Overview

- Gram-Schmidt algorithm, QR factorization
- Projection onto subspaces
- Orthogonal projectors
- Best approximation in the 2-norm
- Linear regression
- Polynomial approximation
- Polynomial interpolation

Definition. *The Dirac delta symbol* δ_{ij} *is defined as*

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

Definition. *A set of vectors* $\{\boldsymbol{u}_1, \dots, \boldsymbol{u}_n\}$ *is said to be orthonormal if*

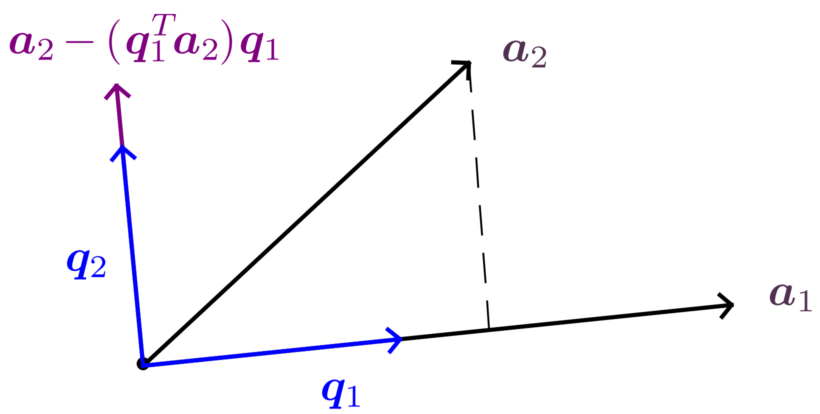
$$\boldsymbol{u}_i^T \boldsymbol{u}_j = \delta_{ij}$$

- The column vectors of the identity matrix are orthonormal

$$\boldsymbol{I} = \begin{pmatrix} \boldsymbol{e}_1 & \dots & \boldsymbol{e}_m \end{pmatrix}$$

$$\boldsymbol{e}_i^T \boldsymbol{e}_j = \delta_{ij}$$

- An arbitrary vector set can be transformed into an orthonormal set by the Gram-Schmidt algorithm
- Idea:
  - Start with an arbitrary direction $\boldsymbol{a}_1$
  - Divide by its norm to obtain a unit-norm vector $\boldsymbol{q}_1 = \boldsymbol{a}_1 / \|\boldsymbol{a}_1\|$
  - Choose another direction $\boldsymbol{a}_2$
  - Subtract off its component along previous direction(s) $\boldsymbol{a}_2 - (\boldsymbol{q}_1^T \boldsymbol{a}_2)\boldsymbol{q}_1$
  - Divide by norm $\boldsymbol{q}_2 = (\boldsymbol{a}_2 - (\boldsymbol{q}_1^T \boldsymbol{a}_2)\boldsymbol{q}_1) / \|\boldsymbol{a}_2 - (\boldsymbol{q}_1^T \boldsymbol{a}_2)\boldsymbol{q}_1\|$
  - Repeat the above



- Consider $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with linearly independent columns. By linear combinations of the columns of $\boldsymbol{A}$ a set of orthonormal vectors $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n$ will be obtained. This can be expressed as a matrix product

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \ldots & \boldsymbol{a}_n \end{pmatrix} = \begin{pmatrix} \boldsymbol{q}_1 & \boldsymbol{q}_2 & \ldots & \boldsymbol{q}_n \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & \ldots & r_{1n} \\ 0 & r_{22} & r_{23} & \ldots & r_{2n} \\ 0 & 0 & r_{33} & \ldots & r_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \ldots & r_{mn} \end{pmatrix} = \boldsymbol{Q}\boldsymbol{R}$$

  with $\boldsymbol{Q} \in \mathbb{R}^{m \times n}$, $\boldsymbol{R} \in \mathbb{R}^{n \times n}$. The matrix $\boldsymbol{R}$ is upper-triangular (also referred to as right-triangular) since to find vector $\boldsymbol{q}_1$ only vector $\boldsymbol{a}_1$ is used, to find vector $\boldsymbol{q}_2$ only vectors $\boldsymbol{a}_1$, $\boldsymbol{a}_2$ are used

- The above is equivalent to the system

$$\begin{cases} \boldsymbol{a}_1 = r_{11}\boldsymbol{q}_1 \\ \boldsymbol{a}_2 = r_{12}\boldsymbol{q}_1 + r_{22}\boldsymbol{q}_2 \\ \vdots \\ \boldsymbol{a}_n = r_{1n}\boldsymbol{q}_1 + r_{2n}\boldsymbol{q}_2 + \ldots + r_{nn}\boldsymbol{q}_n \end{cases}$$

- The system can be solved to find $\boldsymbol{R}, \boldsymbol{Q}$ by:

1. Imposing $\|\boldsymbol{q}_1\| = 1 \Rightarrow r_{11} = \|\boldsymbol{a}_1\|$, $\boldsymbol{q}_1 = a_1 / r_{11}$
2. Computing projections of $\boldsymbol{a}_2, \ldots, \boldsymbol{a}_n$ along $\boldsymbol{q}_1$

$$r_{12} = \boldsymbol{q}_1^T \boldsymbol{a}_2, \ldots, r_{1n} = \boldsymbol{q}_1^T \boldsymbol{a}_n$$

3. Subtracting components along $\boldsymbol{q}_1$ from $\boldsymbol{a}_2, \ldots, \boldsymbol{a}_n$

$$\begin{cases} \boldsymbol{a}_2 - r_{12}\boldsymbol{q}_1 = r_{22}\boldsymbol{q}_2 \\ \vdots \\ \boldsymbol{a}_n - r_{1n}\boldsymbol{q}_1 = r_{2n}\boldsymbol{q}_2 + \ldots + r_{nn}\boldsymbol{q}_n \end{cases}$$

4. The above steps reduced the size of the system by 1. Repeating the steps completes the solution. The overall process is known as the Gram-Schmidt algorithm

Algorithm (Gram-Schmidt)

Given n vectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n \in \mathbb{R}^m$
Initialize $\boldsymbol{q}_1 = \boldsymbol{a}_1, .., \boldsymbol{q}_n = \boldsymbol{a}_n$, $\boldsymbol{R} = \boldsymbol{I}_n \in \mathbb{R}^{n \times n}$
for $i = 1$ to n
  $r_{ii} = (\boldsymbol{q}_i^T \boldsymbol{q}_i)^{1/2}$; $\boldsymbol{q}_i = \boldsymbol{q}_i / r_{ii}$
  for $j = i + 1$ to n
    $r_{ij} = \boldsymbol{q}_i^T \boldsymbol{a}_j$; $\boldsymbol{q}_j = \boldsymbol{q}_j - r_{ij} \boldsymbol{q}_i$
  end
end
return $\boldsymbol{Q}, \boldsymbol{R}$

- For $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with linearly independent columns, the Gram-Schmidt algorithm furnishes a factorization

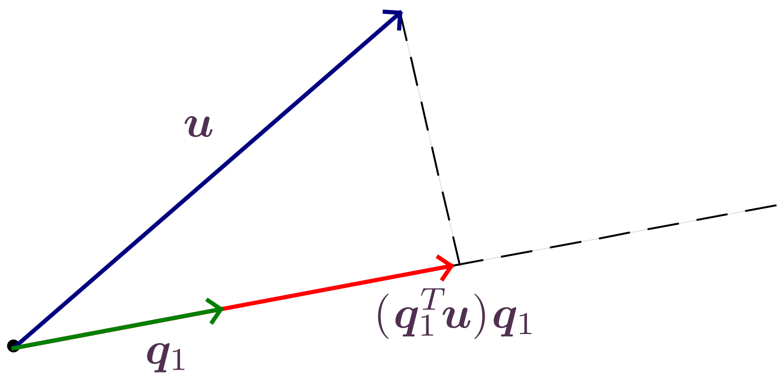
$$\boldsymbol{Q}\boldsymbol{R} = \boldsymbol{A}$$

with $\boldsymbol{Q} \in \mathbb{R}^{m \times n}$ with orthonormal columns and $\boldsymbol{R} \in \mathbb{R}^{n \times n}$ an upper triangular matrix.

- Since the column vectors within $\boldsymbol{Q}$ were obtained through linear combinations of the column vectors of $\boldsymbol{A}$ we have

$$C(\boldsymbol{A}) = C(\boldsymbol{Q})$$

- Consider a vector $\boldsymbol{u} \in \mathbb{R}^m$, and a unit-norm vector $\boldsymbol{q}_1 \in \mathbb{R}^m$

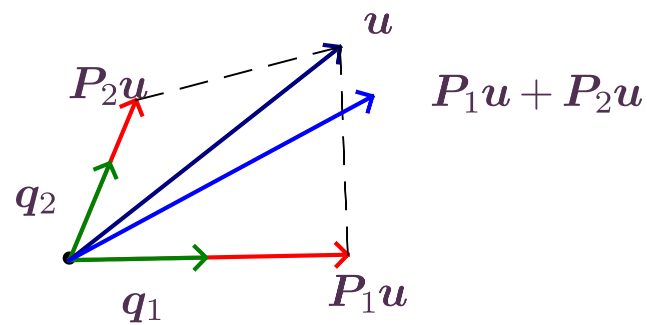


Definition. *The* orthogonal projection *of* $\boldsymbol{u} \in \mathbb{R}^m$ *along direction* $\boldsymbol{q}_1 \in \mathbb{R}^m$, $\|\boldsymbol{q}_1\| = 1$ *is the vector* $(\boldsymbol{q}_1^T\boldsymbol{u})\boldsymbol{q}_1$.

- Scalar-vector multiplication commutativity: $(\boldsymbol{q}_1^T\boldsymbol{u})\boldsymbol{q}_1 = \boldsymbol{q}_1(\boldsymbol{q}_1^T\boldsymbol{u})$
- Matrix multiplication associativity: $\boldsymbol{q}_1(\boldsymbol{q}_1^T\boldsymbol{u}) = (\boldsymbol{q}_1\boldsymbol{q}_1^T)\boldsymbol{u} = \boldsymbol{P}_1\boldsymbol{u}$, with $\boldsymbol{P}_1 \in \mathbb{R}^{m\times m}$

Definition. *The matrix* $\boldsymbol{P}_1 = \boldsymbol{q}_1\boldsymbol{q}_1^T \in \mathbb{R}^{m\times m}$ *is the* orthogonal projector *along direction* $\boldsymbol{q}_1 \in \mathbb{R}^m$, $\|\boldsymbol{q}_1\| = 1$.

- Consider n orthonormal vectors grouped into a matrix $\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{q}_1 & \dots & \boldsymbol{q}_n \end{pmatrix} \in \mathbb{R}^{m \times n}$



- The orthogonal projection of $\boldsymbol{u}$ onto the subspace spanned by $\boldsymbol{q}_1, \dots, \boldsymbol{q}_n$ is

$$\boldsymbol{P}\boldsymbol{u} = \boldsymbol{P}_1\boldsymbol{u} + \dots + \boldsymbol{P}_n\boldsymbol{u} = (\boldsymbol{q}_1\boldsymbol{q}_1^T)\boldsymbol{u} + \dots + (\boldsymbol{q}_n\boldsymbol{q}_n^T)\boldsymbol{u} \Rightarrow$$

$$\boldsymbol{P} = \boldsymbol{q}_1\boldsymbol{q}_1^T + \dots + \boldsymbol{q}_n\boldsymbol{q}_n^T = \begin{pmatrix} \boldsymbol{q}_1 & \dots & \boldsymbol{q}_n \end{pmatrix} \begin{pmatrix} \boldsymbol{q}_1^T \\ \vdots \\ \boldsymbol{q}_n^T \end{pmatrix} = \boldsymbol{Q}\boldsymbol{Q}^T$$

Definition. *The orthogonal projector onto $C(\boldsymbol{Q})$, $\boldsymbol{Q} \in \mathbb{R}^{m \times n}$ with orthonormal column vectors is $\boldsymbol{P} = \boldsymbol{Q}\boldsymbol{Q}^T$*

- Given $\boldsymbol{u} \in \mathbb{R}^m$ and $\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{q}_1 & \dots & \boldsymbol{q}_n \end{pmatrix} \in \mathbb{R}^{m \times n}$ with orthonormal columns

Definition. *The complementary orthogonal projector to $\boldsymbol{P} = \boldsymbol{Q}\boldsymbol{Q}^T$ is $\boldsymbol{I} - \boldsymbol{P}$, where $\boldsymbol{Q} \in \mathbb{R}^{m \times n}$ is a matrix with orthonormal columns.*

- The complementary orthogonal projector projects a vector onto the left null space, $N(\boldsymbol{Q}^T)$

- Consider the linear system $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$ with $\boldsymbol{A}\in\mathbb{R}^{m\times n}$, $\boldsymbol{x}\in\mathbb{R}^n$, $\boldsymbol{b}\in\mathbb{R}^m$. Orthogonal projectors and knowledge of the four fundamental matrix subspaces allows us to succintly express whether there exist no solutions, a single solution of an infinite number of solutions:
  - Consider the factorization $\boldsymbol{Q}\boldsymbol{R}=\boldsymbol{A}$, the orthogonal projector $\boldsymbol{P}=\boldsymbol{Q}\boldsymbol{Q}^T$, and the complementary orthogonal projector $\boldsymbol{I}-\boldsymbol{P}$
  - If $\|(\boldsymbol{I}-\boldsymbol{P})\boldsymbol{b}\|\neq 0$, then $\boldsymbol{b}$ has a component outside the column space of $\boldsymbol{A}$, and $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$ has no solution
  - If $\|(\boldsymbol{I}-\boldsymbol{P})\boldsymbol{b}\|=0$, then $\boldsymbol{b}\in C(\boldsymbol{Q})=C(\boldsymbol{A})$ and the system has at least one solution
  - If $N(\boldsymbol{A})=\{\boldsymbol{0}\}$ (null space only contains the zero vector, i.e., null space of dimension 0) the system has a unique solution
  - If $\dim N(\boldsymbol{A})=n-r>0$, then a vector $\boldsymbol{y}\in N(\boldsymbol{A})$ in the null space is written as

$$\boldsymbol{y}=c_1\boldsymbol{z}_1+\ldots+c_{n-r}\boldsymbol{z}_{n-r}$$

    and if $\boldsymbol{x}$ is a solution of $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$, so is $\boldsymbol{x}+\boldsymbol{y}$, since

$$\boldsymbol{A}(\boldsymbol{x}+\boldsymbol{y})=\boldsymbol{A}\boldsymbol{x}+c_1\boldsymbol{A}\boldsymbol{z}_1+\ldots+c_{n-r}\boldsymbol{A}\boldsymbol{z}_{n-r}=\boldsymbol{b}+\boldsymbol{0}+\ldots+\boldsymbol{0}=\boldsymbol{b}$$

    The linear system has an $(n-r)$-parameter family of solutions

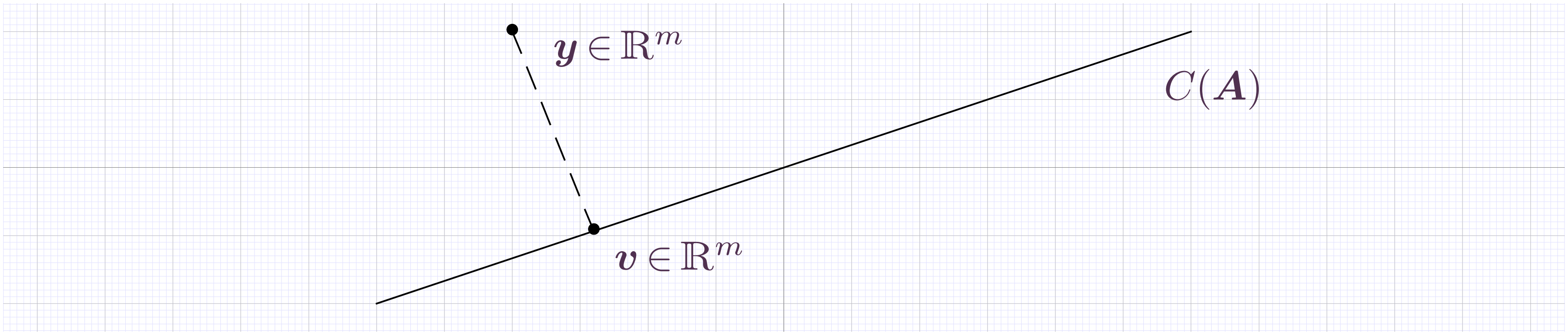


Figure 1. Least squares problem: find $v \in C(A)$, $A \in \mathbb{R}^{m\times n}$ closest to some given y in the 2-norm

- Mathematical statement: solve the minimization problem $\min_{c\in\mathbb{R}^n} \|y - Ac\|$
- Approach: project y onto the column space of A:
  1. Find an orthonormal basis for column space of A by QR factorization, $QR = A$
  2. State that v is the projection of y, $v = P_{C(A)}y = P_Q y = QQ^T y$
  3. State that v is within the column space of A, $v = Ac = QRc$
  4. Set equal the two expressions of v, $QQ^T y = QRc \Rightarrow Rc = Q^T y$
  5. Solve the triangular system to find c (in Julia, Matlab, Octave: `c=R\(Q'y)`)

- In many scientific fields the problem of determining the straight line $y(x) = c_0 + c_1 x$, that best approximate data $\mathcal{D} = \{(x_i, y_i), i = 1, \ldots, m\}$ arises. The problem is to find the coefficients c_0, c_1, and this is referred to as the linear regression problem.
- The calculus approach: Form sum of squared differences between $y(x_i)$ and y_i

$$S(c_0, c_1) = \sum_{i=1}^{m} (y(x_i) - y_i)^2 = \sum_{i=1}^{m} (c_0 + c_1 x_i - y_i)^2$$

and seek (c_0, c_1) that minimize $S(c_0, c_1)$ by solving the equations

$$\frac{\partial S}{\partial c_0} = 0 \Rightarrow 2\sum_{i=1}^{m} (c_0 + c_1 x_i - y_i) = 0 \Leftrightarrow m c_0 + \left(\sum_{i=1}^{m} x_i\right) c_1 = \sum_{i=1}^{m} y_i$$

$$\frac{\partial S}{\partial c_1} = 0 \Rightarrow 2\sum_{i=1}^{m} (c_0 + c_1 x_i - y_i) x_i = 0 \Leftrightarrow \left(\sum_{i=1}^{m} x_i\right) c_0 + \left(\sum_{i=1}^{m} x_i^2\right) c_1 = \sum_{i=1}^{m} x_i y_i$$

- Form a vector of errors with components $e_i = y(x_i) - x_i$. Recognize that $y(x_i)$ is a linear combination of 1 and x_i with coefficients a_0, a_1, or in vector form

$$\boldsymbol{e} = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_m \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} - \boldsymbol{y} = [\,\boldsymbol{1} \;\; \boldsymbol{x}\,]\boldsymbol{c} - \boldsymbol{y} = \boldsymbol{A}\boldsymbol{c} - \boldsymbol{y}$$

- The norm of the error vector $\|\boldsymbol{e}\|$ is smallest when $\boldsymbol{A}\boldsymbol{c}$ is as close as possible to $\boldsymbol{y}$. Since $\boldsymbol{A}\boldsymbol{c}$ is within the column space of $C(\boldsymbol{A})$, $\boldsymbol{A}\boldsymbol{c} \in C(\boldsymbol{A})$, the required condition is for $\boldsymbol{e}$ to be orthogonal to the column space, leading to the normal equations

$$\boldsymbol{e} \perp C(\boldsymbol{A}) \Rightarrow \boldsymbol{A}^T\boldsymbol{e} = \begin{bmatrix} \boldsymbol{1}^T \\ \boldsymbol{x}^T \end{bmatrix} \boldsymbol{e} = \begin{bmatrix} \boldsymbol{1}^T\boldsymbol{e} \\ \boldsymbol{x}^T\boldsymbol{e} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \boldsymbol{0}$$

$$\boldsymbol{A}^T\boldsymbol{e} = \boldsymbol{0} \Leftrightarrow \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{c} - \boldsymbol{y}) = 0 \Leftrightarrow (\boldsymbol{A}^T\boldsymbol{A})\boldsymbol{c} = \boldsymbol{A}^T\boldsymbol{y} \text{ (Normal equations)}$$

- If $\boldsymbol{Q}\boldsymbol{R} = \boldsymbol{A}$ is known, preferable to solve $\boldsymbol{Q}\boldsymbol{Q}^T\boldsymbol{y} = \boldsymbol{Q}\boldsymbol{R}\boldsymbol{c} \Rightarrow \boldsymbol{R}\boldsymbol{c} = \boldsymbol{Q}^T\boldsymbol{y}$.

1 Generate some data on a line and perturb it by some random quantities

```
∴ m=100; x=(0:m-1)./m; c0=2; c1=3; yex=c0.+c1*x; y=(yex.+rand(m,1).-0.5);
∴
```

2 Form the $\boldsymbol{Q}, \boldsymbol{R}$ matrices, $\boldsymbol{Q}\boldsymbol{R} = \boldsymbol{A}$, (`qr(A,0)`)

```
∴ A=ones(m,2); A[:,2]=x[:]; QR=qr(A); Q=QR.Q[:,1:2]; R=QR.R[1:2,1:2];
∴
```

3 Solve the system $\boldsymbol{R}\boldsymbol{x} = \boldsymbol{Q}^T\boldsymbol{y}$

```
∴ c = R\(transpose(Q)*y)
∴
```

4 Form the linear combination $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{x}$ closest to $\boldsymbol{b}$

```
∴ v=A*c;
∴
```

1 Generate some data on a line and perturb it by some random quantities

```
∴ m=100; x=(0:m-1)./m; c0=2; c1=3; yex=c0.+c1*x; y=(yex.+rand(m,1).-0.5);
∴
```

2 Form the $\boldsymbol{Q}, \boldsymbol{R}$ matrices, $\boldsymbol{QR} = \boldsymbol{A}$, (`qr(A,0)`)

```
∴ A=ones(m,2); A[:,2]=x[:]; QR=qr(A); Q=QR.Q[:,1:2]; R=QR.R[1:2,1:2];
∴
```

3 Solve the system $\boldsymbol{Rx} = \boldsymbol{Q}^T\boldsymbol{y}$

```
∴ c = R\(transpose(Q)*y)
```

$$\begin{bmatrix} 1.9812089298039193 \\ 2.9758677046339135 \end{bmatrix} \tag{1}$$

```
∴
```

4 Form the linear combination $\boldsymbol{v} = \boldsymbol{Ax}$ closest to $\boldsymbol{b}$

```
∴ v=A*c;
∴
```

- Plot the perturbed data (black dots), the result of the linear regression (green circles), as well as the line used to generate yex (red line)

```
∴ plot(x,y,".k",x,v,"og",x,yex,"r"); title("Linear␣regression"); xlabel("x");
  ylabel("y,v,yex");
∴ cd(homedir()*"/Desktop/courses/MATH347DS/images"); savefig("L07Fig02.eps");
∴
```

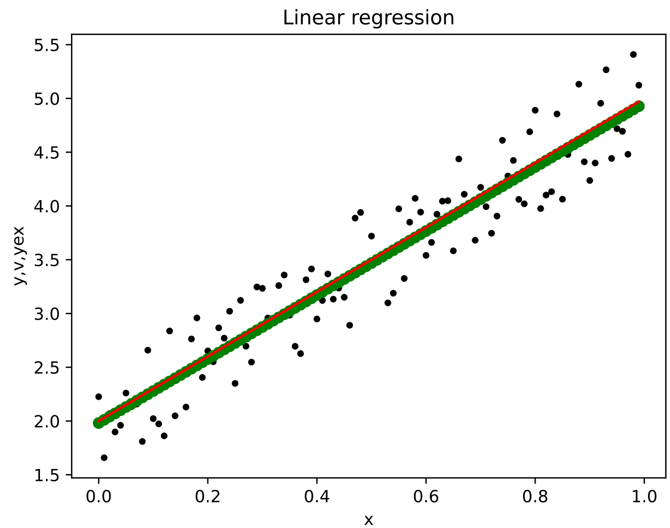


Figure 2. Linear reqression through least squares result

- The calculus approach becomes complex for higher-degree approximation

$$y(x) = c_0 + c_1 x + c_2 x^2 = \begin{bmatrix} 1 & x & x^2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = A(x)\boldsymbol{c}.$$

Note that $y(x)$ is nonlinear.

- The least squares approach retains its simplicity since but $y(c_0, c_1, c_2)$ is linear.

```
∴ m=100; x=(0:m-1)./m; c0=2; c1=3; c2=-5; yex=c0.+c1*x.+c2*x.^2;
∴ y=(yex.+rand(m,1).-0.5);
∴ A=ones(m,3); A[:,2]=x[:]; A[:,3]=x[:].^2; QR=qr(A); Q=QR.Q[:,1:3]; R=QR.R[1:3,1:3];
∴ c = R\(transpose(Q)*y)
∴ v=A*c;
∴
```

- The calculus approach becomes complex for higher-degree approximation

$$y(x) = c_0 + c_1 x + c_2 x^2 = \begin{bmatrix} 1 & x & x^2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = A(x)\mathbf{c}.$$

Note that $y(x)$ is nonlinear.

- The least squares approach retains its simplicity since but $y(c_0, c_1, c_2)$ is linear.

```
∴ m=100; x=(0:m-1)./m; c0=2; c1=3; c2=-5; yex=c0.+c1*x.+c2*x.^2;
∴ y=(yex.+rand(m,1).-0.5);
∴ A=ones(m,3); A[:,2]=x[:]; A[:,3]=x[:].^2; QR=qr(A); Q=QR.Q[:,1:3]; R=QR.R[1:3,1:3];
∴ c = R\(transpose(Q)*y)
```

$$\begin{bmatrix} 2.019352129655483 \\ 2.8933875491171213 \\ -4.773241628594068 \end{bmatrix} \tag{2}$$

```
∴ v=A*c;
∴
```

- Plot the perturbed data (black dots), the result of the linear regression (green circles), as well as the line used to generate yex (red line)

```
∴ plot(x,y,".k",x,v,"og",x,yex,"r"); title("Quadratic␣regression"); xlabel("x");
  ylabel("y,v,yex");
```

```
∴ cd(homedir()*"/Desktop/courses/MATH347DS/images"); savefig("L07Fig03.eps");
```

```
∴
```

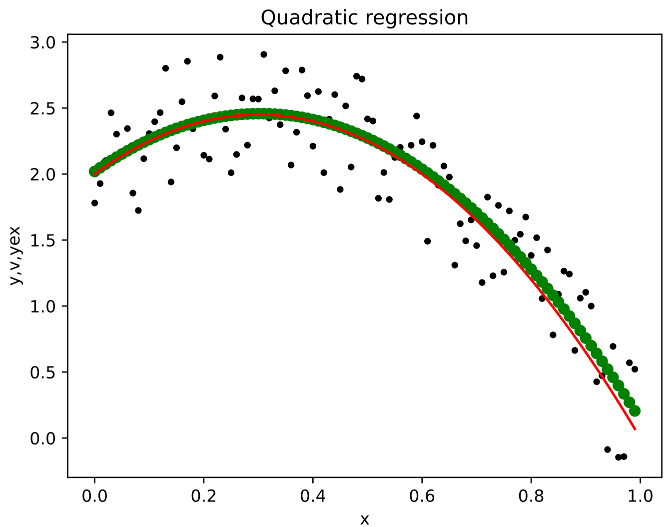


Figure 3. Linear reqression through least squares result

Definition. *The polynomial interpolant of data* $\mathcal{D} = \{(x_i, y_i), i = 1, \ldots, m\}$ *with* $x_i \neq x_j$ *if* $i \neq j$ *is a polynomial of degree* $m - 1$

$$p_{m-1}(x) = c_0 + c_1 x + \ldots + c_{m-1} x^{m-1}$$

that satisfies the conditions $p_{m-1}(x_i) = y_i$, $i = 1, \ldots, m$.

- We can apply the same approach. In this particular case, the error e can be made zero.

```
∴ m=3; x=(0:m-1)./m; c0=2; c1=3; c2=-5; yex=c0.+c1*x.+c2*x.^2;
∴ A=ones(m,3); A[:,2]=x[:]; A[:,3]=x[:].^2; QR=qr(A); Q=QR.Q[:,1:3]; R=QR.R[1:3,1:3];
∴ c = R\(transpose(Q)*yex)
```

Note that the coefficients used to generate the data are recovered exactly.

Definition. *The polynomial interpolant of data* $\mathcal{D} = \{(x_i, y_i), i = 1, \ldots, m\}$ *with* $x_i \neq x_j$ *if* $i \neq j$ *is a polynomial of degree* $m-1$

$$p_{m-1}(x) = c_0 + c_1 x + \ldots + c_{m-1} x^{m-1}$$

that satisfies the conditions $p_{m-1}(x_i) = y_i$, $i = 1, \ldots, m$.

- We can apply the same approach. In this particular case, the error e can be made zero.

```
∴ m=3; x=(0:m-1)./m; c0=2; c1=3; c2=-5; yex=c0.+c1*x.+c2*x.^2;
∴ A=ones(m,3); A[:,2]=x[:]; A[:,3]=x[:].^2; QR=qr(A); Q=QR.Q[:,1:3]; R=QR.R[1:3,1:3];
∴ c = R\(transpose(Q)*yex)
```

$$\begin{bmatrix} 2.0 \\ 2.9999999999999987 \\ -4.999999999999999 \end{bmatrix} \quad (3)$$

Note that the coefficients used to generate the data are recovered exactly.