form. Any higher order ODE can be written as a system of first order equations as follows: We introduce new unknowns

$$\mathbf{u}_1(t) = \mathbf{x}, \mathbf{u}_2(t) = \mathbf{x}^{(1)}, \cdots, \mathbf{u}_k(t) = \mathbf{x}^{(k-1)}, \quad \mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_k)^T$$

so that the k-th order ODE becomes

$$\mathbf{U}' = \begin{bmatrix} \mathbf{u}_1' \\ \mathbf{u}_2' \\ \cdots \\ \mathbf{u}_{k-1}' \\ \mathbf{u}_k' \end{bmatrix} = \begin{bmatrix} \mathbf{u}_2 \\ \mathbf{u}_3 \\ \cdots \\ \mathbf{u}_k \\ \mathbf{f}(t, \mathbf{u}_1, \cdots, \mathbf{u}_k) \end{bmatrix} := \mathbf{G}(t, \mathbf{U}) \tag{4.2}$$

Thus it suffices to consider the following IVP:

$$\begin{cases} \mathbf{x}'(t) &= \mathbf{G}(t, \mathbf{x}) \\ \mathbf{x}(t_0) &= \mathbf{x}_0. \end{cases} \tag{4.3}$$

4.1.1 Existence and Stability

4.2 Numerical Methods

4.2.1 Finite Difference Method

Given a differentiable function f we let

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

This is called a **forward difference**. The approximation

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

is called a **backward difference**.

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f''(\xi_1), \quad \xi_1 \in (x, x+h) \\ \therefore \frac{f(x+h) - f(x)}{h} &= f'(x) + \frac{h}{2}f''(x) + \frac{h^2}{6}f''(\xi_1). \end{aligned}$$

Similarly,

$$\begin{aligned} f(x-h) &= f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f''(\xi_2), \quad \xi_2 \in (x-h, x) \\ \therefore \frac{f(x) - f(x-h)}{h} &= f'(x) - \frac{h}{2}f''(x) + \frac{h^3}{6}f''(\xi_2). \end{aligned}$$

Take the average

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{6}\frac{f'''(\xi_1) + f'''(\xi_2)}{2}.$$

By MVT the average $\frac{f'''(\xi_1)+f'''(\xi_2)}{2}$ is achieved by $f'''(\gamma)$ for some γ. Thus we obtain a second order approximation to $f'(x)$. This is a **central difference**.

$$\frac{f_{i+1} - f_{i-1}}{2h_i} - f'(x_i) = O(\frac{h_i^2}{6}) \quad \text{if } h_i = h_{i+1}.$$

Treating the end points

It is sometimes necessary to have a $O(h^2)$ method at end points. But we cannot use the central difference at end points. Can we derive $O(h^2)$ method at end points without using the central difference? Consider

$$\begin{aligned} f(x) &= f(x) \\ f(x+h) &= f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f''(\gamma_1), \\ f(x+2h) &= f(x) + 2hf'(x) + 2h^2 f''(x) + \frac{4h^3}{3}f''(\gamma_2). \end{aligned}$$

By taking a linear combination of these terms, we want to get an approximation to $f'(x)$. Assume

$$f'(x) = af(x) + bf(x+h) + cf(x+2h) + O(h^2)$$

and use the Taylor expansion to find a, b, c. From

$$\begin{aligned} f'(x) &= af(x) + b(f(x) + hf'(x) + \frac{h^2}{2}f''(x)) \\ &\quad + c(f(x) + 2hf'(x) + 2h^2 f''(x)) + O(h^3) \end{aligned}$$

we obtain

$$a+b+c=0,\ b+2c=\frac{1}{h},\ \frac{b}{2}+2c=0.$$

The solution is

$$a=-\frac{3}{2h},\ b=\frac{2}{h},\ c=-\frac{1}{2h}.$$

So

$$f'(x) = \frac{-3f(x)+4f(x+h)-f(x+2h)}{2h}+O(h^2).$$

4.2.2 One step methods - Euler methods

Let $x^{(n)}=x(t_n)$. Since

$$\frac{x_{n+1}-x_n}{h_n}\approx x'(t)=f(t_n,x^{(n)})$$

we approximate x by the following formula

$$x^{(n+1)}\approx x^{(n)}+h_nf(t_n,x^{(n)}),\quad h_n=t_{n+1}-t_n.$$

So we define a sequence

$$x_{n+1}=x_n+h_nf(t_n,x_n),\quad x_0=x(t_0).$$

This is called **forward Euler method(explicit)**. If we replace $f(t_n,x^{(n)})$ by $f(t_{n+1},x^{(n+1)})$ we obtain

$$x^{(n+1)}\approx x^{(n)}+h_nf(t_{n+1},x^{(n+1)}).$$

This is called the **backward Euler method(implicit)**. This is more stable but need to iterate to find the solution. (Picard) These can be also derived from the integral form:

$$x^{(n+1)} = x^{(n)}+\int_{t_n}^{t_{n+1}}f(t,x(t))\,dt.$$

Errors

Several kinds of errors.

(1) Local truncation error. Depends on the scheme – $O(h^{n+1})$ for n-th order methods.

(2) Local round off error. Depends on the machine – $\sim 10^{-14}$ for each arithmetic for double precision.

(3) Global truncation error. Depends on the scheme – $O(h^n)$ since $n = (T - t_0)/h$.

(4) Global round off error. – $\sim 10^{-14}n$ for each arithmetic for double precision.

(5) Total error — Sum of the above.

Analysis of Euler's Method

We return to the one step methods. A general one step method has the form

$$y_{n+1} = y_n + h\Phi(x_n, y_n, h). \tag{4.4}$$

Define the **local truncation error(LTE)** by

$$LTE_{n+1} = y(x_{n+1}) - [y(x_n) + h\Phi(x_n, y(x_n), h)] \tag{4.5}$$

The **global error(GE)** is defined as

$$GE_{n+1} = y(x_{n+1}) - y_{n+1}. \tag{4.6}$$

Definition 4.2.1. For numerical methods such as (4.4)

(1) we say the scheme is **consistent** if the local truncation error approaches 0 as $h \to 0$.

(2) we say a method is **stable** if for all sufficiently small h and ϵ, there is a $K > 0$ such that the numerical solution of $y' = f(x, y),\ y(x_0) = y_0 + \epsilon$ differs from the exact solution by at most ϵK.

(3) we say a method is **convergent** if the global error approaches 0 as h tends to zero.

For Euler's methods, we have

$$\begin{aligned} LTE_{n+1} &= y(x_{n+1}) - [y(x_n) + hf(x_n, y(x_n))] \\ &= y_n + hy'(x_n) + \frac{h^2}{2}y''(\xi_n) - y(x_n) - hf(x_n, y(x_n)) \\ &= y_n + hf(x_n, y(x_n)) + \frac{h^2}{2}y''(\xi_n) - y(x_n) - hf(x_n, y(x_n)) \\ &= \frac{h^2}{2}y''(\xi_n). \end{aligned}$$

What about the global error?

$$\begin{aligned} y(x_{n+1}) &= y(x_n) + hf(x_n, y(x_n)) + LTE_{n+1} \\ y(x_{n+1}) - y_{n+1} &= y(x_n) + hf(x_n, y(x_n)) + LTE_{n+1} - y_{n+1} \\ &= y(x_n) + hf(x_n, y(x_n)) + LTE_{n+1} - [y_n + hf(x_n, y_n)] \\ &= y(x_n) - y_n + h[f(x_n, y(x_n)) - f(x_n, y_n)] + LTE_{n+1} \\ GE_{n+1} &= GE_n + h[f(x_n, y(x_n)) - f(x_n, y(x_n) - GE_n)] + LTE_{n+1}. \end{aligned}$$

Let us write $e_{n+1} = |GE_{n+1}|$. Then

$$e_{n+1} \leq e_n + h|f(x_n, y(x_n)) - f(x_n, y(x_n) - GE_n)| + M$$

for some positive M which bounds LTE_{n+1}. Let f be a Lipschitz constant for f w.r.t y. Then

$$|f(x_n, y(x_n)) - f(x_n, y(x_n) - GE_n)| \leq L|GE_n| = Le_n.$$

Hence

$$\begin{aligned} e_{n+1} &\leq e_n + hLe_n + M = (1 + hL)e_n + M \\ &\leq (1 + hL)((1 + hL)e_{n-1} + M) + M \\ &\leq (1 + hL)^2 e_{n-1} + M(1 + hL) + M \\ &\leq (1 + hL)^{n+1} e_0 + M(1 + hL)^n + \cdots + M(1 + hL) + M \\ &\leq M\frac{(1 + hL)^{n+1} - 1}{hL} (\because e_0 = 0). \end{aligned}$$

Assume $(n+1)h \leq T$. Since $(1+hL) \leq e^{hL}$, we have

$$\frac{(1+hL)^{n+1}-1}{hL} \leq \frac{e^{h(n+1)L}-1}{hL} \leq \frac{e^{TL}-1}{(n+1)hL}(n+1).$$

$$e_{n+1} \leq M\frac{e^{TL}-1}{TL}(n+1) \leq Ch^2(n+1) \leq CTh.$$

Theorem 4.2.2. *A stable scheme is convergent iff it is consistent.*

Above argument shows that a consistent scheme is convergent. Conversely, a convergent scheme is clearly consistent because $LTE \leq GE$.

Method of undetermined coefficients

Consider the Taylor series

$$x(t+h) = x(t) + hx'(x) + \frac{h^2}{2!}x''(t) + \frac{h^3}{3!}x^{(3)}(t) + \cdots \tag{4.7}$$

By taking derivatives, we see that

$$\begin{aligned} x'(t) &= f \\ x''(t) &= f_t + f_x x' = f_t + f_x f \\ x'''(t) &= f_{tt} + f_{tx}f + (f_t + f_x f)f_x + f(f_{xt} + f_{xx}f), \end{aligned}$$

where f, f_t, f_x etc are evaluated at (t,x). The first three terms in the Taylor series can be written as

$$x(t+h) = x(t) + hf + \frac{1}{2}h^2(f_t + ff_x) + O(h^3) \tag{4.8}$$

$$= x(t) + \frac{1}{2}hf + \frac{1}{2}h[f + hf_t + hff_x] + O(h^3). \tag{4.9}$$

One could define a numerical scheme using this formula. However, we need to provide f_t and f_x. We will try a more efficient way of evaluating the value $x(t+h)$. Using

$$f(t+h, x+hf) = f + hf_t + hff_x + O(h^2), \tag{4.10}$$

we see

$$x(t+h) = x + \frac{1}{2}hf + \frac{1}{2}hf(t+h, x+hf) + O(h^3).$$

Hence we define

$$x_{n+1} = x_n + \frac{1}{2}(F_1 + F_2), \text{ sim. to trapezoid rule} \tag{4.11}$$

where

$$F_1 = hf(t_n, x_n), \quad F_2 = hf(t_n + h, x_n + F_1).$$

This is called a **second order Runge -Kutta methods**(truncation error is $O(h^3)$) or **Heun's Method**. Let us count the cost: For (4.9) we see

3 function evaluations, 4 multi and 3 additions,

while for (4.11) we have

2 function evaluations, 2 multi and 4 additions.

Usually function evaluations require several arithmetic operations.

One can obtain more general second order Runge -Kutta method by setting

$$x(t+h) = x(t) + w_1 hf + w_2 hf(t + \alpha h, x + \beta hf) + O(h^3), \tag{4.12}$$

where w_1, w_2, α, β are parameters to be determined. With the aid of Taylor expansion (4.10), it can be written as

$$x(t+h) = x(t) + w_1 hf + w_2 h[f + \alpha h f_t + \beta h f f_x] + O(h^3).$$

Comparing this with (4.9) we obtain

$$\begin{cases} w_1 + w_2 &= 1 \\ w_2\alpha &= \frac{1}{2} \\ w_2\beta &= \frac{1}{2}. \end{cases}$$

One solution is $w_1 = w_2 = \frac{1}{2}, \alpha = \beta = 1$. Hence we get Heun's method from (4.12). Other solution may exists, such as $w_1 = 0, w_2 = 1, \alpha = \beta = \frac{1}{2}$. The result is called **modified Euler method**:

$$x_{n+1} = x_n + F_2,$$

where

$$\begin{cases} F_1 &= hf(t_n, x_n) \\ F_2 &= hf(t_n + \frac{1}{2}h, x_n + \frac{1}{2}F_1). \text{ sim. to mid point rule} \end{cases}$$

Predictor-corrector methods

We start from the integral form:

$$x^{(n+1)} = x^{(n)} + \int_{t_n}^{t_{n+1}} f(t, x(t))\, dt.$$

If we use trapezoidal rule to approximate the integral, we get

$$x_{n+1} \doteq x_n + \frac{h}{2}[f(t_n, x_n) + f(t_{n+1}, x_{n+1})].$$

The result is a nonlinear equation in x_{n+1}. One way of solving it is to replace(Predict) the term x_{n+1} on the right by some approximation.

If we use Euler's method as a predictor,(i.e, replace x_{n+1} by $x_n + hf(t_n, x_n)$) we obtain the Heun's method:

$$x_{n+1}^{(P)} = x_n + hf(t_n, x_n) \tag{4.13}$$

$$x_{n+1} = x_n + \frac{h}{2}[f(t_n, x_n) + f(t_{n+1}, x_{n+1}^{(P)})]. \tag{4.14}$$

This is another interpretation of the Heun's method. We can go one step further: If we use the result of Heun's method as a corrector then the **corrected Heun's method** is obtained:

$$\begin{aligned} x_{n+1}^{(P)} &= x_n + hf(t_n, x_n) \\ x_{n+1}^{(C)} &= x_n + \frac{h}{2}[f(t_n, x_n) + f(t_{n+1}, x_{n+1}^{(P)})] \\ x_{n+1} &= x_n + \frac{h}{2}[f(t_n, x_n) + f(t_{n+1}, x_{n+1}^{(C)})]. \end{aligned}$$

Fourth order Runge-Kutta method

Higher order Runge-Kutta method can be obtained by similar idea. The following is the fourth order Runge-Kutta method. Composite finite elements for elliptic boundary value problems with discontinuous coefficients. Computing 77, 29

$$x_{n+1} = x_n + \frac{1}{6}(F_1 + 2F_2 + 2F_3 + F_4),$$

where

$$\begin{cases} F_1 &= hf(t, x_n) \\ F_2 &= hf(t + \frac{1}{2}h, x_n + \frac{1}{2}F_1) \\ F_3 &= hf(t + \frac{1}{2}h, x_n + \frac{1}{2}F_2) \\ F_4 &= hf(t + h, x_n + F_3). \end{cases}$$

4.3 Multistep methods

So far, the approximation at $x(t_{n+1})$ depends on the value of f only at $x_n = x(t_n)$. The known values $x_{n-1}, x_{n-2}, \cdots$ are *not* used. More efficient methods can be derived if one utilizes these values.

Consider again

$$\begin{cases} x'(t) &= f(t, x(t)) \\ x(t_0) &= x_0 \end{cases}.$$

We set by integrating

$$x_{n+1} = x_n + \int_{t_n}^{t_{n+1}} f(t, x(t))dt.$$

The integral on the right side can be approximated by a numerical quadrature using several points.

Adams Bashforth formula-Explicit methods

We assume $h_n = h$ for all n. Suppose the formula is of the following form:

$$x_{n+1} = x_n + a_0 f_n + a_1 f_{n-1} + \cdots,$$

where $f_n = f(t_n, x_n)$ and

$$\int_{t_n}^{t_{n+1}} f(t, x(t))dt \approx \sum_i a_i f_{n-i}.$$

Such a formula is called Adams Bashforth Formula. The Adams Bashforth

Formula of order 2,3,4,5(resp.) are as follows: HW. 4.29

$$x_{n+1} = x_n + \frac{h}{2}[3f_n - f_{n-1}], \quad lte = O(h^3) \tag{4.15}$$

$$x_{n+1} = x_n + \frac{h}{12}[23f_n - 16f_{n-1} + 5f_{n-2}], \quad lte = O(h^4) \tag{4.16}$$

$$x_{n+1} = x_n + \frac{h}{24}[55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3}], \quad lte = O(h^5) \tag{4.17}$$

$$x_{n+1} = x_n + \frac{h}{720}[1901f_n - 2774f_{n-1} + 2616f_{n-2} - 1274f_{n-3} + 251f_{n-4}]. \tag{4.18}$$

For example, (4.18) can be derived by considering the following quadrature.

$$\int_{t_n}^{t_{n+1}} f(t, x(t))dt \approx h[Af_n + Bf_{n-1} + Cf_{n-2} + Df_{n-3} + Ef_{n-4}]$$

so that the integral is exact for polynomials of degree four. Without loss of generality, we assume $t_0 = 0$ and $h = 1$ so that $t_{n-4} = -4, \cdots, t_n = 0, t_{n+1} = 1$. We take the following forms for polynomial basis(recall Newton form).

$$\begin{aligned} p_0(t) &= 1 \\ p_1(t) &= t \\ p_2(t) &= t(t+1) \\ p_3(t) &= t(t+1)(t+2) \\ p_4(t) &= t(t+1)(t+2)(t+3) \end{aligned}$$

We require the quadrature to be exact for these polynomials, i,e, for $n = 0, 1, \cdots, 4$,

$$\int_0^1 p_n(t)\, dt = Ap_n(0) + Bp_n(-1) + Cp_n(-2) + Dp_n(-3) + Ep_n(-4).$$

Solving these equations, we obtain coefficients in (4.18).

Remark 4.3.1. To use multistep methods, we need starting values since only x_0 is known. One choice is to use Runge-Kutta method in the first few steps to get $x_n, n = 1, 2, 3, \cdots$. Usually, formula of the same order are used.

Adams-Moulton method-predictor-corrector

A more precise formula can be derived if **backward scheme** is used, i.e, f_{n+1} is used in integrating the integral, we obtain

$$x_{n+1} = x_n + af_{n+1} + bf_n + cf_{n-1} + \cdots$$

which can also be obtained by the method described above. But this formula cannot be used as is since right hand side contains an *unknown*. Instead, if a reasonable approximation to f_{n+1} is known, we can derive an efficient method. Here we introduce so called **predictor-corrector** method. First, use Adams -Bashforth formula to find a tentative value $x_{n+1}^{(P)}$, then use Adams-Moulton Formula to compute the corrected value of x_{n+1} by $f(t_{n+1}, x_{n+1}^{(P)})$.

First, Adams-Moulton Formula of order 2,3,4, 5 are as follows:

$$x_{n+1} = x_n + \frac{h}{2}[f_{n+1} + f_n], \quad lte = O(h^3) \tag{4.19}$$

$$x_{n+1} = x_n + \frac{h}{12}[5f_{n+1} + 8f_n - f_{n-1}], \quad lte = O(h^4) \tag{4.20}$$

$$x_{n+1} = x_n + \frac{h}{24}[9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2}], \quad lte = O(h^5) \tag{4.21}$$

$$x_{n+1} = x_n + \frac{h}{720}[251f_{n+1} + 646f_n - 264f_{n-1} + 106f_{n-2} - 19f_{n-3}]. \tag{4.22}$$

Combining these with Adams-Bashforth formula (4.16),(4.17),(4.18), we can derive Adams-Moulton predictor- corrector methods: For example Adams-Moulton method of order 4 is

$$\begin{cases} x_{n+1}^{(P)} &= x_n + \frac{h}{24}[55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3}] \\ x_{n+1} &= x_n + \frac{h}{24}[9f_{n+1}(t_{n+1}, x_{n+1}^{(P)}) + 19f_n - 5f_{n-1} + f_{n-2}]. \end{cases}$$

Another method to approximate x_{n+1} is to use fixed point iteration:

$$z_{k+1} = \phi(z_k).$$

For example, to find x_{n+1} in (4.22), we let $\phi(z) = \frac{251}{720}hf(t_{n+1}, z) + F(x_n, x_{n-1}, \cdots)$, etc. F denotes the rest of terms. It is guaranteed to converge to the fixed point if $|\phi'(z)| < 1$ which can be assured if we take h sufficiently small.

There are other methods: The 3-th order Milne-Thompson Formula is as

follows:

$$x_{n+1} = x_{n-3} + h\left[\frac{8}{3}f_n - \frac{4}{3}f_{n-1} + \frac{8}{3}f_{n-2}\right]. \tag{4.23}$$