Scientific Computing

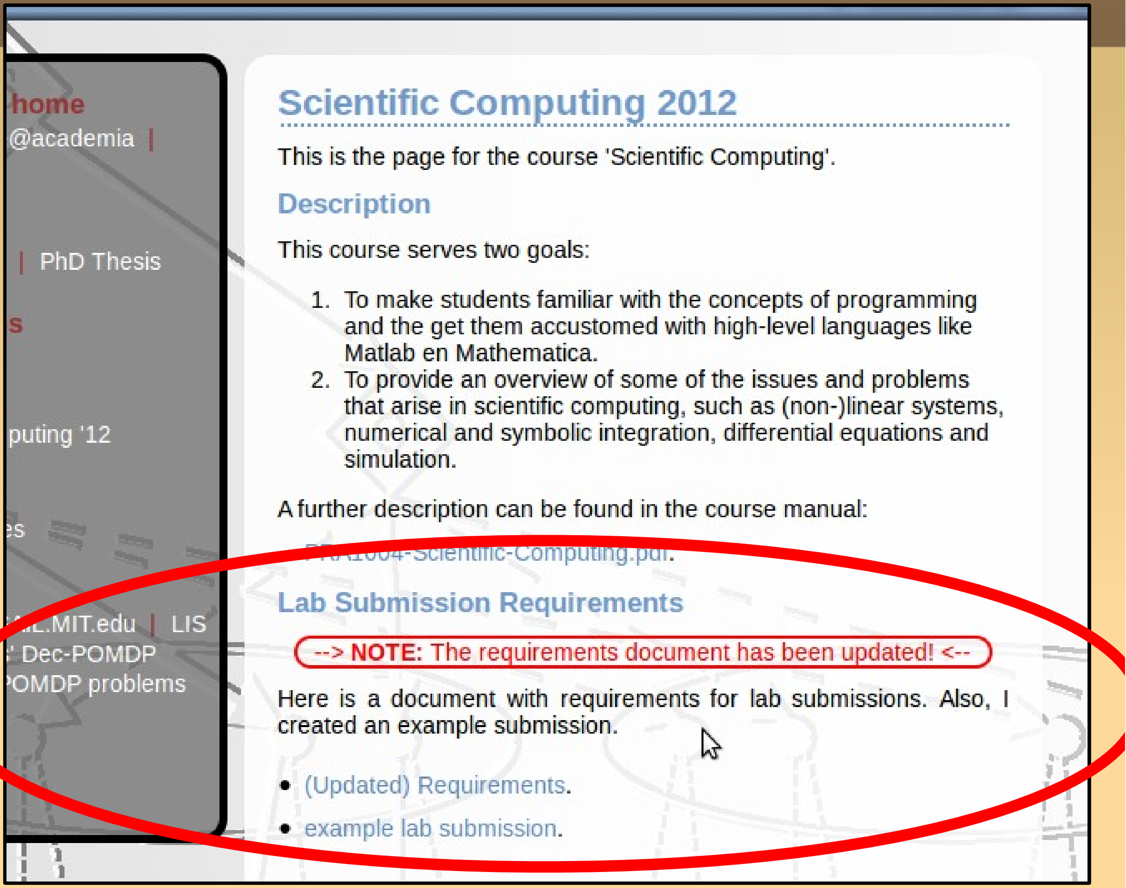
Maastricht Science Program

Week 5

Frans Oliehoek
<frans.oliehoek@maastrichtuniversity.nl>

Announcements

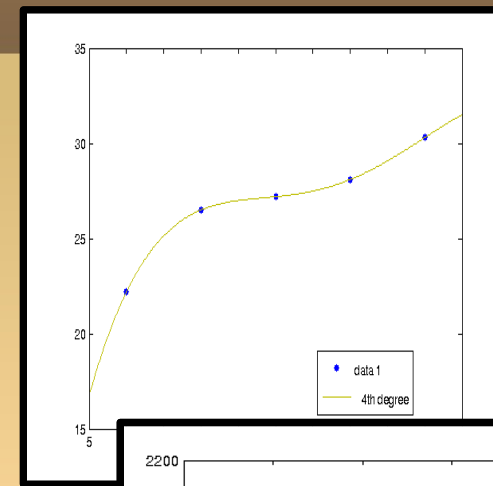
- I will be more strict!

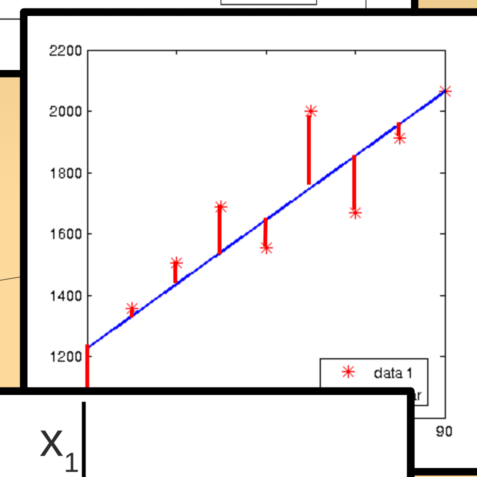


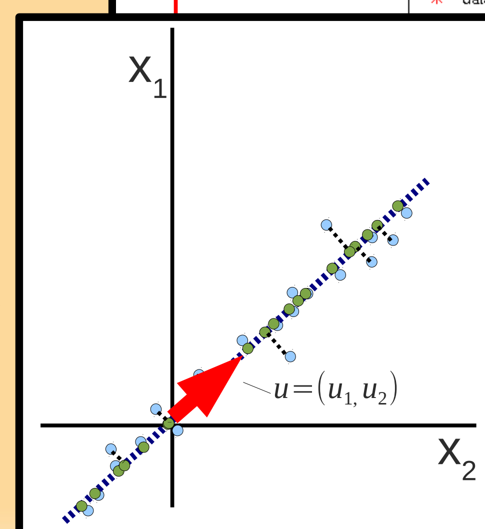
- Requirements updated...
- **YOU** are responsible that the submission satisfies the requirements!!!
  - I will not email you until the rest has their mark.

Recap Last Two Week

- Supervised Learning
  - find *f* that maps $\{x_1^{(j)},\ldots,x_D^{(j)}\} \rightarrow y^{(j)}$
  - Interpolation
    - *f* goes through the data points
  - linear regression
    - lossy fit, minimizes 'vertical' SSE
- Unsupervised Learning
  - We just have data points $\{x_1^{(j)},\ldots,x_D^{(j)}\}$
  - PCA
    - minimizes orthogonal projection

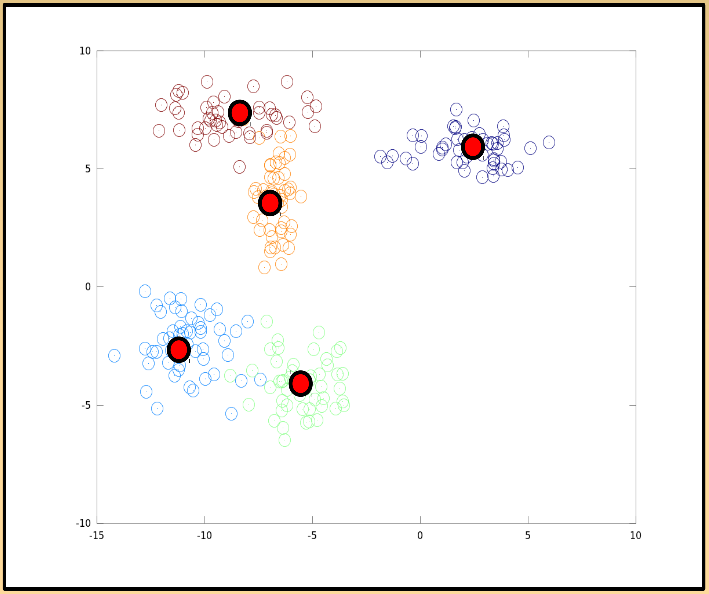






Recap: Clustering

- *Clustering* or *Cluster Analysis* has many applications
- Understanding
  - Astronomy, Biology, etc.
- Data (pre)processing
  - summarization of data set
  - compression



- Are there questions about k-means clustering?

This Lecture

- Last week: *unlabeled* data (also 'unsupervised learning')
  - data: just x
  - Clustering
  - Principle Components analysis (PCA) – what?
- This week
  - Principle Components analysis (PCA) – how?
  - Numerical differentiation and integration.

Part 1: Principal Component Analysis

- Recap
- How to do it?

PCA – Intuition

- How would you summarize this data using 1 dimension?

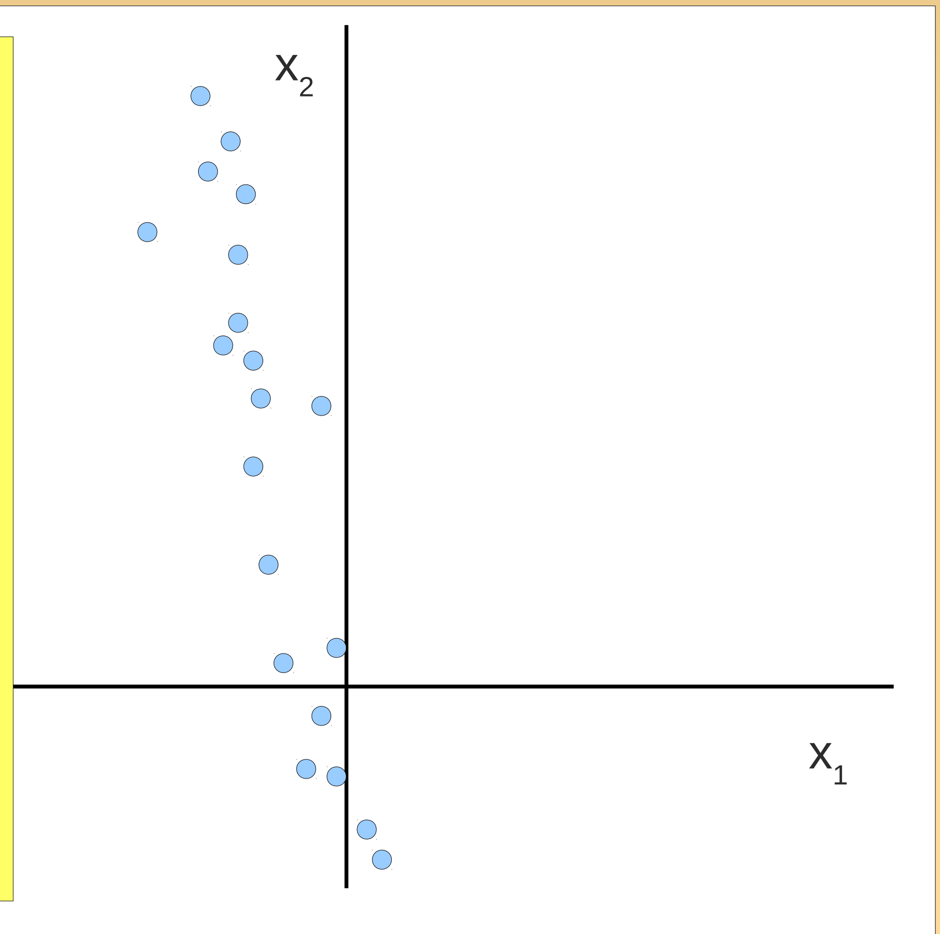
  (what variable contains the most information?)

Very important idea

The most information is contained by the variable with the largest spread.

- i.e., highest variance

(Information Theory)



PCA – Intuition

- How would you summarize this data using 1 dimension?

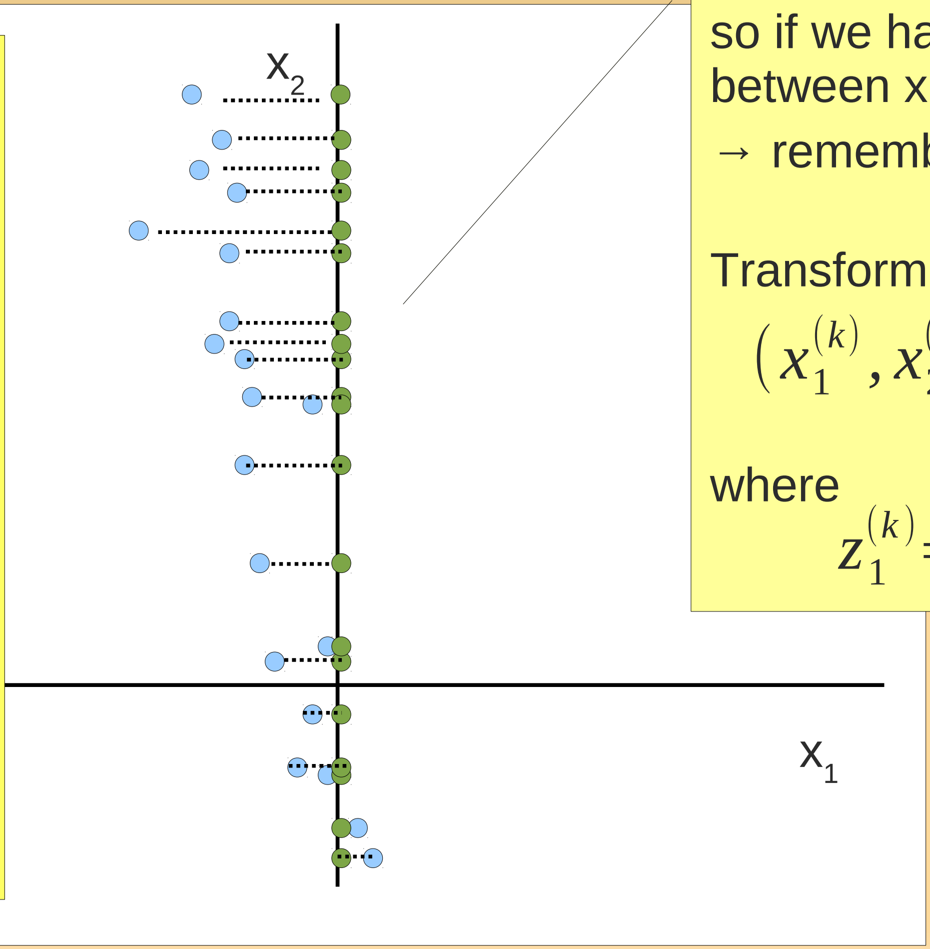
  (what variable contains the most information?)

Very important idea

The most information is contained by the variable with the largest spread.

- i.e., highest variance

(Information Theory)



so if we have to chose between x_1 and x_2
→ remember x_2

Transform of *k*-th point:

$$\left(x_1^{(k)}, x_2^{(k)}\right) \rightarrow \left(z_1^{(k)}\right)$$

where

$$z_1^{(k)} = x_2^{(k)}$$

PCA – Intuition

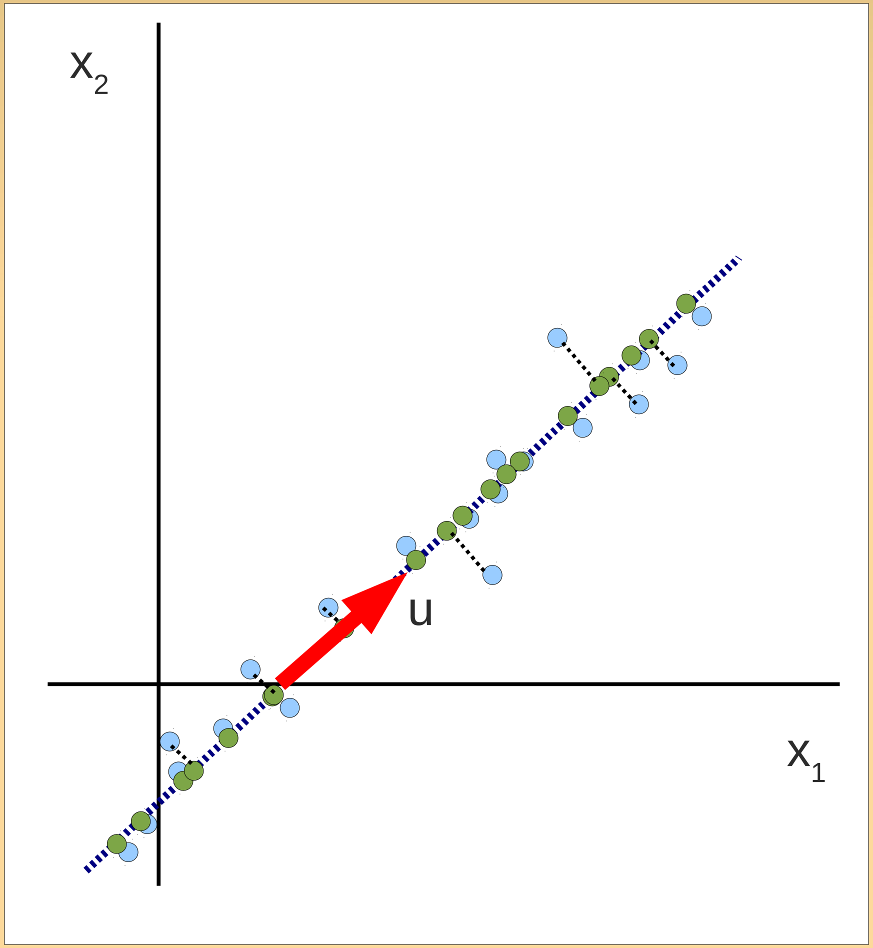
- How would you summarize this data using 1 dimension?

Transform of *k*-th point:

$$\left(x_1^{(k)}, x_2^{(k)}\right) \rightarrow \left(z_1^{(k)}\right)$$

where z_1 is the
orthogonal scalar projection
on (unit vector) $u^{(1)}$:

$$z_1^{(k)} = u_1^{(1)} x_1^{(k)} + u_2^{(1)} x_2^{(k)} = \left(u^{(1)}, x^{(k)}\right)$$



More Principle Components

- $u^{(2)}$ is the direction with most 'remaining' variance
  - orthogonal to $u^{(1)}$!

In general

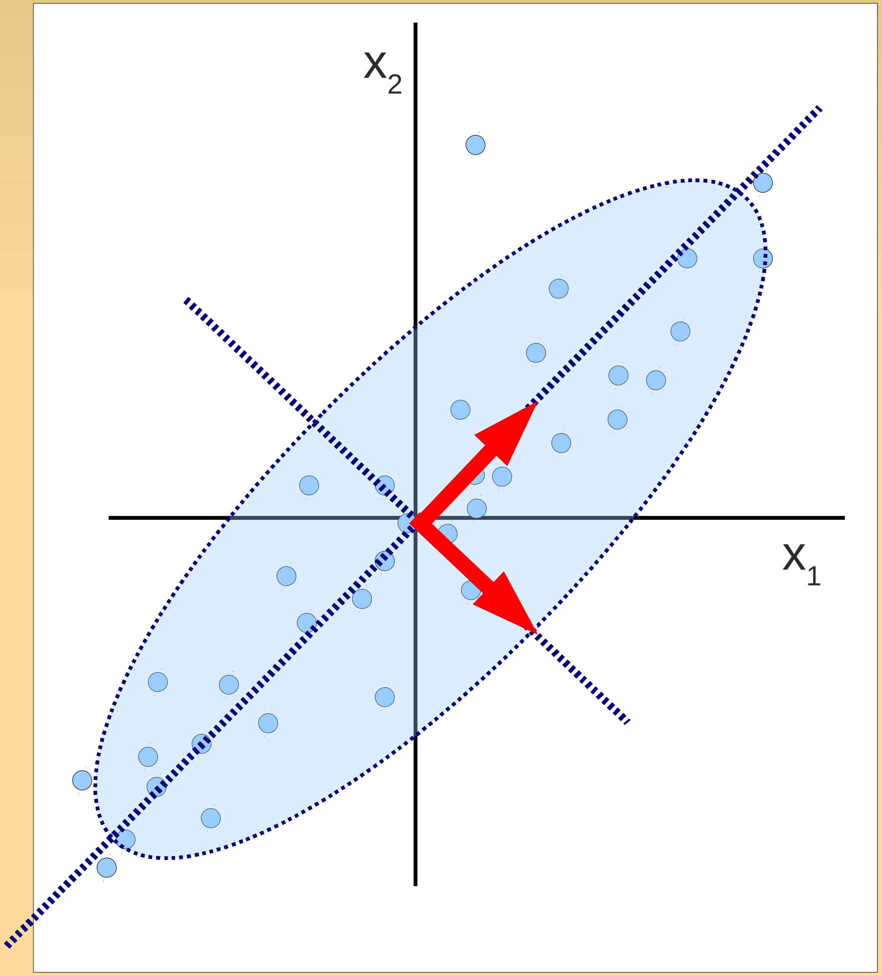
- If the data is D-dimensional
- We can find D directions $u^{(1)}, \dots, u^{(D)}$
- Each direction itself is a D-vector:

$$u^{(i)} = (u_{1,}^{(i)} \dots, u_D^{(i)})$$

- Each direction is orthogonal to the others:

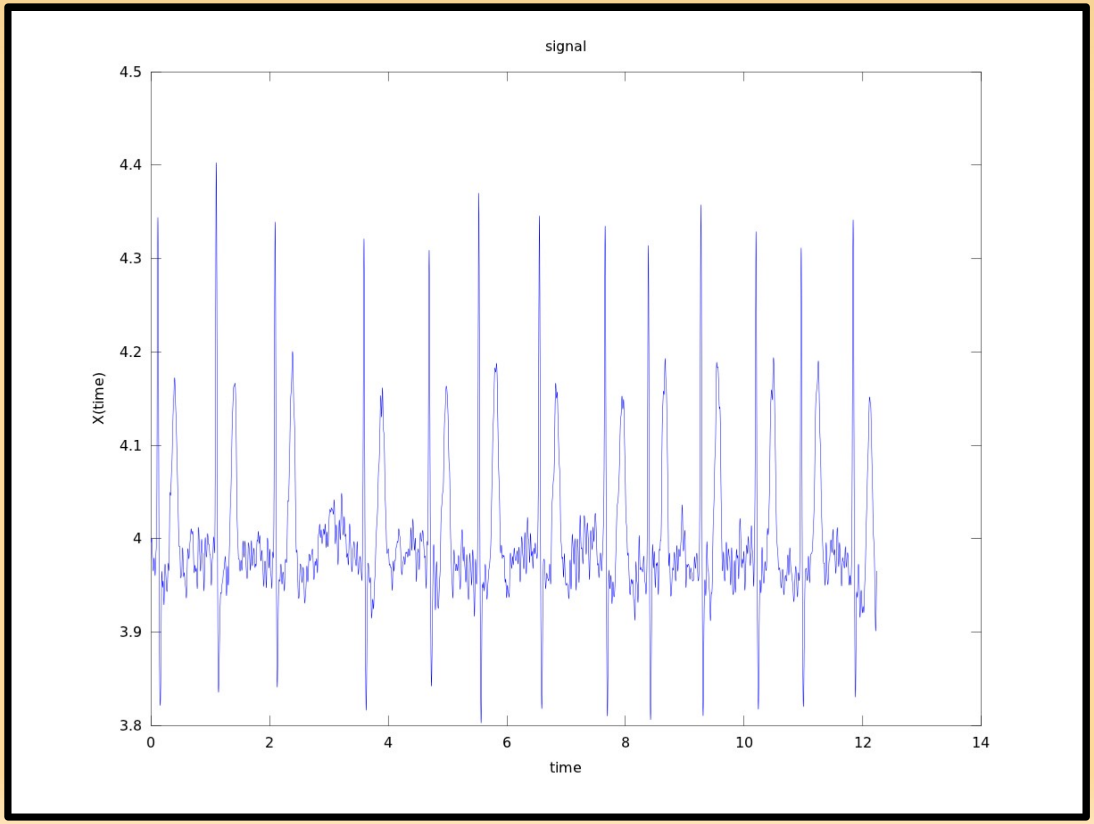
$$(u^{(i)}, u^{(j)}) = 0$$

- The first direction is has most variance
- The least variance is in direction $u^{(D)}$



PCA – Goals

- All directions of high variance might be useful in itself
  - Analysis of data
  - In the lab you will analyze the ECG signal of a patient with a heart disease.



PCA – Goals

- All directions of high variance might be useful in itself
- But not for dimension reduction...
  - Given X (N data points of D variables)
    → Convert to Z (N data points of d variables)

$$\left(x_1^{(0)}, x_2^{(0)}, \dots, x_D^{(0)}\right) \rightarrow \left(z_1^{(0)}, z_2^{(0)}, \dots, z_d^{(0)}\right)$$
$$\left(x_1^{(1)}, x_2^{(1)}, \dots, x_D^{(1)}\right) \rightarrow \left(z_1^{(1)}, z_2^{(1)}, \dots, z_d^{(1)}\right)$$
$$\dots$$
$$\left(x_1^{(n)}, x_2^{(n)}, \dots, x_D^{(n)}\right) \rightarrow \left(z_1^{(n)}, z_2^{(n)}, \dots, z_d^{(n)}\right)$$

The vector $\left(z_i^{(0)}, z_i^{(1)}, \dots, z_i^{(n)}\right)$

is called the *i*-th **principal component** (of the data set)

PCA – Dimension Reduction

- Approach
- Step 1:
  - find all directions (and principal components)

$$\left(x_1^{(0)}, x_2^{(0)}, \dots, x_D^{(0)}\right) \rightarrow \left(z_1^{(0)}, z_2^{(0)}, \dots, z_D^{(0)}\right)$$
$$\left(x_1^{(1)}, x_2^{(1)}, \dots, x_D^{(1)}\right) \rightarrow \left(z_1^{(1)}, z_2^{(1)}, \dots, z_D^{(1)}\right)$$
$$\dots$$
$$\left(x_1^{(n)}, x_2^{(n)}, \dots, x_D^{(n)}\right) \rightarrow \left(z_1^{(n)}, z_2^{(n)}, \dots, z_D^{(n)}\right)$$

- Step 2: …?

PCA – Dimension Reduction

- Approach
- Step 1:
  - find all directions (and principal components)

$$\left(x_1^{(0)}, x_2^{(0)}, \dots, x_D^{(0)}\right) \rightarrow \left(z_1^{(0)}, z_2^{(0)}, \dots, z_D^{(0)}\right)$$
$$\left(x_1^{(1)}, x_2^{(1)}, \dots, x_D^{(1)}\right) \rightarrow \left(z_1^{(1)}, z_2^{(1)}, \dots, z_D^{(1)}\right)$$
$$\dots$$
$$\left(x_1^{(n)}, x_2^{(n)}, \dots, x_D^{(n)}\right) \rightarrow \left(z_1^{(n)}, z_2^{(n)}, \dots, z_D^{(n)}\right)$$

first $d<D$ PCs contain most information!

- Step 2:
  - keep only the directions with high variance.

    → the principal components with much information

$$\left(x_1^{(0)}, x_2^{(0)}, \dots, x_D^{(0)}\right) \rightarrow \left(z_1^{(0)}, z_2^{(0)}, \dots, z_d^{(0)}\right)$$
$$\left(x_1^{(1)}, x_2^{(1)}, \dots, x_D^{(1)}\right) \rightarrow \left(z_1^{(1)}, z_2^{(1)}, \dots, z_d^{(1)}\right)$$
$$\dots$$
$$\left(x_1^{(n)}, x_2^{(n)}, \dots, x_D^{(n)}\right) \rightarrow \left(z_1^{(n)}, z_2^{(n)}, \dots, z_d^{(n)}\right)$$

PCA – Dimension Reduction

- Approach
- Step 1:
  - find all directions
    (and principal components)

$$\left(x_1^{(0)}, x_2^{(0)}, \ldots, x_D^{(0)}\right) \rightarrow \left(z_1^{(0)}, z_2^{(0)}, \ldots, z_D^{(0)}\right)$$
$$\left(x_1^{(1)}, x_2^{(1)}, \ldots, x_D^{(1)}\right) \rightarrow \left(z_1^{(1)}, z_2^{(1)}, \ldots, z_D^{(1)}\right)$$
$$\ldots$$
$$\ldots, x_2^{(n)}, \ldots, x_D^{(n)}\right) \rightarrow \left(z_1^{(n)}, z_2^{(n)}, \ldots, z_D^{(n)}\right)$$

first d<D PCs contain most information!

$$x_1^{(0)}, x_2^{(0)}, \ldots, x_D^{(0)}\right) \rightarrow \left(z_1^{(0)}, z_2^{(0)}, \ldots, z_d^{(0)}\right)$$
$$x_1^{(1)}, x_2^{(1)}, \ldots, x_D^{(1)}\right) \rightarrow \left(z_1^{(1)}, z_2^{(1)}, \ldots, z_d^{(1)}\right)$$
$$\ldots$$
$$x_1^{(n)}, x_2^{(n)}, \ldots, x_D^{(n)}\right) \rightarrow \left(z_1^{(n)}, z_2^{(n)}, \ldots, z_d^{(n)}\right)$$

PCA – More Concrete

- PCA
  - finding all the directions, and
  - principle components
- Data compression using PCA
  - computing compressed representation
  - computing reconstruction

PCA – More Concrete

- PCA
  - finding all the directions, and
  - principle components
- Data compression using PCA
  - computing compressed representation
  - computing reconstruction

still to be shown
(using eigen decomposition of cov. matrix)

Easy! for *k*-th point:

$$z_j^{(k)} = \left(u^{(j)}, x^{(k)}\right)$$

Easy!
For *k*-th point just keep

$$\left(z_1^{(k)}, \ldots, z_d^{(k)}\right)$$

still to be shown
(we show that data is a linear combination of the PCs)

PCA – More Concrete

- PCA
  - finding all the directions, and
  - principle components
- Data compression using PCA
  - computing compressed representation
  - computing reconstruction

still to be shown
(using eigen decomposition of cov. matrix)

Easy! for *k*-th point:

$$z_j^{(k)} = \left(u^{(j)}, x^{(k)}\right)$$

Easy!
For *k*-th point just keep
$\left(z_1^{(k)}, \ldots, z_d^{(k)}\right)$

still to be shown
(we show that data is a linear combination of the PCs)

Computing the directions U

Note: X is now D x N (before N x D)

Algorithm

- X is the DxN data matrix

1) Preprocessing:
   - **scale** the features
   - make X **zero mean**
2) Compute the **data covariance matrix**
3) Perform **eigen decomposition**
   - directions u_i are the eigenvectors of C
   - variance of u_i is the corresponding eigenvalue

Computing the directions U

Algorithm

- X is the DxN data matrix

1) Preprocessing:
   - **scale** the features
   - make X **zero mean**

$$x_i^{(k)} = \frac{2 x_i^{(k)}}{max_l x_i^{(l)} - min_m x_i^{(l)}}$$

2) Compute the **data covariance matrix**
3) Perform **eigen decomposition**
   - directions u_i are the eigenvectors of C
   - variance of u_i is the corresponding eigenvalue

Computing the directions U

Algorithm

- X is the DxN data matrix

1) Preprocessing:
   - **scale** the features
   - make X **zero mean**

2) Compute the **data covariance matrix**

3) Perform **eigen decomposition**
   - directions u_i are the eigenvectors of C
   - variance of u_i is the corresponding eigenvalue

- Compute μ (the mean data point) $$\mu_i = \frac{1}{N} \sum_{k=1}^{N-1} x_i^{(k)}$$
- subtract the mean from each point $$x^{(k)} = x^{(k)} - \mu$$

Computing the directions U

Algorithm

- X is the DxN data matrix

1) Preprocessing:
   - **scale** the features
   - make X **zero mean**
2) Compute the **data covariance matrix**
3) Perform **eigen decomposition**
   - directions u_i are the eigenvectors of C
   - variance of u_i is the corresponding eigenvalue

- Data covariance matrix

$$C = \frac{1}{N} X X^T$$

Computing the directions U

Algorithm

- X is the DxN data matrix

1) Preprocessing:
   - **scale** the features
   - make X **zero mean**
2) Compute the **data covariance matrix**
3) Perform **eigen decomposition**
   - directions u_i are the eigenvectors of C
   - variance of u_i is the corresponding eigenvalue

- A square matrix has eigenvectors: map to a multiple of themselves

$$C x = \lambda x$$

eigenvector

(scalar) eigenvalue

Computing the directions U

Algorithm

- X is the DxN data matrix

1) Preprocessing:
   - **scale** the features
   - make X **zero mean**
2) Compute the **data covariance matrix**
3) Perform **eigen decomposition**
   - directions u_i are the eigenvectors of C
   - variance of u_i is the corresponding eigenvalue

```
[eigenvectors, eigenvals] = eig(C)
% 'eig' delivers eigenvectors with
% the wrong order
% so we flip the matrix
U = fliplr(eigenvectors)

% U(i, :) now is the i-th direction
```

- A square matrix has eigenvectors: map to a multiple of themselves

$$Cx = \lambda x$$

eigenvector

(scalar) eigenvalue

PCA – More Concrete

- PCA
  - finding all the directions, and
  - principle components
- Data compression using PCA
  - computing compressed representation
  - computing reconstruction

still to be shown
(using eigen decomposition of cov. matrix)

Easy! for *k*-th point:

$$z_j^{(k)} = \left(u^{(j)}, x^{(k)}\right)$$

Easy!
For *k*-th point just keep
$\left(z_1^{(k)}, \ldots, z_d^{(k)}\right)$

still to be shown
(we show that data is a linear combination of the PCs)

Data as Linear Combination of The Principal Components

- Starting from $z_i^{(k)} = u_1^{(i)} x_1^{(k)} + \ldots + u_D^{(i)} x_D^{(k)}$
- In matrix form $Z = U^T X$
  - Note: X is still D x N (before N x D)

$$\begin{bmatrix} z_{11} & \ldots & z_{1n} \\ \ldots & \ldots & \ldots \\ z_{D1} & \ldots & z_{Dn} \end{bmatrix} = \begin{bmatrix} u_{11}^T & \ldots & u_{1D}^T \\ \ldots & \ldots & \ldots \\ u_{D1}^T & \ldots & u_{DD}^T \end{bmatrix} \begin{bmatrix} x_{11} & \ldots & x_{1n} \\ \ldots & \ldots & \ldots \\ x_{D1} & \ldots & x_{Dn} \end{bmatrix}$$

$$\begin{bmatrix} \begin{pmatrix} \vdots \\ z^{(1)} \\ \vdots \end{pmatrix} \cdots \begin{pmatrix} \vdots \\ z^{(n)} \\ \vdots \end{pmatrix} \end{bmatrix} = \begin{bmatrix} \begin{pmatrix} \ldots & u^{(1)} & \ldots \end{pmatrix} \\ \ldots \\ \begin{pmatrix} \ldots & u^{(D)} & \ldots \end{pmatrix} \end{bmatrix} \begin{bmatrix} \begin{pmatrix} \vdots \\ x^{(1)} \\ \vdots \end{pmatrix} \cdots \begin{pmatrix} \vdots \\ x^{(n)} \\ \vdots \end{pmatrix} \end{bmatrix}$$

Data as Linear Combination of The Principal Components

- Starting from $z_i^{(k)} = u_1^{(i)} x_1^{(k)} + \ldots + u_D^{(i)} x_D^{(k)}$
- In matrix form $Z = U^T X$

$$\begin{bmatrix} z_{11} & \ldots & z_{1n} \\ \ldots & \ldots & \ldots \\ z_{D1} & \ldots & z_{Dn} \end{bmatrix} = \begin{bmatrix} u_{11}^T & \ldots & u_{1D}^T \\ \ldots & \ldots & \ldots \\ u_{D1}^T & \ldots & u_{DD}^T \end{bmatrix} \begin{bmatrix} x_{11} & \ldots & x_{1n} \\ \ldots & \ldots & \ldots \\ x_{D1} & \ldots & x_{Dn} \end{bmatrix}$$

$$\left[\begin{pmatrix} \vdots \\ z^{(1)} \\ \vdots \end{pmatrix} \ldots \begin{pmatrix} \vdots \\ z^{(n)} \\ \vdots \end{pmatrix} \right] = \begin{bmatrix} (\ldots & u^{(1)} & \ldots) \\ & \ldots & \\ (\ldots & u^{(D)} & \ldots) \end{bmatrix} \left[\begin{pmatrix} \vdots \\ x^{(1)} \\ \vdots \end{pmatrix} \ldots \begin{pmatrix} \vdots \\ x^{(n)} \\ \vdots \end{pmatrix} \right]$$

Data as Linear Combination of The Principal Components

- Starting from $z_i^{(k)} = u_1^{(i)} x_1^{(k)} + \ldots + u_D^{(i)} x_D^{(k)}$
- In matrix form $Z = U^T X$

$$\begin{bmatrix} z_{11} & \ldots & z_{1n} \\ \ldots & \ldots & \ldots \\ z_{D1} & \ldots & z_{Dn} \end{bmatrix} = \begin{bmatrix} u_{11}^T & \ldots & u_{1D}^T \\ \ldots & \ldots & \ldots \\ u_{D1}^T & \ldots & u_{DD}^T \end{bmatrix} \begin{bmatrix} x_{11} & \ldots & x_{1n} \\ \ldots & \ldots & \ldots \\ x_{D1} & \ldots & x_{Dn} \end{bmatrix}$$

$$\begin{bmatrix} \begin{pmatrix} \vdots \\ z^{(1)} \\ \vdots \end{pmatrix} \cdots \begin{pmatrix} \vdots \\ z^{(n)} \\ \vdots \end{pmatrix} \end{bmatrix} = \begin{bmatrix} \begin{pmatrix} \ldots & u^{(1)} & \ldots \end{pmatrix} \\ \ldots \\ \begin{pmatrix} \ldots & u^{(D)} & \ldots \end{pmatrix} \end{bmatrix} \begin{bmatrix} \begin{pmatrix} \vdots \\ x^{(1)} \\ \vdots \end{pmatrix} \cdots \begin{pmatrix} \vdots \\ x^{(n)} \\ \vdots \end{pmatrix} \end{bmatrix}$$

Linear Algebra:

$$Z = U^T X$$

$$(U^T)^{-1} Z = X$$

$$\{U \text{ is orthonormal}\}$$

$$U Z = X$$

Data as Linear Combination of The Principal Components

- Starting from $z_i^{(k)} = u_1^{(i)} x_1^{(k)} + \ldots + u_D^{(i)} x_D^{(k)}$
- In matrix form $Z = U^T X$

$$\begin{bmatrix} z_{11} & \ldots & z_{1n} \\ \ldots & \ldots & \ldots \\ z_{D1} & \ldots & z_{Dn} \end{bmatrix} = \begin{bmatrix} u_{11}^T & \ldots & u_{1D}^T \\ \ldots & \ldots & \ldots \\ u_{D1}^T & \ldots & u_{DD}^T \end{bmatrix} \begin{bmatrix} x_{11} & \ldots & x_{1n} \\ \ldots & \ldots & \ldots \\ x_{D1} & \ldots & x_{Dn} \end{bmatrix}$$

$$\left[\begin{pmatrix} \vdots \\ z^{(1)} \\ \vdots \end{pmatrix} \cdots \begin{pmatrix} \vdots \\ z^{(n)} \\ \vdots \end{pmatrix} \right] = \begin{bmatrix} (\ldots & u^{(1)} & \ldots) \\ & \ldots & \\ (\ldots & u^{(D)} & \ldots) \end{bmatrix} \left[\begin{pmatrix} \vdots \\ x^{(1)} \\ \vdots \end{pmatrix} \cdots \begin{pmatrix} \vdots \\ x^{(n)} \\ \vdots \end{pmatrix} \right]$$

Linear Algebra:

$$Z = U^T X$$
$$(U^T)^{-1} Z = X$$
$$\{U \text{ is orthonormal}\}$$
$$U Z = X$$

$$\left[\begin{pmatrix} \vdots \\ x^{(1)} \\ \vdots \end{pmatrix} \cdots \begin{pmatrix} \vdots \\ x^{(n)} \\ \vdots \end{pmatrix} \right] = \left[\begin{pmatrix} \vdots \\ u^{(1)} \\ \vdots \end{pmatrix} \cdots \begin{pmatrix} \vdots \\ u^{(D)} \\ \vdots \end{pmatrix} \right] \left[\begin{pmatrix} \vdots \\ z^{(1)} \\ \vdots \end{pmatrix} \cdots \begin{pmatrix} \vdots \\ z^{(n)} \\ \vdots \end{pmatrix} \right]$$

Data as Linear Combination of The Principal Components

- Starting from $z_i^{(k)} = u_1^{(i)} x_1^{(k)} + \ldots + u_D^{(i)} x_D^{(k)}$
- In matrix form $Z = U^T X$

$$\begin{bmatrix} z_{11} & \ldots & z_{1n} \\ \ldots & \ldots & \ldots \\ z_{D1} & \ldots & z_{Dn} \end{bmatrix} = \begin{bmatrix} u_{11}^T & \ldots & u_{1D}^T \\ \ldots & \ldots & \ldots \\ u_{D1}^T & \ldots & u_{DD}^T \end{bmatrix} \begin{bmatrix} x_{11} & \ldots & x_{1n} \\ \ldots & \ldots & \ldots \\ x_{D1} & \ldots & x_{Dn} \end{bmatrix}$$

$$\begin{bmatrix} \begin{pmatrix} \vdots \\ z^{(1)} \\ \vdots \end{pmatrix} \cdots \begin{pmatrix} \vdots \\ z^{(n)} \\ \vdots \end{pmatrix} \end{bmatrix} = \begin{bmatrix} (\ldots & u^{(1)} & \ldots) \\ & \ldots & \\ (\ldots & u^{(D)} & \ldots) \end{bmatrix} \begin{bmatrix} \begin{pmatrix} \vdots \\ x^{(1)} \\ \vdots \end{pmatrix} \cdots \begin{pmatrix} \vdots \\ x^{(n)} \\ \vdots \end{pmatrix} \end{bmatrix}$$

Linear Algebra:

$$Z = U^T X$$

$$(U^T)^{-1} Z = X$$

$$\{U \text{ is orthonormal}\}$$

$$U Z = X$$

$$\begin{bmatrix} \begin{pmatrix} \vdots \\ x^{(1)} \\ \vdots \end{pmatrix} \cdots \begin{pmatrix} \vdots \\ x^{(n)} \\ \vdots \end{pmatrix} \end{bmatrix} = \begin{bmatrix} \begin{pmatrix} \vdots \\ u^{(1)} \\ \vdots \end{pmatrix} \cdots \begin{pmatrix} \vdots \\ u^{(D)} \\ \vdots \end{pmatrix} \end{bmatrix} \begin{bmatrix} \begin{pmatrix} \vdots \\ z^{(1)} \\ \vdots \end{pmatrix} \cdots \begin{pmatrix} \vdots \\ z^{(n)} \\ \vdots \end{pmatrix} \end{bmatrix}$$

1st PC

Dth PC

Data as Linear Combination of The Principal Components

- Starting from $z_i^{(k)} = u_1^{(i)} x_1^{(k)} + \ldots + u_D^{(i)} x_D^{(k)}$
- In matrix form $Z = U^T X$

$$\begin{bmatrix} z_{11} & \ldots & z_{1n} \\ \ldots & \ldots & \ldots \\ z_{D1} & \ldots & z_{Dn} \end{bmatrix} = \begin{bmatrix} u_{11}^T & \ldots & u_{1D}^T \\ \ldots & \ldots & \ldots \\ u_{D1}^T & \ldots & u_{DD}^T \end{bmatrix} \begin{bmatrix} x_{11} & \ldots & x_{1n} \\ \ldots & \ldots & \ldots \\ x_{D1} & \ldots & x_{Dn} \end{bmatrix}$$

$$\left[\begin{pmatrix} \vdots \\ z^{(1)} \\ \vdots \end{pmatrix} \cdots \begin{pmatrix} \vdots \\ z^{(n)} \\ \vdots \end{pmatrix} \right] = \begin{bmatrix} (\ldots & u^{(1)} & \ldots) \\ & \ldots & \\ (\ldots & u^{(D)} & \ldots) \end{bmatrix} \left[\begin{pmatrix} \vdots \\ x^{(1)} \\ \vdots \end{pmatrix} \cdots \begin{pmatrix} \vdots \\ x^{(n)} \\ \vdots \end{pmatrix} \right]$$

Linear Algebra:

$$Z = U^T X$$

$$(U^T)^{-1} Z = X$$

$$\{U \text{ is orthonormal}\}$$

$$U Z = X$$

$$x_i^{(k)} = x_{ik}$$

$$x_i^{(k)} = u_{i1} z_{1k} + \ldots + u_{iD} z_{Dk}$$

$$x_i^{(k)} = u_i^{(1)} z_1^{(k)} + \ldots + u_i^{(D)} z_D^{(k)}$$

$$\left[\begin{pmatrix} \vdots \\ x^{(1)} \\ \vdots \end{pmatrix} \cdots \begin{pmatrix} \vdots \\ x^{(n)} \\ \vdots \end{pmatrix} \right] = \left[\begin{pmatrix} \vdots \\ u^{(1)} \\ \vdots \end{pmatrix} \cdots \begin{pmatrix} \vdots \\ u^{(D)} \\ \vdots \end{pmatrix} \right] \left[\begin{pmatrix} \vdots \\ z^{(1)} \\ \vdots \end{pmatrix} \cdots \begin{pmatrix} \vdots \\ z^{(n)} \\ \vdots \end{pmatrix} \right]$$

1st PC

Dth PC

Reconstruction from PCs

- Compression: only keep first *d* PCs
- Reconstruction from those...?
  - just by the previous formulas

$$Z = U^T X$$
$$\left(U^T\right)^{-1} Z = X$$
$$\{U \text{ is orthonormal}\}$$
$$U Z = X$$

Data as Linear Combination of The Principal Components

- Compression: only keep first *d* PCs
- Reconstruction from those...?
  - just by the previous formulas

$$Z = U^T X$$
$$\left(U^T\right)^{-1} Z = X$$
$$\{U \text{ is orthonormal}\}$$
$$U Z = X$$

$$x_i^{(k)} = u_i^{(1)} z_1^{(k)} + \ldots + u_i^{(d)} z_d^{(k)} + \ldots + u_i^{(D)} z_D^{(k)}$$

$$\left[\begin{pmatrix} \vdots \\ x^{(1)} \\ \vdots \end{pmatrix} \cdots \begin{pmatrix} \vdots \\ x^{(n)} \\ \vdots \end{pmatrix}\right] = \left[\begin{pmatrix} \vdots \\ u^{(1)} \\ \vdots \end{pmatrix} \cdots \begin{pmatrix} \vdots \\ u^{(D)} \\ \vdots \end{pmatrix}\right]\left[\begin{pmatrix} \vdots \\ z^{(1)} \\ \vdots \end{pmatrix} \cdots \begin{pmatrix} \vdots \\ z^{(n)} \\ \vdots \end{pmatrix}\right]$$

1^{st} PC

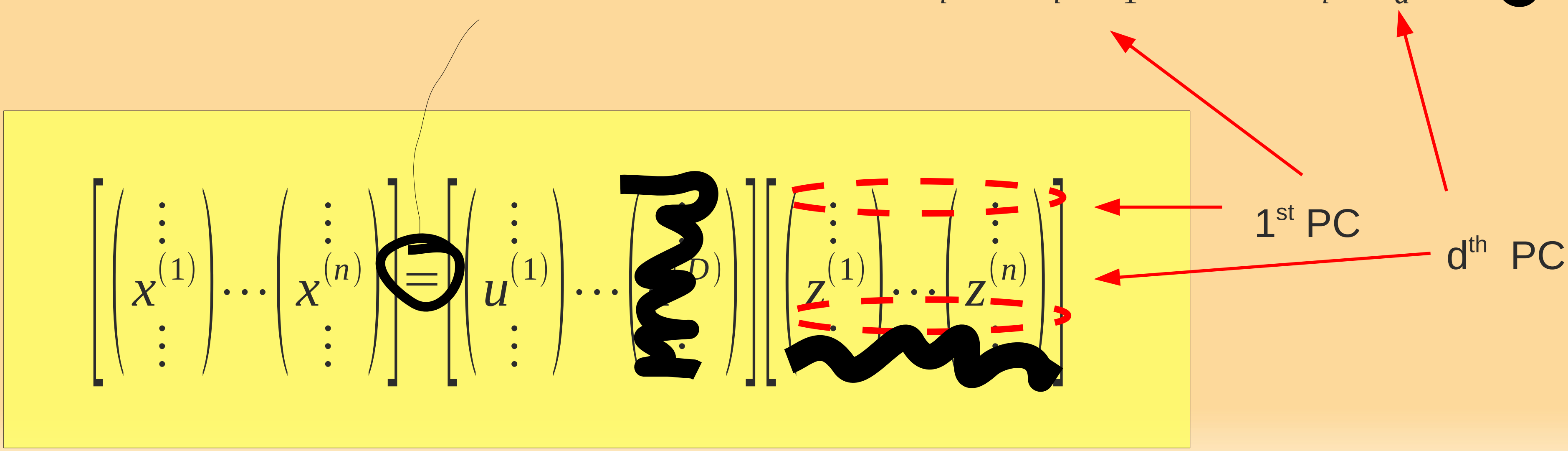
d^{th} PC

Data as Linear Combination of The Principal Components

- Compression: only keep first *d* PCs
- Reconstruction from those...?
  - just by the previous formulas

$$Z = U^T X$$
$$(U^T)^{-1} Z = X$$
$$\{U \text{ is orthonormal}\}$$
$$U Z = X$$

$$x_i^{(k)} = u_i^{(1)} z_1^{(k)} + \dots + u_i^{(d)} z_d^{(k)}$$ [redacted]



Reconstruction from PCs

- Compression: only keep first *d* PCs
- Reconstruction from those...?
  - just by the previous formulas

$$Z = U^T X$$
$$(U^T)^{-1} Z = X$$
$$\{U \text{ is orthonormal}\}$$
$$U Z = X$$

Dimension reduction

- rather than using all D directions $u^{(i)}$,
- use only the *d* first $u^{(1)},\ldots,u^{(d)}$
- so now $\hat{U}$ is a $D \times d$ matrix

$$\{\hat{Z} \text{ is } d \times N\}$$

$$\hat{Z} = \hat{U}^T X$$
$$\hat{U}\hat{Z} \approx X$$

Reconstruction from PCs

- Compression: only keep first *d* PCs
- Reconstruction from those...?
  - just by the previous formulas

$$Z = U^T X$$
$$(U^T)^{-1} Z = X$$
$$\{U \text{ is orthonormal}\}$$
$$U Z = X$$

Dimension reduction

- rather than using all D directions $u^{(i)}$,
- use only the d first $u^{(1)},\ldots,u^{(d)}$
- so now $\hat{U}$ is a $D \times d$ matrix

$\{\hat{Z} \text{ is } d \times N\}$ →

$$\hat{Z} = \hat{U}^T X$$
$$\hat{U}\hat{Z} \approx X$$

this is the reconstruction of the data from only the first *d* principal components

Finally: How many components?

- Compression: only keep first *d* PCs
  - but how to decide how many?!

Finally: How many components?

- Compression: only keep first *d* PCs
  - but how to decide how many?!

- eigenvector j (direction $u^{(j)}$) ↔ associated eigenvalue
  - indicates the amount of variance in $u^{(j)}$
  - sum of eigenvalues is the total variance
  - typically pick *d* to preserve, e.g., 90% of the variance.

Numerical Differentiation and Integration

Numerical Differentiation and Integration

- Finding derivatives or primitives of a function *f*
- not always easy or possible....
  - no closed form solution exists
  - the solution is a very complex expression that is hard to evaluate
  - we may not know *f* (as before!)

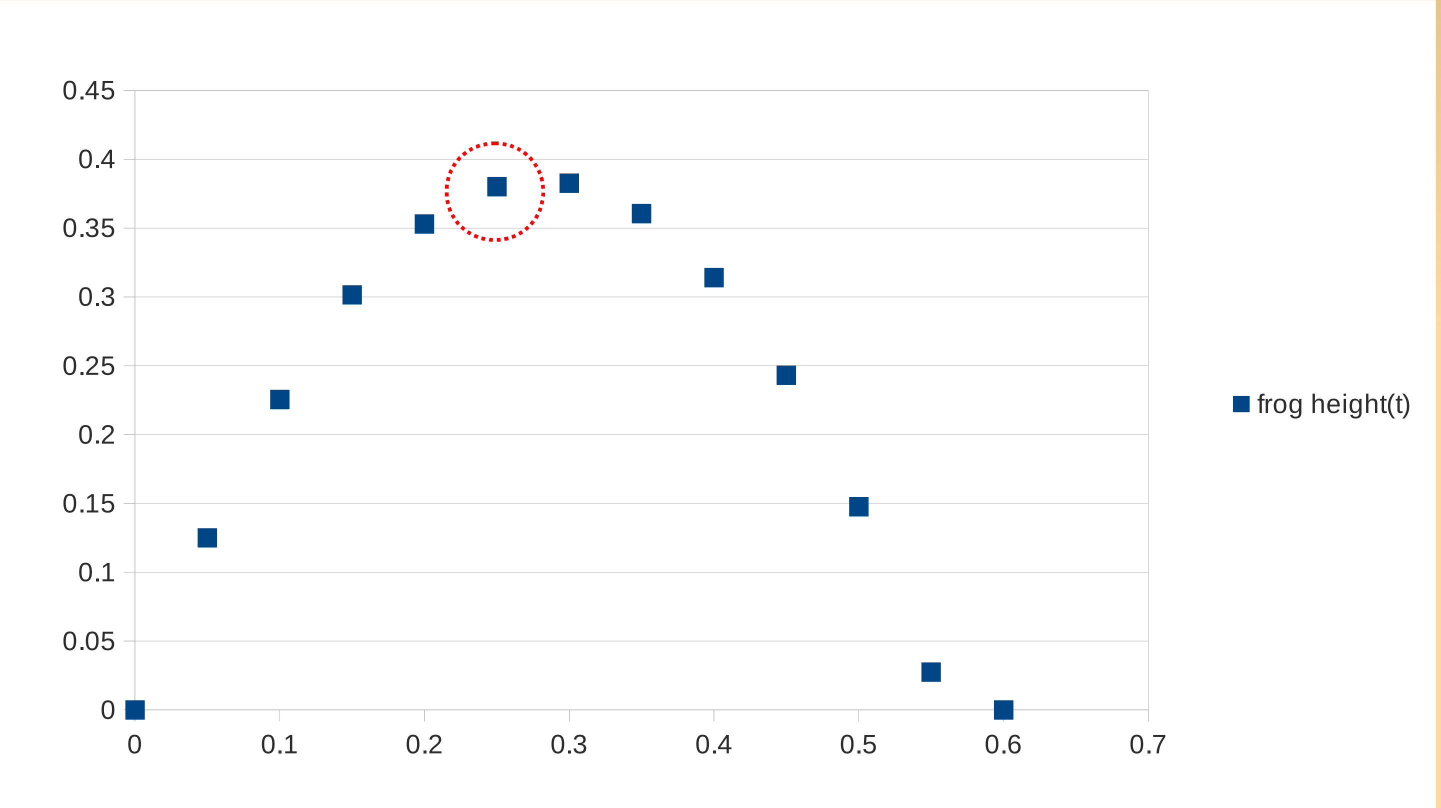
→ numerical methods

Numerical Differentiation

- If we want to know the rate of change...
- E.g.:
  - [QSG] fluid in a cylinder with a hole in the bottom, measured every 5 seconds.
  - High-speed camera images of animal movements, (jumping in frogs and insects, suction feeding in fish, and the strikes of mantis shrimp)
    - determine speed
    - and acceleration

Numerical Differentiation

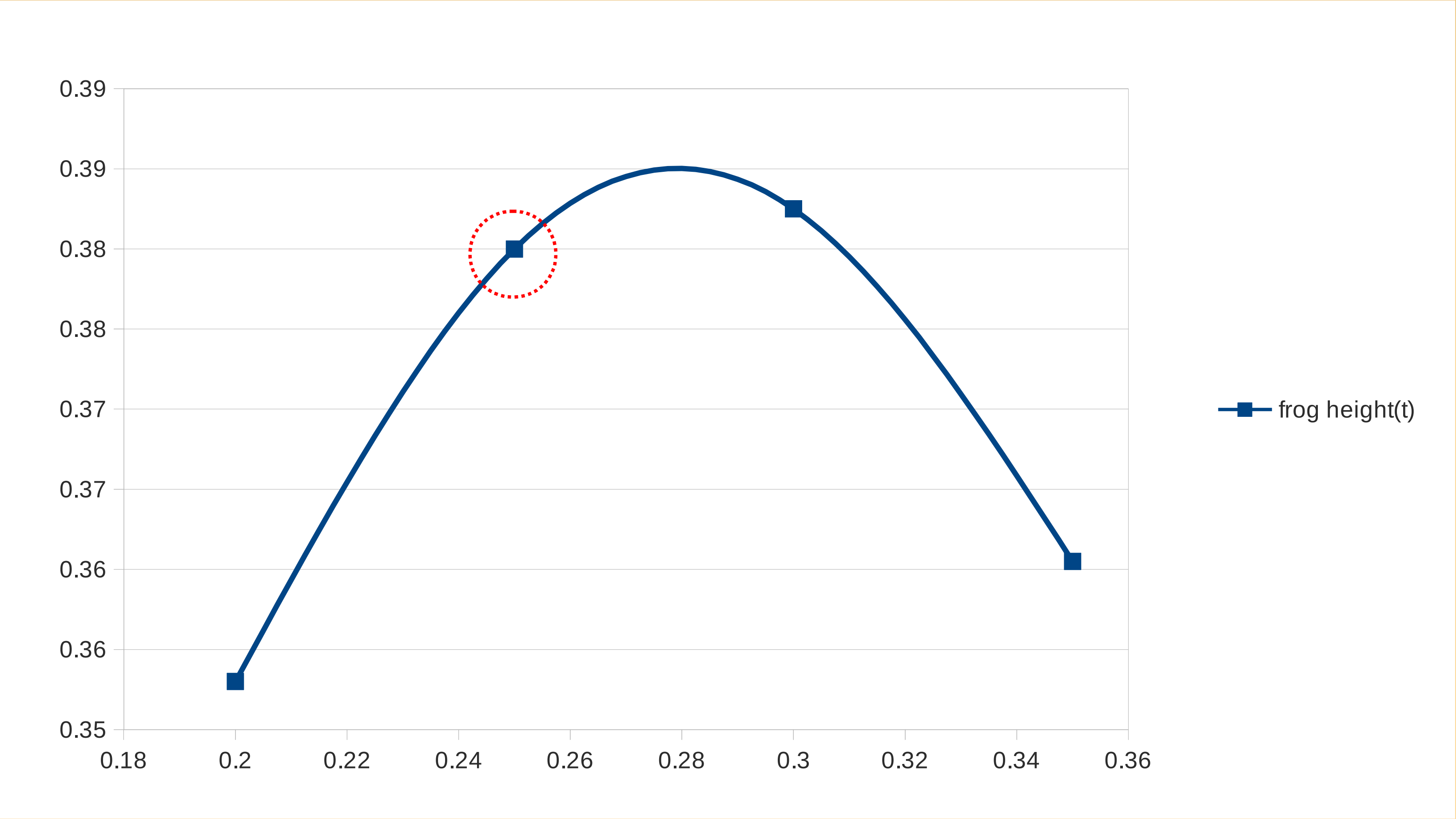
- Determine the vertical speed at t=0.25



- what would you do?

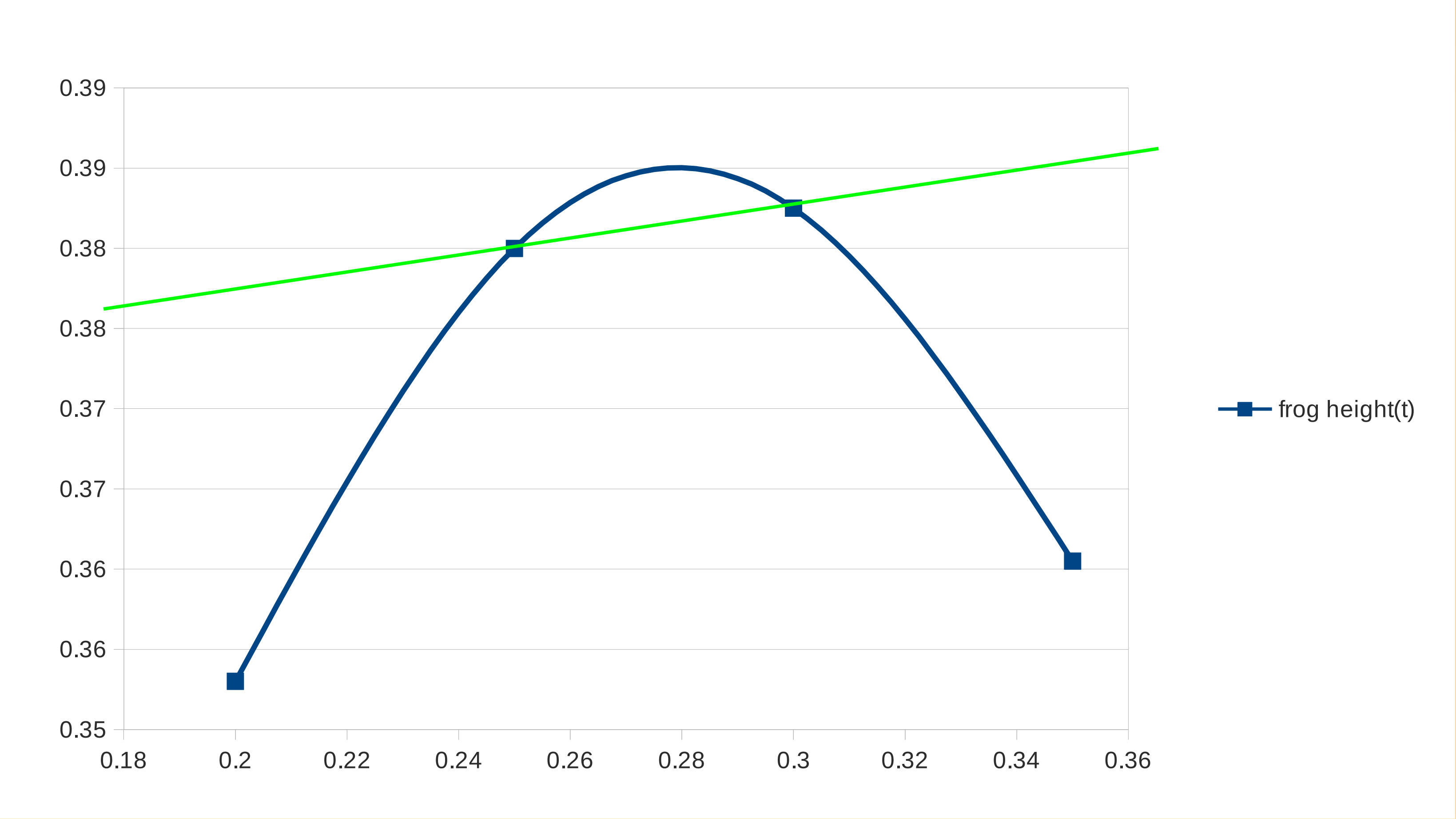
Numerical Differentiation

- Determine the vertical speed at t=0.25...
  - a few options...



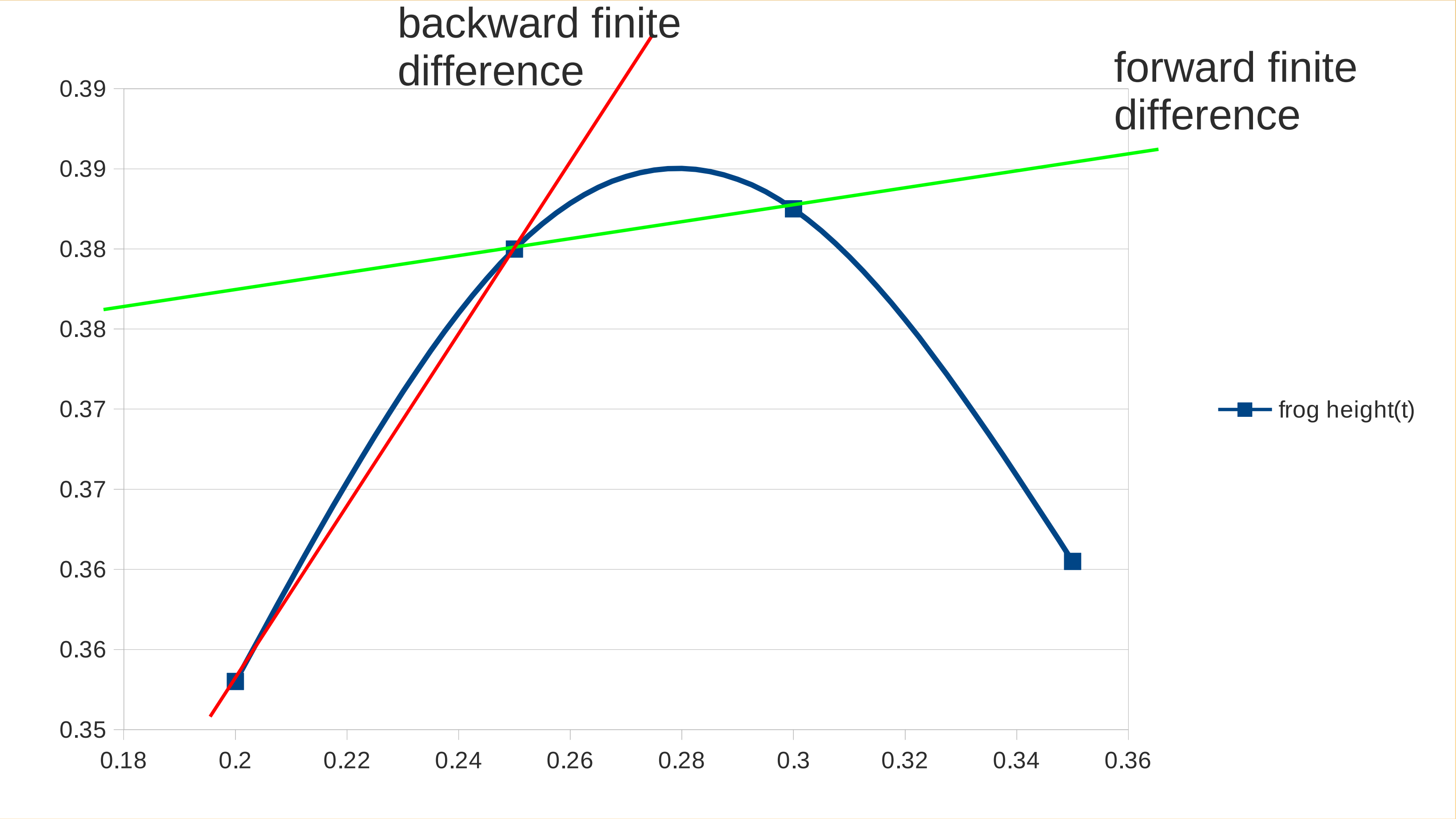
Numerical Differentiation

- Determine the vertical speed at t=0.25...
  - a few options...



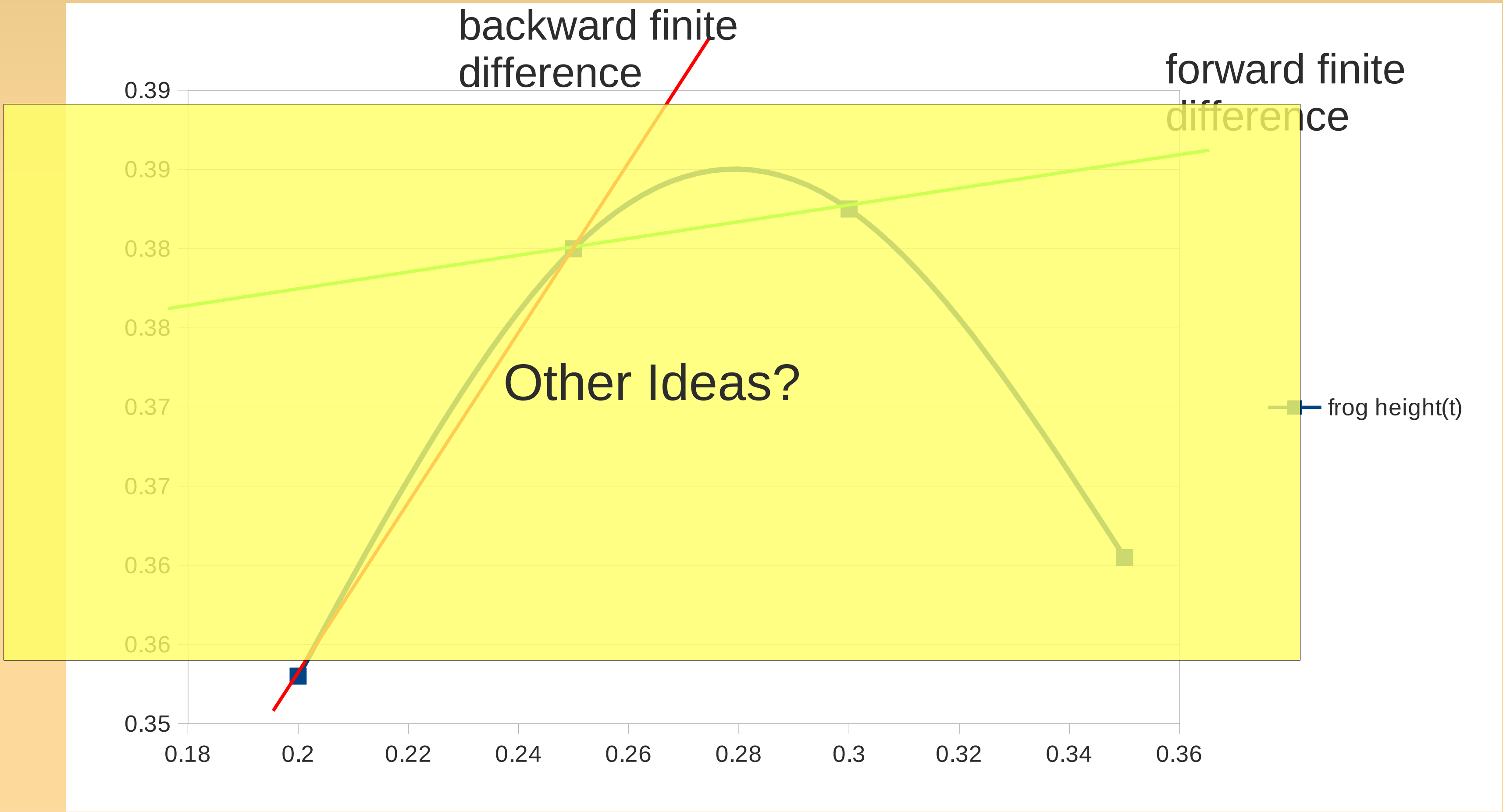
Numerical Differentiation

- Determine the vertical speed at t=0.25...
  - a few options...



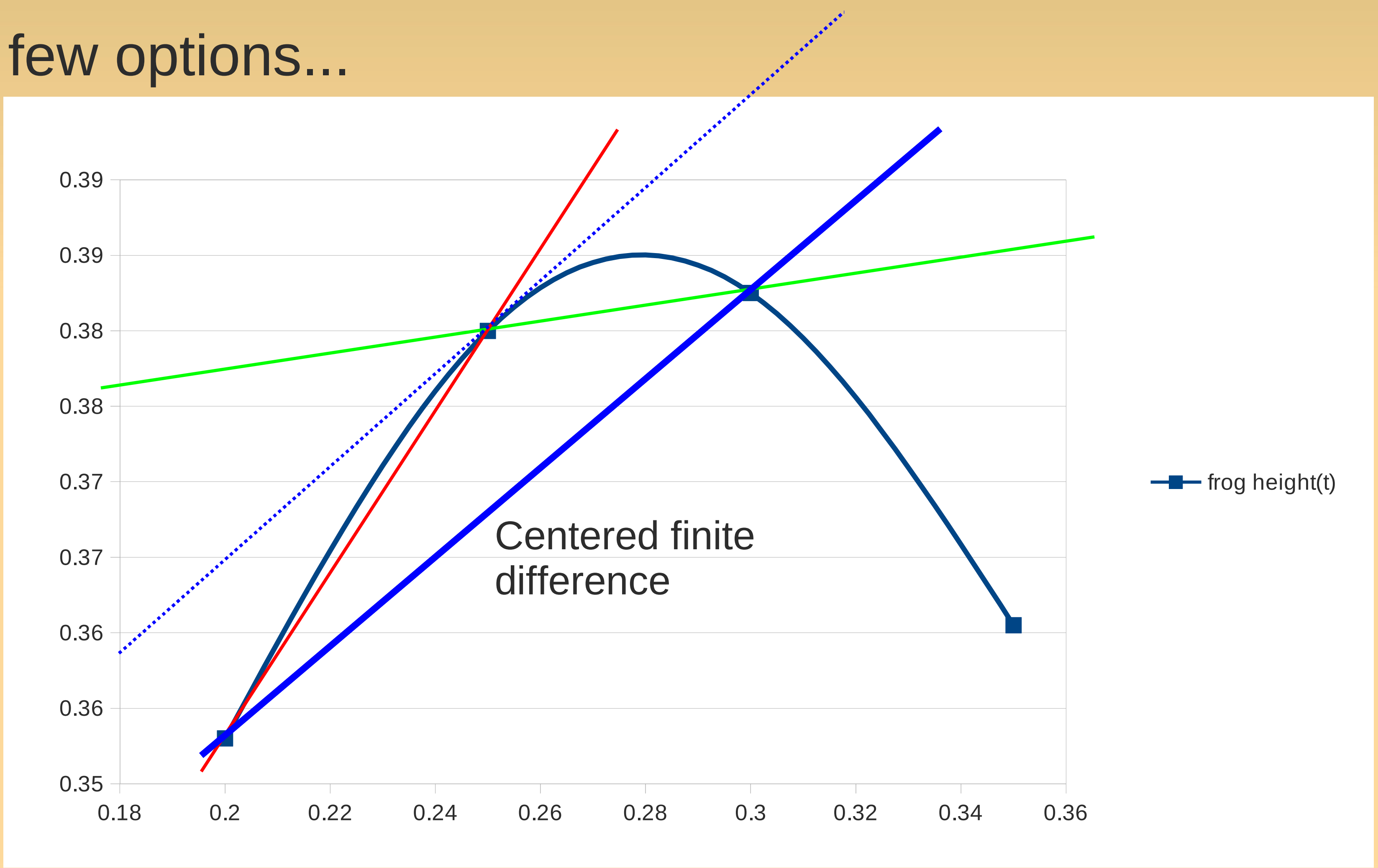
Numerical Differentiation

- Determine the vertical speed at t=0.25...
  - a few options...



Numerical Differentiation

- Determine the vertical speed at t=0.25...
  - a few options...



Numerical Integration

- Integration: the reversed problem...
- Suppose we travel in a car with a broken odometer
- Speedometer is working...

Numerical Integration

- maintain speeds, to figure out traveled distance

t	v(t) km/h
0	80
30	120
65	128
120	122
728	120
733	0
798	20
836	20
941	70
970	120
1350	123
1404	90

enter highway ramp (→ row t = 0)

traffic jam (→ row t = 733)

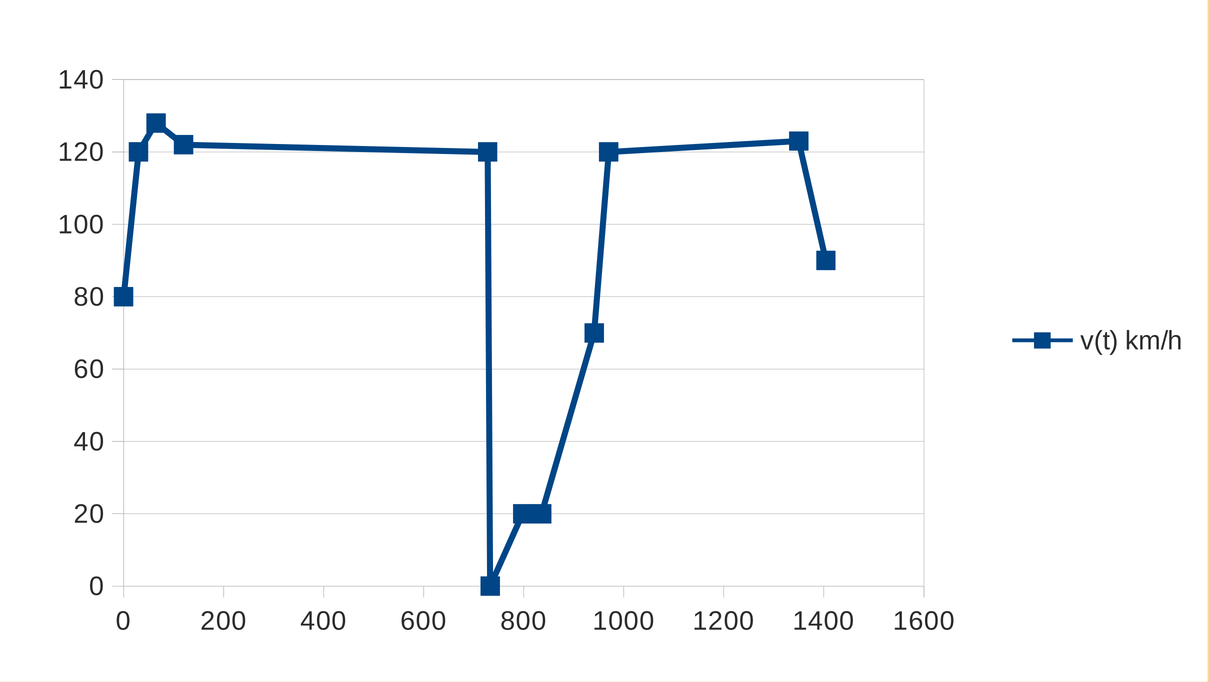
exit highway ramp (→ row t = 1404)

Numerical Integration

- maintain speeds, to figure out traveled distance

t	v(t) km/h
0	80
30	120
65	128
120	
728	
733	
798	
836	
941	
970	
1350	
1404	

enter highway ramp



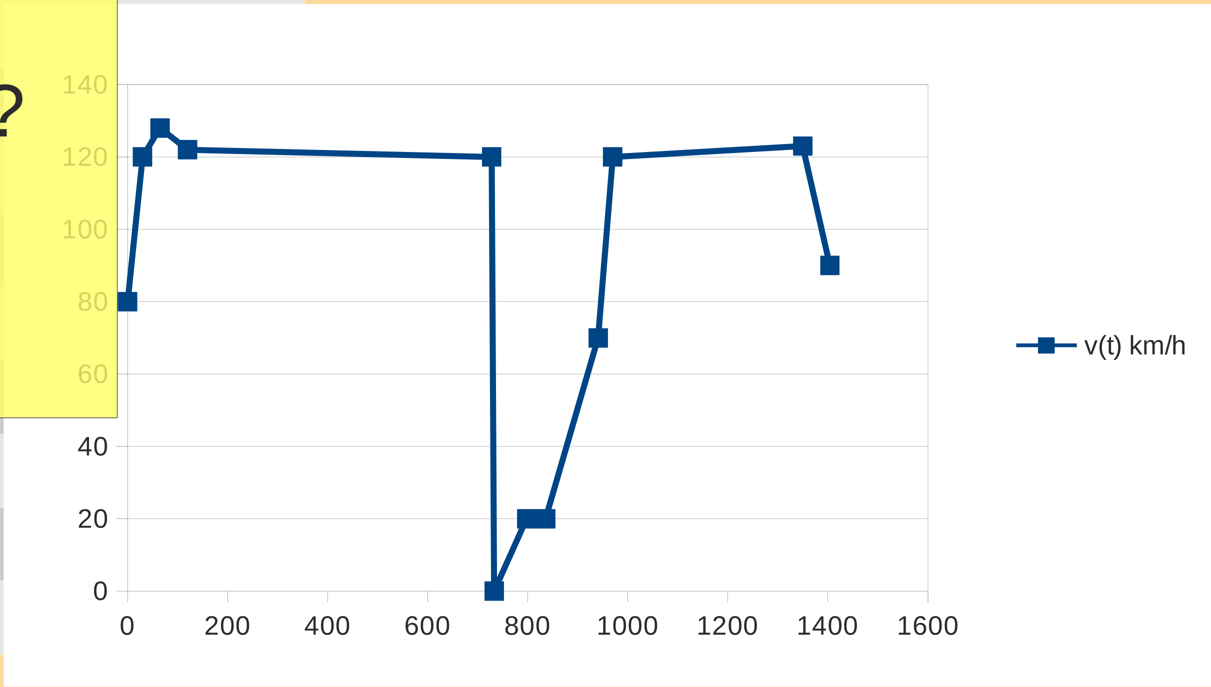
Numerical Integration

- maintain speeds, to figure out traveled distance

t	v(t) km/h
0	80
30	120
65	128
120	
728	
733	
798	
836	
941	
970	
1350	
1404	

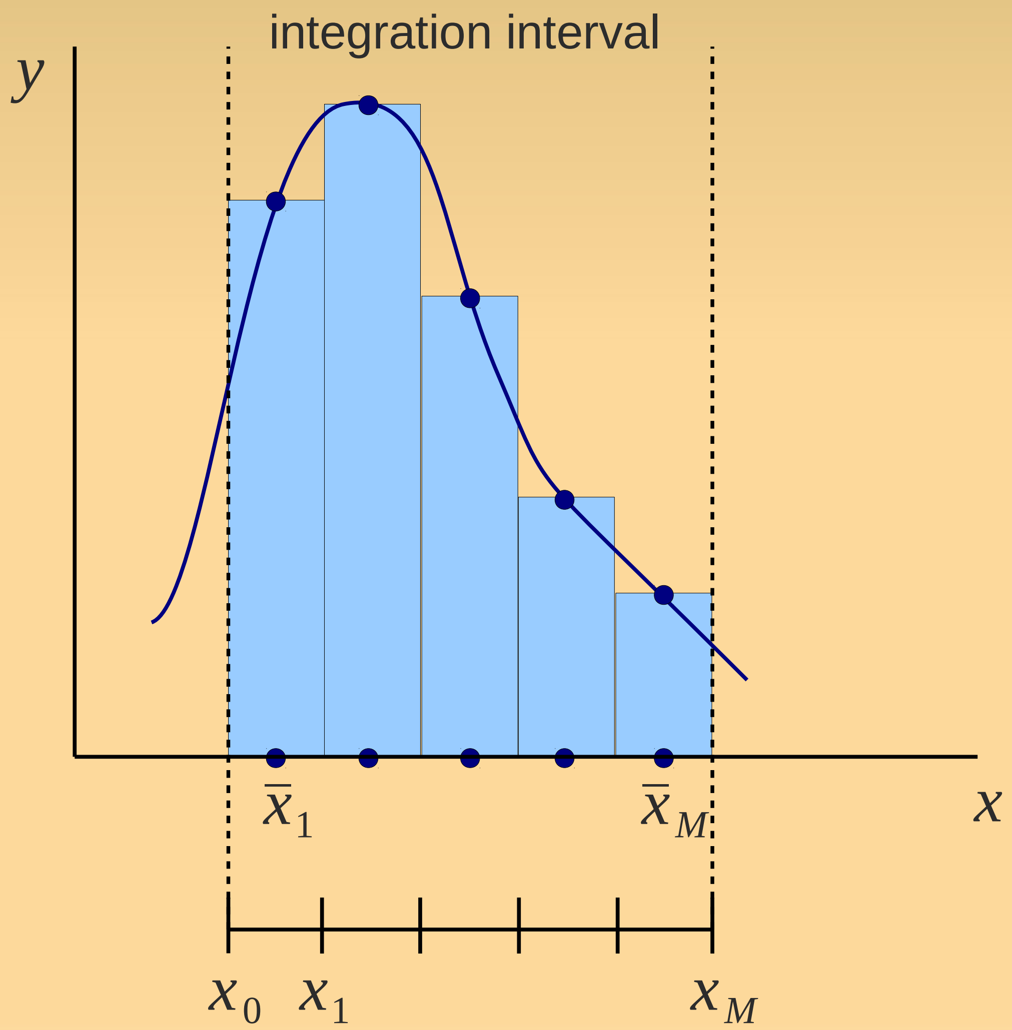
enter highway ramp

How far did we travel?

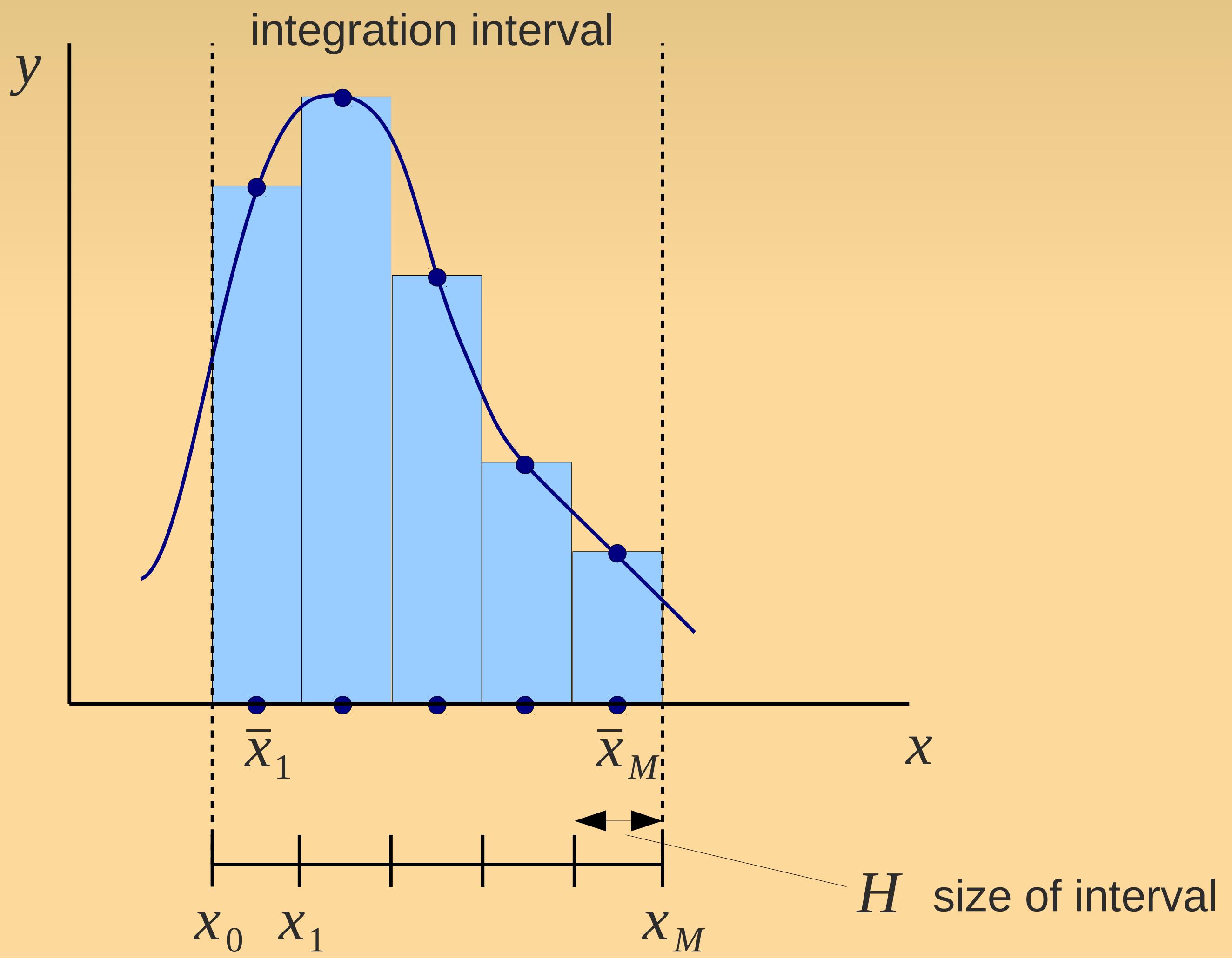


Midpoint Formula

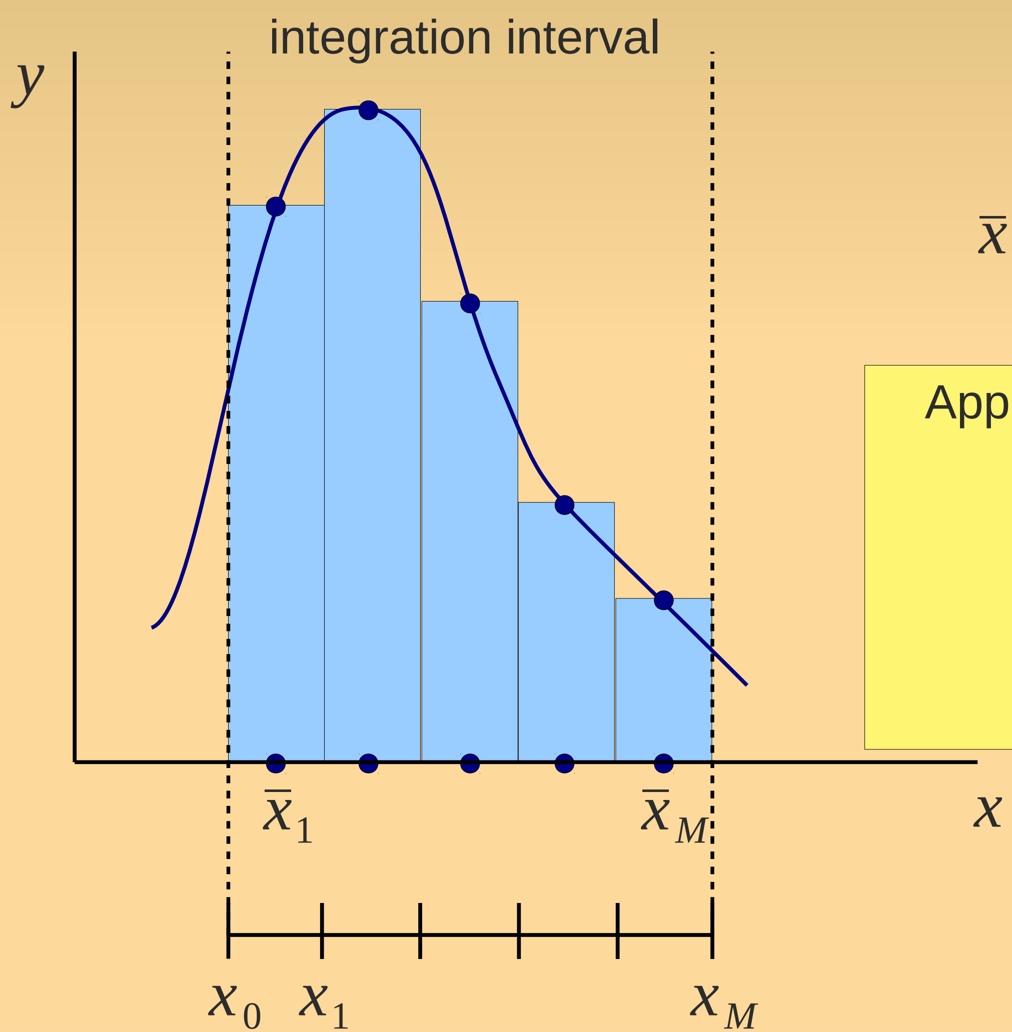
- Approximate the integral with a finite sum



Midpoint Formula



Midpoint Formula

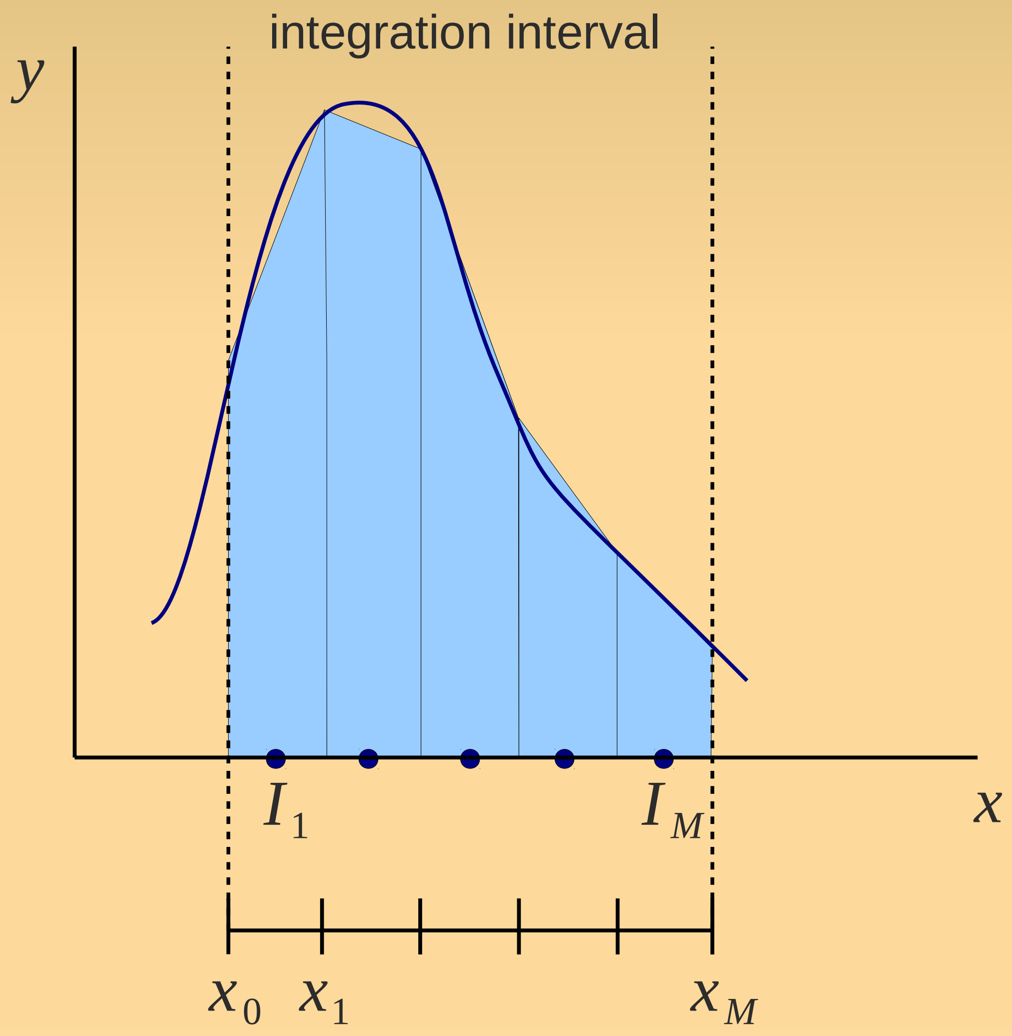


$$\bar{x}_k = \frac{x_{k-1} + x_k}{2}$$

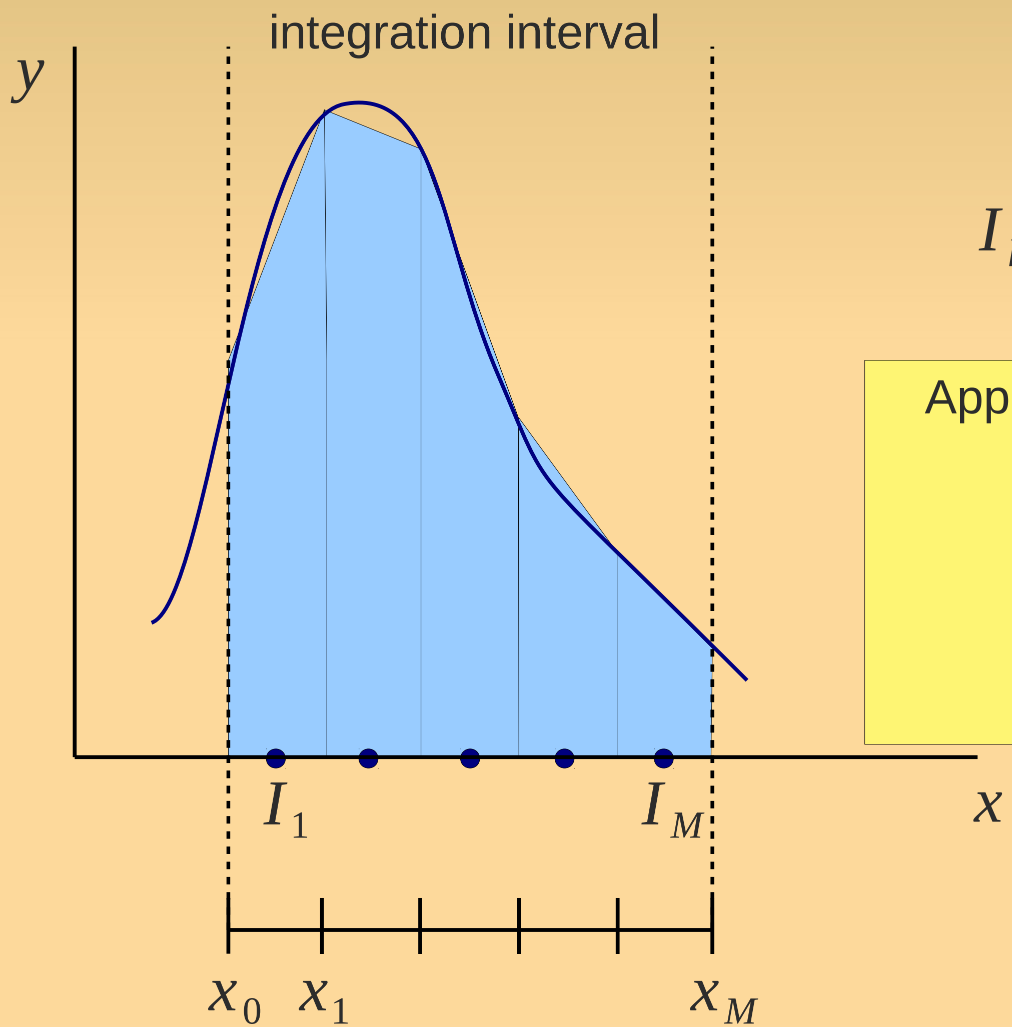
Approximation of the integral:

$$I_{MP}(f) = H \sum_{k=1}^{M} f(\bar{x}_k)$$

Trapezoid Formula



Trapezoid Formula

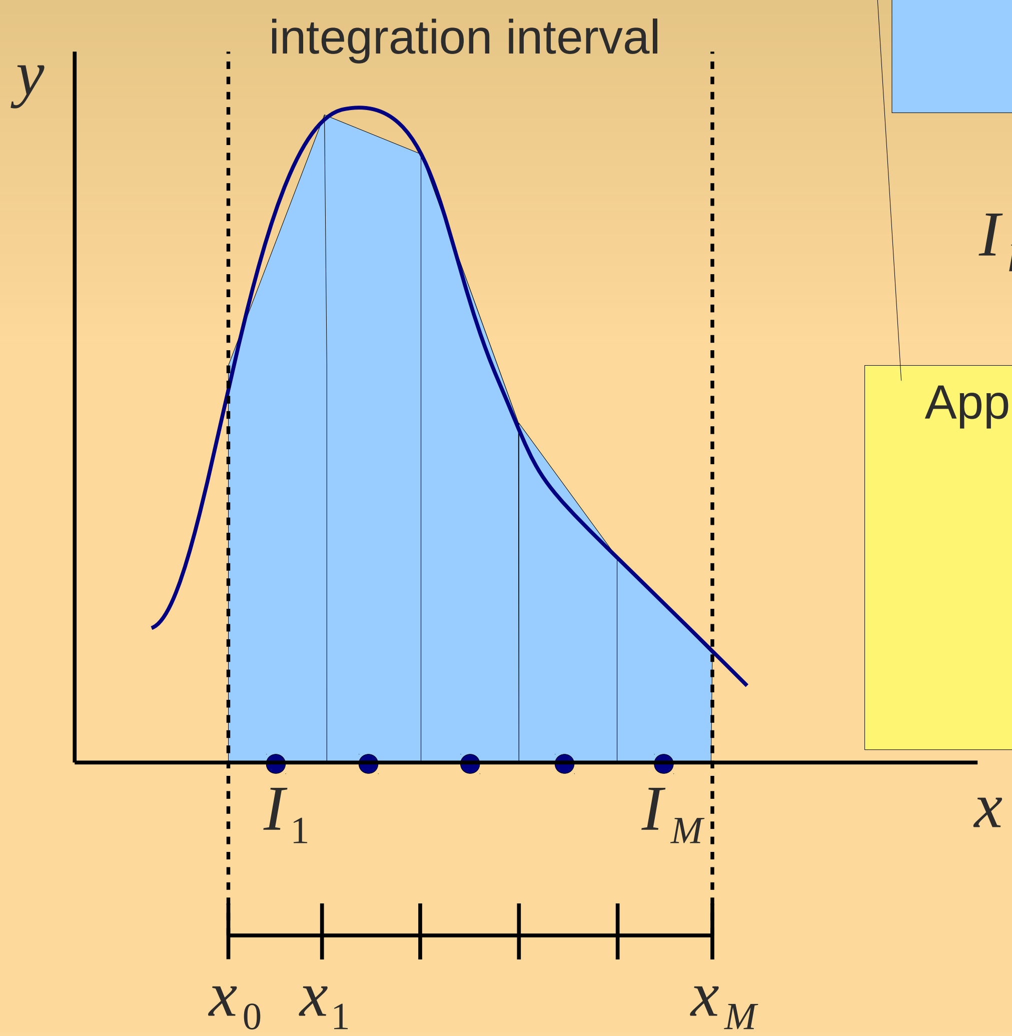


$$I_k = H\,\frac{f(x_{k-1}) + f(x_k)}{2}$$

Approximation of the integral:

$$I_{MP}(f) = \sum_{k=1}^{M} I_k$$

Trapezoid Formula



Possible to avoid some work by using different formula [QSG]

$$I_k = H \frac{f(x_{k-1}) + f(x_k)}{2}$$

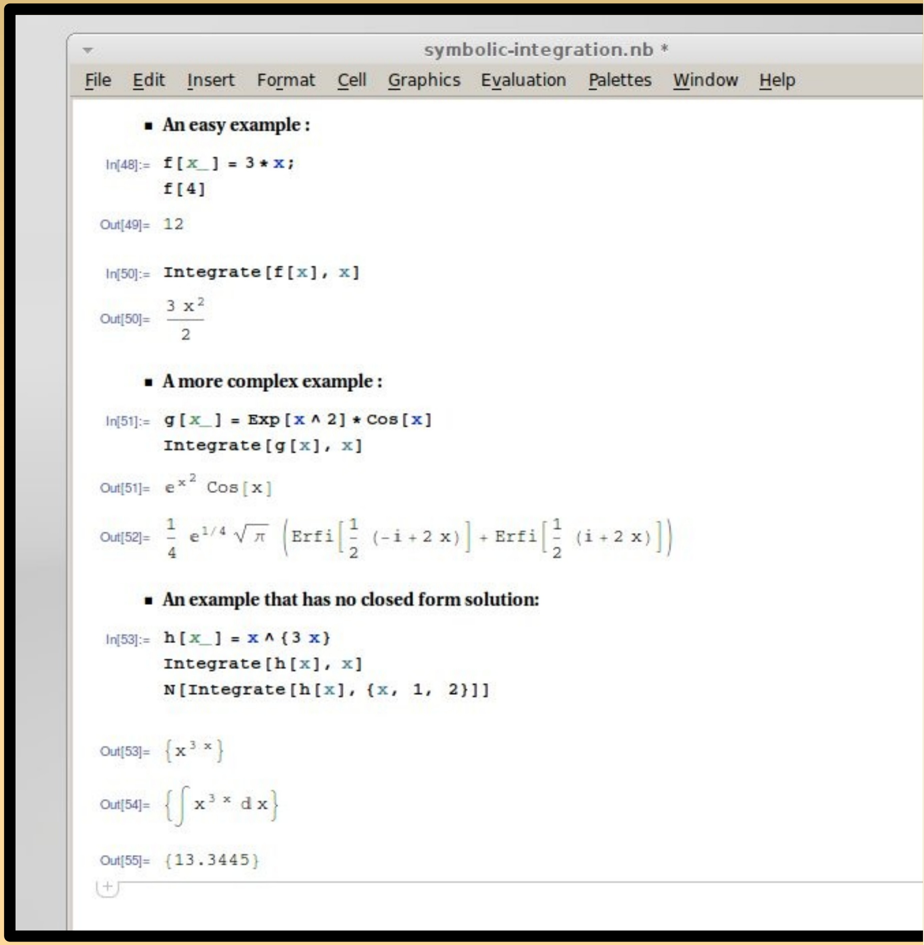
Approximation of the integral:

$$I_{MP}(f) = \sum_{k=1}^{M} I_k$$

Symbolic Integration

- Finally: when faced with a difficult integral...

  → try 'symbolic' packages!



symbolic-integration.nb *

File Edit Insert Format Cell Graphics Evaluation Palettes Window Help

- **An easy example :**

In[48]:=
```
f[x_] = 3 * x;
f[4]
```

Out[49]= 12

In[50]:=
```
Integrate[f[x], x]
```

Out[50]= $\frac{3\,x^2}{2}$

- **A more complex example :**

In[51]:=
```
g[x_] = Exp[x ^ 2] * Cos[x]
Integrate[g[x], x]
```

Out[51]= $e^{x^2}\,\text{Cos}[x]$

Out[52]= $\frac{1}{4}\,e^{1/4}\sqrt{\pi}\left(\text{Erfi}\left[\frac{1}{2}\,(-i+2\,x)\right]+\text{Erfi}\left[\frac{1}{2}\,(i+2\,x)\right]\right)$

- **An example that has no closed form solution:**

In[53]:=
```
h[x_] = x ^ {3 x}
Integrate[h[x], x]
N[Integrate[h[x], {x, 1, 2}]]
```

Out[53]= $\{x^{3\,x}\}$

Out[54]= $\left\{\int x^{3\,x}\,dx\right\}$

Out[55]= {13.3445}

Reading

- PCA – Not in book
  - but: computation of eigenvalues – Ch. 6
- Numerical differentiation / integration
  - Ch. 4 up to 4.4