Introduction to Computational Science

Georgy Gimel'farb & David Welch

(with contributions by Michael J. Dinneen and Alexei Drummond

+ selected materials from slides by Prof. Michael T. Heath)

February 27, 2012

Objectives of the lecture

- Outline the course contents
- Introduce the lecturers and the tutor
- Outline sources of errors in numerical analysis (optional)

COMPSCI 369 Computational Science

Computational science (scientific computing; numerical analysis): Design, analysis, and application of numerical algorithms, solving math problems

- Digital simulation of natural phenomena
- Virtual prototyping of engineering designs

THIS COURSE overviews popular algorithms and modelling techniques to solve problems that are met in a wide range of applications

- Systems of linear equations (matrix decompositions: SVD, PCA)
- Unconstrained and constrained optimisation; least squares
- Finite difference and finite element methods
- Dynamic programming
- Local search; backtracking; branch and bound search
- String algorithms
- Probability models: Markov chains, hidden Markov models

A large number of scientific and engineering questions are answered by combining these computational tools

Your Lecturers

Georgy Gimel'farb

Scientific Computing and Computational Engineering

David Welch

Scientific Computing and Computational Biology

Class Tutor

Andrew Probert

(apro002 @ aucklanduni.ac.nz)

Attending tutorials should be regarded as **essential** *to successful completion for all students, particularly those unfamiliar with the math techniques used in the course*

Problem sets and exercises will be provided to help to cement your understanding of scientific questions underlying examples in the course

Course Administration and Assessment

- Course supervisor:
  `georgy@cs.auckland.ac.nz` (room 303.389)
  [*Office hours: Open door policy*]
- Course organization and assessment:
  - 32 lectures: Part 1 – 16 (Computational Science (Engineering));
    Part 2 – 16 (Computational Science (Biology))
  - 2 assignments (15% each): one for Part 1 and one for Part 2
  - 1 midterm test — April 3rd (during the lecture time)
  - 1 final exam (60%): closed book, no calculators; 3 hours
- Tutorial times: Wednesday 2–3 pm in Science Centre 303S-G75 (starting next week)
  These are **optional** but **recommended**!
- Class Representative - who will be a volunteer?

Recommended Texts

Computational science and engineering:

- **Scientific Computing: An Introductory Survey** by M. T. Heath (McGraw-Hill, 2002)
- **Algorithm Design** by J. Kleinberg and E. Tardos (2006)
- **Computational Science and Engineering** by G. Strang (Wellesley-Cambridge Press, 2007)

Computational biology:

- **Biological sequence analysis** by R. Durbin, S. R. Eddy, A. Krogh and G. Mitchinson (Cambridge University Press, 1998)
- **Bioinformatics and Molecular Evolution** by P. G. Higgs and T. K. Attwood (Blackwell Publ., 2005)
- **An Introduction to Bioinformatics Algorithms** by N. C. Jones and P. A. Pevzner (MIT Press, 2004)

COMPSCI 369 Curriculum in 2012

1. Introduction to computational science and engineering
2. Solving linear systems; SVD; PCA; multilinear models
3. Unconstrained and constrained non-linear optimisation, including also
   - Least squares methods
   - Local search algorithms
   - String matching algorithms
   - Dynamic programming
   - Introduction to backtracking and branch-and-bound
4. Basics of probabilistic modelling and Bayesian inference
   - Maximum likelihood
   - Hidden Markov models
5. Finite difference methods
6. Computational biology and evolution
   - Introduction to sequences and genetic distance
   - Pairwise and multiple sequence alignment
   - Parsimony and molecular evolution models
   - Introduction to phylogenetics

What Is Expected to Be Learned Earlier? (Prerequisites)

STATS 101 – 125:

- Probability models, random walks, Markov chains
- Data analysis, statistical inference, regression

General math (b.t.w., it is quite necessary for studying STATS 101 – 125):

- Linear algebra: vector-matrix operations
- Differentiation / integration of functions
- First/second-order ordinary differential equations

  Basic details, which you are expected to know, will be provided in optional course materials

COMPSCI 220:

- $O(n)$, $\Theta(n)$, $\Omega(n)$ time/space complexity
- Searching and sorting (especially, hash tables and hashing)
- Graphical models (graph algorithms)

Learning Outcomes:

Be familiar with and understand basic numerical methods and models for:

- Finding roots of equations
- Solving systems of linear equations, including
  - Standard matrix decompositions (SVD, QR, LU)
  - Eigen-vectors and principal component analysis (PCA)
- Unconstrained and constrained optimisation
  - Method of Lagrange for a system of equality constraints
  - Unconstrained gradient search for an optimum
  - Dynamic programming, including the edit-distance problem
  - Heuristic strategies when no good exact algorithm is known
- Probabilistic modelling and inference, including
  - Markov chains and hidden Markov models (HMM)
  - Maximum likelihood (ML) and least squares frameworks
- String matching
- Finite difference models and Taylor series

Understand how a wide range of scientific questions in biology and engineering can be answered by combining these methods

A Brief Quiz: Do These Formulas Scare You?

$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$: a vector -column ; $\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}$: an $m \times n$- matrix

$\mathbf{x}^\mathsf{T} = [x_1 \ \dots \ x_n]$ – a transposed vector-column, or a vector-row

$\mathbf{A}^\mathsf{T} = \begin{bmatrix} a_{11} & \dots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \dots & a_{mn} \end{bmatrix}$: a transposed $m \times n$-matrix, i.e. an $n \times m$ matrix

$\mathbf{x}^\mathsf{T}\mathbf{y} = c$ – Dot product of vectors, e.g. $[3\ 2\ 5] \begin{bmatrix} -2 \\ 4 \\ 1 \end{bmatrix} = 7$

$\mathbf{A}\mathbf{x} = \mathbf{y}$ – Matrix-vector product, e.g. $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$f'(x) \equiv \frac{df(x)}{dx}$: the first derivative (e.g. $f(x) = x^n \rightarrow f'(x) = nx^{n-1}$)

A Brief Quiz: Do These Formulas Scare You?

$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$: a vector -column ; $\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}$: an $m \times n$-matrix

$\mathbf{x}^\mathsf{T} = [x_1 \ \dots \ x_n]$ – a transposed vector-column, or a vector-row

$\mathbf{A}^\mathsf{T} = \begin{bmatrix} a_{11} & \dots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \dots & a_{mn} \end{bmatrix}$: a transposed $m \times n$-matrix, i.e. an $n \times m$ matrix

$\mathbf{x}^\mathsf{T}\mathbf{y} = c$ – Dot product of vectors, e.g. $[3\ 2\ 5] \begin{bmatrix} -2 \\ 4 \\ 1 \end{bmatrix} = 7$

$\mathbf{A}\mathbf{x} = \mathbf{y}$ – Matrix-vector product, e.g. $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$f'(x) \equiv \frac{df(x)}{dx}$: the first derivative (e.g. $f(x) = x^n \rightarrow f'(x) = nx^{n-1}$)

A Brief Quiz: Do These Formulas Scare You?

$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$: a vector -column ; $\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}$: an $m \times n$-matrix

$\mathbf{x}^\mathsf{T} = [x_1 \ \dots \ x_n]$ – a transposed vector-column, or a vector-row

$\mathbf{A}^\mathsf{T} = \begin{bmatrix} a_{11} & \dots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \dots & a_{mn} \end{bmatrix}$: a transposed $m \times n$-matrix, i.e. an $n \times m$ matrix

$\mathbf{x}^\mathsf{T}\mathbf{y} = c$ – Dot product of vectors, e.g. $[3\ 2\ 5] \begin{bmatrix} -2 \\ 4 \\ 1 \end{bmatrix} = 7$

$\mathbf{A}\mathbf{x} = \mathbf{y}$ – Matrix-vector product, e.g. $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$f'(x) \equiv \frac{df(x)}{dx}$: the first derivative (e.g. $f(x) = x^n \rightarrow f'(x) = nx^{n-1}$)

A Brief Quiz: Do These Formulas Scare You?

$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$: a vector -column ; $\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}$: an $m \times n$-matrix

$\mathbf{x}^\mathsf{T} = [x_1 \ \dots \ x_n]$ – a transposed vector-column, or a vector-row

$\mathbf{A}^\mathsf{T} = \begin{bmatrix} a_{11} & \dots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \dots & a_{mn} \end{bmatrix}$: a transposed $m \times n$-matrix, i.e. an $n \times m$ matrix

$\mathbf{x}^\mathsf{T}\mathbf{y} = c$ – Dot product of vectors, e.g. $[3\ 2\ 5] \begin{bmatrix} -2 \\ 4 \\ 1 \end{bmatrix} = 7$

$\mathbf{A}\mathbf{x} = \mathbf{y}$ – Matrix-vector product, e.g. $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$f'(x) \equiv \frac{df(x)}{dx}$: the first derivative (e.g. $f(x) = x^n \rightarrow f'(x) = nx^{n-1}$)

A Brief Quiz: Do These Formulas Scare You?

$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$: a vector -column ; $\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}$: an $m \times n$-matrix

$\mathbf{x}^\mathsf{T} = [x_1 \ \dots \ x_n]$ – a transposed vector-column, or a vector-row

$\mathbf{A}^\mathsf{T} = \begin{bmatrix} a_{11} & \dots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \dots & a_{mn} \end{bmatrix}$: a transposed $m \times n$-matrix, i.e. an $n \times m$ matrix

$\mathbf{x}^\mathsf{T}\mathbf{y} = c$ – Dot product of vectors, e.g. $[3\ 2\ 5] \begin{bmatrix} -2 \\ 4 \\ 1 \end{bmatrix} = 7$

$\mathbf{A}\mathbf{x} = \mathbf{y}$ – Matrix-vector product, e.g. $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$f'(x) \equiv \frac{df(x)}{dx}$: the first derivative (e.g. $f(x) = x^n \rightarrow f'(x) = nx^{n-1}$)

A Brief Quiz: Do These Formulas Scare You?

$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$: a vector -column ; $\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}$: an $m \times n$-matrix

$\mathbf{x}^\mathsf{T} = [x_1 \ \dots \ x_n]$ – a transposed vector-column, or a vector-row

$\mathbf{A}^\mathsf{T} = \begin{bmatrix} a_{11} & \dots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \dots & a_{mn} \end{bmatrix}$: a transposed $m \times n$-matrix, i.e. an $n \times m$ matrix

$\mathbf{x}^\mathsf{T}\mathbf{y} = c$ – Dot product of vectors, e.g. $[3\ 2\ 5] \begin{bmatrix} -2 \\ 4 \\ 1 \end{bmatrix} = 7$

$\mathbf{A}\mathbf{x} = \mathbf{y}$ – Matrix-vector product, e.g. $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$f'(x) \equiv \frac{df(x)}{dx}$: the first derivative (e.g. $f(x) = x^n \rightarrow f'(x) = nx^{n-1}$)

A Brief Quiz: Do These Formulas Scare You?

$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$: a vector -column ; $\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}$: an $m \times n$- matrix

$\mathbf{x}^\mathsf{T} = [x_1 \ \dots \ x_n]$ – a transposed vector-column, or a vector-row

$\mathbf{A}^\mathsf{T} = \begin{bmatrix} a_{11} & \dots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \dots & a_{mn} \end{bmatrix}$: a transposed $m \times n$-matrix, i.e. an $n \times m$ matrix

$\mathbf{x}^\mathsf{T}\mathbf{y} = c$ – Dot product of vectors, e.g. $[3\ 2\ 5] \begin{bmatrix} -2 \\ 4 \\ 1 \end{bmatrix} = 7$

$\mathbf{A}\mathbf{x} = \mathbf{y}$ – Matrix-vector product, e.g. $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$f'(x) \equiv \frac{df(x)}{dx}$: the first derivative (e.g. $f(x) = x^n \rightarrow f'(x) = nx^{n-1}$)

A Brief Quiz: Do These Formulas Scare You?

$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$: a vector -column ; $\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}$: an $m \times n$-matrix

$\mathbf{x}^\mathsf{T} = [x_1 \ \dots \ x_n]$ – a transposed vector-column, or a vector-row

$\mathbf{A}^\mathsf{T} = \begin{bmatrix} a_{11} & \dots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \dots & a_{mn} \end{bmatrix}$: a transposed $m \times n$-matrix, i.e. an $n \times m$ matrix

$\mathbf{x}^\mathsf{T}\mathbf{y} = c$ – Dot product of vectors, e.g. $[3\ 2\ 5] \begin{bmatrix} -2 \\ 4 \\ 1 \end{bmatrix} = 7$

$\mathbf{A}\mathbf{x} = \mathbf{y}$ – Matrix-vector product, e.g. $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$f'(x) \equiv \frac{df(x)}{dx}$: the first derivative (e.g. $f(x) = x^n \rightarrow f'(x) = nx^{n-1}$)

A Brief Quiz: Do These Formulas Scare You?

$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$: a vector -column ; $\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}$: an $m \times n$-matrix

$\mathbf{x}^\mathsf{T} = [x_1 \ \dots \ x_n]$ – a transposed vector-column, or a vector-row

$\mathbf{A}^\mathsf{T} = \begin{bmatrix} a_{11} & \dots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \dots & a_{mn} \end{bmatrix}$: a transposed $m \times n$-matrix, i.e. an $n \times m$ matrix

$\mathbf{x}^\mathsf{T}\mathbf{y} = c$ – Dot product of vectors, e.g. $[3\ 2\ 5] \begin{bmatrix} -2 \\ 4 \\ 1 \end{bmatrix} = 7$

$\mathbf{A}\mathbf{x} = \mathbf{y}$ – Matrix-vector product, e.g. $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$f'(x) \equiv \frac{df(x)}{dx}$: the first derivative (e.g. $f(x) = x^n \rightarrow f'(x) = nx^{n-1}$)

A Brief Quiz: Do These Formulas Scare You?

$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$: a vector -column ; $\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}$: an $m \times n$-matrix

$\mathbf{x}^\mathsf{T} = [x_1 \ \dots \ x_n]$ – a transposed vector-column, or a vector-row

$\mathbf{A}^\mathsf{T} = \begin{bmatrix} a_{11} & \dots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \dots & a_{mn} \end{bmatrix}$: a transposed $m \times n$-matrix, i.e. an $n \times m$ matrix

$\mathbf{x}^\mathsf{T}\mathbf{y} = c$ – Dot product of vectors, e.g. $[3\ 2\ 5] \begin{bmatrix} -2 \\ 4 \\ 1 \end{bmatrix} = 7$

$\mathbf{A}\mathbf{x} = \mathbf{y}$ – Matrix-vector product, e.g. $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$f'(x) \equiv \frac{df(x)}{dx}$: the first derivative (e.g. $f(x) = x^n \rightarrow f'(x) = nx^{n-1}$)

A Brief Quiz: Do These Formulas Scare You?

$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$: a vector -column ; $\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}$: an $m \times n$- matrix

$\mathbf{x}^\mathsf{T} = [x_1 \ \dots \ x_n]$ – a transposed vector-column, or a vector-row

$\mathbf{A}^\mathsf{T} = \begin{bmatrix} a_{11} & \dots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \dots & a_{mn} \end{bmatrix}$: a transposed $m \times n$-matrix, i.e. an $n \times m$ matrix

$\mathbf{x}^\mathsf{T}\mathbf{y} = c$ – Dot product of vectors, e.g. $[3\ 2\ 5] \begin{bmatrix} -2 \\ 4 \\ 1 \end{bmatrix} = 7$

$\mathbf{A}\mathbf{x} = \mathbf{y}$ – Matrix-vector product, e.g. $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$f'(x) \equiv \frac{df(x)}{dx}$: the first derivative (e.g. $f(x) = x^n \rightarrow f'(x) = nx^{n-1}$)

A Brief Quiz: Do These Formulas Scare You?

$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$: a vector -column ; $\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}$: an $m \times n$-matrix

$\mathbf{x}^\mathsf{T} = [x_1 \ \dots \ x_n]$ – a transposed vector-column, or a vector-row

$\mathbf{A}^\mathsf{T} = \begin{bmatrix} a_{11} & \dots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \dots & a_{mn} \end{bmatrix}$: a transposed $m \times n$-matrix, i.e. an $n \times m$ matrix

$\mathbf{x}^\mathsf{T}\mathbf{y} = c$ – Dot product of vectors, e.g. $[3\ 2\ 5] \begin{bmatrix} -2 \\ 4 \\ 1 \end{bmatrix} = 7$

$\mathbf{A}\mathbf{x} = \mathbf{y}$ – Matrix-vector product, e.g. $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$f'(x) \equiv \frac{df(x)}{dx}$: the first derivative (e.g. $f(x) = x^n \rightarrow f'(x) = nx^{n-1}$)

A Brief Quiz: Do These Formulas Scare You?

$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$: a vector -column ; $\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}$: an $m \times n$-matrix

$\mathbf{x}^\mathsf{T} = [x_1 \ \dots \ x_n]$ – a transposed vector-column, or a vector-row

$\mathbf{A}^\mathsf{T} = \begin{bmatrix} a_{11} & \dots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \dots & a_{mn} \end{bmatrix}$: a transposed $m \times n$-matrix, i.e. an $n \times m$ matrix

$\mathbf{x}^\mathsf{T}\mathbf{y} = c$ – Dot product of vectors, e.g. $[3\ 2\ 5] \begin{bmatrix} -2 \\ 4 \\ 1 \end{bmatrix} = 7$

$\mathbf{A}\mathbf{x} = \mathbf{y}$ – Matrix-vector product, e.g. $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$f'(x) \equiv \frac{df(x)}{dx}$: the first derivative (e.g. $f(x) = x^n \rightarrow f'(x) = nx^{n-1}$)

A Brief Quiz: Do These Formulas Scare You?

$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$: a vector -column ; $\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}$: an $m \times n$-matrix

$\mathbf{x}^\mathsf{T} = [x_1 \ \dots \ x_n]$ – a transposed vector-column, or a vector-row

$\mathbf{A}^\mathsf{T} = \begin{bmatrix} a_{11} & \dots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \dots & a_{mn} \end{bmatrix}$: a transposed $m \times n$-matrix, i.e. an $n \times m$ matrix

$\mathbf{x}^\mathsf{T}\mathbf{y} = c$ – Dot product of vectors, e.g. $[3\ 2\ 5] \begin{bmatrix} -2 \\ 4 \\ 1 \end{bmatrix} = 7$

$\mathbf{A}\mathbf{x} = \mathbf{y}$ – Matrix-vector product, e.g. $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$f'(x) \equiv \frac{df(x)}{dx}$: the first derivative (e.g. $f(x) = x^n \rightarrow f'(x) = nx^{n-1}$)

A Brief Quiz: Do These Formulas Scare You?

$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$: a vector -column ; $\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}$: an $m \times n$-matrix

$\mathbf{x}^\mathsf{T} = [x_1 \ \dots \ x_n]$ – a transposed vector-column, or a vector-row

$\mathbf{A}^\mathsf{T} = \begin{bmatrix} a_{11} & \dots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \dots & a_{mn} \end{bmatrix}$: a transposed $m \times n$-matrix, i.e. an $n \times m$ matrix

$\mathbf{x}^\mathsf{T}\mathbf{y} = c$ – Dot product of vectors, e.g. $[3\ 2\ 5] \begin{bmatrix} -2 \\ 4 \\ 1 \end{bmatrix} = 7$

$\mathbf{A}\mathbf{x} = \mathbf{y}$ – Matrix-vector product, e.g. $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$f'(x) \equiv \frac{df(x)}{dx}$: the first derivative (e.g. $f(x) = x^n \rightarrow f'(x) = nx^{n-1}$)

A Brief Quiz: Do These Formulas Scare You?

$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$: a vector -column ; $\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}$: an $m \times n$-matrix

$\mathbf{x}^\mathsf{T} = [x_1 \ \dots \ x_n]$ – a transposed vector-column, or a vector-row

$\mathbf{A}^\mathsf{T} = \begin{bmatrix} a_{11} & \dots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \dots & a_{mn} \end{bmatrix}$: a transposed $m \times n$-matrix, i.e. an $n \times m$ matrix

$\mathbf{x}^\mathsf{T}\mathbf{y} = c$ – Dot product of vectors, e.g. $[3\ 2\ 5] \begin{bmatrix} -2 \\ 4 \\ 1 \end{bmatrix} = 7$

$\mathbf{A}\mathbf{x} = \mathbf{y}$ – Matrix-vector product, e.g. $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$f'(x) \equiv \frac{df(x)}{dx}$: the first derivative (e.g. $f(x) = x^n \rightarrow f'(x) = nx^{n-1}$)

A Brief Quiz: Do These Formulas Scare You?

$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$: a vector -column ; $\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}$: an $m \times n$- matrix

$\mathbf{x}^\mathsf{T} = [x_1 \ \dots \ x_n]$ – a transposed vector-column, or a vector-row

$\mathbf{A}^\mathsf{T} = \begin{bmatrix} a_{11} & \dots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \dots & a_{mn} \end{bmatrix}$: a transposed $m \times n$-matrix, i.e. an $n \times m$ matrix

$\mathbf{x}^\mathsf{T}\mathbf{y} = c$ – Dot product of vectors, e.g. $[3\ 2\ 5] \begin{bmatrix} -2 \\ 4 \\ 1 \end{bmatrix} = 7$

$\mathbf{A}\mathbf{x} = \mathbf{y}$ – Matrix-vector product, e.g. $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$f'(x) \equiv \frac{df(x)}{dx}$: the first derivative (e.g. $f(x) = x^n \rightarrow f'(x) = nx^{n-1}$)

A Brief Quiz: Do These Formulas Scare You?

$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$: a vector -column ; $\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}$: an $m \times n$-matrix

$\mathbf{x}^\mathsf{T} = [x_1 \ \dots \ x_n]$ – a transposed vector-column, or a vector-row

$\mathbf{A}^\mathsf{T} = \begin{bmatrix} a_{11} & \dots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \dots & a_{mn} \end{bmatrix}$: a transposed $m \times n$-matrix, i.e. an $n \times m$ matrix

$\mathbf{x}^\mathsf{T}\mathbf{y} = c$ – Dot product of vectors, e.g. $[3\ 2\ 5] \begin{bmatrix} -2 \\ 4 \\ 1 \end{bmatrix} = 7$

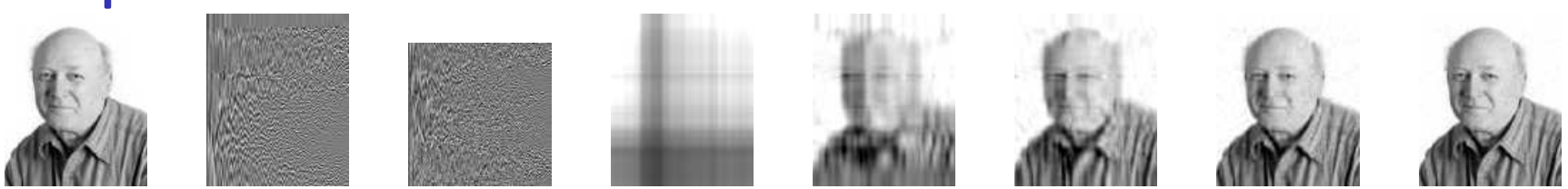
$\mathbf{A}\mathbf{x} = \mathbf{y}$ – Matrix-vector product, e.g. $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$

$f'(x) \equiv \frac{df(x)}{dx}$: the first derivative (e.g. $f(x) = x^n \rightarrow f'(x) = nx^{n-1}$)

What Might You Expect in Assignment 1?

- Take your own grayscale digital image 120×100
- Consider it as a 120×100 matrix $\mathbf{Y}_{\text{ou}}$
- Perform the SVD (singular value decomposition): $\mathbf{Y}_{\text{ou}} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathsf{T}}$
  (C-subroutines of SVD and image i/o in pgm format will be provided)
- Approximate $\mathbf{Y}_{\text{ou}}$ with ρ singular values and columns of $\mathbf{U}$ and $\mathbf{V}$; $1 \leq \rho \leq 100$
- Assess the approximations for different ρ and decide how much this image can be compressed with retention of its visual quality

Example:



$\mathbf{M}_{\text{e}}$	$\mathbf{U}$	$\mathbf{V}$	$\rho = 1$	$\rho = 5$	$\rho = 10$	$\rho = 20$	$\rho = 30$
	Compression (%):		98.2	90.8	81.6	63.2	44.7

Trying to Deal with a Problem or Stressful Situation?

Personal Support for Computer Science Students

Radu Nicolescu
Tamaki Campus : 731.332
Ext: 86831
E-mail: r.nicolescu@auckland.ac.nz

Ann Cameron
Room: 303S.594
Ext: 84947
E-mail: ann@cs.auckland.ac.nz

Pat Riddle
Room: 303S.392
Ext: 87093
Email: pat@cs.auckland.ac.nz

Need to talk to someone?

We are here to listen & help!

Come and talk to us.

Paul Denny
Room: 303S.465
Ext: 87087
Email: paul@cs.auckland.ac.nz

Angela Chang
Room 303S.585
Ext: 86620
Email: angela@cs.auckland.ac.nz

Adriana Ferraro
Room: 303S.592
Ext: 87113
Email: adriana@cs.auckland.ac.nz

Andrew Luxton-Reilly.
Room: 303S.479
Ext: 85654
Email: andrew@cs.auckland.ac.nz

Patricia Rood
Room: 303S.379
Ext: 85720
Email: p.rood@auckland.ac.nz

Computation + Science (optional)

In the 21st century almost all science is computational. **Why**?

- Computation: the information processing in the human brain, a digital computer, or elsewhere
  - For certain well-defined algorithms this processing can be done a lot faster in a computer
- Science: the effort to expand our understanding of the world around us and discover the underpinnings of physical reality
  - It is an attempt to ask the question: how do things work?
- The most successful approach to science: a combination of

1. observation
2. modelling
3. inference or estimation
4. prediction, and
5. experimentation

Mathematical Modelling

An attempt to describe essential components of a system in order to better understand it

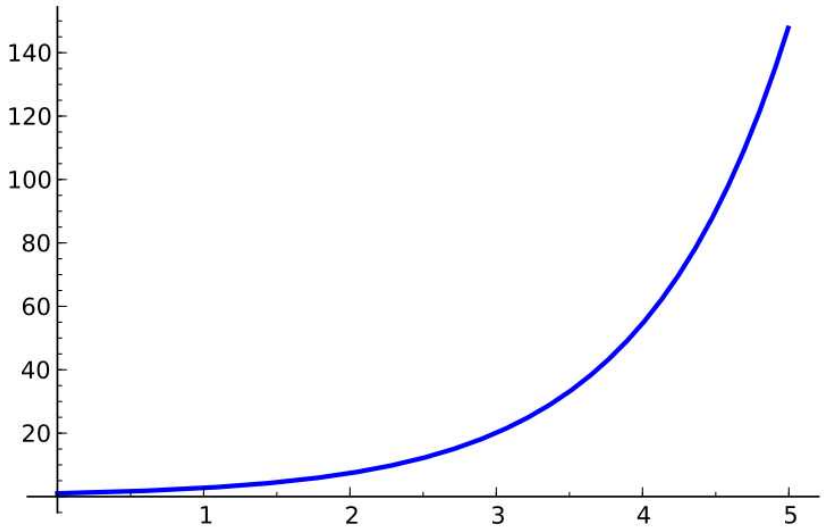
† *A model is worthless if it doesn't produce some insight beyond what was available from direct observation*

- The observed data, together with the scientific question at hand, suggest the essential features of the math model
- Together with the observed data, the model can be used to estimate parameter values and infer unobserved phenomena
- The mathematical model often provides predictions of future outcomes that can be tested experimentally
- Experimental observation can evaluate the accuracy of the model and suggest refinements

Examples of Linear and Non-linear Models

Exponential growth (Malthus, 1798)

$$N(t) = N_0 e^{rt}$$

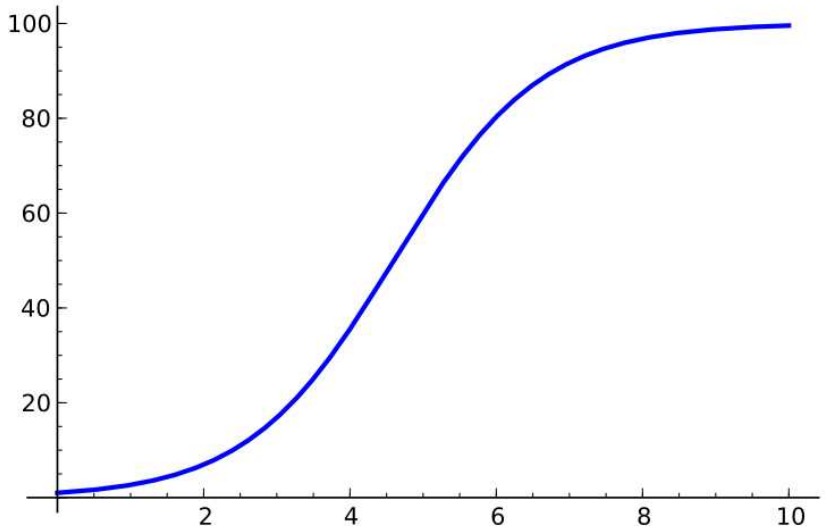


Can be rewritten as a linear model:

$$\underbrace{\phi(t)}_{\log N(t)} = \underbrace{\phi_0}_{\log N_0} + rt$$

Logistic growth (Verhulst, 1838)

$$N(t) = \frac{K N_0 e^{rt}}{K + N_0(e^{rt} - 1)}$$



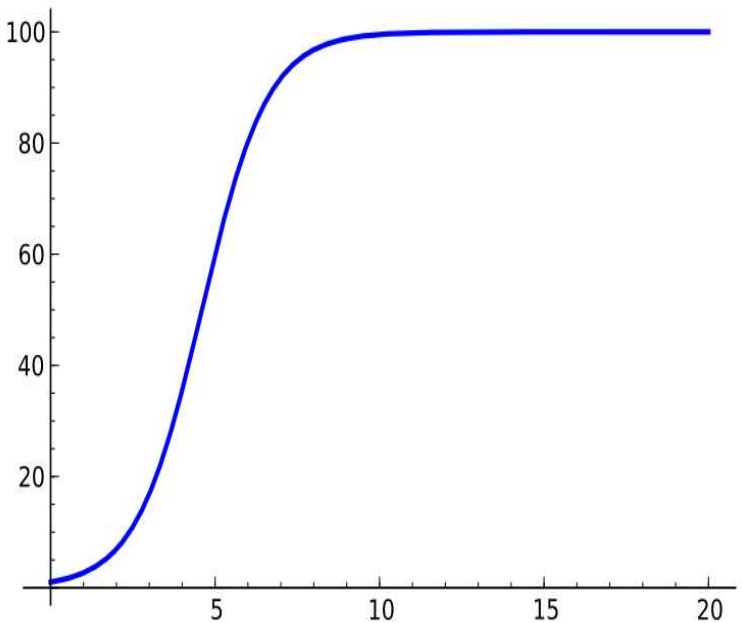
No such simple linearisation...

Examples of Deterministic and Stochastic Models

Deterministic logistic growth

Just the same model as on Slide 16, but written as a differential equation:

$$\frac{dN}{dt} = rN\left(1 - \frac{N}{K}\right)$$



Stochastic logistic growth of the population size N_i; $0 \leq N_i \leq K$

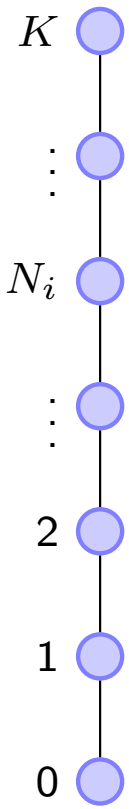
1D biased random walk to model each i-th birth or death event:

$$\begin{cases} \Pr(N_{i+1} = N_i - 1) = p_i \\ \Pr(N_{i+1} = N_i + 1) = 1 - p_i \end{cases}$$

where $p_i = \frac{\mu}{\mu + \lambda(K - N_i)}$ and N_0 is the initial population size

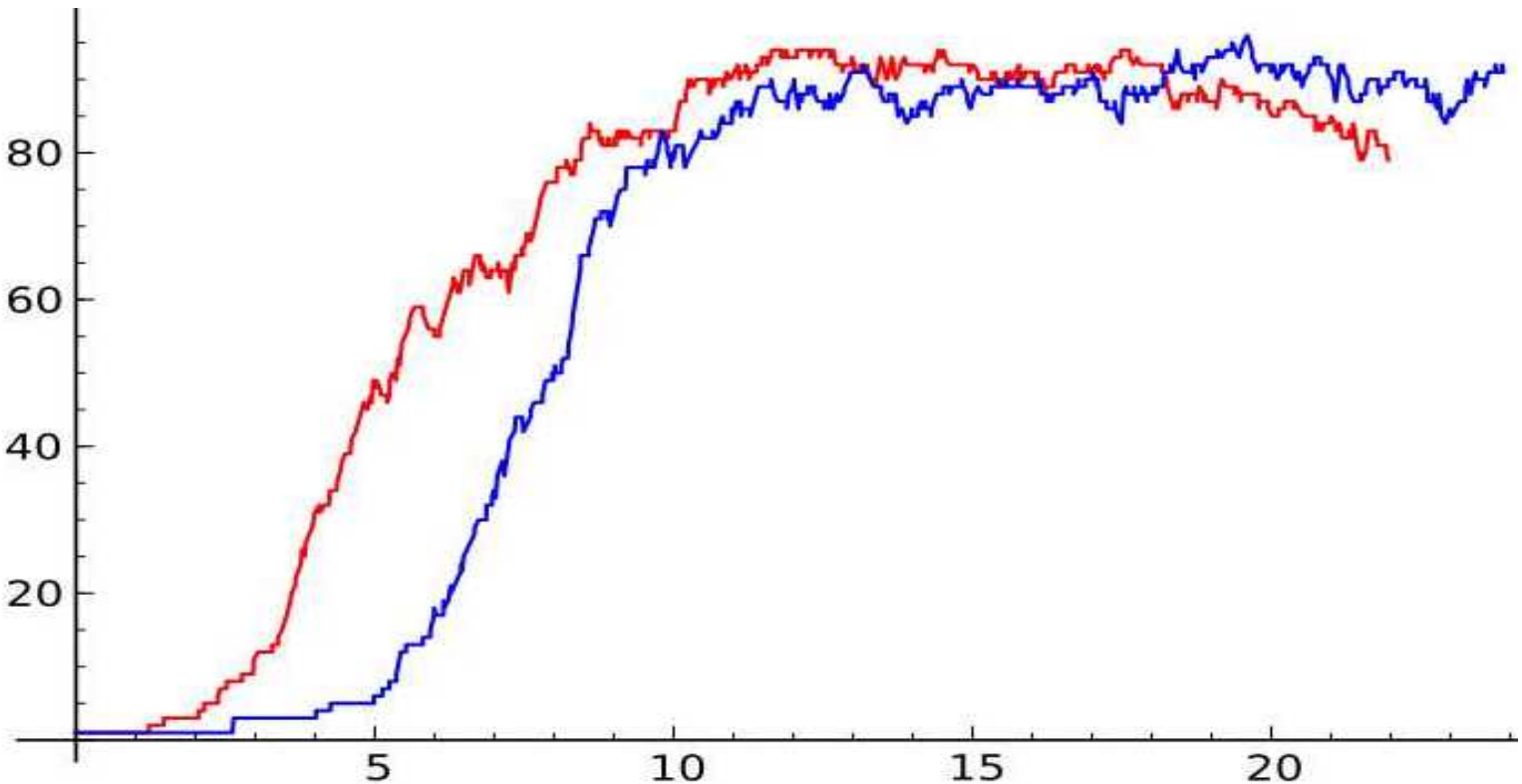
The time t_i of the i-th event:

$$t_{i+1} \approx t_i + \exp\left(\frac{p_i}{\mu}\right);\ t_0 = 0$$



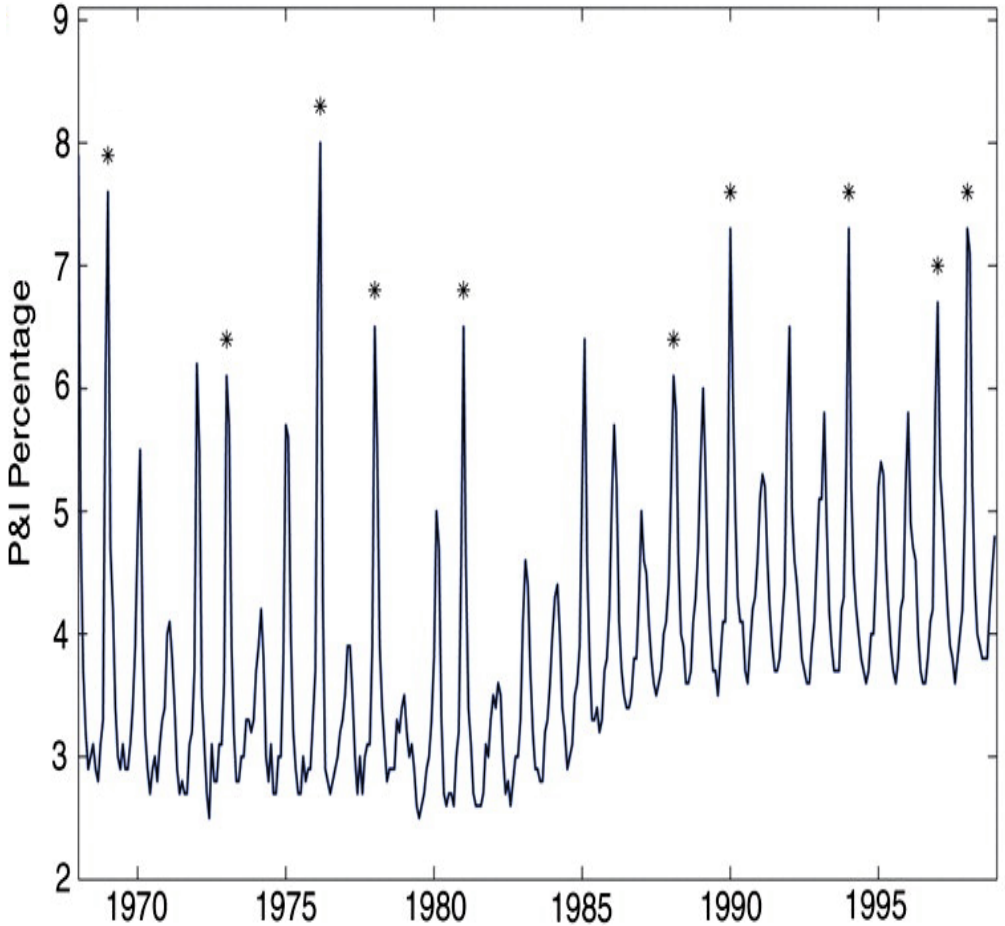
Stochastic Logistic Growth

- Process is characterised by a probability distribution over population trajectories
- New behaviour: Population extinction due to no birth if the last individual dies
- Early chance events can have a disproportionately large downstream effect

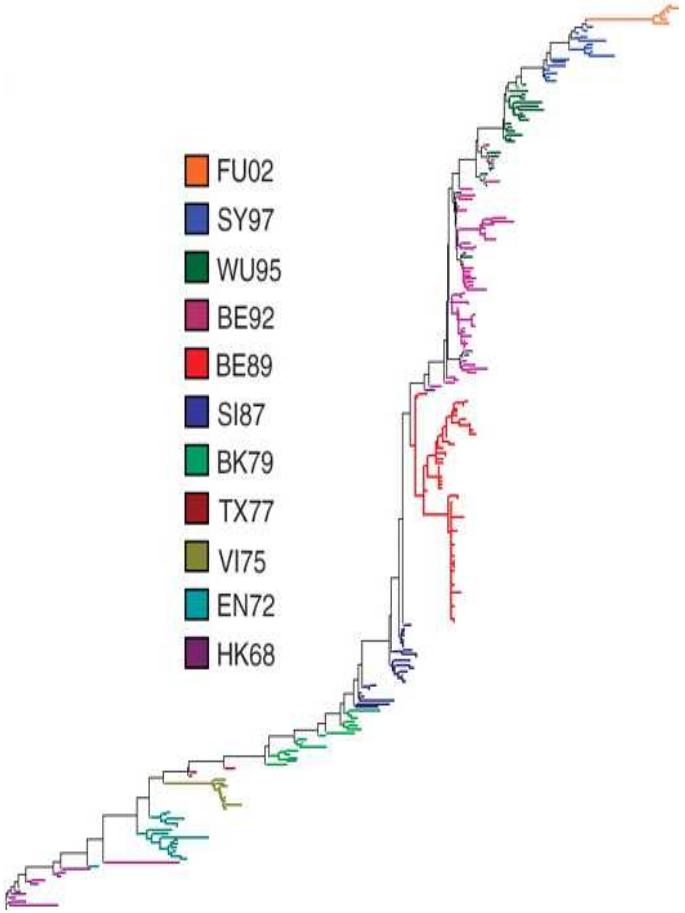


Continuous and Combinatorial Models

Index of death rates for people aged > 65 attributed to Pneumonia and Influenza over 31 years



Estimated "phylogenetic tree" of a sample of human influenza virus sequences collected over 34 years



From Mathematical to Computational Models

Two main reasons:

1. Too much data
   - Dramatic growth of the amount of digital data (observations, experiments, etc.)
   - Quantities too large for direct analysis by a human brain
2. Availability of powerful fast computers
   - Possible use of models being too complex to analyze directly with traditional mathematical tools
   - Possible release of scientists from the shackles of mathematical convenience
     - A push towards more realistic models, but often with the sacrifice of closed-form mathematical solutions

Numerical Computations: Ill- and Well-Posed Problems

- Well-posed problem (http://en.wikipedia.org/wiki/...) if (*i*) its solution exists, (*ii*) is unique, and (*iii*) depends continuously on problem data
- Otherwise, problem is ill-posed
- Solution of well-posed problem may still be sensitive to input data
  - Computational algorithm should not make sensitivity worse

Well-posed problem	Ill-posed problem
Multiplication by a small number a: $ax = y$	Division by a small number: $x = \frac{y}{a} \equiv a^{-1}y$ $(a \ll 1)$
Multiplication by a matrix $\mathbf{A}$: $\mathbf{Ax} = \mathbf{y}$	Inversion $\mathbf{x} = \mathbf{A}^{-1}\mathbf{y}$ for ill-conditioned, $\det(\mathbf{A}) \ll 1$; singular, $\det(\mathbf{A}) = 0$; or rectangular, $m \times n$ $(m \neq n)$, matrix $\mathbf{A}$
Integration $y(t) = y(0) + \int_0^t x(\xi)d\xi$	Differentiation $x(t) = y'(t) \equiv \frac{d}{dt}y(t)$

S.I.Kabanikhin: Definitions and examples of inverse and ill-posed problems, *J. Inv. Ill-Posed Problems*, vol.16, pp.317–357,2008

Numerical Computations: Sensitivity and Conditioning

- Insensitive, or well-conditioned problem:
  if relative change in input x causes similar relative change in solution y
- Sensitive, or ill-conditioned problem:
  if relative change in solution y can be much larger than in input data x
- Condition number:

$$\text{cond} = \frac{|\text{relative change in solution}|}{|\text{relative change in input data}|} \equiv \frac{|\Delta y/y|}{|\Delta x/x|}$$

- Problem is sensitive, or ill-conditioned, if $\text{cond} \gg 1$

Sensitivity and Conditioning: An Example

- Evaluating function $f : \mathbb{R} \to \mathbb{R}$ for approximate, $\widehat{x} = x + \Delta x$, instead of true, x, input:

$$\text{cond} = \left| \frac{(f(x+\Delta x)-f(x))/f(x)}{\Delta x/x} \right|$$

$$\approx \left| \frac{f'(x)\Delta x/f(x)}{\Delta x/x} \right| = \left| \frac{xf'(x)}{f(x)} \right|$$

- Relative error in function value can be much larger or smaller than that in input, depending on particular f and x
  - $f(x) = x^n$; $f'(x) = nx^{n-1} \implies \text{cond} = \left| \frac{x \cdot nx^{n-1}}{x^n} \right| = n$
  - $f(x) = \mathrm{e}^x$; $f'(x) = \mathrm{e}^x \implies \text{cond} = \left| \frac{x\mathrm{e}^x}{\mathrm{e}^x} \right| = x$

Algorithm Stability and Accuracy

Stable algorithm: if result is relatively insensitive to perturbations *during* computation

- Stability of algorithms is similar to conditioning of problems
- Computational error for stable algorithm is no worse than small input data error

Accuracy of algorithm: closeness of computed solution to true solution of problem

- Stability alone does not guarantee accurate results
- Accuracy depends on conditioning of problem and stability of algorithm
- **Inaccurate solution**: from applying (*i*) stable algorithm to ill-conditioned problem or (*ii*) unstable one to well-conditioned problem
- **Accurate solution**: from applying stable algorithm to well-conditioned problem

General Strategy of Numerical Solution

- Replacing difficult (complicated) problem by easier (simplified) one with the same or closely related solution:

  Infinite $\rightarrow$ Finite
  Differential $\rightarrow$ Algebraic
  Non-linear $\rightarrow$ Linear
  ...

- Solution obtained may only approximate that of original problem
  - Approximation before computation (input, or data errors)
    - Modelling; empirical measurements; previous computations
  - Approximation during computation (computational errors)
    - Truncation, or discretisation; rounding
  - Accuracy of final result reflects all these errors
    - Uncertainty in input may be amplified by problem
    - Computational errors may be amplified by algorithm

Example: Computing Earth's Surface Area: $A = 4\pi r^2$

- **Approximations:**
  - Idealised Earth shape model: a sphere
  - Value for radius r by empirical measurements and previous computations
  - Truncated value for π $(= 3.1415926536\ldots)$
  - Rounded in computer values for input data and results of arithmetic operations
- Absolute error $=$ approximate value $-$ true value
- Relative error $= \dfrac{\text{absolute error}}{\text{true value}}$
  - True value is usually unknown: error estimate or bound is used
  - Relative error - usually, relative to approximate, rather than (unknown) true value

Data Error and Computational Error

Problem: compute value of function $f : \mathbb{R} \to \mathbb{R}$ for given argument

x : true input value	$f(x)$: desired result
$\widehat{x}$: approximate (inexact) input	$\widehat{f}$: approximate function actually computed

Total error: $\widehat{f}(\widehat{x}) - f(x) = \underbrace{\widehat{f}(\widehat{x}) - f(\widehat{x})}_{\text{computational error}} + \underbrace{f(\widehat{x}) - f(x)}_{\text{propagated data error}}$

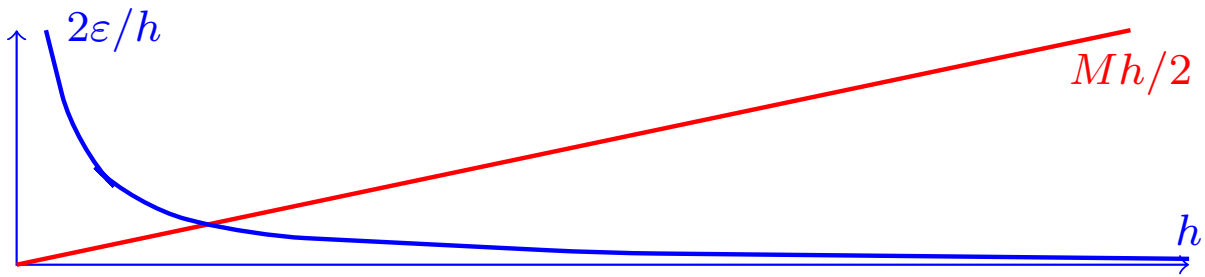
- Algorithm has no effect on propagated data error
- Computational error: sum of **truncation** and **rounding** errors
- One of these two error types usually dominates
  - **Truncation error**: difference between true result $f(\widehat{x})$ for actual input and result of given algorithm using exact arithmetic
    - Truncated infinite series; iterations terminated before convergence...
  - **Rounding error**: difference between results of given algorithm using exact arithmetic and using limited precision arithmetic
    - Inexact representation of real numbers and operations upon them

Example: Finite Difference Approximation

- Finite difference approximation of first derivative

$$f'(x) \approx \frac{f(x+h) - f(x)}{h};\ h > 0$$

- Truncation error bound: $Mh/2$, where M bounds $|f''(t)|_{t \text{ near } x}$
- Rounding error bound: $2\varepsilon/h$, where ε bounds error on function values
- Minimum total error when $h \approx 2\sqrt{\varepsilon/M}$
  - Total error increases for smaller h due to rounding error and for larger h due to truncation error



Floating-Point Numbers in Computers (optional)

$$x = \pm \left(d_0 + \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \ldots + \frac{d^{p-1}}{\beta^{p-1}} \right) \beta^E$$

β base, or radix; $0 \leq d_i \leq \beta - 1;\ i = 0, 1, \ldots, p-1$
$\beta = 2$ for binary computer arithmetics) :

$$x = \pm \left(d_0 + \frac{d_1}{2} + \frac{d_2}{2^2} + \ldots + \frac{d^{p-1}}{2^{p-1}} \right) 2^E$$

p precision : 24 (IEEE Single Precision)
53 (IEEE Double Precision)

E exponent $-126 \leq E \leq 127$ (IEEE Single Precision)
$-1022 \leq E \leq 1023$ (IEEE Double Precision)

Sign, $\pm$, exponent, E, and mantissa, $d_0 d_1 \ldots d_{p-1}$, are stored in separate fixed-width *fields* of each floating-point *word*

Floating-Point Numbers in Computers, continued

Normalised floating-point system: leading digit d_0 is always nonzero unless number represented is zero

- Normalised binary system: mantissa m of nonzero floating-point number always satisfies $1 \leq m < 2$
- Unique representation of each number; leading bit need not be stored
- $254 \cdot 2^{24} + 1$ single precision floating-point numbers
- $2026 \cdot 2^{53} + 1$ double precision floating-point numbers

Rounding: if real number x is not exactly representable, then it is approximated by the nearest machine number $\mathrm{fl}(x)$

Accuracy of Floating-Point System

Characterised by unit roundoff (machine precision): $\varepsilon_{\text{mach}} = 2^{-p}$

$\varepsilon_{\text{mach}} = 2^{-24} \approx 10^{-7}$ IEEE Single Precision (7 decimal digits)
$\varepsilon_{\text{mach}} = 2^{-53} \approx 10^{-16}$ IEEE Double Precision (16 decimal digits)

- Maximum relative error in representing real x: $\left|\frac{\text{fl}(x)-x}{x}\right| \leq \varepsilon_{\text{mach}}$
- Smallest positive normalised floating-point number:

$$\text{UFL} = 2^{-126} \text{ (SP) or } 2^{-1022} \text{ (DP)}$$

- Largest floating-point number:

$$\text{OFL} = 2^{128} - 2^{104} \text{ (SP) or } 2^{1024} - 2^{971} \text{ (DP)}$$

Accuracy of Floating-Point System, continued

Exceptional situations (through special reserved exponent values):

- Inf – "infinity" (results from dividing a finite number by zero; e.g. $1/0$)
- NaN – "not a number" (results from undefined or indeterminate operations such as $0/0$, $0 \cdot \mathrm{Inf}$, or $\mathrm{Inf}/\mathrm{Inf}$

Floating-point arithmetic – result of operation may differ from what corresponding real arithmetic operation produces on same operands:

SP: $x = \frac{1}{3} = 0.3333333$; $y = 3$

$\Rightarrow xy = 0.9999999$, rather than 1.0

Floating-Point Arithmetic

- **Addition** or **subtraction**:
  Shifting of mantissa to make exponents match may cause loss of some digits of smaller number (possibly, all of them)
- **Multiplication**:
  Product of two p-digit mantissas contains up to $2p$ digits, so result may not be representable
- **Division**:
  Quotient of two p-digit mantissas may contain more than p digits, e.g. non-terminating binary expansion of $1/10$
- Real result may fail to be representable if its exponent is beyond available range
- Overflow is usually fatal, but underflow is silently set to zero

Floating-Point Arithmetic, continued

Ideally, $x \text{ flop } y = \text{fl}(x \text{ op } y)$, i.e. floating-point arithmetic operations produce correctly rounded results

- IEEE standard achieves this ideal if $x \text{ op } y$ is within the range of numbers
- Laws of real arithmetic may not be valid
  - Example: Addition and multiplication are now commutative but **not** associative: if $\epsilon < \varepsilon_{\text{mach}} < 2\epsilon$, then $(1 + \epsilon) + \epsilon = 1$, but $1 + (\epsilon + \epsilon) > 1$

Cancellation:

Subtraction between two p-digit numbers having same sign and similar magnitudes yield result with *fewer* than p digits because leading digits of two numbers cancel (i.e. have zero difference)

Floating-Point Arithmetic, continued

Cancellation:
If $\epsilon < \varepsilon_{\text{mach}} < 2\epsilon$, then $(1+\epsilon) - (1-\epsilon) = 1 - 1 = 0$

- It is correct for actual operands of final subtraction, but true result, 2ϵ, has been completely lost
- Digits lost to cancellation are *most* significant, leading digits
- Digits lost in rounding are *least* significant, trailing digits

Do not compute any small quantity as difference of large quantities!

Example: Solutions of quadratic equation $ax^2 + bx + c = 0$

Floating-Point Arithmetic, continued

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (*)$$

- Naïve use of the formula $(*)$ can suffer overflow, or underflow, or severe cancellation
- Rescaling coefficients avoids overflow or harmful underflow
- To avoid cancellation between $-b$ and square root, compute one root with alternative formula $x = \frac{2c}{-b \mp \sqrt{b^2 - 4ac}} \quad (**)$
- But cancellation inside square root is not avoided without higher precision

Precision 7 digits; $a = 0.05$; $b = -100.0$; $c = 5.0$

Correctly rounded roots: **1999.950** and **0.05000125**

Computed by $(*)$: **1999.950** and **0.05000000**

Computed by $(**)$: **0.05000125** and **2000.000**