# Overdetermined systems

We examined the case of systems $Ax = b$ where $A \in \mathbb{R}^{m\times n}$ and $\underline{m > n}$. In general, a true solution does not exist. We define however the least squares solution as the vector $\underline{x}$ that minimizes

$$\underline{x} = \arg\min \|\underline{b} - A\underline{x}\|_2^2$$

$\underline{x}$ is given by the solution to the system of normal equations

$$A^T A \underline{x} = A^T \underline{b} \qquad (1)$$

System (1) is square ($n\times n$) and invertible (if $A$ has linearly independent columns). However, the condition number of $A^TA$ could be very poor... for example, if $A$ was square, we would have $\text{cond}(A^TA) = [\text{cond}(A)]^2$.

An alternative method that does not suffer from this problematic conditioning is the $QR$ factorization.

**Def** An $n\times n$ matrix $Q$ is called orthogonal iff

$$Q^T Q = Q Q^T = I.$$

Thm Let $A \in \mathbb{R}^{m \times n}$ $(m > n)$ have linearly independent columns. Then a decomposition

$$A = QR$$

exists, such that $Q \in \mathbb{R}^{m \times m}$ is orthogonal and $R \in \mathbb{R}^{m \times n}$ is upper triangular (i.e. $R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & \cdots & r_{2n} \\ & 0 & \ddots & \\ 0 & 0 & \cdots & r_{nn} \\ \vdots & \vdots & & 0 \\ 0 & 0 & \cdots & 0 \end{bmatrix}$)

Additionally, given that $A$ has linearly independent columns, we guarantee that $r_{ii} \neq 0$.

In Matlab, the QR decomposition is obtained via the qr function, i.e

```
[Q, R] = qr(A);
```

Now, let us write

$$Q = \left[ \hat{Q} \;\middle|\; Q^* \right]$$

where $\hat{Q} \in \mathbb{R}^{m \times m}$ contains the first $m$ columns of $Q$

and $Q^* \in \mathbb{R}^{(m-n) \times m}$ contains the last $(m-n)$ columns

Respectively, we write:

$$R = \begin{bmatrix} \hat{R} \\ \text{-----} \\ 0_{(m-n)\times n} \end{bmatrix}$$

where $\hat{R} \in \mathbb{R}^{n\times n}$ (and upper triangular) contains the first $n$ rows of $R$.
$\hat{R}$ is also nonsingular (for lin. ind. columns of

We can verify the following:

→ $\hat{Q}^T\hat{Q} = I_n$ (although $\hat{Q}\hat{Q}^T \neq I_m$ !)

Proof: $[\hat{Q}^T\hat{Q}]_{ij} = \sum_{k=1}^{m} [\hat{Q}^T]_{ik} [\hat{Q}]_{kj}$
$1 \le i,j \le n$

$$= \sum_{k=1}^{m} [\hat{Q}]_{ki} [\hat{Q}]_{kj} = \sum_{k=1}^{m} [Q]_{ki} [Q]_{kj}$$

$$= [Q^TQ]_{ij} = [I_m]_{ij}$$

→ $\underset{(m\times n)}{A} = \underset{(m\times m)(m\times n)}{QR} = \underset{(m\times n)}{\hat{Q}} \cdot \underset{(n\times n)}{\hat{R}}$.

Proof: Similar.

The factorization $A = \hat{Q}\cdot\hat{R}$ is the so-called economy-size QR factorization, and computed in Matlab as:

$$[\hat{Q}, \hat{R}] = qr(A, 0);$$

Once we have $\hat{Q}$ & $\hat{R}$ computed, we observe that the normal equations can be written as:

$$A^T A \underline{x} = A^T b$$

using $A = \hat{Q}\hat{R}$ $\Longrightarrow$

$$\hat{R}^T \underbrace{\hat{Q}^T \hat{Q}}_{= I_n} \hat{R} \underline{x} = \hat{R}^T \hat{Q}^T b$$

$$\hat{R}^T \hat{R} \underline{x} = \hat{R}^T \hat{Q}^T \underline{b}$$

$\hat{R}$ is invertible $\Longrightarrow$

$$(\hat{R}^{-T})(\hat{R}^T \hat{R} x) = (\hat{R}^{-T})(\hat{R}^T \hat{Q}^T \underline{b})$$

$$\Longrightarrow \boxed{\hat{R}\underline{x} = \hat{Q}^T \underline{b}} \quad (*)$$

Least squares solution using QR decomposition

<u>Matlab</u>:

```
[Q̂, R̂] = qr(A,0)
z = Q̂' * b;
x = R̂ \ z;
```

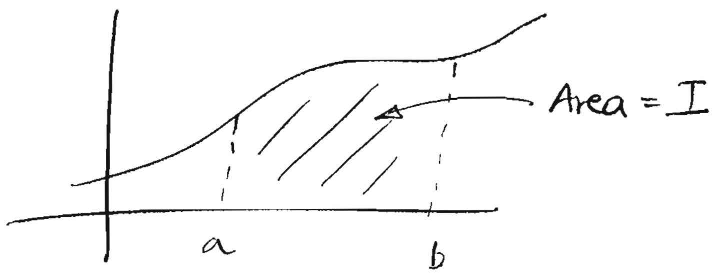
<u>Benefit</u>: We can show that $\text{cond}(A^T A) = [\text{cond}(\hat{R})]^2$, thus, equation (*) is <u>much better</u> conditioned than the normal equations system.

## Numerical integration

We seek an algorithm to approximate the definite integral :

$$I = \int_a^b f(x)\,dx$$

(or, the area below the graph of $y = f(x)$)




Of course, in the fortuitous case where we know a function $F(x)$ (the anti-derivative of $f$), s.t. $F'(x) = f(x)$,

we can write :

$$\int_a^b f(x)\,dx = F(b) - F(a)$$

e.g $[\arctan(x)]' = \dfrac{1}{1+x^2}$ , thus

$$\int_a^b \frac{dx}{1+x^2} = \arctan(b) - \arctan(a).$$

However, this is not a practical algorithm, since:

→ The anti-derivative is not generally known.

→ Often, the anti-derivative may be significantly more expensive to evaluate than $f(x)$ itself, (e.g. compare $f(x) = \frac{1}{1+x^2}$ (easy) with $F(x) = \arctan(x)$ (expensive)).

General methodology

→ Subdivide the interval of integration using the $n+1$ points $\{x_i\}_{i=0}^{n}$, with

$$a = x_0 < x_1 < x_2 < \dots < x_{n-1} < x_n = b$$

→ In each interval $[x_i, x_{i+1}]$ approximate $f(x)$ with some simpler function, say a polynomial $p^{(i)}(x)$, which is easy to integrate. Approximate

$$I_i = \int_{x_i}^{x_{i+1}} f(x)\,dx \approx \int_{x_i}^{x_{i+1}} p^{(i)}(x)\,dx.$$

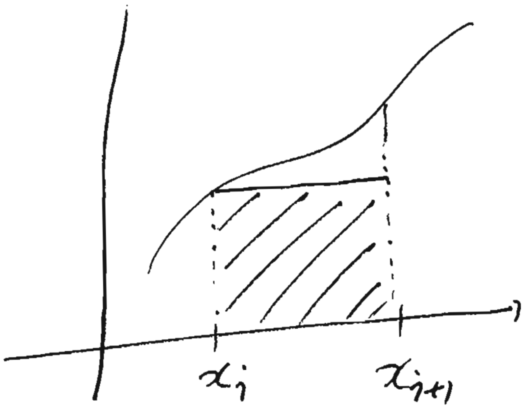
→ Compute the integral $I = \int_a^b f(x)\,dx$

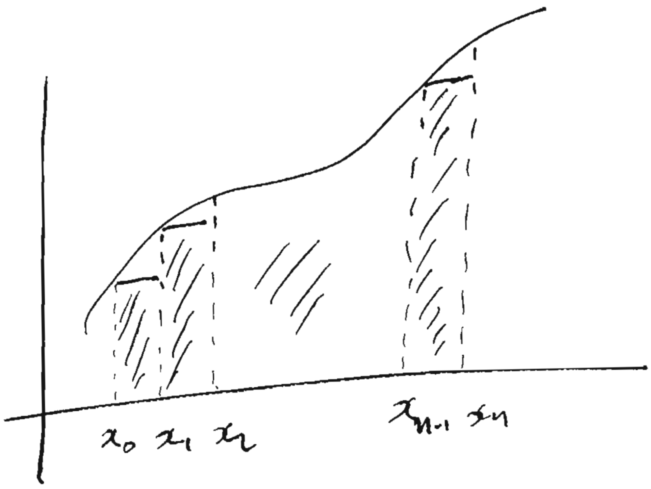
as $$I = \sum_{i=0}^{n-1} I_i \approx \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} p^{(i)}(x)\,dx.$$

Example: The rectangle rule

At each interval $[x_i, x_{i+1}]$ use the approximation

$p^{(i)}(x) = f(x_i)$ (the left endpoint!)




Thus we approximate:

$$I_i = \int_{x_i}^{x_{i+1}} f(x)\,dx \approx \int_{x_i}^{x_{i+1}} f(x_i)\,dx = f(x_i)\,(x_{i+1} - x_i)$$

(We often present this rule on a single interval $[a,b]$, as

$$\int_a^b f(x)\,dx \approx f(a)\cdot(b-a)$$

)

In the case where $x_{i+1} - x_i = h = \text{const}$, we can write

$$I = \int_a^b f(x)dx = \sum_{i=0}^{n-1} I_i \cong \sum_{i=0}^{n-1} f(x_i) \cdot h \Rightarrow$$

$$\boxed{I \cong \frac{b-a}{n} \sum_{i=0}^{n-1} f(x_i)}$$

As in the case of interpolation, we can assess the error incurred by this approximation. There are 2 errors we actually focus on:

→ The local error $\left| \int_{x_i}^{x_{i+1}} (f(x) - p^{(i)}(x))\, dx \right|$ at each subinterval

→ The global error for the entire integral $\int_a^b f(x)\, dx$.