Computational Methods (PHYS 2030)

Instructors: Prof. Christopher Bergevin (cberge@yorku.ca)

Schedule: Lecture: MWF 11:30-12:30 (CLH M)

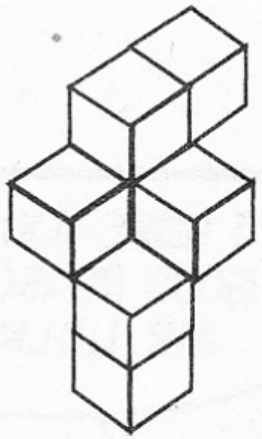
Website: http://www.yorku.ca/cberge/2030W2018.html

York University
Winter 2018
Lecture 1

Ex. 1

Below is a 7-cube configuration (one cube is hidden from this view).

If you were to pick it up and look at it from every angle, how many faces would you find?



→ Now that you have (or have not) figured this out, determine how to write a computer code to "solve" this problem (e.g., computer vision")

Quora THE BEST ANSWER TO ANY QUESTION. | DEC. 28 2015 7:06 AM

Slate

Will Coding Still Be Relevant in a Decade?

By Quora Contributor

100 19

Students learn how to code at an Apple Store through Apple's Hour of Code workshop program on Dec. 9, 2015, in New York City.

Photo by Andrew Burton/Getty Images

"Absolutely. Not only will coding be relevant in 10 years, it will be *more* relevant than it is today. However, the syntax of coding languages will continue becoming easier. When it started, coding was about holes in pieces of cardboard. Then it looked like this: 00101010101. It now looks a lot more like English. As coding languages become more English-like, they will be easier to learn, less arcane, and thus more popular. And as computing systems permeate our lives, telling these devices what we want them to do, and inventing new uses for them, will continue to be more popular."

http://www.slate.com/blogs/quora/2015/12/28/will_coding_still_be_relevant_in_a_decade.html

Slate

"But to teach a computer to do something it never has before will still require specialized understanding of how to communicate with the specificity of a computer programmer, as well as the computational thinking needed to describe an algorithm. The syntax of how to design a loop or a conditional for a computer to do something or make a decision—that syntax may change, but the fundamental concepts underneath are unlikely to go away for many many decades."

→ Some key conceptual ideas are present here. Not only we will build off of such, but PHYS 2030 will allow these to become a key foundational component in your scientific thinking....

http://www.slate.com/blogs/quora/2015/12/28/will_coding_still_be_relevant_in_a_decade.html

Tangent (re downstream “job qualifications”)

Postdoctoral Research Scientist, Acoustics & Audio

Oculus is a world leader in the design of virtual reality systems. We are currently seeking innovative researchers with a passion for technology to push the state-of-the-art for VR audio in our research location in Redmond, WA. This role is focused on developing advanced audio techniques to solve hard unsolved problems in the area of acoustics, spatial audio, audio processing and signal processing. The position is a fixed-term (postdoctoral) contract for 24 months and requires a PhD in audio engineering and signal processing, acoustics, electrical engineering, computer science or similar, and a strong background in solving hard problems in audio.

https://www.facebook.com/careers/resume?req=a0I1200000JXrgMEAT

Tangent (re downstream "job qualifications")

Minimum Qualification

- PhD and/or postdoctoral assignment in the field of Audio Engineering and Signal Processing, Acoustics, Electrical Engineering, Computer Science, or a related field
- Graduating with a PhD, or completing a university postdoctoral assignment, by spring 2017
- 5+ years experience with working on audio-focused simulations, algorithm development, and measurement prototyping
- 3+ years experience in mathematical optimization, including software libraries and implementation from first principles
- 3+ years experience in acoustic simulation and modeling techniques
- Demonstrated ability to construct audio-specific proof-of-concept systems performance evaluations
- 5+ years experience with scientific programming languages such as Matlab, C++, or similar
- Interpersonal skills: cross-group and cross-culture collaboration
- Able to obtain work authorization in the US for a two-year period beginning in 2017

https://www.facebook.com/careers/resume?req=a0I1200000JXrgMEAT

Tangent (re downstream “job qualifications”)

Research Intern, Audio

Oculus Research Redmond is looking for exceptional interns to help us make audio in virtual reality realistic. Expertise in audio, acoustics, signal processing, CPU-GPU parallelization, machine learning, psychoacoustics, and computer science are especially appreciated, but smart, motivated people with strong coding or hardware backgrounds are more than welcome. Strong math skills are a huge plus. We want to build a broad virtual reality research community, so we encourage publishing. Come join us as we make VR happen!

We are accepting applications from students interested in (but not limited to) the following areas: -Audio -Signal processing -Acoustics -Speech -Psychoacoustics -CPU-GPU parallelization -Machine learning -Numerical solvers

Our internships are sixteen (16) to twenty four (24) weeks long and we have various start dates throughout the year.

“Strong math skills are a huge plus”

https://www.facebook.com/careers/resume?req=a0I1200000JXdHzEAL

Tangent (re downstream "job qualifications")

Minimum Qualification

Pursuing a PhD or Masters in Computer Science or related STEM field
Must be currently enrolled in a full time degree program and returning to the program after the completion of the internship
Excellent interpersonal skills, cross-group and cross-culture collaboration
High levels of creativity and quick problem solving capabilities
Proven track record of achieving significant results
Ability to obtain work authorization in the United States in 2017

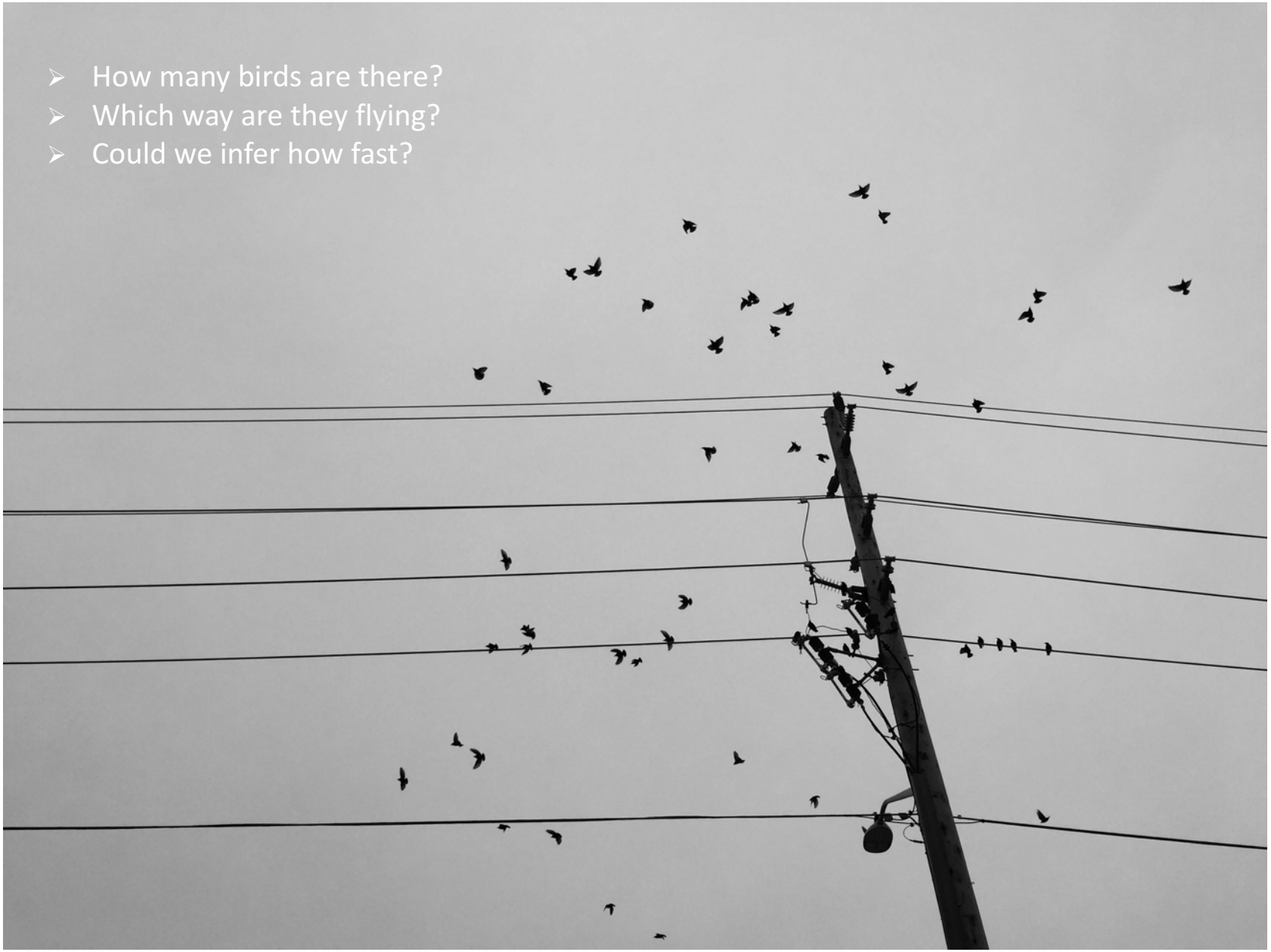
Preferred Qualifications

Demonstrated software engineer experience via an internship, work experience, coding competitions, or PhD papers
Experience with C/C++, MATLAB or CUDA

→ PHYS 2030 is providing you w/ some of the skills to pay the bills!

https://www.facebook.com/careers/resume?req=a0I1200000JXdHzEAL

Ex. 2

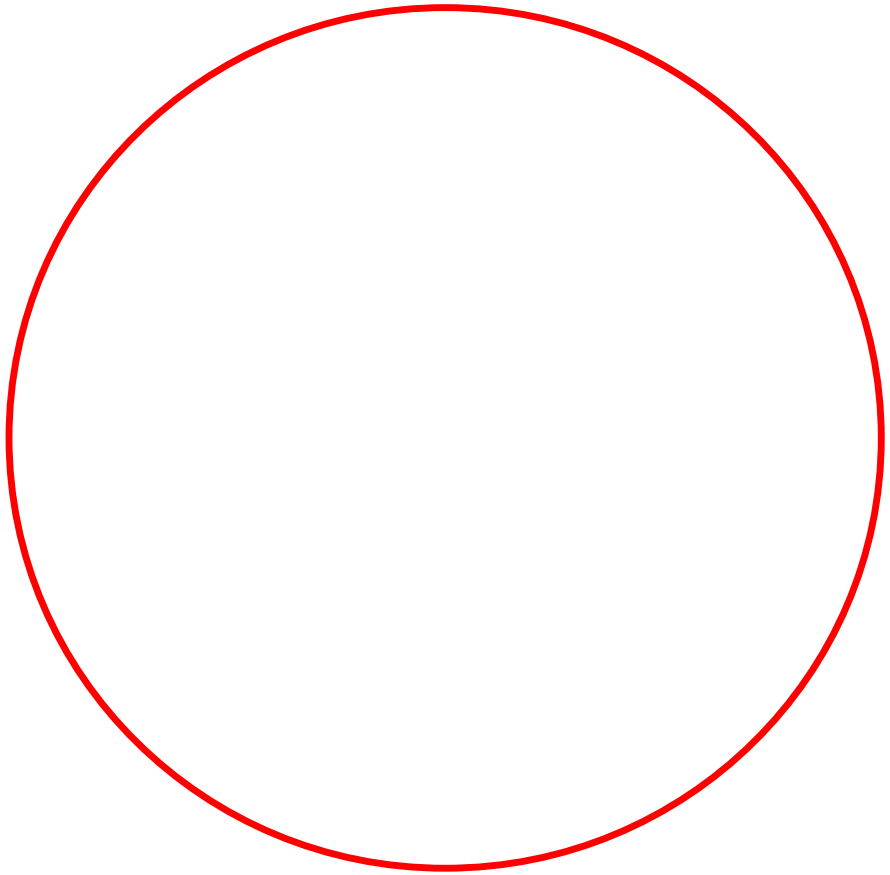

How many birds are there?
Which way are they flying?
Could we infer how fast?

Ex. 3

Heuristic starting point:

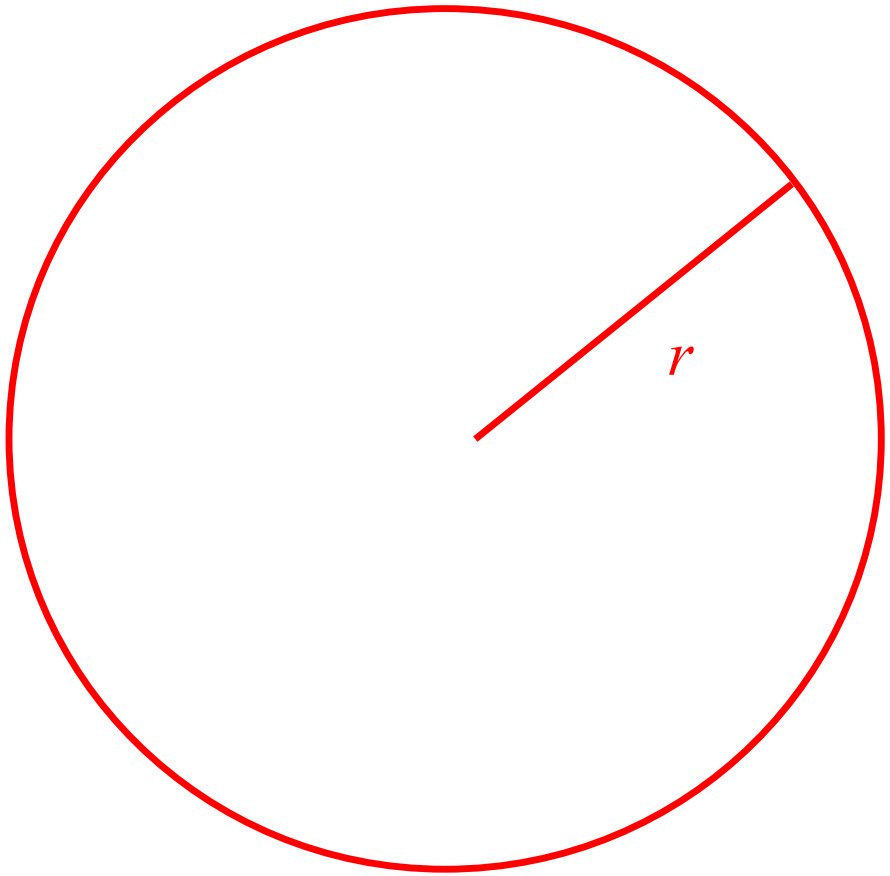
Always a good idea to have ***three*** independent ways to back up ideas/thoughts

Consider a (seemingly trivial) example:



What is the area of a circle?

What is the area of a circle?



Approach 1

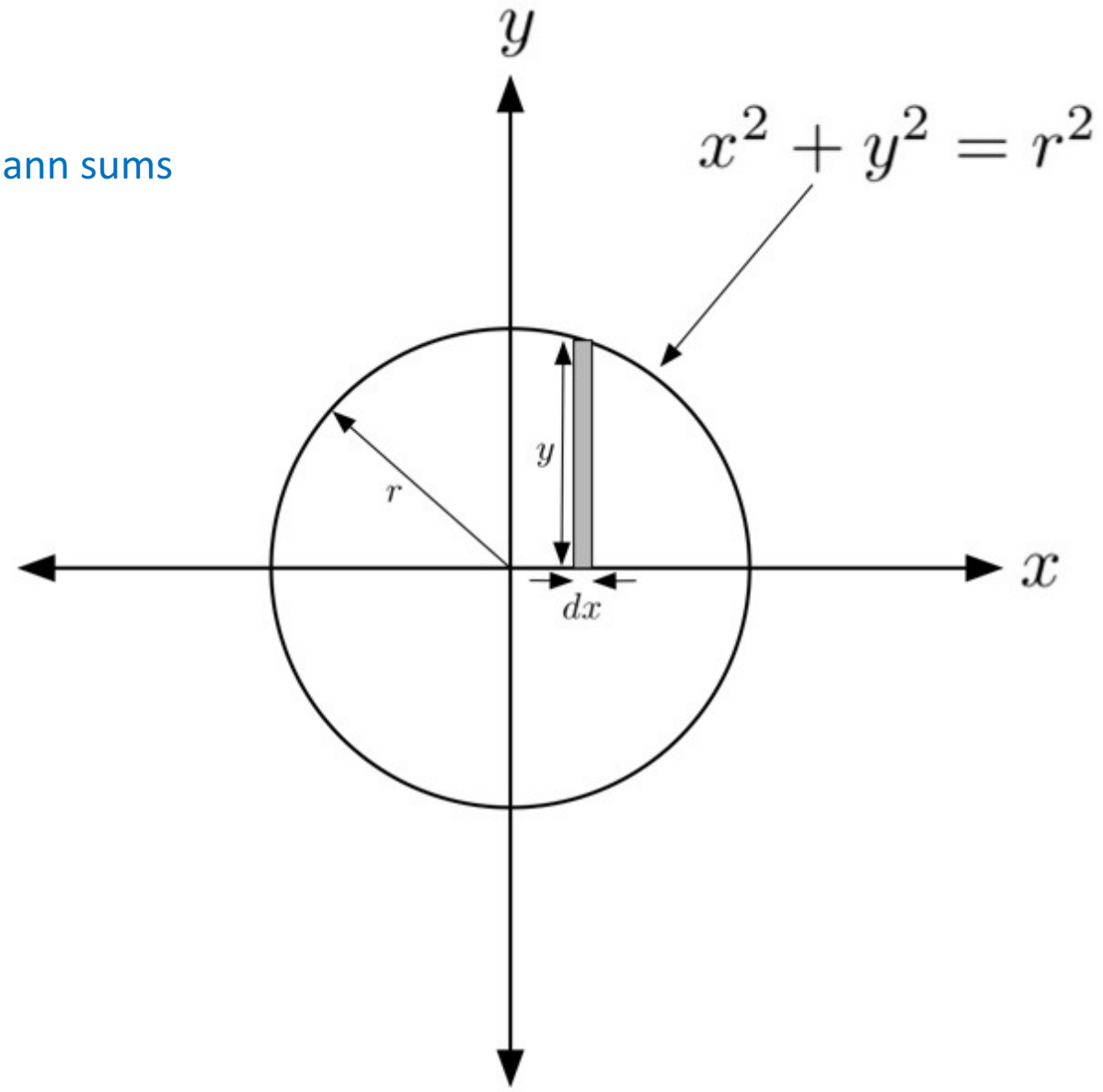
Area = πr^2

Begs the question though: What in the world is this mysterious number π?

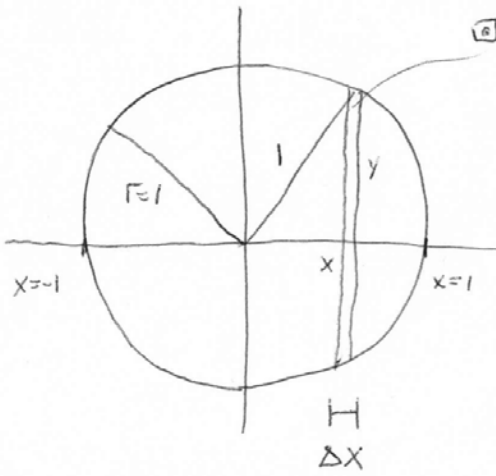
What is the area of a circle?

Approach 2

Riemann sums



Area of a circle via Riemann Sums



- area of "strip" ($\equiv A_s$) $\simeq 2 \cdot y \cdot \Delta x = 2\sqrt{1-x^2}\,\Delta x$
  (since $x^2+y^2=1$)
- area of circle (A_c) $\simeq \sum_{x=-1}^{x=1} A_s = \sum 2\sqrt{1-x^2}\,\Delta x$
- taking the limit where our strip get infinitesimally small:

$$A_c = \lim_{\Delta x \to 0} \sum_{x=-1}^{1} 2\sqrt{1-x^2}\,\Delta x = \int_{-1}^{1} 2\sqrt{1-x^2}\,dx$$

- Using a trusty "table of integrals", we find: $\int \sqrt{a^2-x^2}\,dx = \frac{x\sqrt{a^2-x^2}}{2} + \frac{a^2}{2}\sin^{-1}\left(\frac{x}{a}\right)$
- Back to our problem:

$$A_c = \int_{-1}^{1} 2\sqrt{1-x^2}\,dx = 2\left[\frac{x\sqrt{1-x^2}}{2} + \frac{1}{2}\sin^{-1}x\right]_{-1}^{1}$$

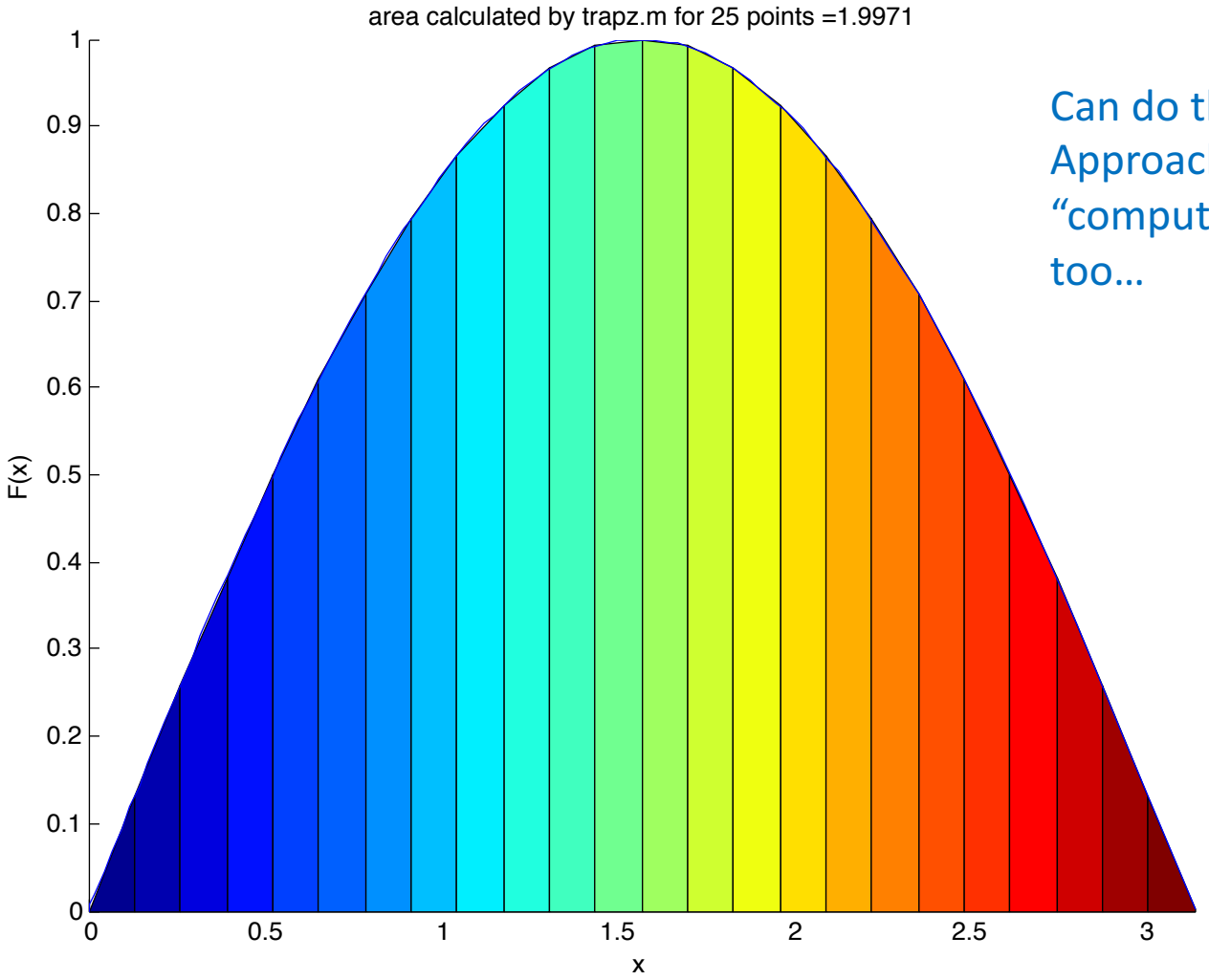
$$= 1\sqrt{1-1} + \sin^{-1}(1) - \frac{-1\sqrt{1-1}}{1} - \sin^{-1}(-1) = \sin^{-1}(1) - \sin^{-1}(-1)$$

$$= \frac{\pi}{2} - \left(-\frac{\pi}{2}\right) = \pi = \pi \cdot 1^2$$

$\therefore$

Note: these are the "inverse trig functions" (if $x = \sin y$, then $y = \sin^{-1}x$, i.e. the angle whose sine is x; $\sin^{-1}(1) = \frac{\pi}{2}$ and $\sin^{-1}(-x) = -\sin^{-1}x$)

```
Trapezoid method (Method B)
np= 25
```



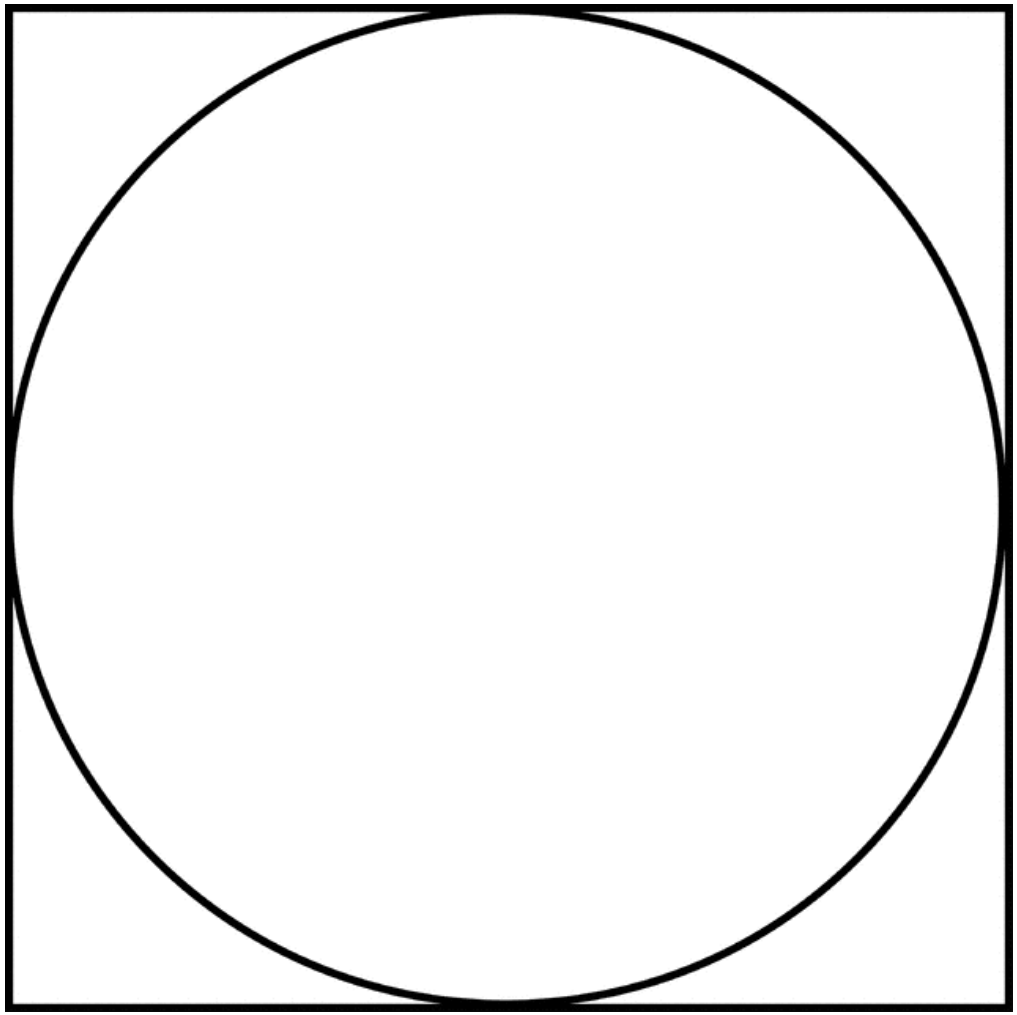
Can do this (i.e., Approach 2) "computationally" too…

What is the area of a circle?

Approach 3

Computational “Monte carlo” approach

→ Turn the problem into a different one *(e.g., what is that mystery number π?)*



Ex. Estimating π

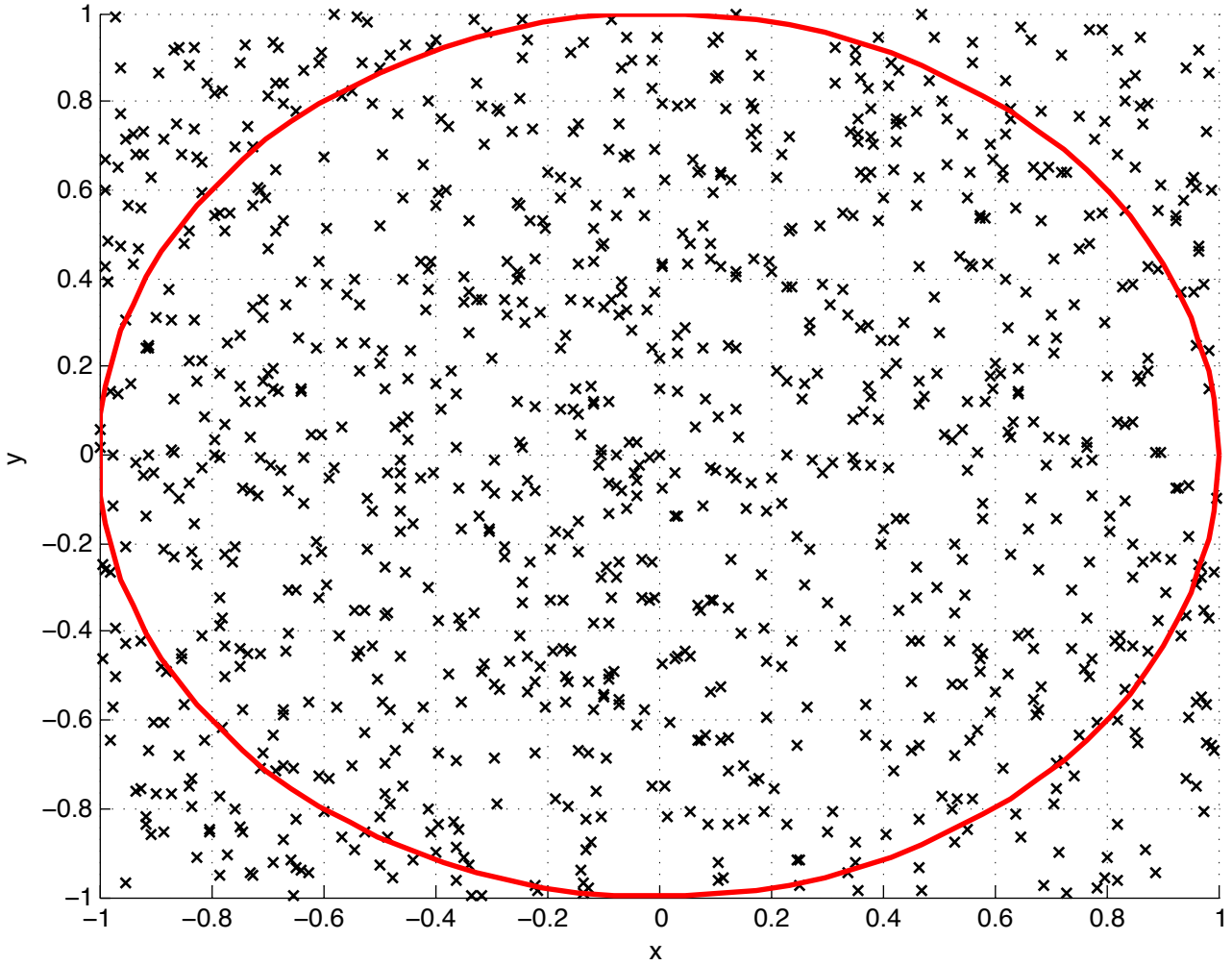
EXestimatePI.m

```
% ### EXestimatePI.m ###      04.22.11 {C. Bergevin}
% Use a random # generator to estimate pi by considering the ratio of areas
% of a circle to a square

clear
% -----------
N=1000;      % # of points to use
% -----------
figure(1); clf; hold on; grid on;
% +++
% generate array of (uniformly distributed) x and y values
A= 2*rand(N,1)-1;   % x coord.
B= 2*rand(N,1)-1;   % y coord.
% +++
% loop thru to test if each coordinate pair is inside or out
Ac= 0;  % indexer
for nn=1:N
    x= A(nn); y= B(nn);
    % test to see if the point falls within the unit circle (i.e., within
    % it area); if so, update indexer
    if (sqrt(x^2+y^2) <= 1),    Ac= Ac+1;   end
    plot(x,y,'kx');     % also plot for visual purposes
end
% +++
% estimate pi as the ratio of the areas
piEST= 4*(Ac/N); fprintf('\n estimate for pi = %g (using %g points) \n\n',piEST, N);
% +++
axis([-1 1 -1 1]);  xlabel('x'); ylabel('y')
% also draw a unit circle
theta= linspace(0,2*pi,100); xC= cos(theta); yC= sin(theta); plot(xC,yC,'r-');
```

Ex. Estimating π

```
N=1000;        % # of points to use
```

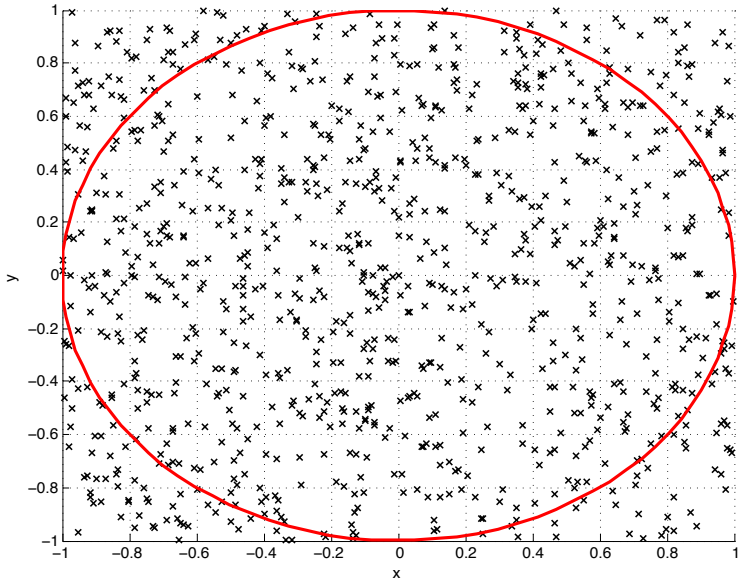


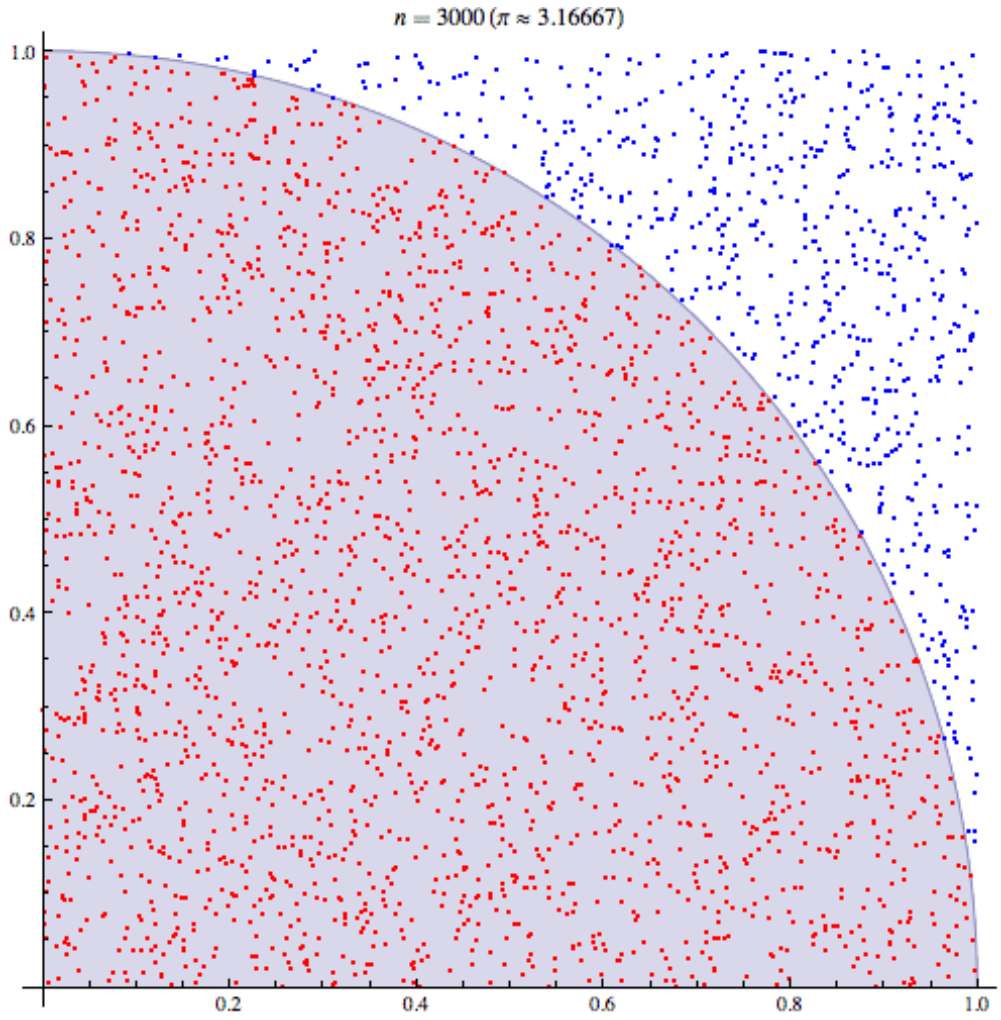
→ Simply just using random numbers and ratios of areas

```
estimate for pi = 3.12 (using 1000 points)
```

Ex. Estimating π

EXestimatePI.m





wikipedia (Monte Carlo method)

PHYS 2030: Course philosophy

"*What I cannot create, I do not understand.*" - R. Feynman

"Programming is a science based on limited resources; one can program only within the range of power available in a computer's hardware and software."

- D. Kushner
(Masters of Doom: How Two Guys Created an Empire and Transformed Pop Culture)

→ Our focus is on conceptual understanding, not optimization per se. That's not to say that efficiency is not important!

PHYS 2030 Overview

- Numeric differentiation
- Numeric integration
- ODEs
- DAQ
- Data visualization
- Linear systems theory
- Regression
- Nonlinear system
- Fourier/spectral analysis
- Linear systems (revisited)
- Monte Carlo methods

Quora THE BEST ANSWER TO ANY QUESTION. DEC. 28 2015 7:06 AM

Will Coding Still Be Relevant in a Decade?

By Quora Contributor

100 19

Students learn how to code at an Apple Store through Apple's Hour of Code workshop program on Dec. 9, 2015, in New York City.

Photo by Andrew Burton/Getty Images

PHYS 2030 Overview: Course themes

1. **Programming/computing skills**
   - o knowledge of syntax & capabilities
   - o comfortability
   - o debugging (pay attention to error messages)
   - o style: modular versus comprehensive programming
   - o finding online resources (incl. scientific journals)

2. **Numerical methods**
   - o solving differential equations —> harmonic oscillator
   - o eigenvalue problems —> Principle component analysis
   - o Monte Carlo
   - o regression
   - o complex numbers —> fractals
   - o Fourier transforms
   - o bifurcation analysis (period doubling cascade and chaos)

3. **Data acquisition/analysis**
   - o measuring data
   - o visualizing data
   - o analyzing data & signal processing
   - o statistics —> Anscombe's data
   - o compressed sensing

Caveat: As time permits
(i.e., we might not get to all this)

York University
PHYS 2030: Computational Methods (3 credits)
Winter 2018

Time & Location
Lecture: MWF 11:30-12:30 (CLH M)
Drop-in Lab I: M 2:30-3:30 (Ross S110; "Gauss Lab") *Drop-in Lab II*: W 3:30-4:30 (Ross S110)

Instructor: Christopher Bergevin (cberge@yorku.ca)
Office: Petrie 240
Office Hours: TBD (and by appt.)

TAs:

- TBD (tbd@yorku.ca)

Graders:

- TBD (tbd@yorku.ca)
- TBD (tbd@yorku.ca)

Course Website:
`http://www.yorku.ca/cberge/2030W2018.html`

Textbook:
Basic Concepts in Computational Physics, Stickler, BA & Schachinger, E (*Springer, 2014*)*

Prerequisites: SC/PHYS 1010 6.00 or a minimum grade of C in SC/PHYS 1410 6.00 or SC/PHYS 1420 6.00; One of LE/CSE 1020 3.00, LE/CSE 1540 3.00; SC/MATH 1014 3.00 or equivalent. Corequisite: SC/MATH 2015 3.00 or equivalent, SC/MATH 2271. Basically/ideally, you should have some familiarity with Matlab (or similar programming language) and differential equations (or in the process of concurrently doing such).

First Day of Class	Jan. 5
Add Deadline	Jan. 31
Midterm Exam	Feb. 9
Winter Reading Week	Feb. 19–23 (**no classes**)
Drop Deadline	Mar. 9
Good Friday	Mar. 30 (**no class**)
Last Day of Class	Apr. 6
Final Exam	TBD

Grading

There will be 100 total possible points in the course. Point breakdowns are as follows:

- Homework – **20 points**
- Midterm – **25 points**
- Final Exam – **40 points**
- Class attendance – **15 points**

*This book is freely available as a "soft copy" to York students via SpringerLink: `https://link.springer.com/book/10.1007/978-3-319-02435-6`

http://www.yorku.ca/cberge/2030W2018.html

→ This is the course website and will be the main "go to" place for all course-related info (e.g., syllabus, slides, chapters for reading, exam info, etc…)

‘Golden rules’ of (2030) computing

1. **The computer only does what you tell it to do**

2. Think ahead when writing code: **try to minimize future work**
   (e.g., even remotely archaic code now will certainly equal headache later on)

3. Stay organized. Find a **style** that works for you

4. Comment. Comment. **Comment**. [clearly and efficiently]

Other tips:

- 'Preserve all 'versions' of workable code.'
  [Caveat: Don't modify someone else's code! Copy and make your own version.]

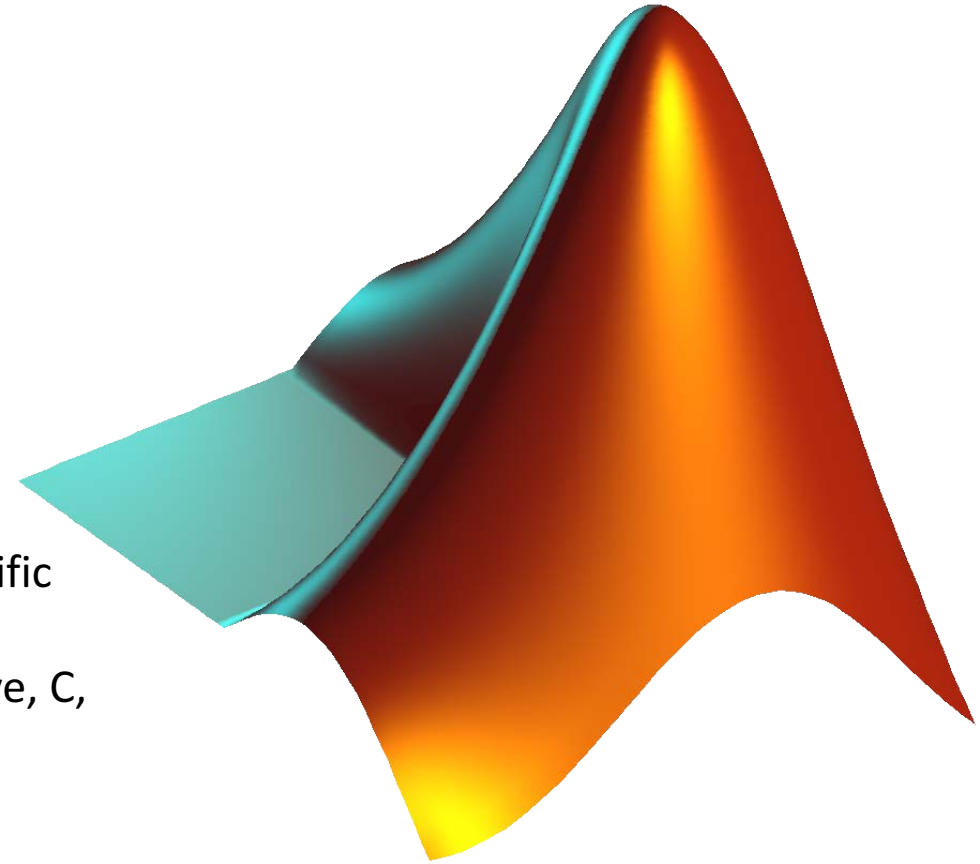
- When using external routines (i.e., code you didn't write yourself), keep the spectre of the 'blackbox' in mind at all time.

PHYS 2030 Logistics

- Want some sort of programmable interface

- **Matlab** is what we will use here, but....

- ... there are many options for scientific computing/coding available (e.g., Mathematica, Maple, Python, Octave, C, Java, etc....)

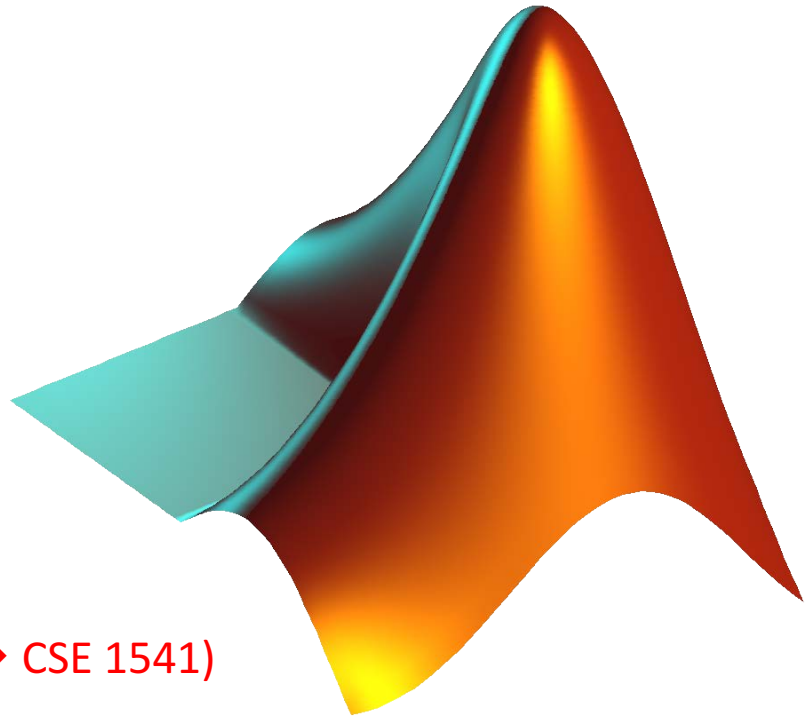


Important message here!

→ The true value of PHYS 2030 lies not in any Matlab syntax you'll pick up, but in the basic/core programming concepts, practice in such, and a new "attitude" re computing

PHYS 2030 Logistics

- Think of “computer coding” as learning to read & write



- First, you pick a language (e.g., Matlab)
- Second, you learn basic grammar (“syntax” → CSE 1541)
- Next you learn language and read Shakespeare (PHYS 2030)
- Eventually you get on with your life and actually do something, like write a novel or a FAQ page for a video game

  (though you never stop learning along the way!)

Accessing Matlab

Two possible options:

- Purchase (or 'rent') a student version via Mathworks

- Access remotely via secure login (webfas) to York server (a bit slow/clunky)

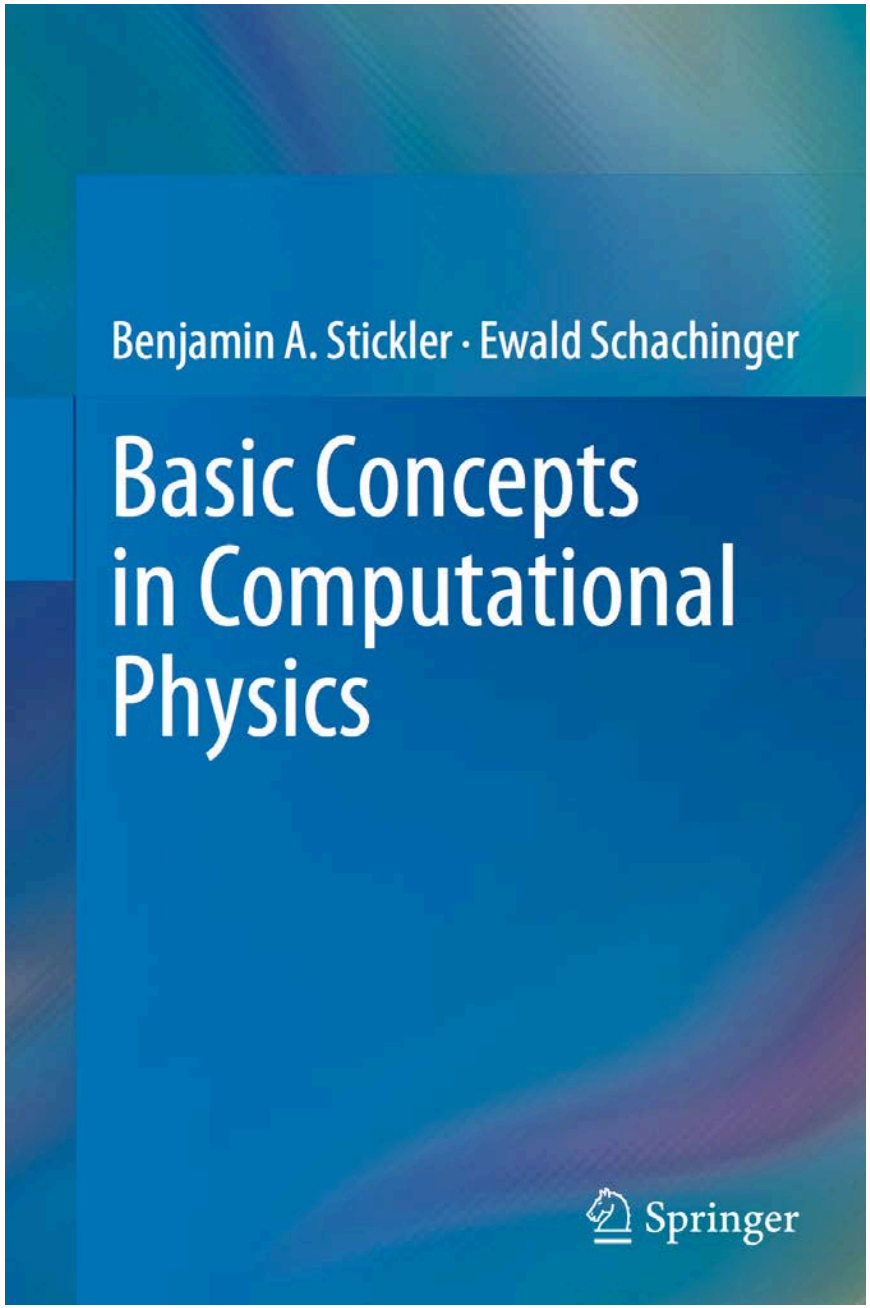
1. Go to this URL and and click on the install button for the citrix receiver: https://webfas.yorku.ca/Citrix/WEBFASWeb/

2. Download the citrix reciever and install the executable

3. After installation continue to the login and use your passport york username and password

4. On the left should be a tab with a "+" sign, open it and click All Apps and add the matlab app

5. Click the matlab app and a download will begin. Once finished loading, doubleclick the download and from there it should take care of the rest



Course text

(which we will only loosely follow; the bulk of the content will be contained in the course notes/codes)

→ You can (legitimately!) get a "soft copy" of the book

Home – Springer

link.springer.com

Search

Google Maps | Geocaching | Google Scholar | Dropbox | Pitchfork | Thesaurus | York | Other

» Sign up / Log in | English | Academic edition

Springer Link

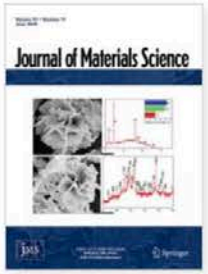
Search

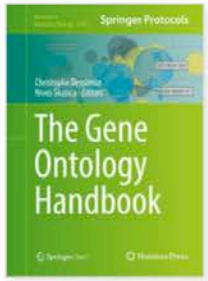
Home • Contact Us

Browse by discipline

- » Architecture & Design
- » Astronomy
- » Biomedical Sciences
- » Business & Management
- » Chemistry
- » Computer Science
- » Earth Sciences & Geography
- » Economics
- » Education & Language
- » Energy
- » Engineering
- » Environmental Sciences
- » Food Science & Nutrition

Providing researchers with access to millions of scientific documents from journals, books, series, protocols and reference works.





New books and journals are available every day.

Book 2014

Basic Concepts in Computational Physics

Authors: Benjamin A. Stickler, Ewald Schachinger

ISBN: 978-3-319-02434-9 (Print) 978-3-319-02435-6 (Online)

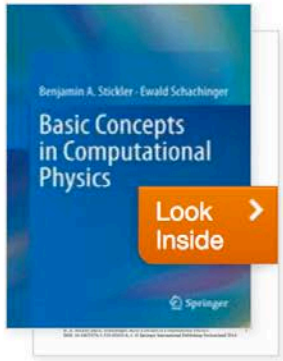


Table of contents (20 chapters)

Chapter

Numerical Differentiation

Benjamin A. Stickler, Ewald Schachinger

» **Download PDF** (222KB) » **View Chapter** Pages 17-28

Chapter

Numerical Integration

Benjamin A. Stickler, Ewald Schachinger

» **Download PDF** (309KB) » **View Chapter** Pages 29-50

Chapter

The Kepler Problem

Benjamin A. Stickler, Ewald Schachinger

» **Download PDF** (186KB) » **View Chapter** Pages 51-59

Chapter

Ordinary Differential Equations: Initial Value Problems

Benjamin A. Stickler, Ewald Schachinger

» **Download PDF** (449KB) » **View Chapter** Pages 61-79

Chapter

The Double Pendulum

Benjamin A. Stickler, Ewald Schachinger

» **Download PDF** (2474KB) » **View Chapter** Pages 81-96

Chapter

Molecular Dynamics

Benjamin A. Stickler, Ewald Schachinger

» **Download PDF** (316KB) » **View Chapter** Pages 97-109

Chapter

Numerics of Ordinary Differential Equations: Boundary Value Problems

Benjamin A. Stickler, Ewald Schachinger

» **Download PDF** (176KB) » **View Chapter** Pages 111-122

Chapter

The One-Dimensional Stationary Heat Equation

Benjamin A. Stickler, Ewald Schachinger

» **Download PDF** (297KB) » **View Chapter** Pages 123-129

Chapter

The One-Dimensional Stationary SCHRÖDINGER Equation

Benjamin A. Stickler, Ewald Schachinger

» **Download PDF** (445KB) » **View Chapter** Pages 131-146

Chapter

Partial Differential Equations

Benjamin A. Stickler, Ewald Schachinger

» **Download PDF** (831KB) » **View Chapter** Pages 147-168

Stochastic Methods

Front Matter

» **Download PDF** (34KB) Pages 169-169

Chapter

Pseudo Random Number Generators

Benjamin A. Stickler, Ewald Schachinger

» **Download PDF** (364KB) » **View Chapter** Pages 171-183

Chapter

Random Sampling Methods

Benjamin A. Stickler, Ewald Schachinger

» **Download PDF** (271KB) » **View Chapter** Pages 185-195

Chapter

A Brief Introduction to Monte-Carlo Methods

Benjamin A. Stickler, Ewald Schachinger

» **Download PDF** (263KB) » **View Chapter** Pages 197-208

Chapter

The Ising Model

Benjamin A. Stickler, Ewald Schachinger

» **Download PDF** (618KB) » **View Chapter** Pages 209-228

Chapter

Some Basics of Stochastic Processes

Benjamin A. Stickler, Ewald Schachinger

» **Download PDF** (339KB) » **View Chapter** Pages 229-250

Chapter

The Random Walk and Diffusion Theory

Benjamin A. Stickler, Ewald Schachinger

» **Download PDF** (993KB) » **View Chapter** Pages 251-273

Chapter

Markov-Chain Monte Carlo and the Potts Model

Benjamin A. Stickler, Ewald Schachinger

» **Download PDF** (667KB) » **View Chapter** Pages 275-286

Chapter

Data Analysis

Benjamin A. Stickler, Ewald Schachinger

→ We will cover some of these topics, plus (plenty of) others not listed here

Review (re CSE 1541)

http://www.eecs.yorku.ca/course_archive/2013-14/W/1541/calendar.shtml

- basics of computing: digital versus analog, binary
- overview of Matlab and basic format/syntax
- numbers versus strings
- logicals & relational operators
- vectors, arrays, structures, etc…. (+ indicies)
- basic programming, including loops and conditionals
- basic etiquette (e.g., naming variable, commenting code)
- data types
- data file input/output
- basic statistics
- plotting data
- user-defined functions
- Scripts
- matrix manipulations (e.g., Gaussian elimination)
- random numbers
- (basic) curve fitting
- recursive relations, bisection
- numerical differentiation, integration
- solving ODEs

→ We are going to start here and try to ensure everyone gets up to speed (but it's going to be hard!)

Post-class exercises

- Make sure you have a comfortable means to run Matlab
- Write a code to read in an image. How does Matlab store the numbers?
- Outline a means as to how you might tackle this problem posed earlier....
- Write your own "estimate π" code

Course website: http://www.yorku.ca/cberge/2030W2018.html

OPEN ACCESS Freely available online

PLOS | BIOLOGY

Community Page

Best Practices for Scientific Computing

Greg Wilson[1]*, D. A. Aruliah[2], C. Titus Brown[3], Neil P. Chue Hong[4], Matt Davis[5], Richard T. Guy[6¤], Steven H. D. Haddock[7], Kathryn D. Huff[8], Ian M. Mitchell[9], Mark D. Plumbley[10], Ben Waugh[11], Ethan P. White[12], Paul Wilson[13]

1 Mozilla Foundation, Toronto, Ontario, Canada, **2** University of Ontario Institute of Technology, Oshawa, Ontario, Canada, **3** Michigan State University, East Lansing, Michigan, United States of America, **4** Software Sustainability Institute, Edinburgh, United Kingdom, **5** Space Telescope Science Institute, Baltimore, Maryland, United States of America, **6** University of Toronto, Toronto, Ontario, Canada, **7** Monterey Bay Aquarium Research Institute, Moss Landing, California, United States of America, **8** University of California Berkeley, Berkeley, California, United States of America, **9** University of British Columbia, Vancouver, British Columbia, Canada, **10** Queen Mary University of London, London, United Kingdom, **11** University College London, London, United Kingdom, **12** Utah State University, Logan, Utah, United States of America, **13** University of Wisconsin, Madison, Wisconsin, United States of America

January 2014 | Volume 12 | Issue 1 | e1001745

Box 1. Summary of Best Practices

1. Write programs for people, not computers.

(a) A program should not require its readers to hold more than a handful of facts in memory at once.
(b) Make names consistent, distinctive, and meaningful.
(c) Make code style and formatting consistent.

2. Let the computer do the work.

(a) Make the computer repeat tasks.
(b) Save recent commands in a file for re-use.
(c) Use a build tool to automate workflows.

3. Make incremental changes.

(a) Work in small steps with frequent feedback and course correction.
(b) Use a version control system.
(c) Put everything that has been created manually in version control.

4. Don't repeat yourself (or others).

(a) Every piece of data must have a single authoritative representation in the system.
(b) Modularize code rather than copying and pasting.
(c) Re-use code instead of rewriting it.

5. Plan for mistakes.

(a) Add assertions to programs to check their operation.
(b) Use an off-the-shelf unit testing library.
(c) Turn bugs into test cases.
(d) Use a symbolic debugger.

6. Optimize software only after it works correctly.

(a) Use a profiler to identify bottlenecks.
(b) Write code in the highest-level language possible.

7. Document design and purpose, not mechanics.

(a) Document interfaces and reasons, not implementations.
(b) Refactor code in preference to explaining how it works.
(c) Embed the documentation for a piece of software in that software.

8. Collaborate.

(a) Use pre-merge code reviews.
(b) Use pair programming when bringing someone new up to speed and when tackling particularly tricky problems.
(c) Use an issue tracking tool.

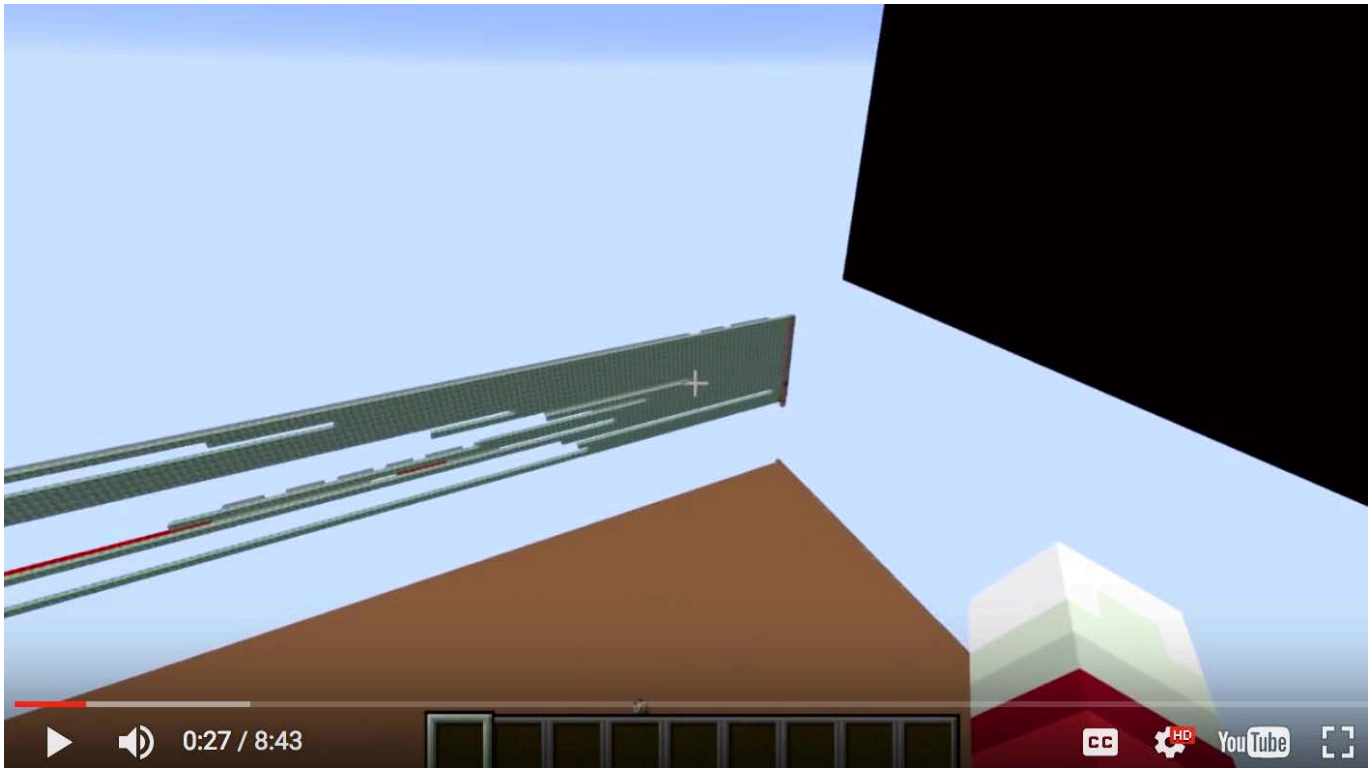
Polygon

VIDEO

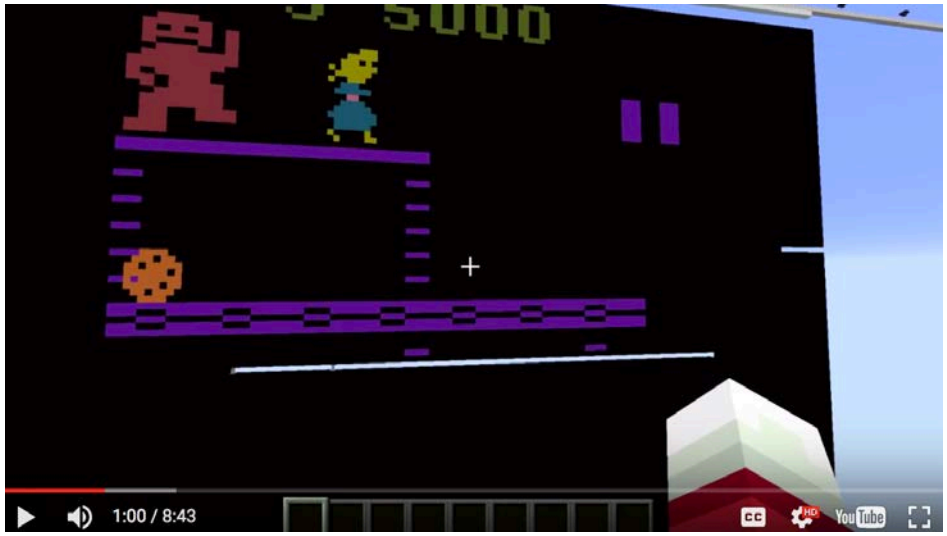
You can play Atari games inside Minecraft

Clear your calendar

by Charlie Hall | @Charlie_L_Hall | Dec 9, 2016, 5:30pm EST



http://www.polygon.com/2016/12/9/13902832/you-can-play-atari-games-inside-minecraft



But don't get too excited.

While SethBling has provided a tool to load Atari ROMs into *Minecraft*, once they're in there you're only going to be playing at around 60 frames per... four hours.

Instead, think of his project as a tremendous way to explain how games are made.

http://www.polygon.com/2016/12/9/13902832/you-can-play-atari-games-inside-minecraft

Let’s cook up a few (more oddball) examples to motivate how we might want to use a computer in a programmable fashion to address a question....

For simplicity, we’ll confine ourselves to the topic of “image analysis”

Note: This sort of thing is a relative “hot topic” (e.g., this is the sort of thing Google has hired extensively in) and leads nicely into other areas such as *deep learning*

Ex. 2

Ex. 2

Keep in mind that there is an issue of “resolution” here that complicates matters...

Ex. 3

EyeLock intros iris authentication reference designs for IoT device integration

http://www.biometricupdate.com/201612/eyelock-intros-iris-authentication-reference-designs-for-iot-device-integration

IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 14, NO. 1, JANUARY 2004

An Introduction to Biometric Recognition

Anil K. Jain, *Fellow, IEEE*, Arun Ross, *Member, IEEE*, and Salil Prabhakar, *Member, IEEE*

Ex. 3

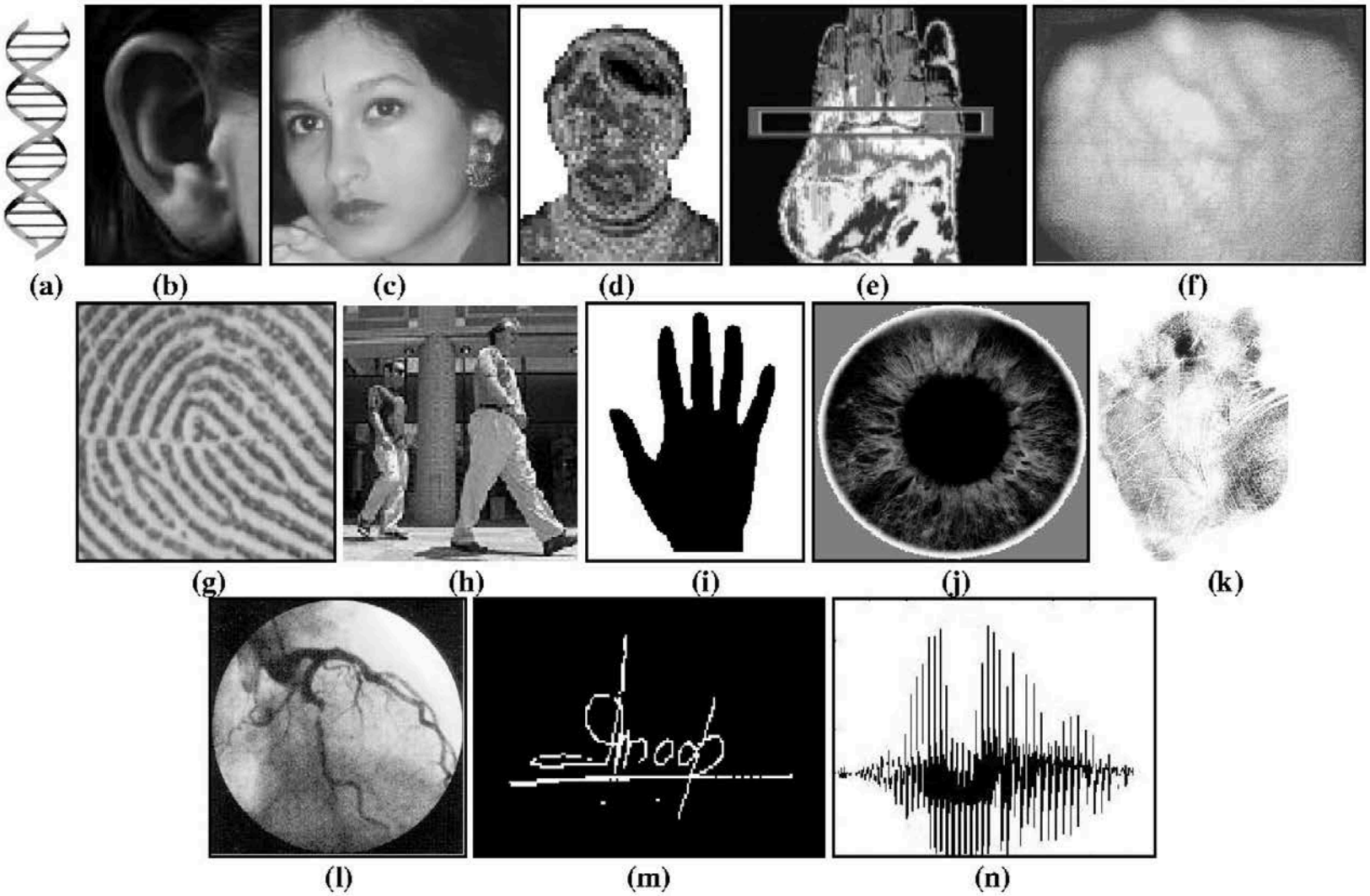


Fig. 3. Examples of biometric characteristics: (a) DNA, (b) ear, (c) face, (d) facial thermogram, (e) hand thermogram, (f) hand vein, (g) fingerprint, (h) gait, (i) hand geometry, (j) iris, (k) palmprint, (l) retina, (m) signature, and (n) voice.

Ex. 3

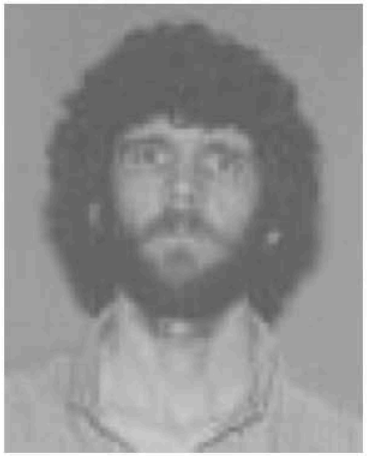
Fig. 4. Examples of biometric application. (a) Fingerprint verification system manufactured by Digital Persona, Inc., is used for computer and network login. (b) Fingerprint-based point of sale (POS) terminal manufactured by Indivos, Inc., that verifies the customers before charging their credit cards and speeds up payment in retail shops, restaurants and cafeterias. (c) Fingerprint-based door lock manufactured by BioThentica Corporation used to restrict access to premises is shown. (d) Immigration and naturalization service accelerated service system (INSPASS), which is installed at major airports in the U.S., is based on hand geometry verification technology developed by Recognition Systems, Inc., and significantly reduces the immigration processing time. (e) Border passage system using iris recognition at London's Heathrow airport. (f) Ben Gurion airport in Tel Aviv (Israel) uses Express Card entry kiosks fitted with hand geometry systems for security and immigration. (g) The FacePass system from Viisage is used in POS verification applications like ATMs, therefore obviating the need for PINs. (h) The Identix TouchClock fingerprint system is used in time and attendance applications.

Ex. 3

Iris Recognition: An Emerging Biometric Technology

RICHARD P. WILDES, MEMBER, IEEE

PROCEEDINGS OF THE IEEE, VOL. 85, NO. 9, SEPTEMBER 1997



Richard P. Wildes (Member, IEEE) received the Ph.D. degree from the Massachusetts Institute of Technology (MIT), Cambridge, in 1989.

In 1984–1988, he was a Research Assistant in the MIT Artificial Intelligence Laboratory. During that time, he was a National Science Foundation Graduate Fellow. In 1988, he joined the Department of Computer Science at the State University of New York at Buffalo as an Assistant Professor. During 1990, he joined The Sarnoff Corporation, where he is a Member of the Technical Staff. His main areas of research interest are machine and biological perception (especially vision), robotics, and artificial intelligence.

Ex. 3

LASSONDE YORK U EECS ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Current Students | Community | Programs | Activities | Research | Our Home | Future Students

Richard Wildes

Department Chair, Associate Professor

Home / Richard Wildes

RESEARCH INTERESTS

- Machine and Biological perception, especially vision;
- Allied aspects of image processing, robotics, and artificial intelligence;
- The analysis and understanding of binocular and motion imagery;
- Gesture recognition and aerial image interpretation applications.

Ex. 3

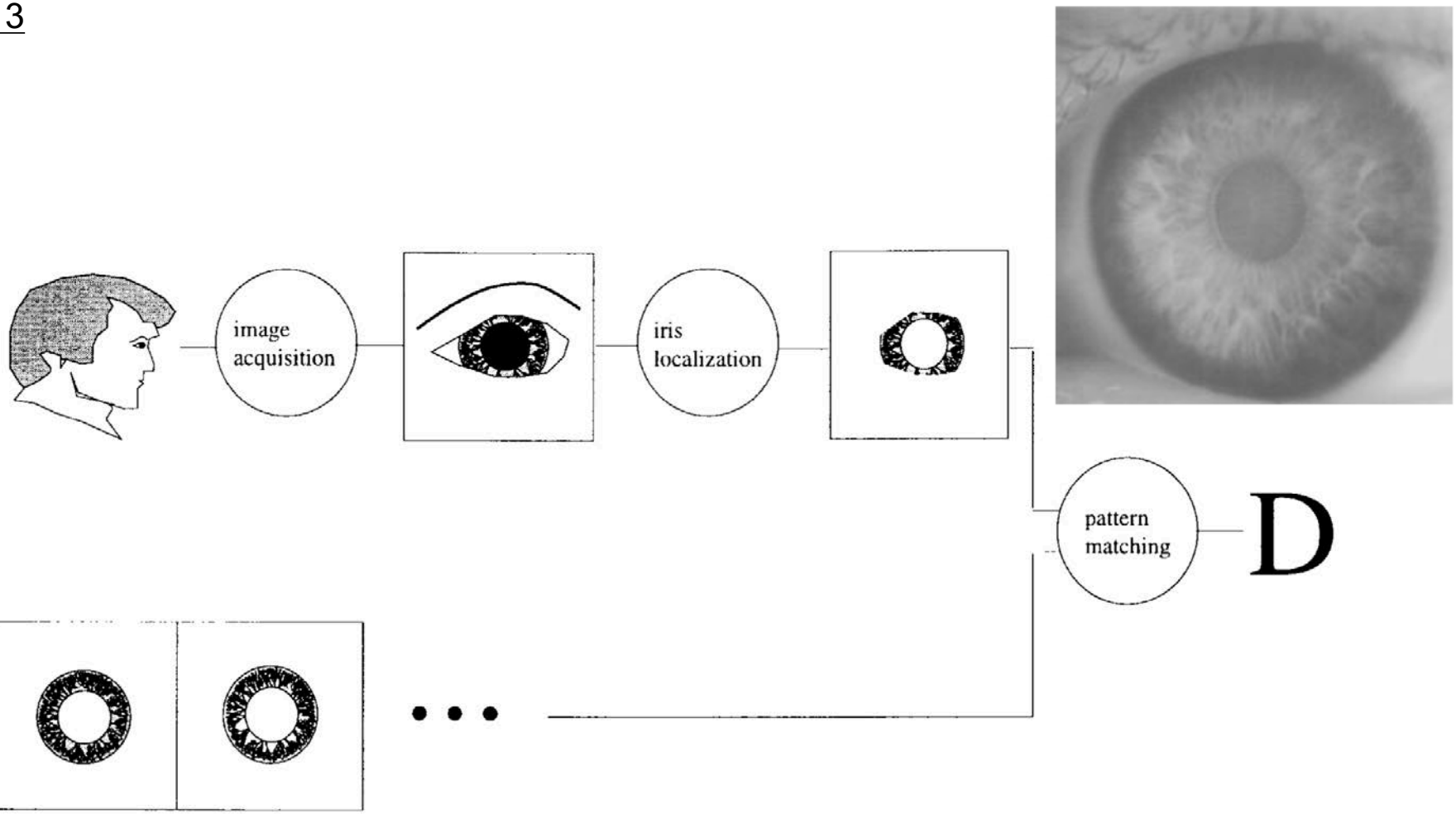
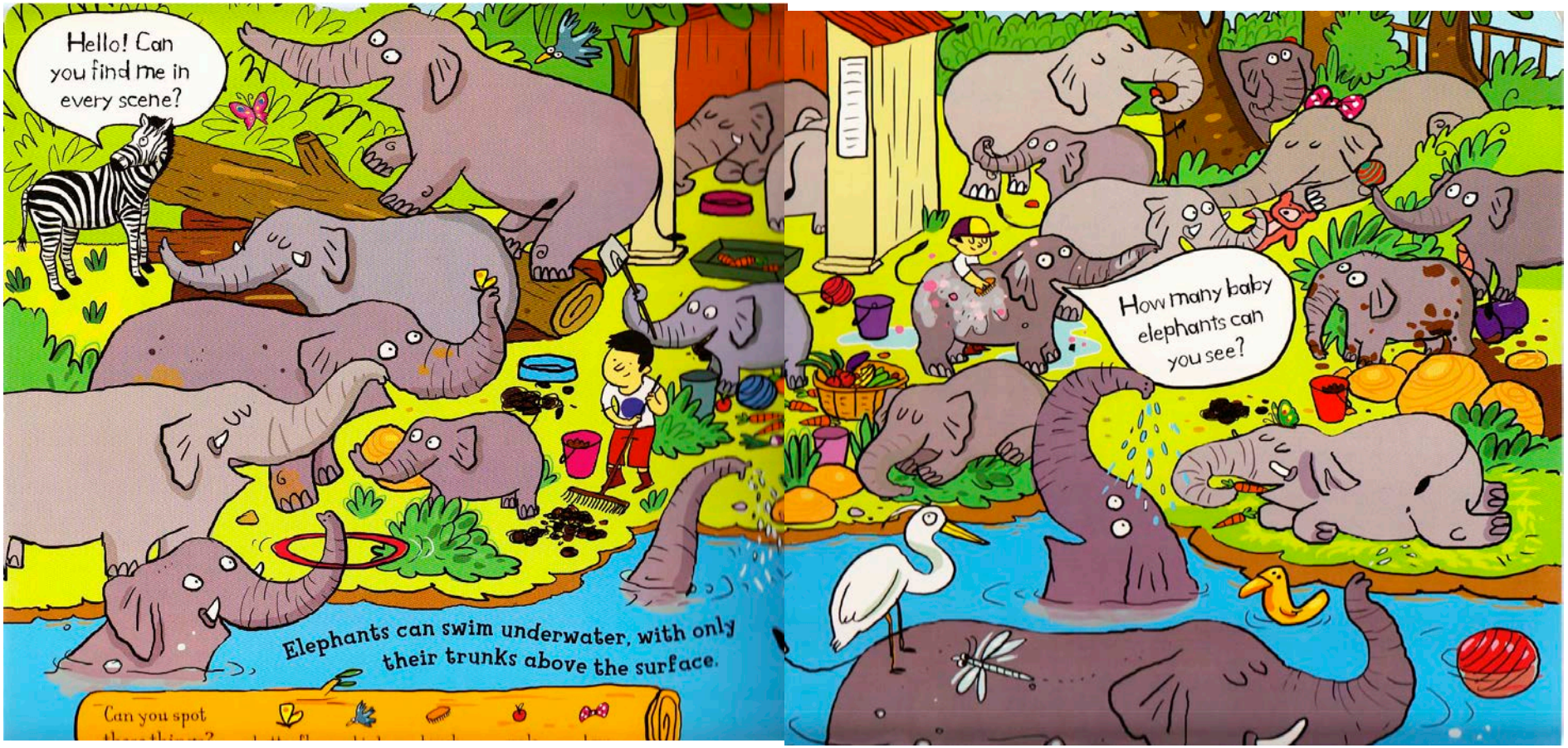


Fig. 3. Schematic diagram of iris recognition. Given a subject to be evaluated (left of upper row) relative to a data base of iris records (left of lower row), recognition proceeds in three steps. The first step is image acquisition, which yields an image of the subject's eye region. The second step is iris localization, which delimits the iris from the rest of the acquired image. The third step is pattern matching, which produces a decision, "D." For verification, the decision is a yes/no response relative to a particular prespecified data base entry; for identification, the decision is a record (possibly null) that has been indexed relative to a larger set of entries.

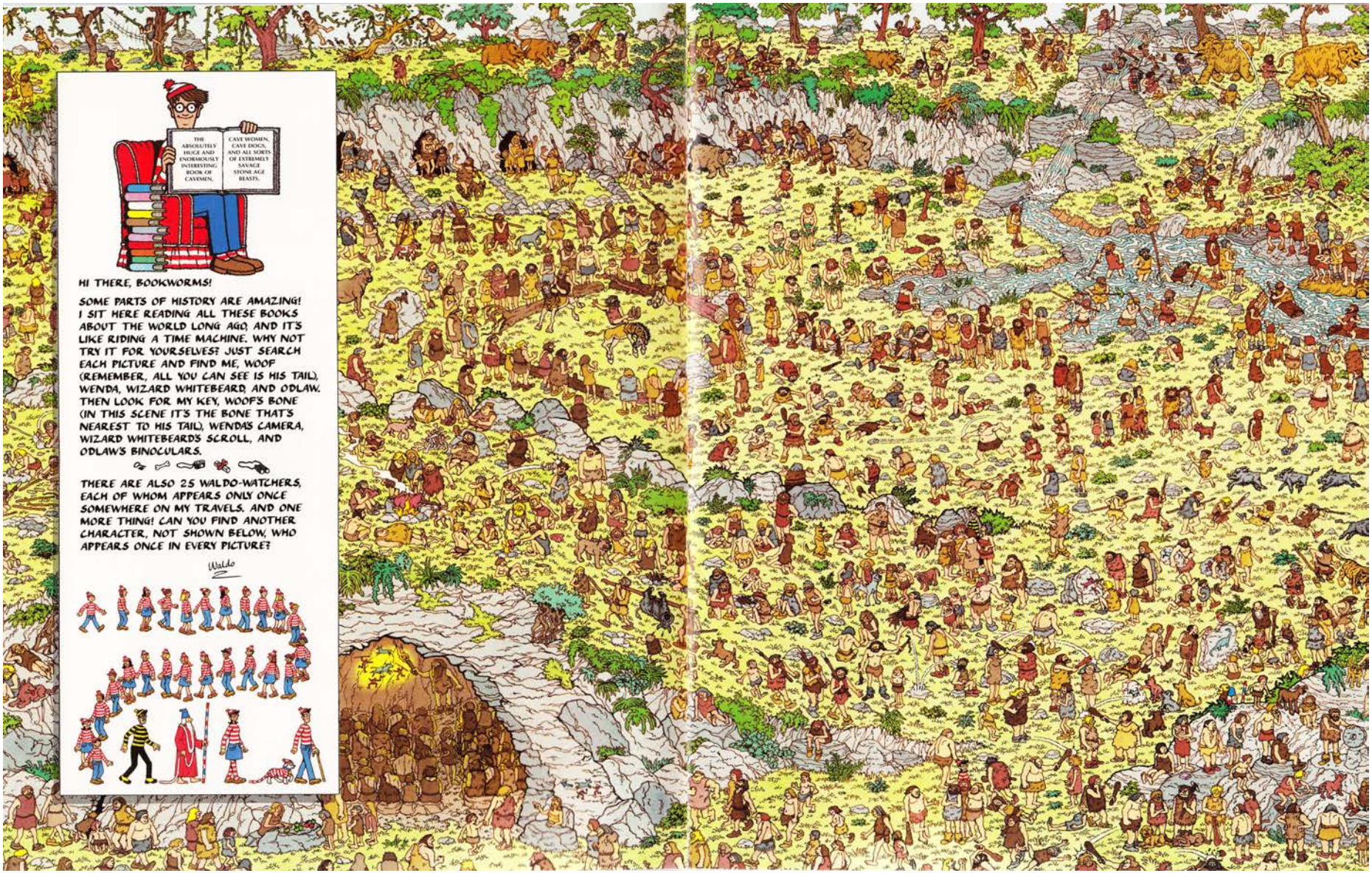
Ex. 3 (cont. re “Search & Find”)



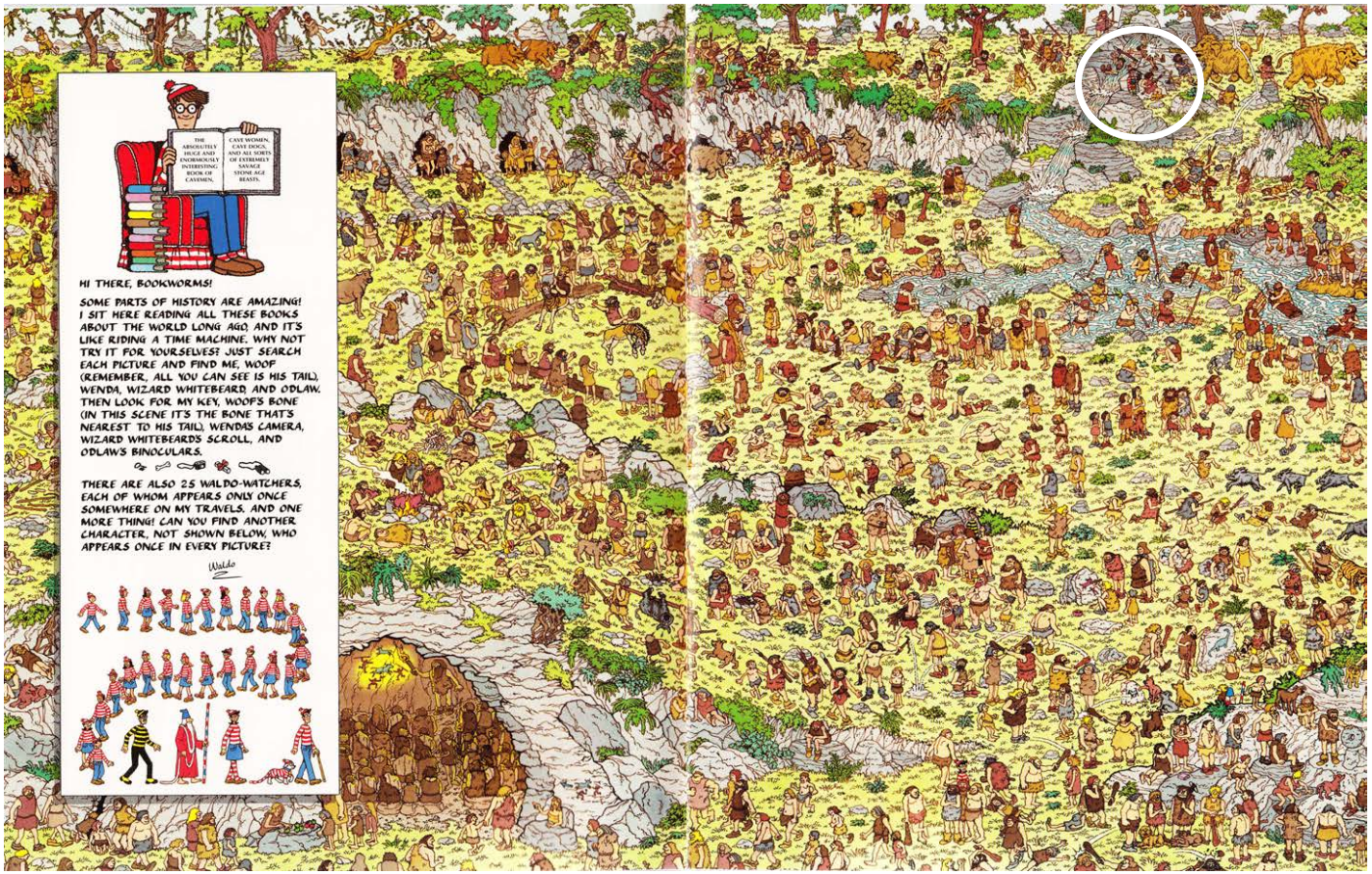
→ Essentially a problem in pattern recognition (e.g., computer vision)

Ex. 3 (cont. re “Search & Find”)

Ex. 3 (cont. re “Search & Find”)



Ex. 3 (cont. re “Search & Find”)



→ Desired item is different than the ‘template’ (or kernel)

Ex. 3 (cont. re “Search & Find”)

To further illustrate how hard this sort of “problem” is...

“An atlas of where proteins are found in cells will help work out what they do”

“This, being quite a task (and not one easily delegated to automatic optical-recognition systems), was carried out in part by a bunch of 120,000 enthusiastic amateurs.”

The Economist December 10th 2016

Molecular biology

Body of knowledge

An atlas of where proteins are found in cells will help work out what they do

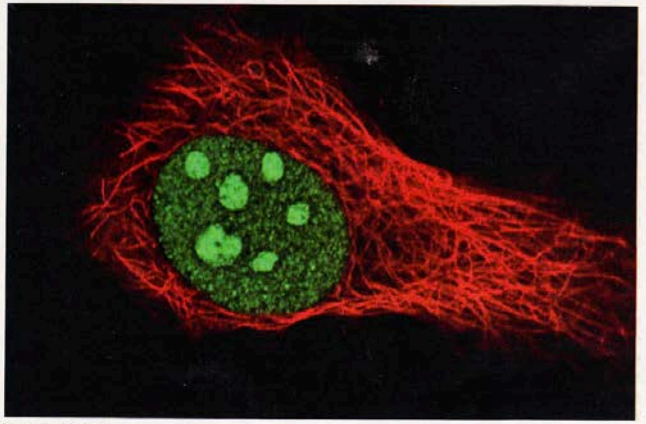
ONE of the most important concepts in biology is compartmentalisation. Different organs do different jobs within bodies. Different tissues do different jobs within organs. Different cells within tissues, likewise. And within cells, different organelles—as subcellular components such as nuclei, mitochondria and Golgi bodies are known—are also specialised for particular functions. Each of these levels of organisation has, over the years, been catalogued in what have come to be known as atlases, beginning in 1543 with Andreas Vesalius's "De Humani Corporis Fabrica" (On the Fabric of the Human Body), the founding text of modern anatomy.

The latest level of detail is to look at different proteins within organelles. Proteins are the molecules that do most of the work within a cell. They range from things like actin and myosin, which collaborate to flex muscle cells—and thus the muscles of which those cells are part—to the enzymes of the Krebs cycle, which disassemble glucose to release the energy therein. The Cell Atlas, a database launched on December 4th at a meeting of the American Society of Cell Biology, records which proteins are found in which organelles. The result, like "De Humani Corporis Fabrica", is both pleasing to the eye and important to the field. Proteins located together are likely to work together, so knowing a protein's whereabouts within a cell will help researchers to determine its job.

The authors of the Cell Atlas, led by Mathias Uhlen of the Royal Institute of Technology, in Stockholm, have pinned 12,051 proteins down in this way using immunofluorescence. This technique employs specially created antibodies (protein molecules, produced by immune-system cells, that bind specifically to a particular target protein) to hunt down that target within a cell. The antibodies themselves have fluorescent tags attached to them, so that they will glow when exposed to ultraviolet light. By applying these tagged antibodies to cells from 22 human-cell lines derived from a range of original tissues, the atlas's authors gave themselves the best possible chance of detecting a particular protein in cells of at least one line. That done, each sample was examined microscopically to determine which of the 13 generally recognised organelles each protein appeared in. This, being quite a task (and not one easily delegated to automatic optical-recognition systems), was carried out in part by a bunch of 120,000 enthusiastic amateurs.

The example in the picture is of the distribution of a protein (tagged green) called ZNF554. This belongs to a group, the zinc-finger transcription factors, whose job is to activate and regulate genes. As the picture shows, ZNF554 is restricted to the cell's nucleus. Within that nucleus there are several places which glow particularly brightly, and where it is therefore particularly abundant. These are the nucleoli—zones where genes are especially active. (The red areas are the cell's microtubules. These act as its skeleton and are tagged in all Cell Atlas pictures, in order to outline a cell's limits.)

What fraction of human proteins the Cell Atlas currently covers is open to debate. The number of protein-coding genes in the human genome is currently reckoned at 19,628. Some genes, however, can yield more than one protein—either be-

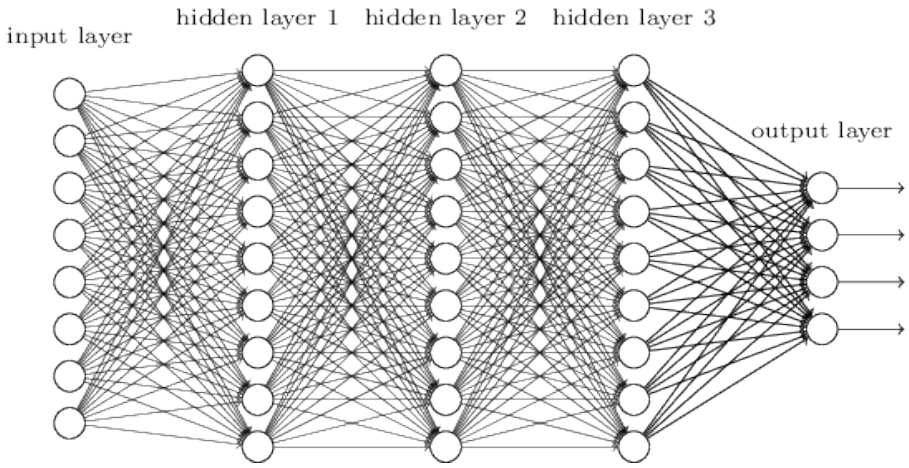


Spot the nucleoli

Aside: Deep Learning

The cover of the January 28, 2016 issue of Nature, which features Google's groundbreaking AI research.

- Machine learning
- Neural networks
- "Deep learning"



http://neuralnetworksanddeeplearning.com/chap6.html

Note: This stuff is beyond the scope of PHYS 2030

Ex. 4

"the staple problem"

→ An easier challenge?

→ Empty bulletin board

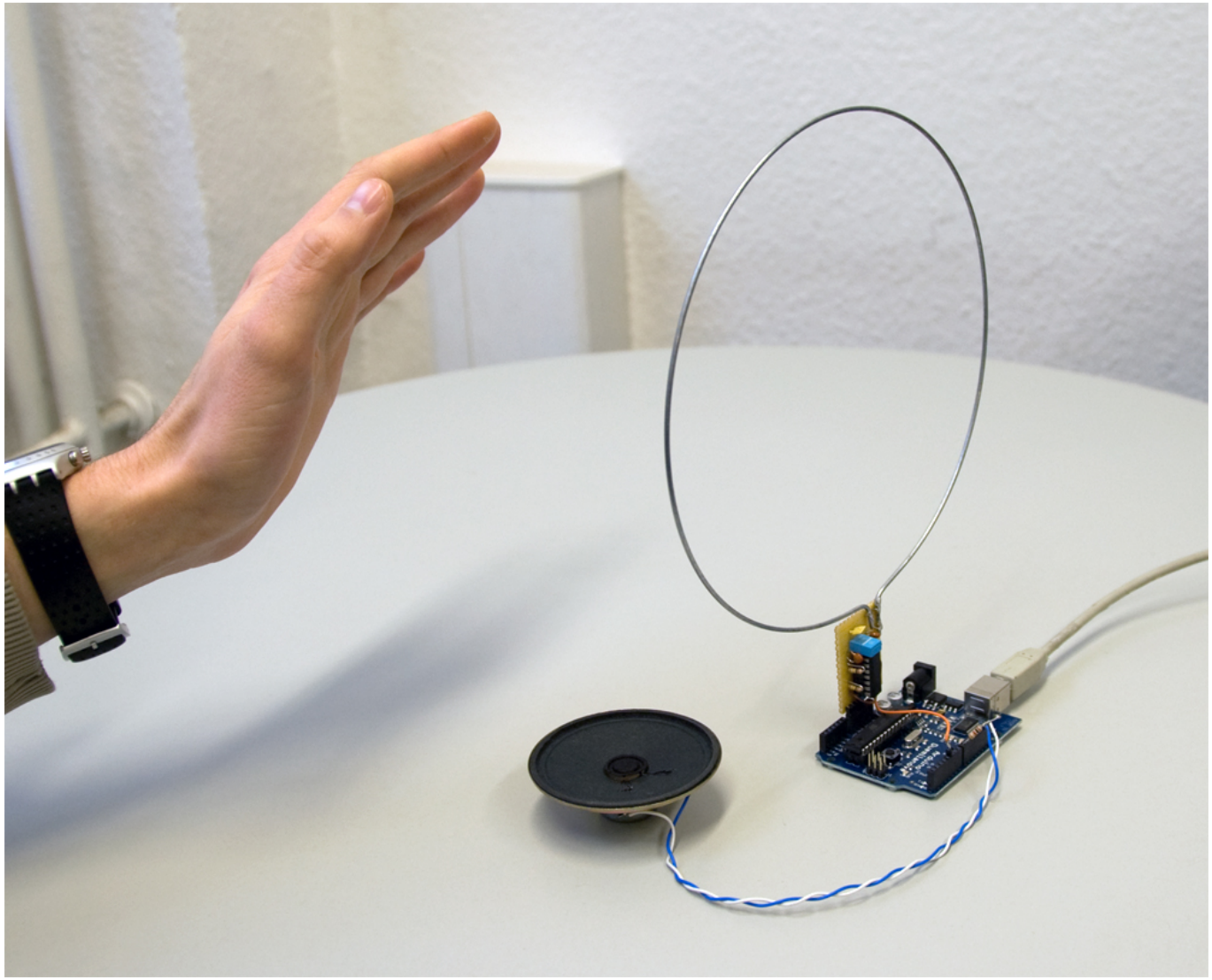
Ex. 4

- How many staple are there?
- Distribution of angles?
- How to write a code to objectively measure such?

Aside

→ More details later in when we deal w/ “DAQ”

ARDUINO™



http://interface.khm.de/index.php/lab/interfaces-advanced/theremin-as-a-capacitive-sensing-device/

Review (re CSE 1541)

- basics of computing: digital versus analog, binary
- overview of Matlab and basic format/syntax
- numbers versus strings
- logicals & relational operators
- vectors, arrays, structures, etc…. (+ indicies)
- basic programming, including loops and conditionals
- basic etiquette (e.g., naming variable, commenting code)
- data types
- data file input/output
- basic statistics
- plotting data
- user-defined functions
- Scripts
- matrix manipulations (e.g., Gaussian elimination)
- random numbers
- (basic) curve fitting
- recursive relations, bisection
- (basic) numerical differentiation, integration
- (basic) solving ODEs

Function functions

- let's find a root of

$$f(x) = \sqrt{x}\tan(\pi\sqrt{x}) - \sqrt{1-x}$$

which has the derivative

$$f'(x) = \frac{1}{2}\left(\frac{1}{\sqrt{1-x}} + \frac{\tan(\pi\sqrt{x})}{\sqrt{x}} + \pi\sec^2(\pi\sqrt{x})\right)$$

Function functions

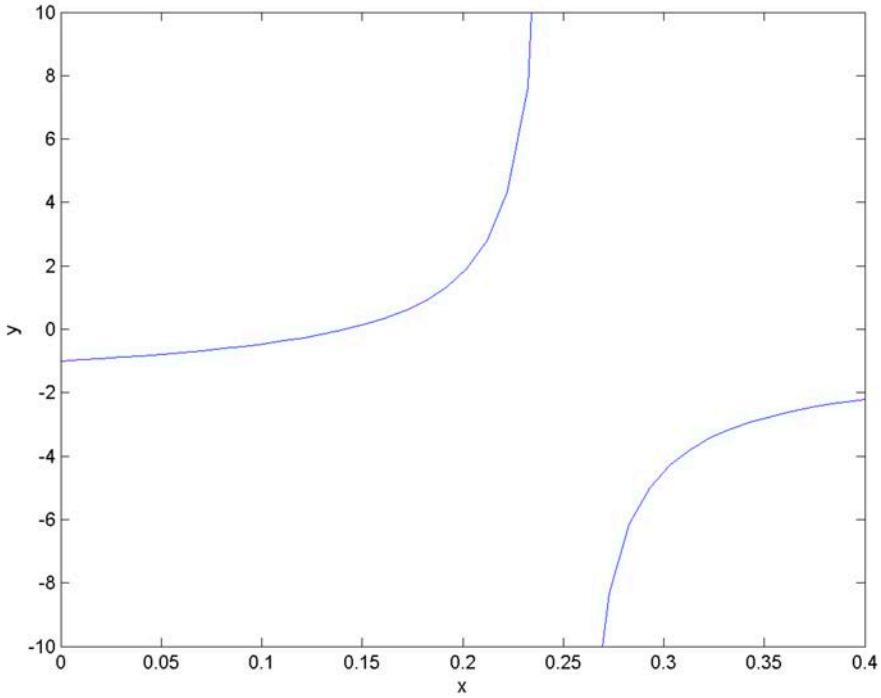
- let's find a root of

$$f(x) = \sqrt{x}\tan(\pi\sqrt{x}) - \sqrt{1-x}$$

which has the derivative

$$f'(x) = \frac{1}{2}\left(\frac{1}{\sqrt{1-x}} + \frac{\tan(\pi\sqrt{x})}{\sqrt{x}} + \pi\sec^2(\pi\sqrt{x})\right)$$

plot of $f(x)$



Review (re CSE 1541)

```
function [y] = myf(x)
%MYF Function to find the root of

y = sqrt(x) .* tan(pi * sqrt(x)) - sqrt(1 - x);

end
```

```
function [y] = myfprime(x)
%MYFPRIME Derivative of MYF

sqrtx = sqrt(x);
a = 1 / sqrt(1 - x);
b = tan(pi * sqrtx) / sqrtx;
c = pi * (sec(pi * sqrtx))^2;
y = 0.5 * (a + b + c);

end
```

Review (re CSE 1541)

function handles

```
function [ root, xvals ] = newton(f, fprime, x0, epsilon)
%NEWTON Newton's method for root finding
%   ROOT = NEWTON(F, FPRIME, X0, EPSILON) finds a root of the
%   function F having derivative FPRIME using Newton's method
%   starting from an initial estimate X0 and a tolerance EPSILON
%
%   [ROOT, XVALS] = NEWTON(F, FPRIME, X0, EPSILON) also returns
%   the iterative estimates in XVALS

xvals = x0;
xi = x0;
while abs(f(xi)) > epsilon
   xj = xi - f(xi) / fprime(xi);
   xi = xj;
   xvals = [xvals xi];
end
root = xi;

end
```

local functions have been removed

Function functions

- we can now use our Newton's method implementation by passing in function handles for **`f`** and **`fprime`**

```
newton(@myf, @myfprime, 0.1, 1e-6)
```

if-statement

- create a function that has an optional last parameter

```
function [x, v] = dhmotion(A, m, k, b, phi, t)
%DHMOTION Damped harmonic motion from Lab 4
%    x = dhmotion(A, m, k, b, phi) computes
%    the position of a damped harmonic oscillator
%    at time t = linspace(0, 1)

if nargin == 5
  t = linspace(0,1);
end

% the rest of your function goes here...
```

nargin is the number of input arguments supplied by the caller

Ex. 4 "staple problem" REDUX

→ As an aside, we'll outline a "blackbox" method that makes use of Matlab's "toolboxes" (something you may or may not have access to)

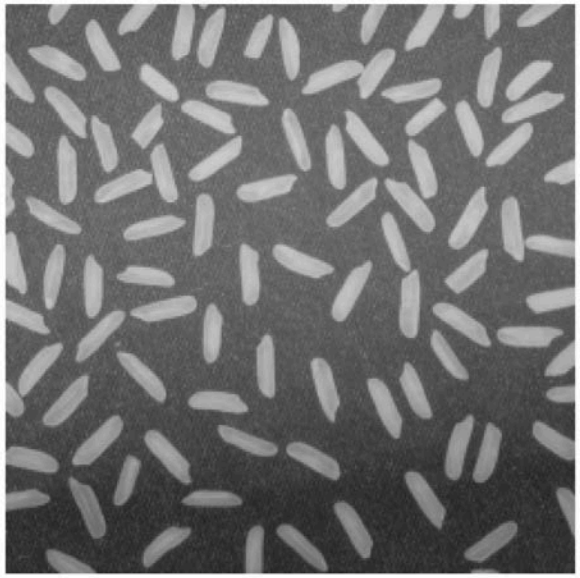
Image Processing Toolbox

https://www.mathworks.com/help/images/examples.html

Ex. 4 "staple problem" REDUX

Step 1: Read Image

```
I = imread('rice.png');
imshow(I)
```



Step 4: Increase the Image Contrast

```
I3 = imadjust(I2);
imshow(I3);
```

https://www.mathworks.com/help/images/examples/correcting-nonuniform-illumination.html

Ex. 4 "staple problem" REDUX

Step 5: Threshold the Image

Create a new binary image by thresholding the adjusted image. Remove background noise with `bwareaopen`.

```
bw = imbinarize(I3);
bw = bwareaopen(bw, 50);
imshow(bw)
```

→ Now up to this point, the underlying "computations" are relatively straight-forward. But things are about to get mysterious...

https://www.mathworks.com/help/images/examples/correcting-nonuniform-illumination.html

Ex. 4 "staple problem" REDUX

Step 6: Identify Objects in the Image

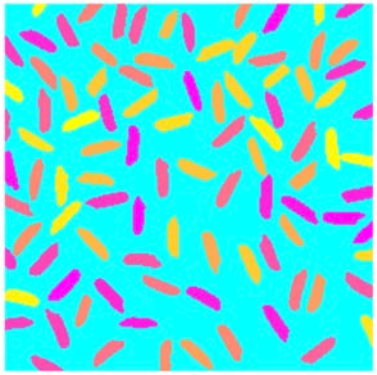
The function bwconncomp finds all the connected components (objects) in the binary image. The accuracy of your results depend on the size of the objects, the connectivity parameter (4,8,or arbitrary), and whether or not any objects are touching (in which case they may be labeled as one object). Some of the rice grains in bw are touching.

```
cc = bwconncomp(bw, 4)
```

```
cc =

  struct with fields:

    Connectivity: 4
       ImageSize: [256 256]
      NumObjects: 95
    PixelIdxList: {1×95 cell}
```



Step 8: View All Objects

One way to visualize connected components is to create a label matrix and then display it as a pseudo-color indexed image.

Use `labelmatrix` to create a label matrix from the output of `bwconncomp`. Note that `labelmatrix` stores the label matrix in the smallest numeric class necessary for the number of objects.

```
labeled = labelmatrix(cc);
whos labeled
```

```
  Name           Size              Bytes  Class    Attributes

  labeled      256x256             65536  uint8
```

In the pseudo-color image, the label identifying each object in the label matrix maps to a different color in the associated colormap matrix. Use `label2rgb` to choose the colormap, the background color, and how objects in the label matrix map to colors in the colormap.

```
RGB_label = label2rgb(labeled, @spring, 'c', 'shuffle');
imshow(RGB_label)
```

→ Beware the mysterious "blackbox" at work here!!

https://www.mathworks.com/help/images/examples/correcting-nonuniform-illumination.html