Computational Simulations in Bioengineering

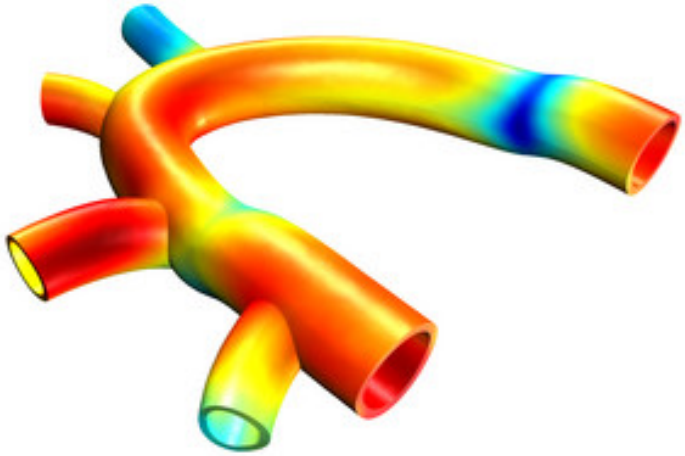
Never Stand Still | Faculty of Engineering | Graduate School of Biomedical Engineering

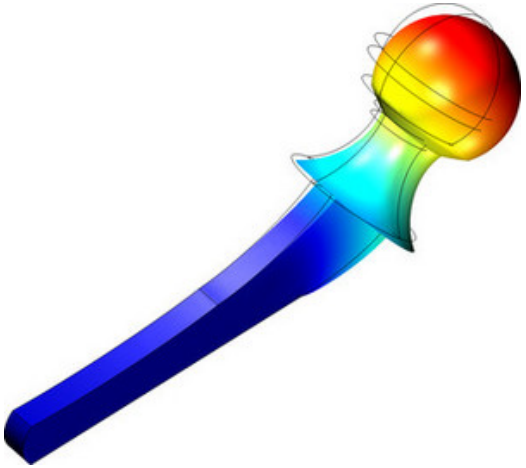
Socrates Dokos, Associate Professor
Graduate School of Biomedical Engineering

BIOM1010, 2 October 2018

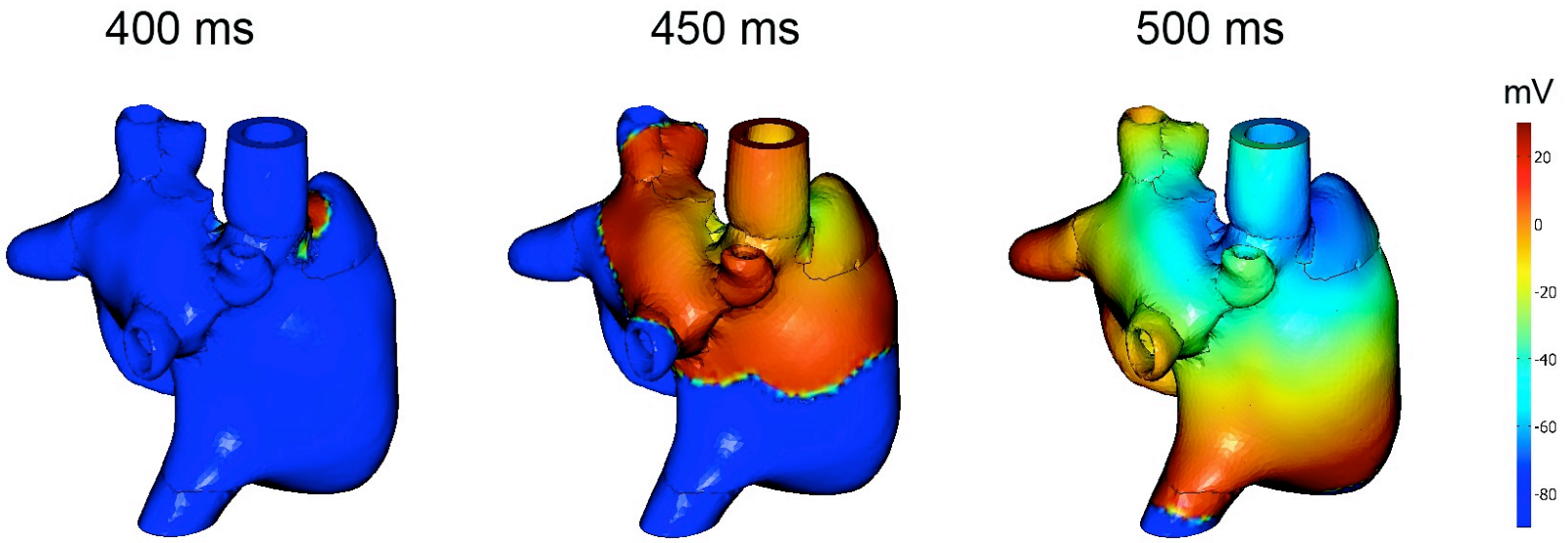
Computational Simulations - Modelling

- *Modelling* is defined as the formulation of a mathematical (or computational) representation of a physical system.
- Used to predict behaviour of the system.
- Models are typically solved by computer.
- In bioengineering, modelling is used to study the behaviour of complex physiological systems and their interaction with external devices.

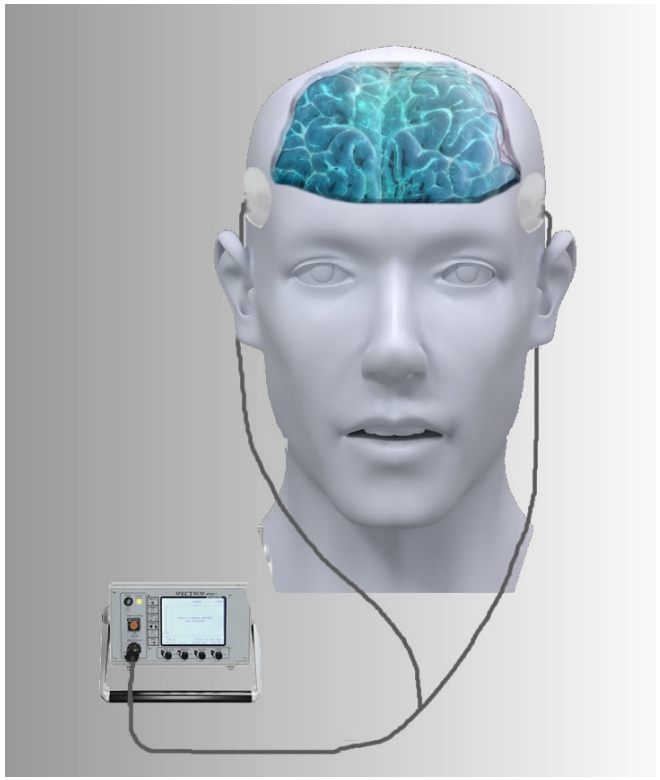


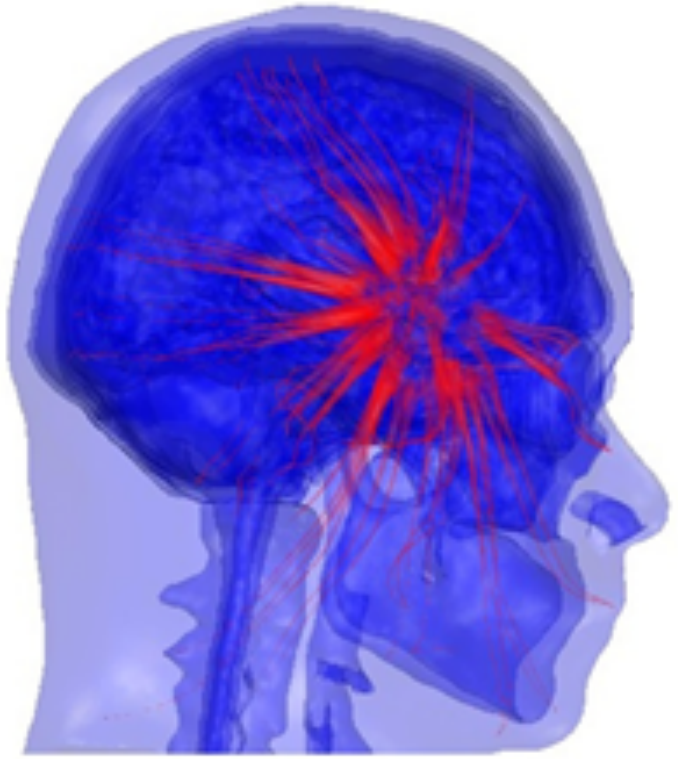


Modelling Applications - Simulations of Cardiac Electrical Activity



Other Modelling Applications - Electroconvulsive Therapy (ECT)





Finite Element (FE) Model Reconstruction

CT/MRI Image

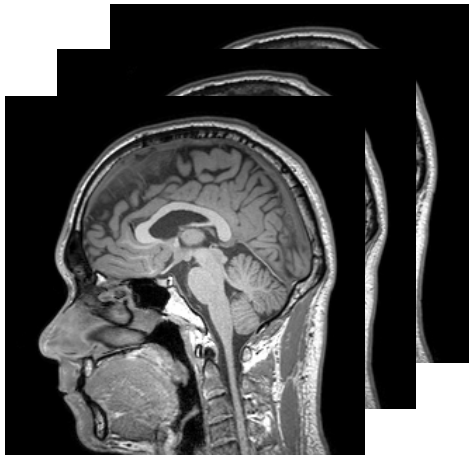
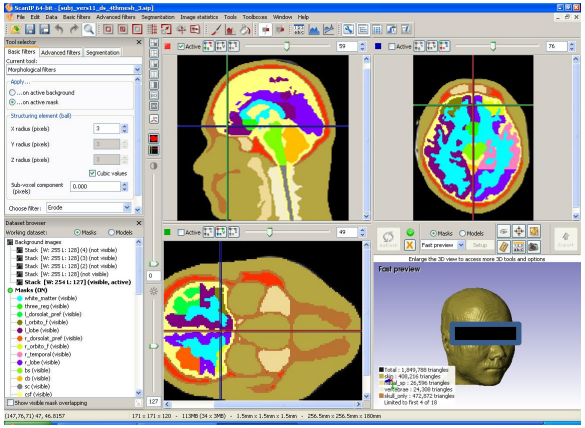
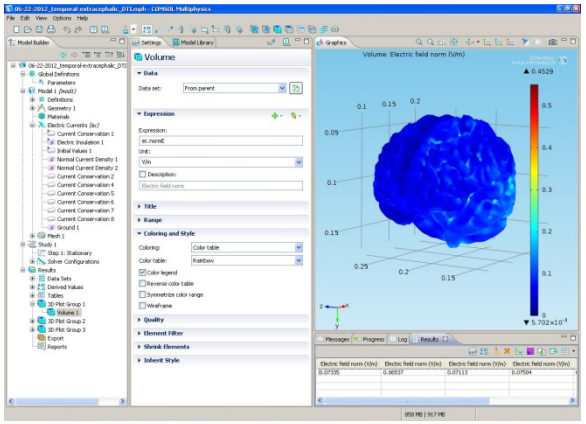


Image segmentation
& FE mesh generation



FE mesh

FE model solver

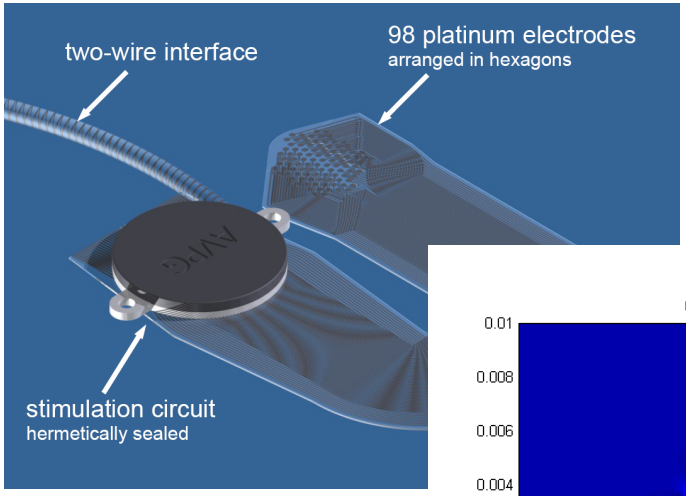


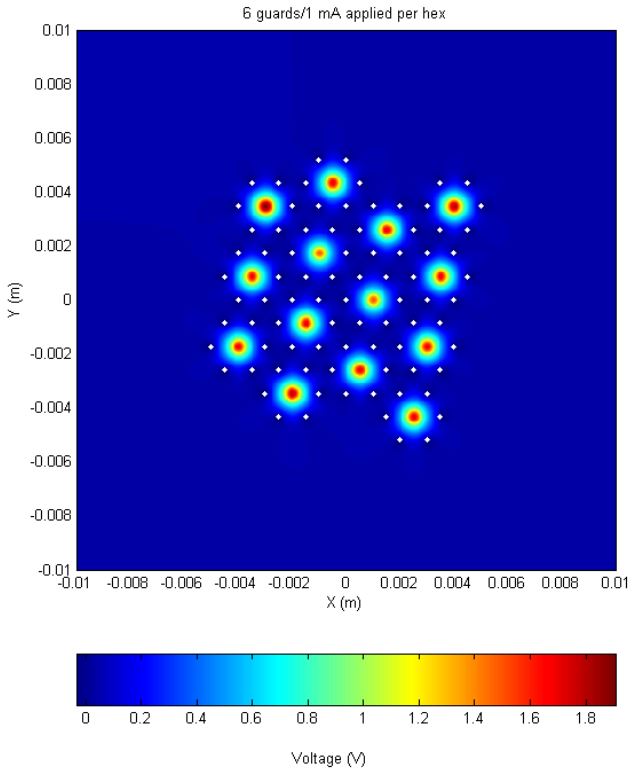
Partial differential equations

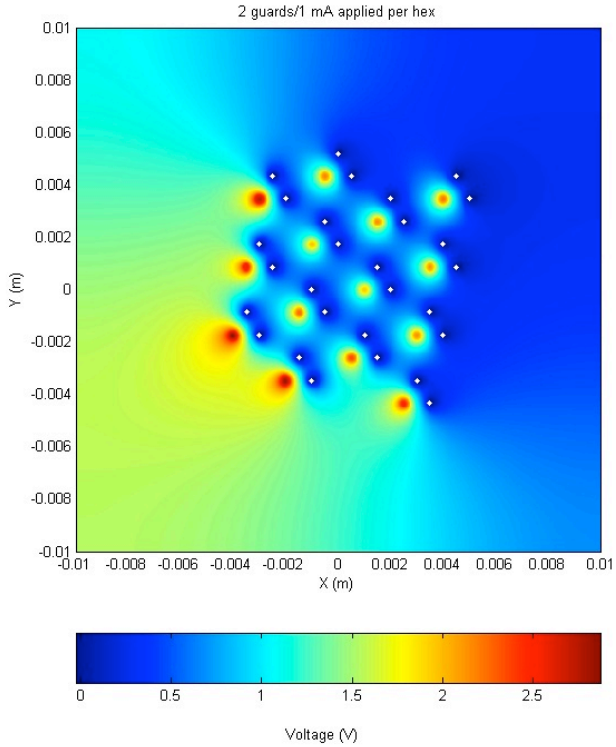
$$\nabla \cdot (-\boldsymbol{\sigma} \nabla V) = 0$$

Parameters including stimulus
parameters and conductivities

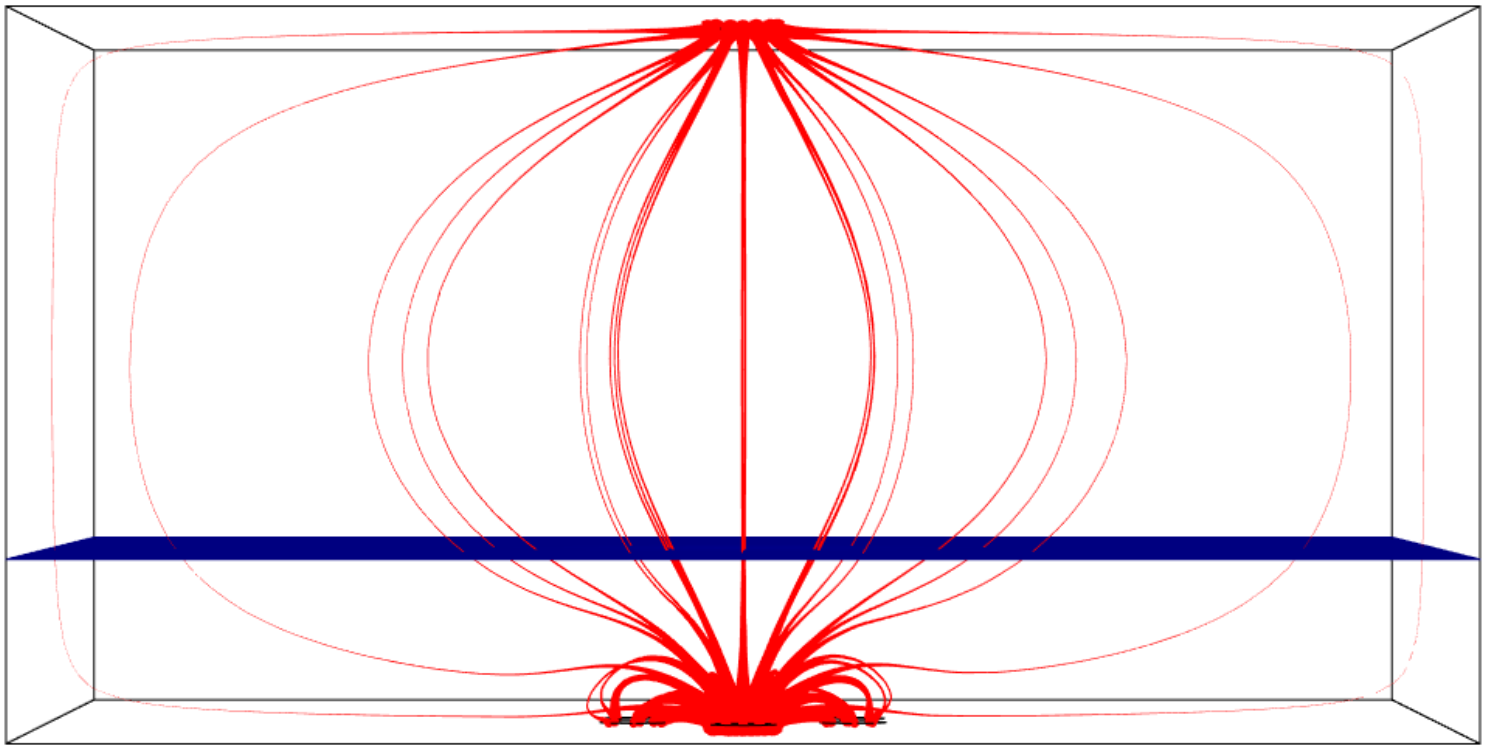
Modelling Applications – Vision Prosthesis



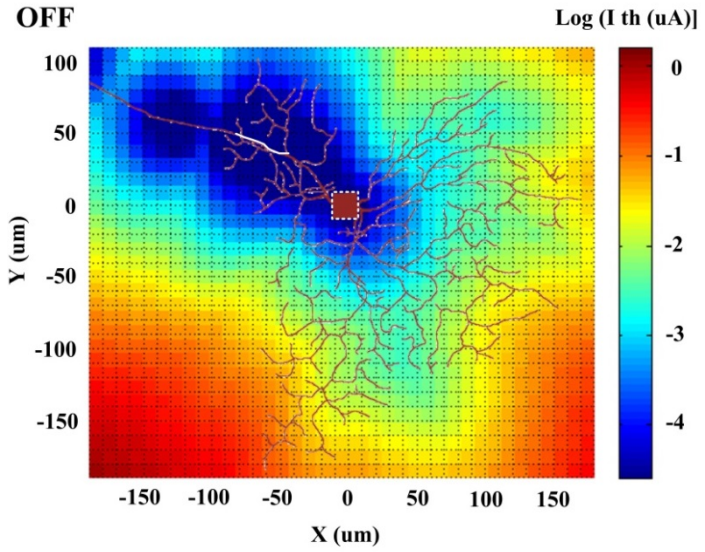


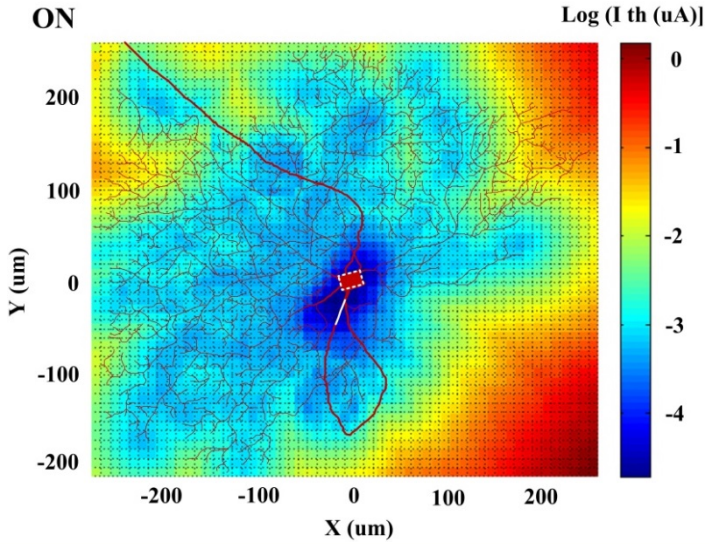


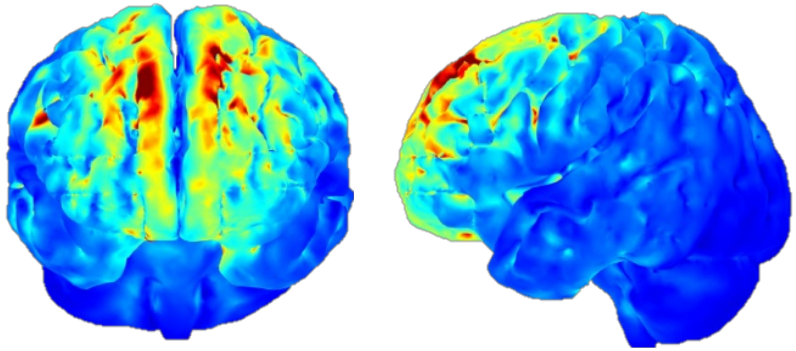
Quasi-Monopolar Stimulation

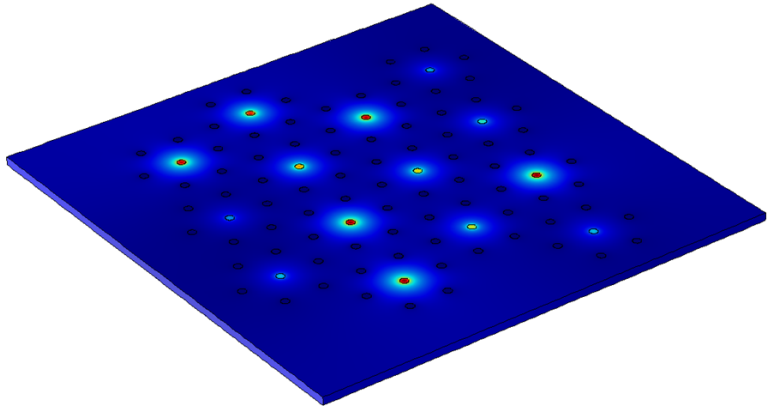


Modelling Neural Stimulation



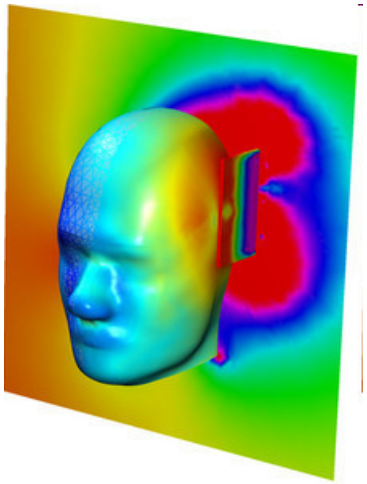




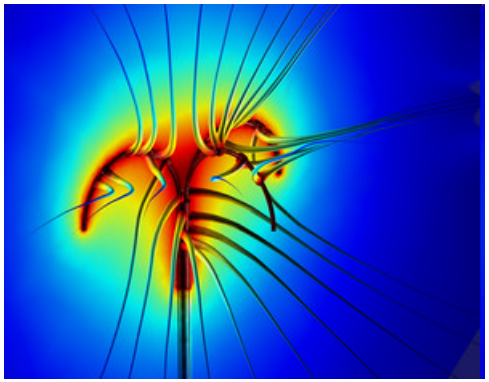


Other Examples…

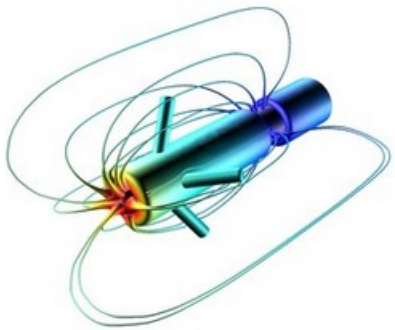
Radiation Exposure



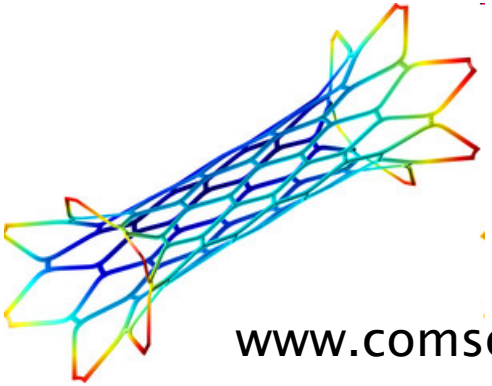
Hepatic Tumour Ablation



Pacemaker Electrode



Stent Deformation



www.comsol.com

Computational Fluid Dynamics (CFD) Modelling

- **Fluid motion governed by incompressible Navier-Stokes equations:**

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla p + \mu\nabla^2\mathbf{u}$$
$$\nabla\cdot\mathbf{u} = 0$$

Variables

$\mathbf{u}$: fluid velocity vector

p: fluid pressure

Parameters

ρ: fluid density

μ: fluid viscosity

Mathematical Operators

∂: partial derivative

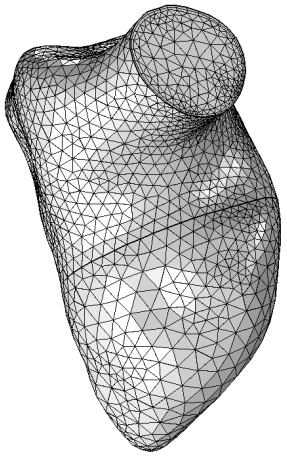
$\cdot$: vector dot product

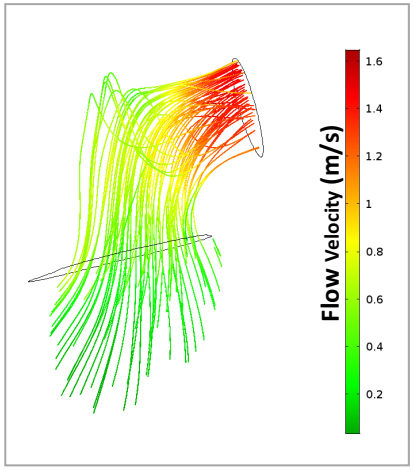
∇: nabla (or del) operator

∇^2: Laplacian operator

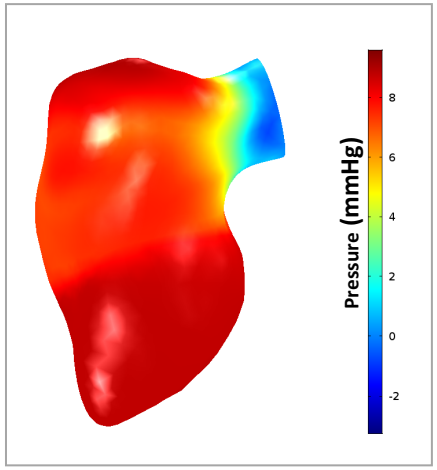
$$\nabla \equiv \begin{pmatrix} \partial/\partial x \\ \partial/\partial y \\ \partial/\partial z \end{pmatrix}$$

$$\nabla^2 \equiv \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$$



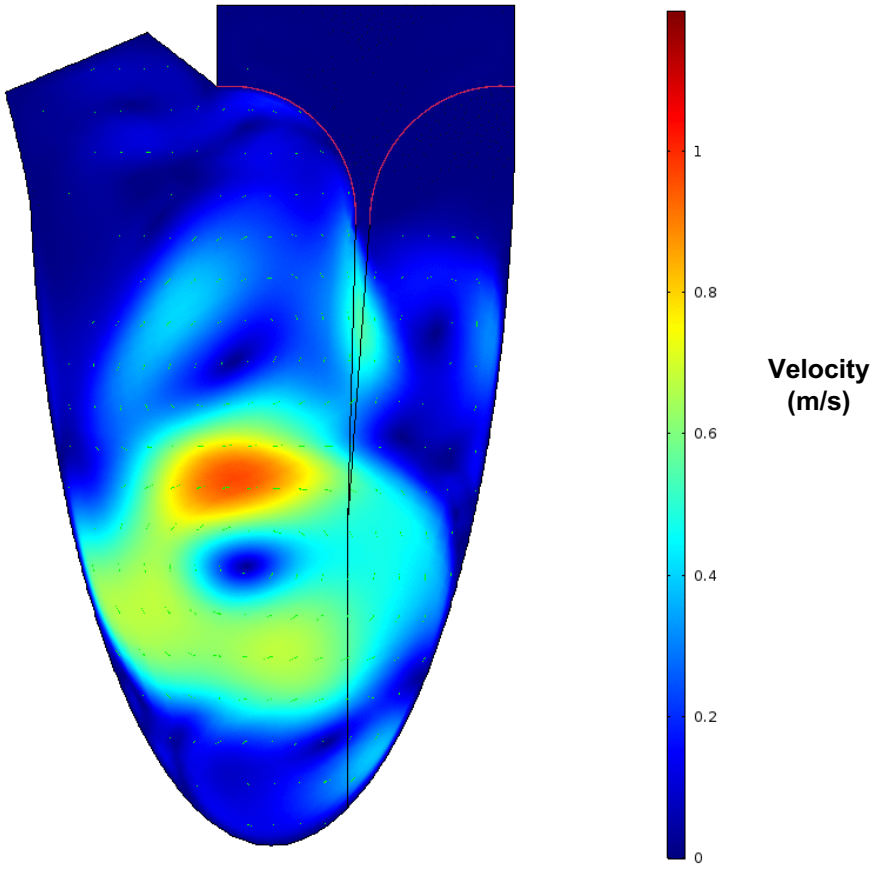


Streamline plot of velocity field



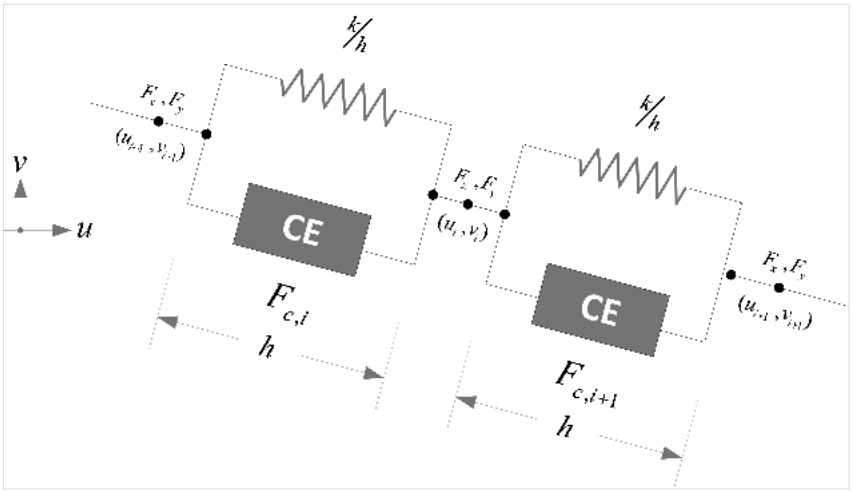
Pressure

Modelling Fluid-Structure Interaction in the Heart

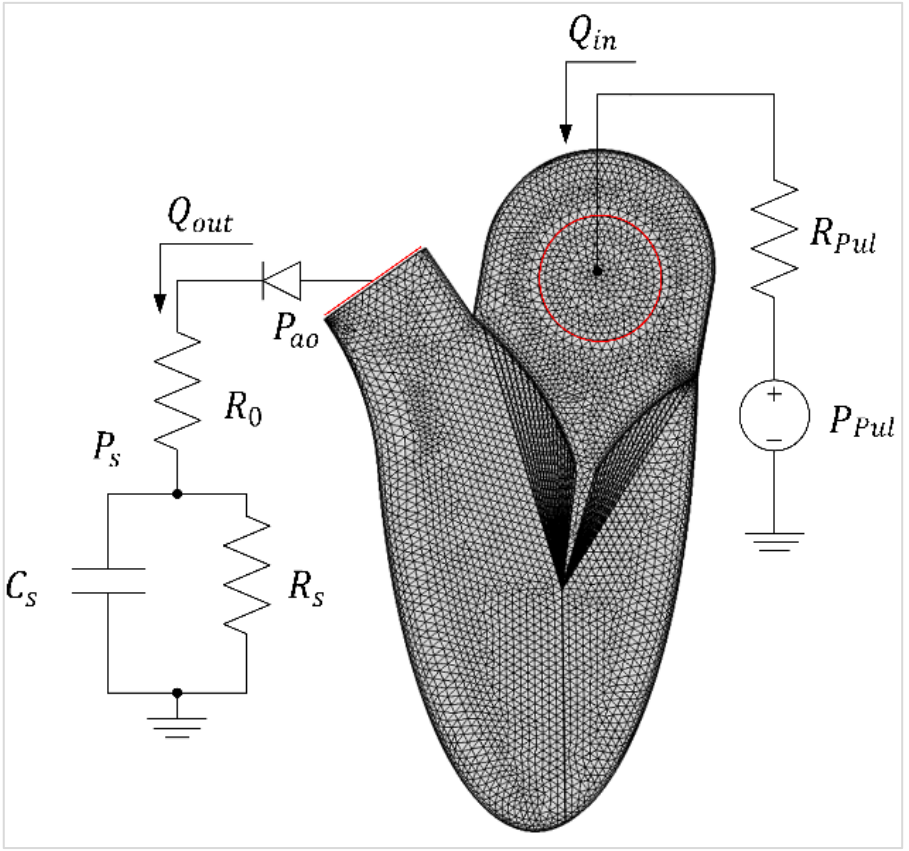


2D Model of the Left Heart

- **Muscular model of LV, LA wall:**
  (MV leaflets is spring-like model only)

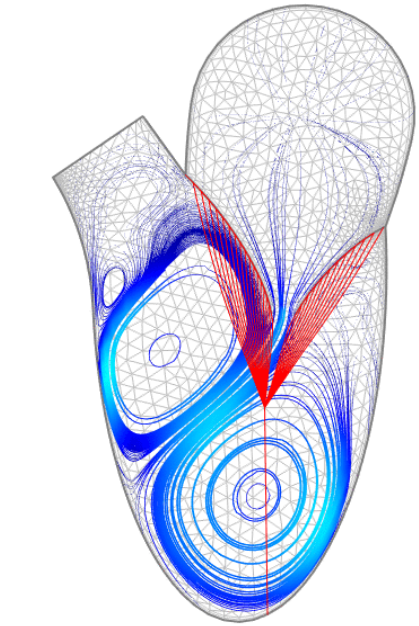


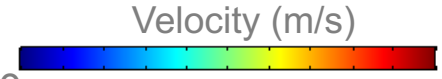
- **2D fully coupled model:**
  with pre- and after- load circuits:

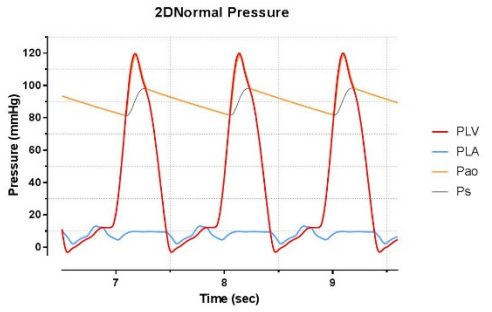


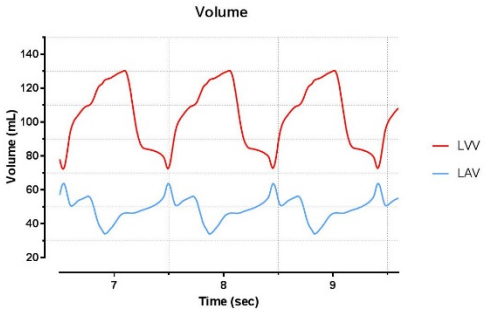
2D Model of the Left Heart

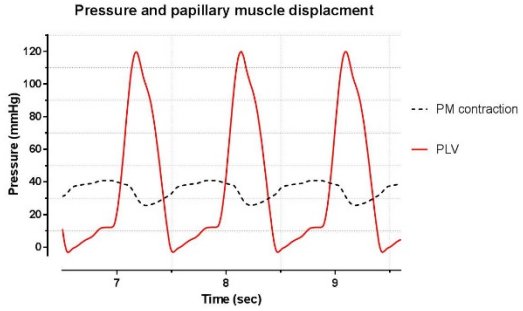
Model Results:

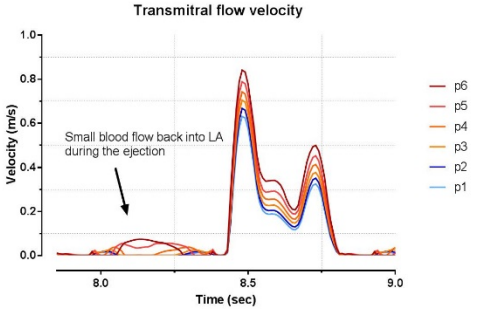


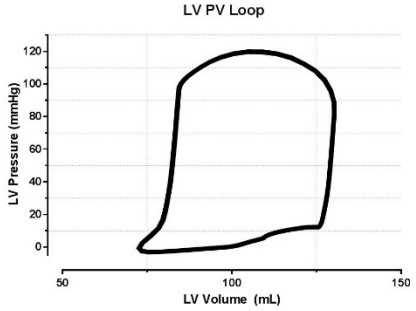


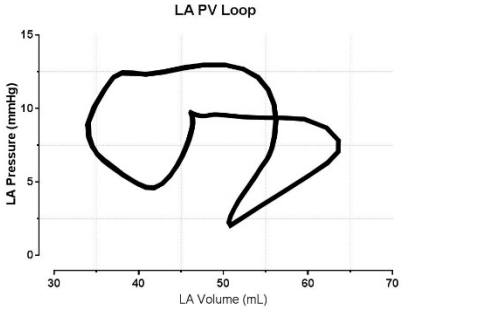






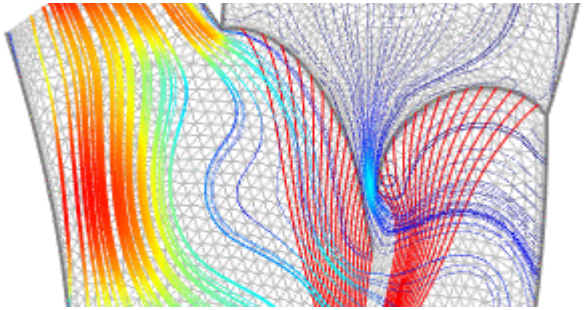


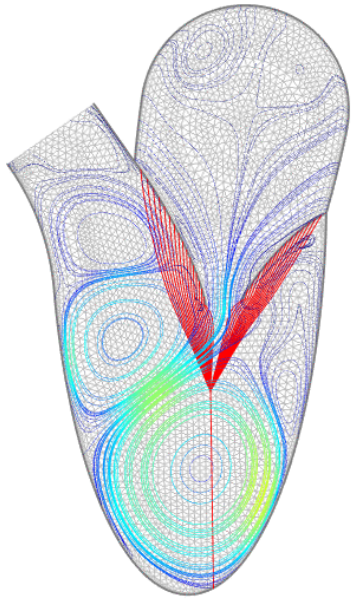


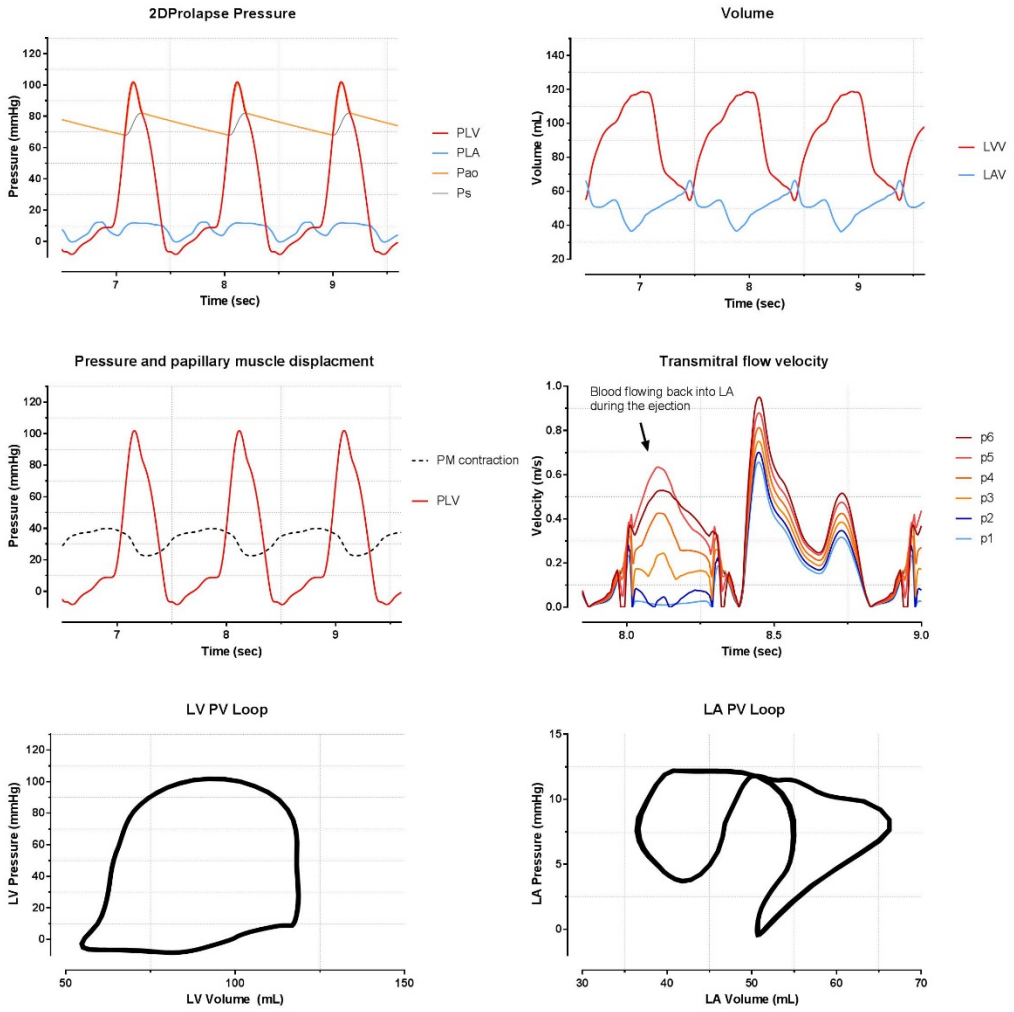


2D Model of the Left Heart
Applications

Mitral Valve (MV) prolapse:
By elongating the posterior chords

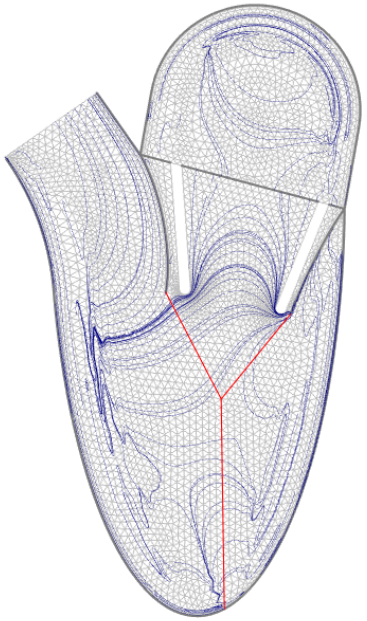


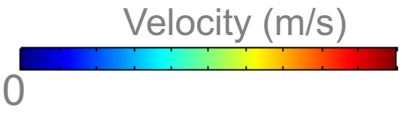


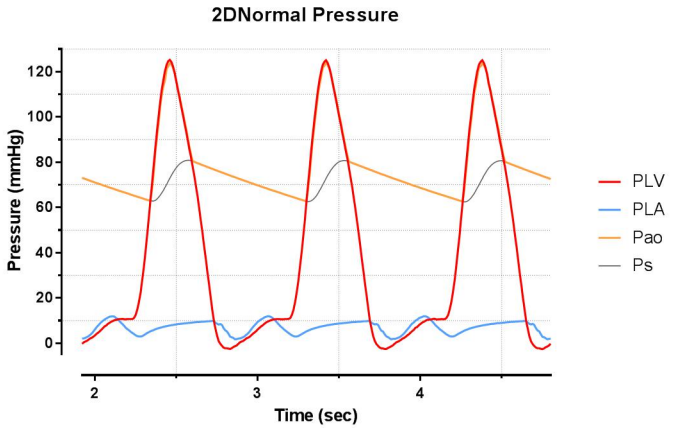


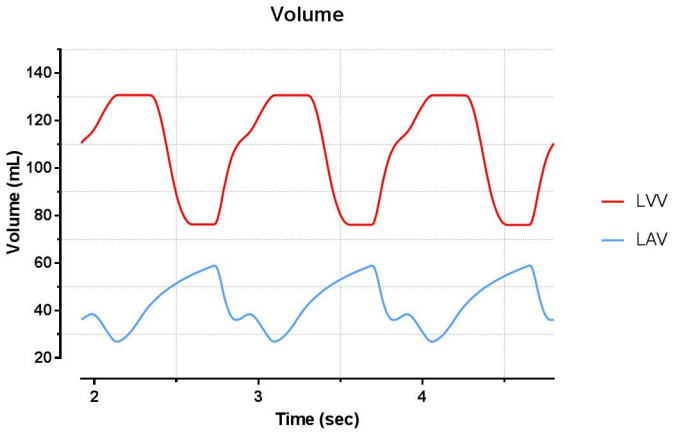
2D Model of the Left Heart:
Applications

With MV prosthesis



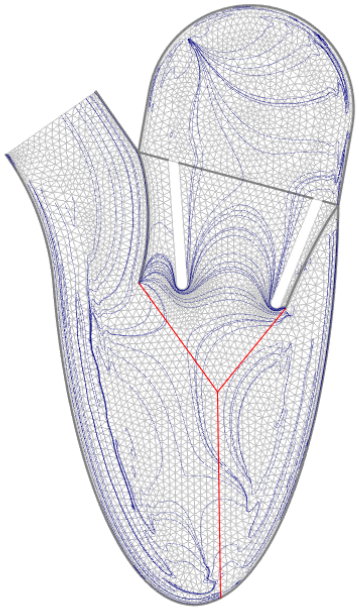


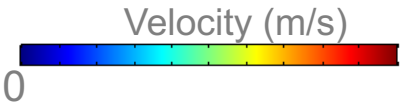


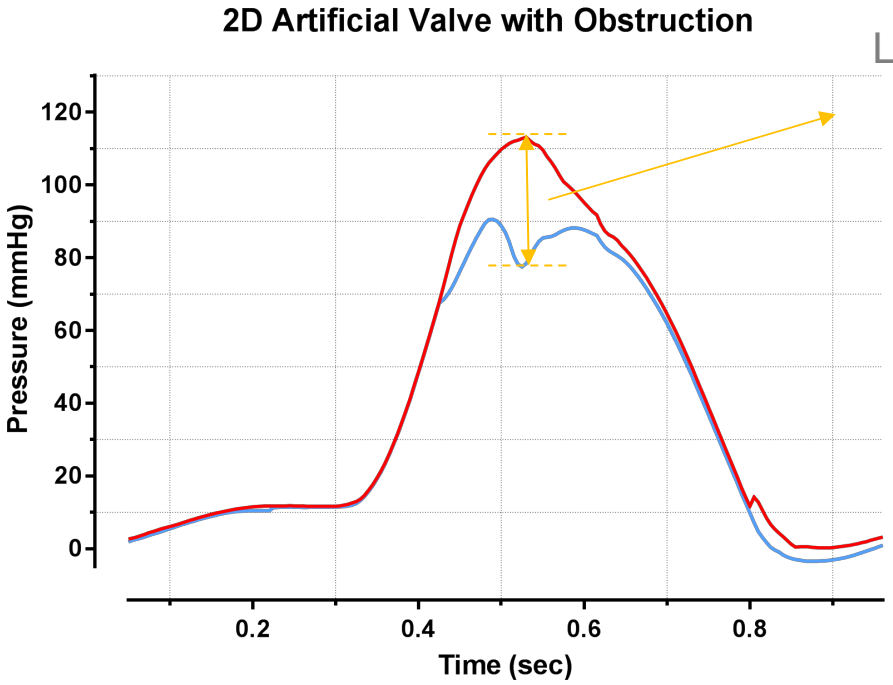


2D Model of the Left Heart:
Applications

Prosthesis with obstruction:
By cutting the anterior chord attachments



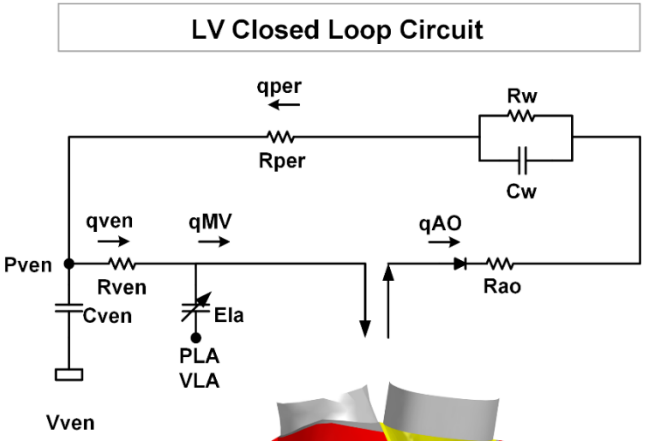




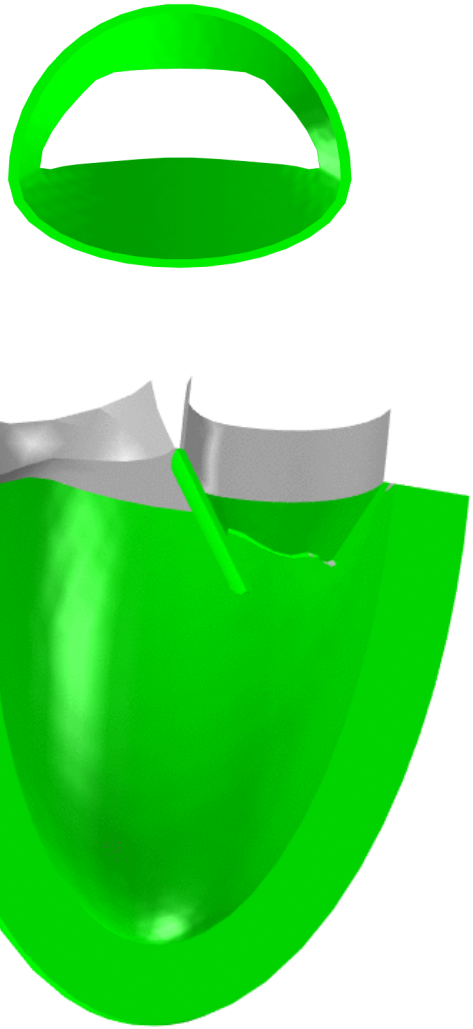
LVOT Pressure gradient

Near to Apex
Near to Ao

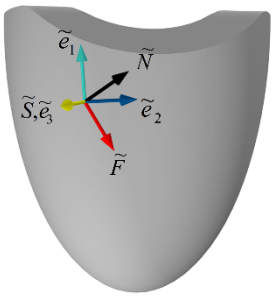
3D Idealized model of LV MV

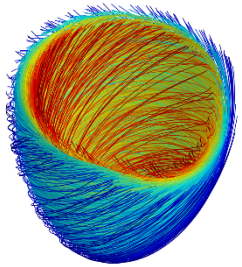


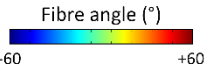
- Geometry

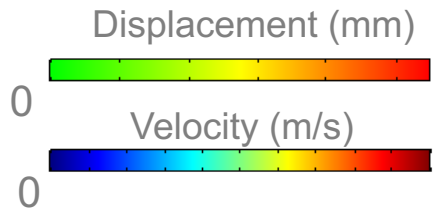


- Myocardial fibre orientation

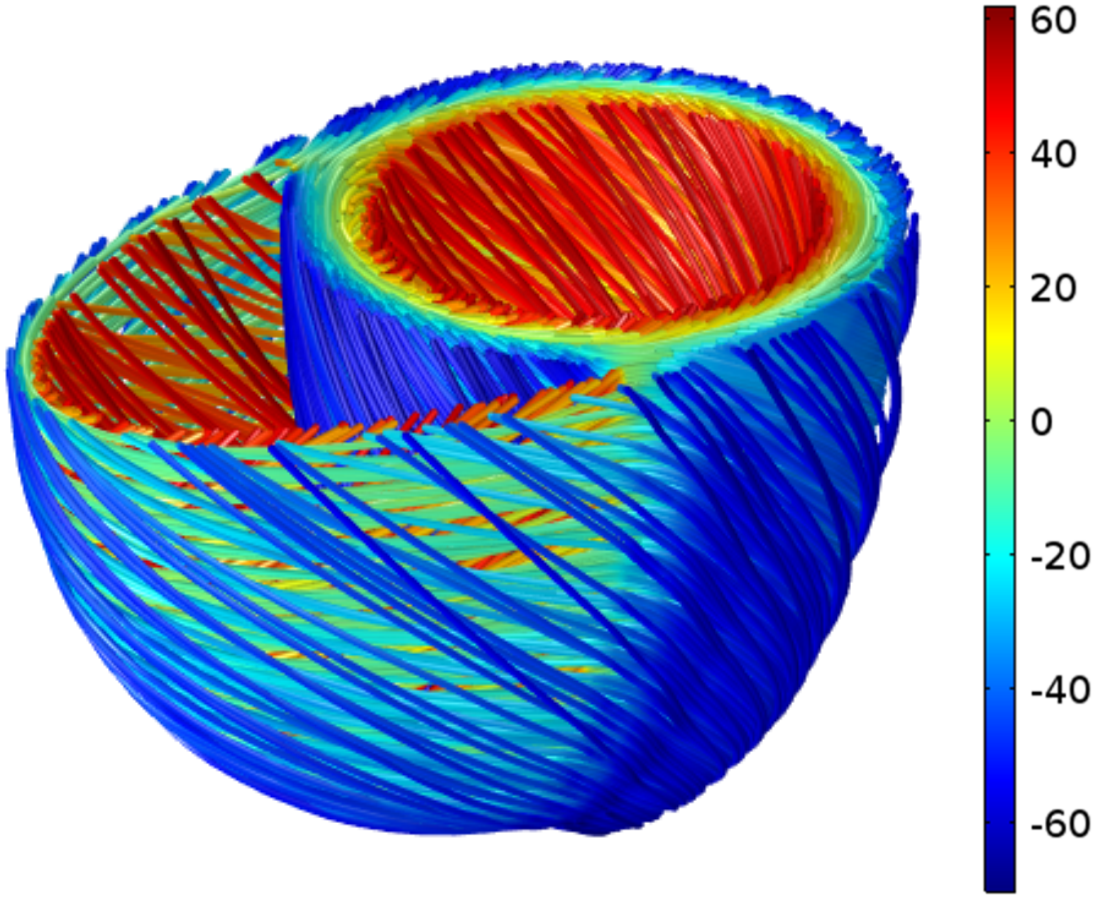




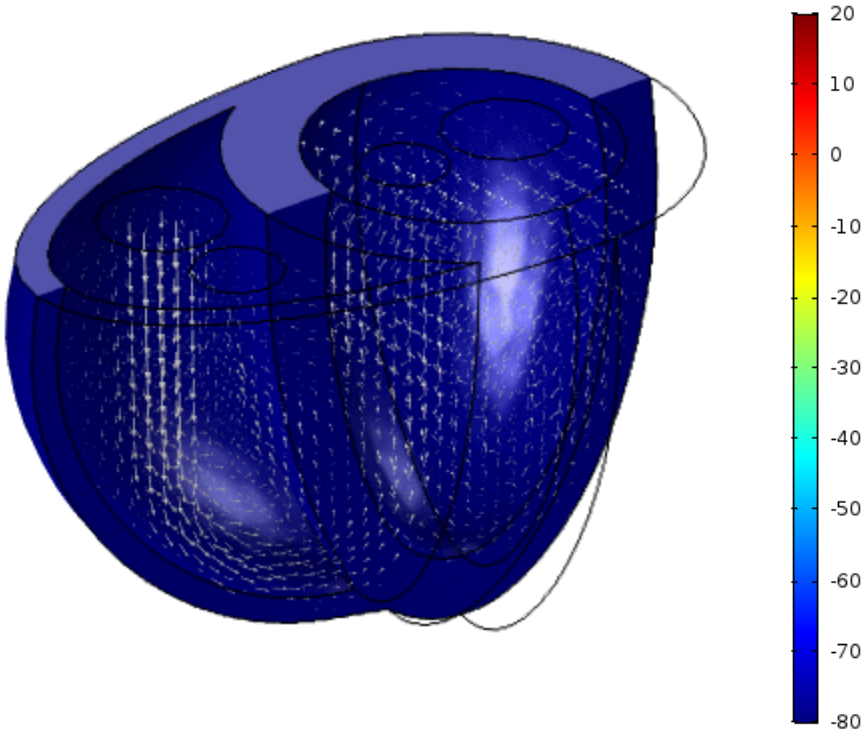




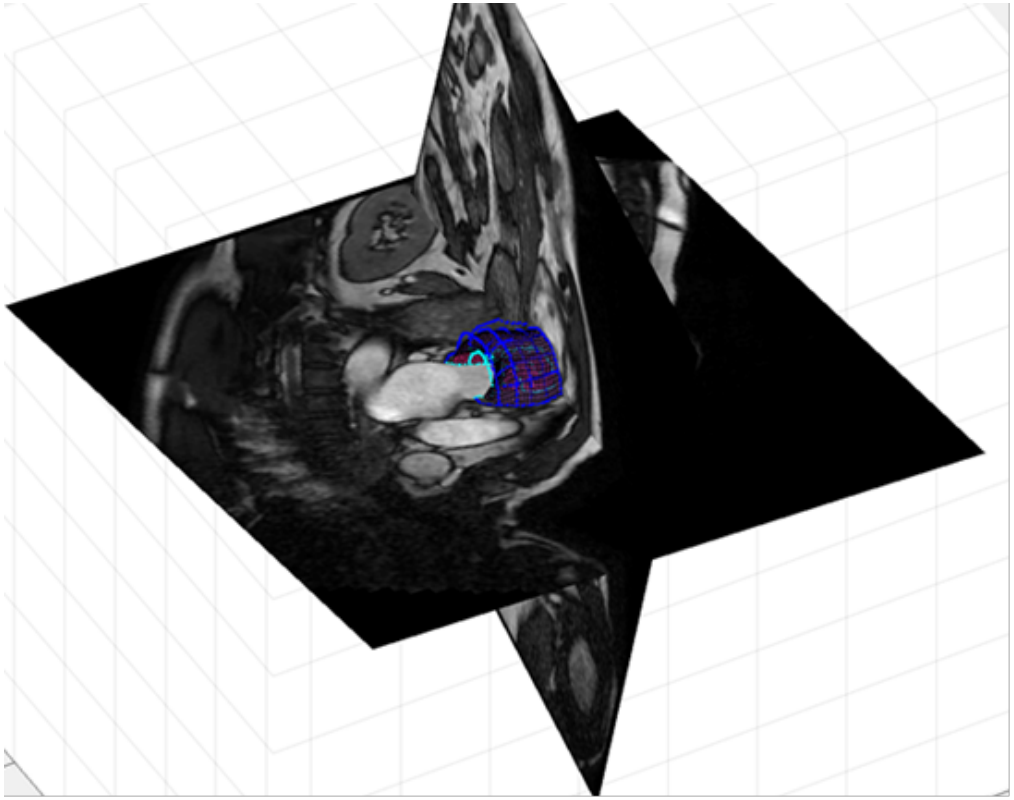
Biventricular Fibre Orientation

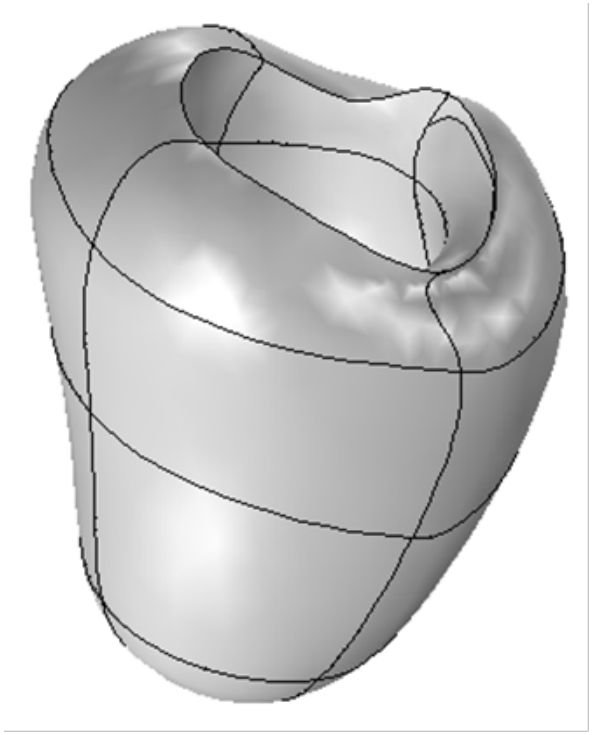


Simulating Biventricular Cardiac Contraction

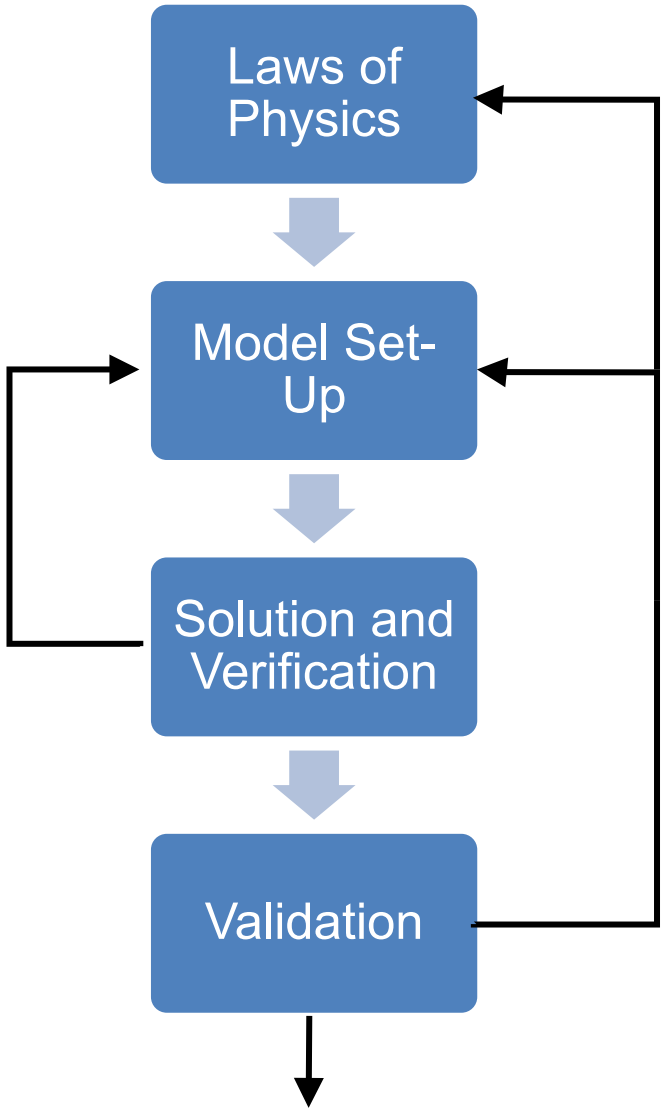


Developing Patient-Specific Models





The Modelling Process



Laws of Physics

- In physical systems, natural phenomena are described by fundamental laws of physics, which include:
  - conservation of mass
  - conservation of energy
- Laws of physics are expressed in terms of ***differential equations***.

e.g. $$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}.\nabla\mathbf{u}\right) = -\nabla p + \mu\nabla^2\mathbf{u} + \mathbf{f}$$

- Differential equations describe the change in a quantity over time and space.

Another Example - Electromagnetism

- The theory of electromagnetism is completely described by Maxwell's equations:

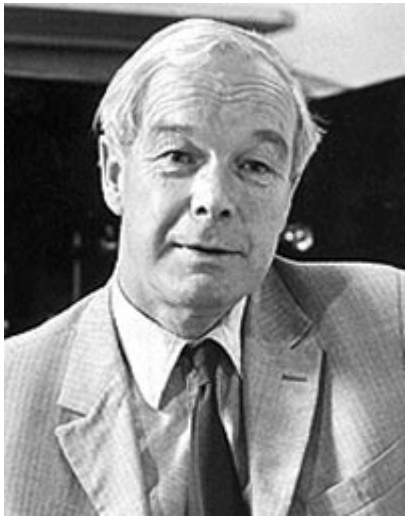
$$\nabla.\mathbf{E} = \frac{\rho}{\varepsilon_0} \qquad \nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

$$c^2 \nabla \times \mathbf{B} = \frac{\partial \mathbf{E}}{\partial t} + \frac{\mathbf{J}}{\varepsilon_0} \qquad \nabla.\mathbf{B} = 0$$

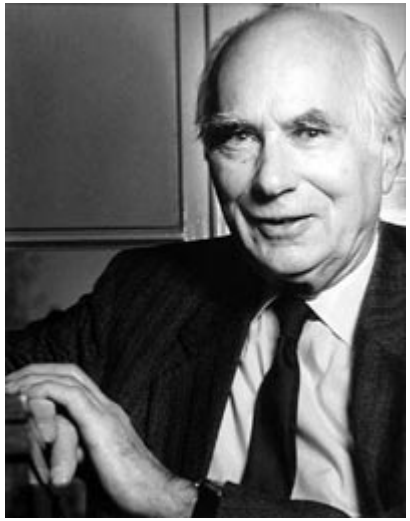
where ρ is the electric charge density (charge per unit volume) and J is the electric current density (Amperes per unit area). E and B denote the electric and magnetic fields, and $\varepsilon_0 = 10^7 c^2/4\pi$ [$C.N^{-2}.m^{-2}$].

Modelling Neural Activation

- Alan Hodgkin and Andrew Huxley shared the 1963 Noble Prize in Physiology or Medicine (jointly with John Eccles) "*for their discoveries concerning the ionic mechanisms involved in excitation and inhibition in the peripheral and central portions of the nerve cell membrane".*



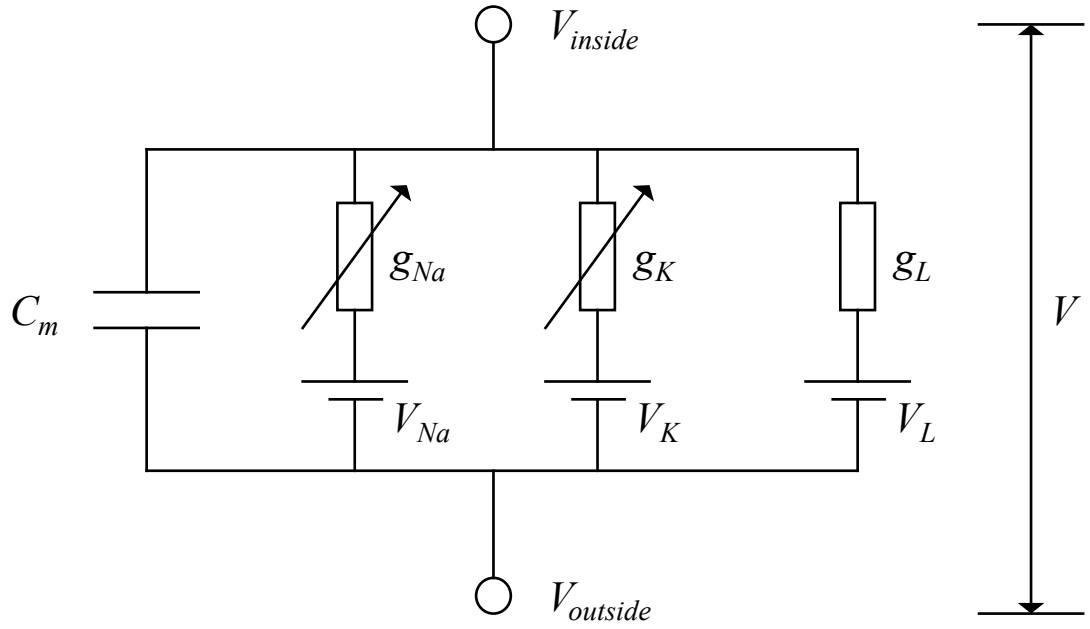
Alan Lloyd Hodgkin
(1914-1998)



Andrew Fielding Huxley
(1917-2012)

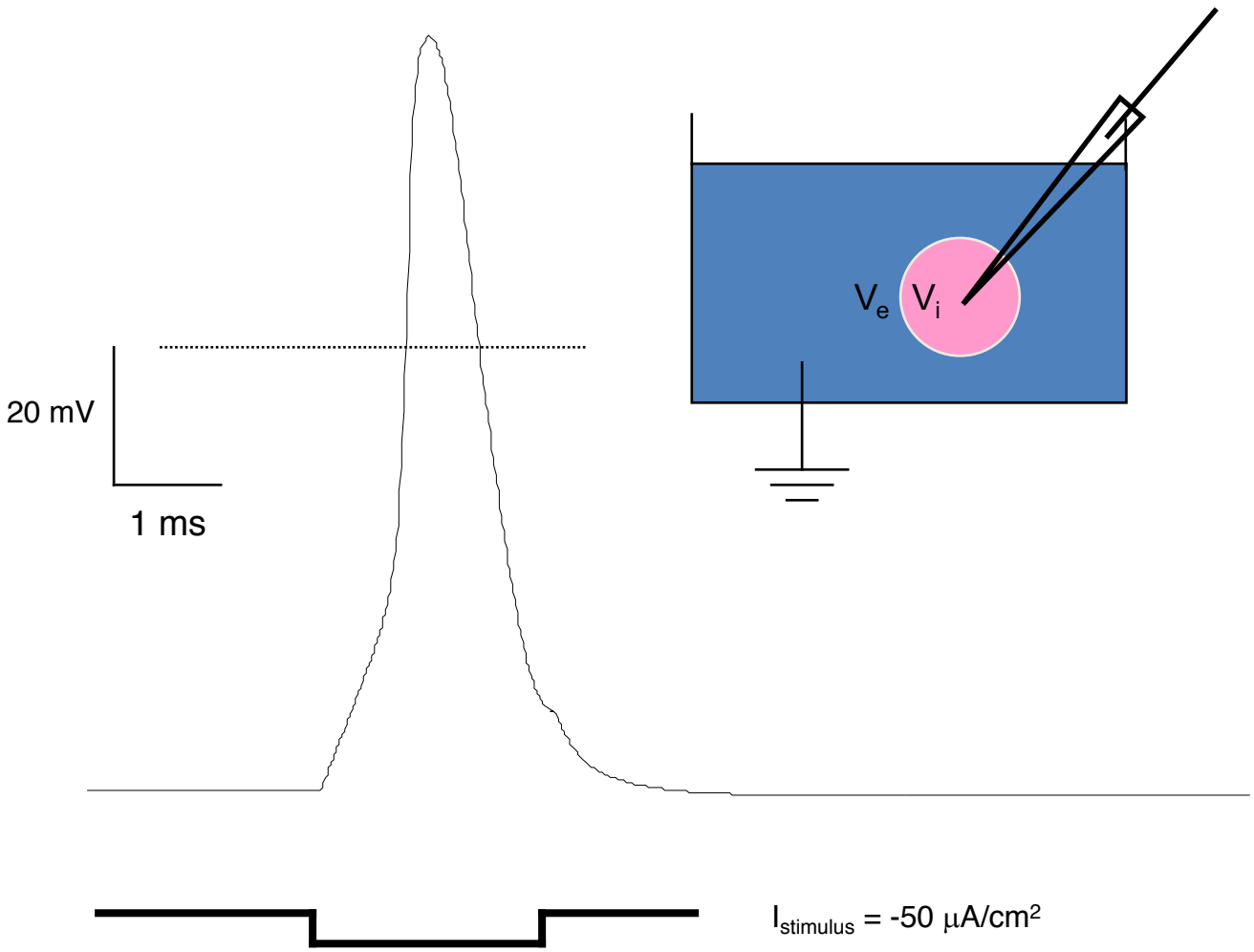
Hodgkin-Huxley (HH) Model

- Based on experiments in the giant axon of the squid.
- The **space-clamped** version (ignoring spatial propagation) of the Hodgkin-Huxley model described by:

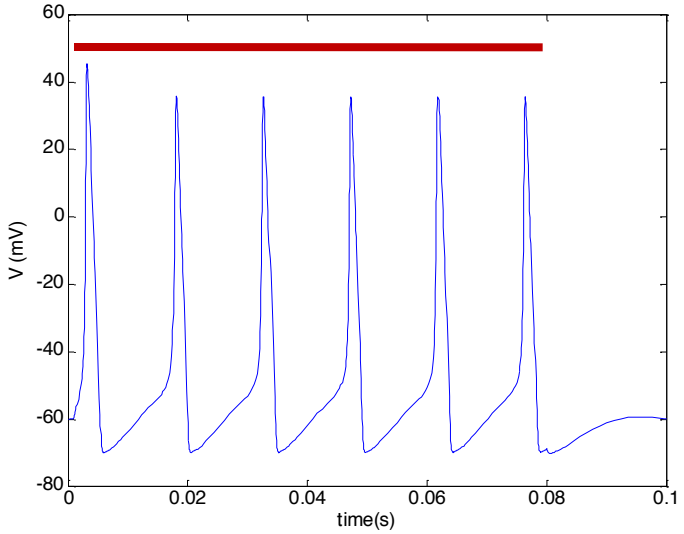


Hodgkin AL, Huxley AF (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. J. Physiol. (Lond.) 117:500-544.

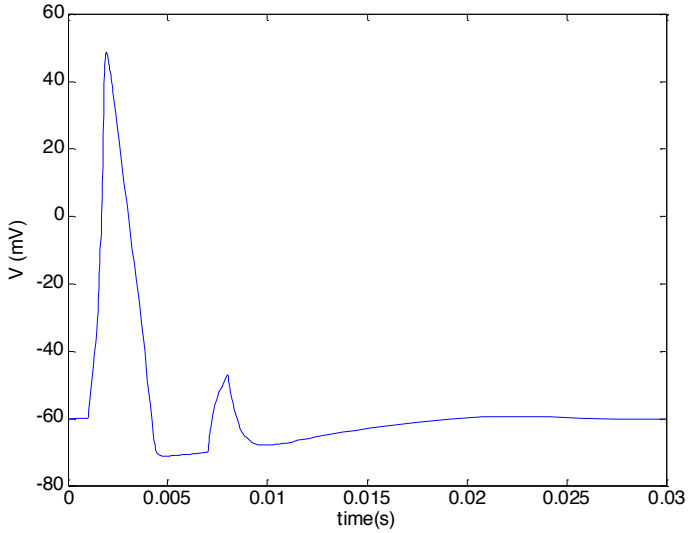
Simulating Neural Activation



HH Model Predictions

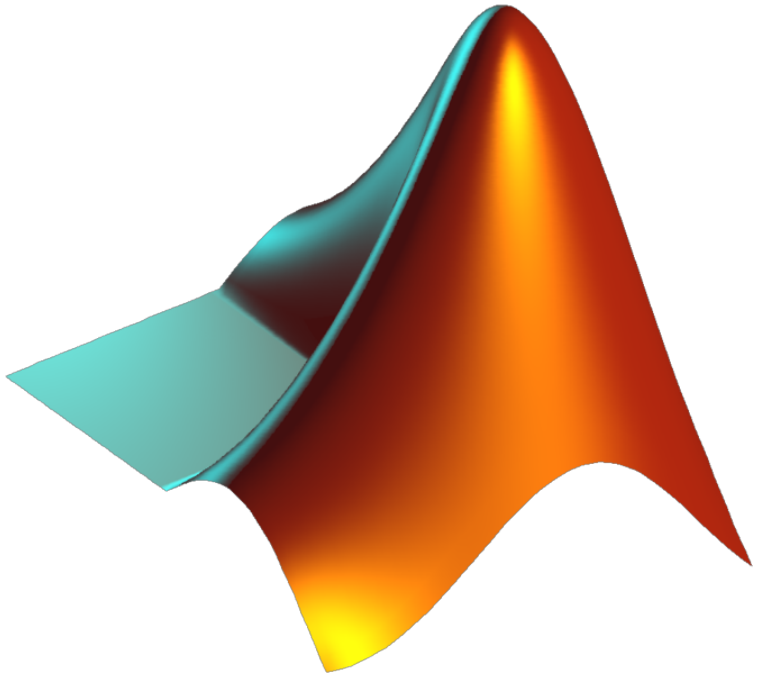


Action potential spike train due to sustained inward stimulus current of 10 $\mu A/cm^2$



Refractoriness. Two equal stimulus currents are applied 6 ms apart. The second stimulus does not elicit an action potential.

Matlab[1] Introduction



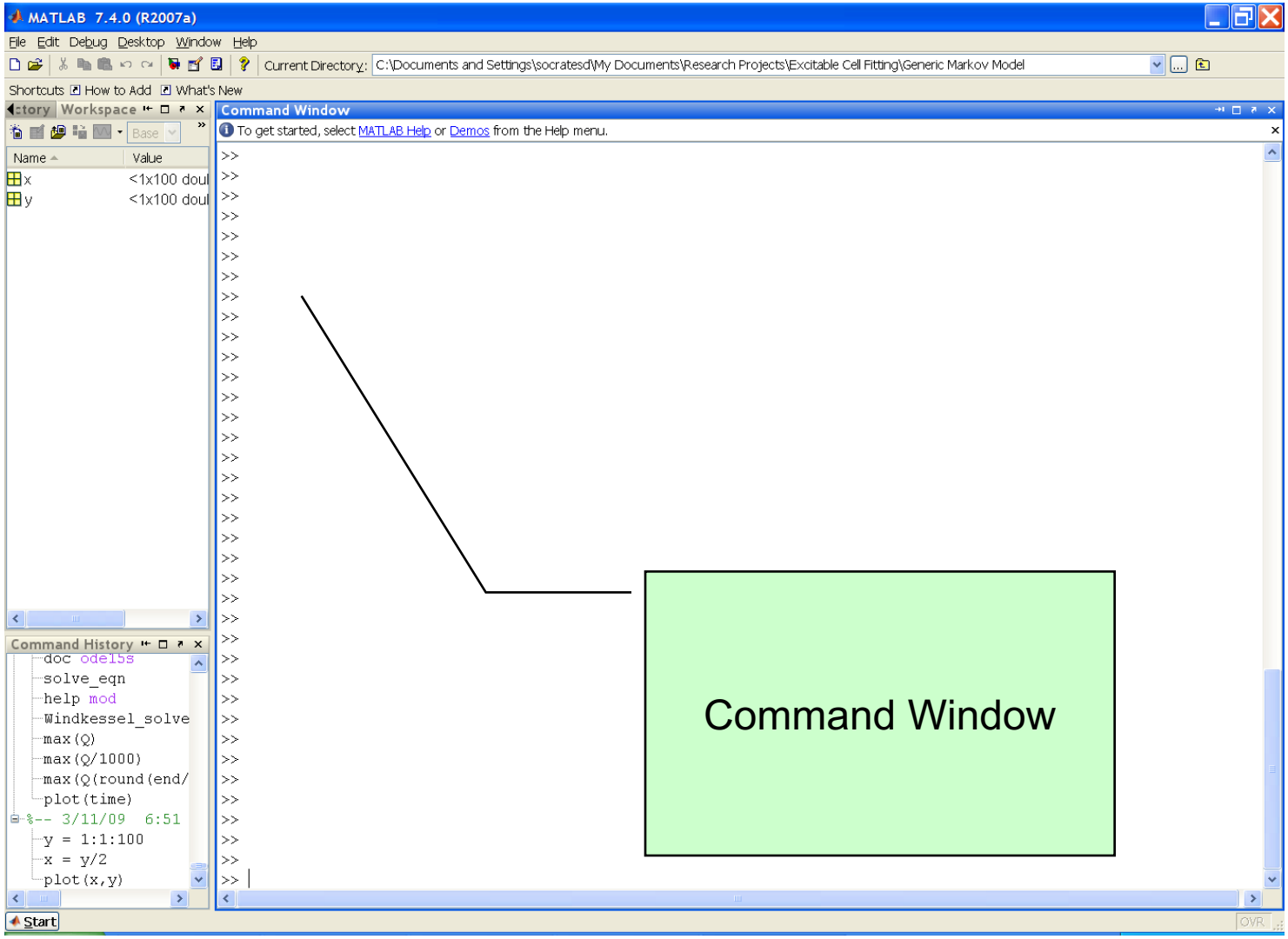
[1]The MathWorks Inc.

Access to MATLAB

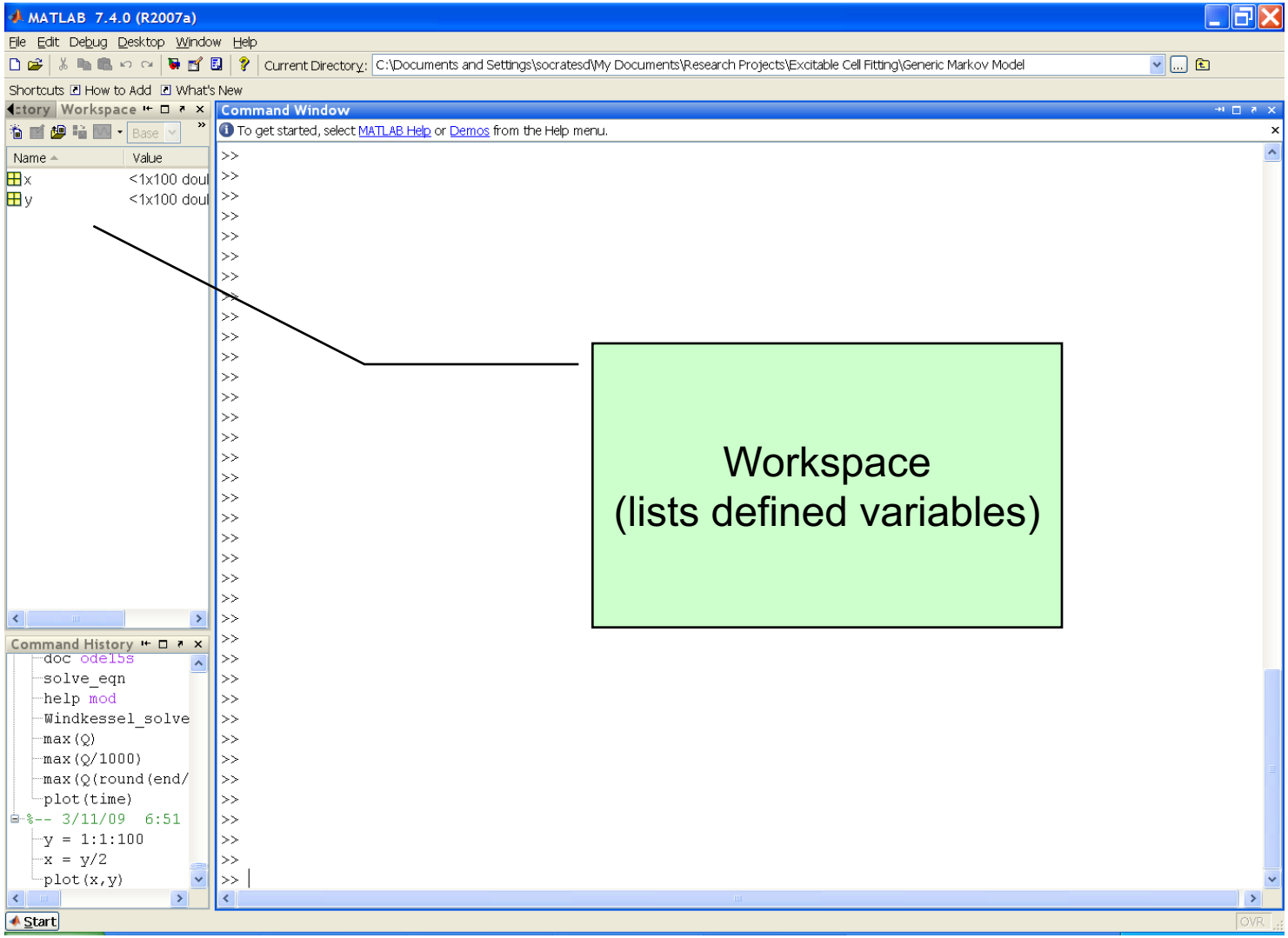
- Via UNSW myAccess at https://www.myaccess.unsw.edu.au

- Direct download through UNSW IT https://www.it.unsw.edu.au/students/software/matlab.html (Free UNSW-wide site licence for non-commercial research and teaching purposes)

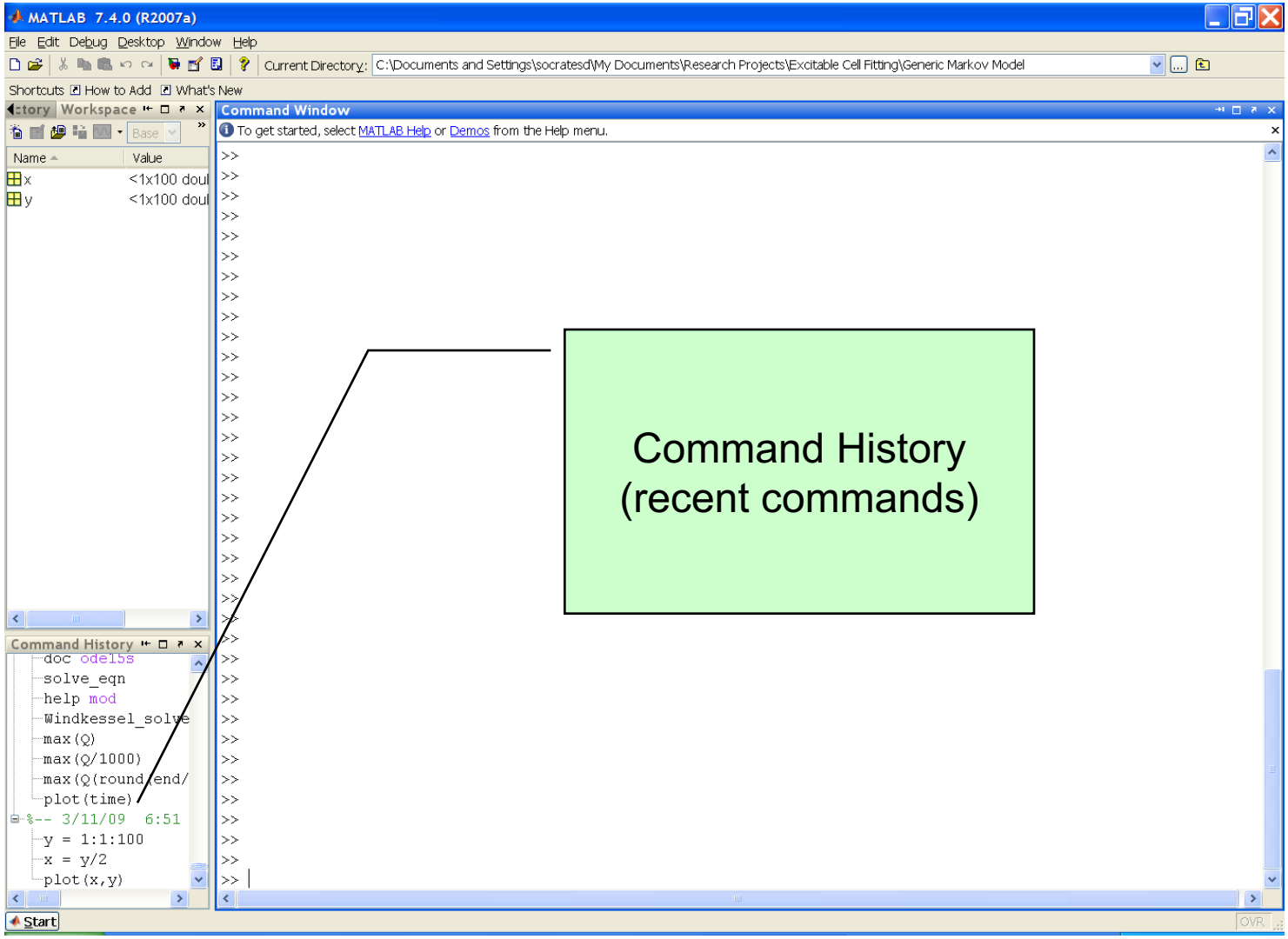
MATLAB Interface



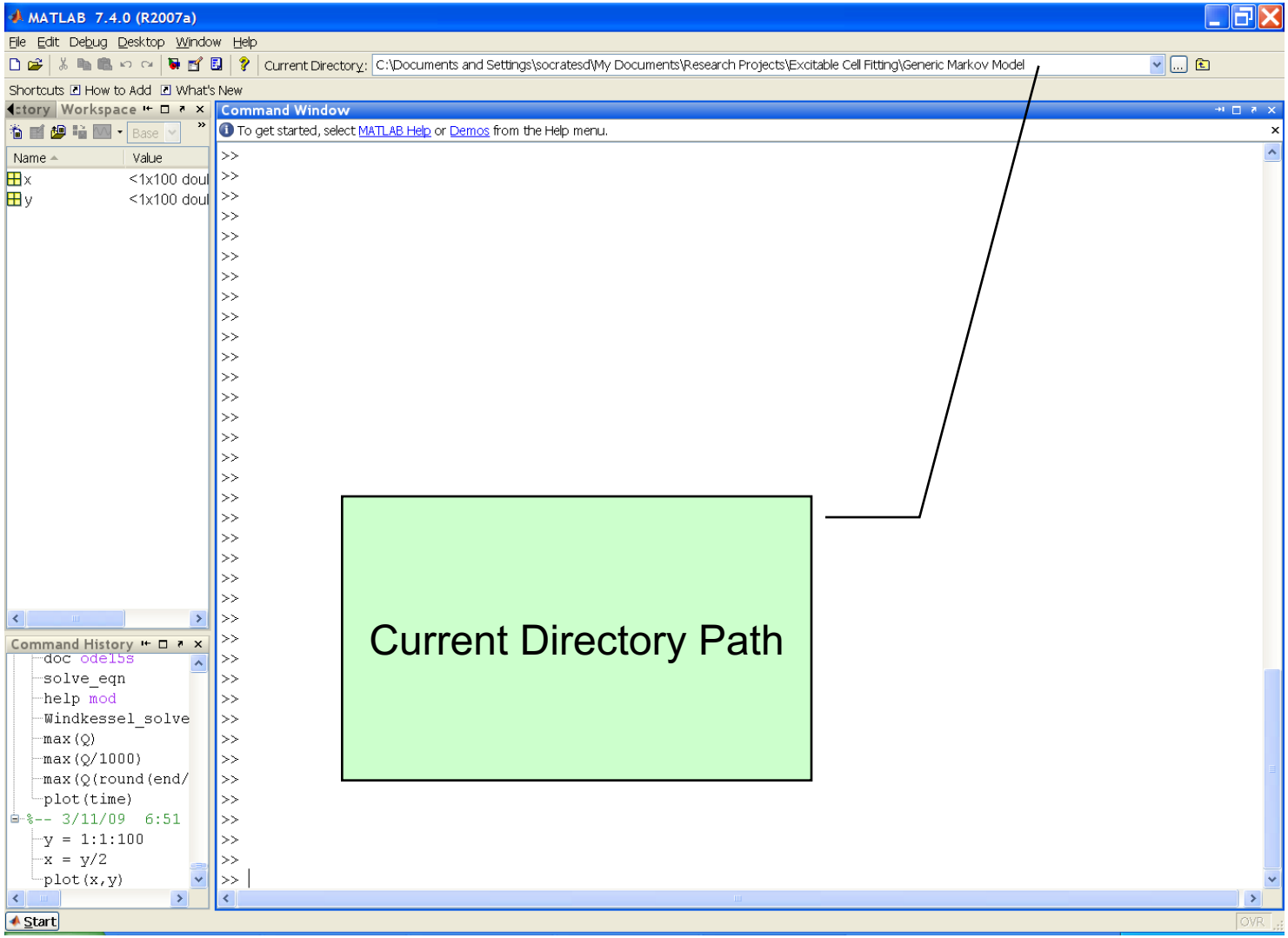
MATLAB Interface



MATLAB Interface



MATLAB Interface



Working with Variables

At the command window, simply assign variables their value. e.g.

```
>> r = 2;
>> c = 2*pi*r;
```

To define a 1D array, use a command like

```
>> x = [0:0.001:1];
```

This creates a row array of 1001 elements, with 0 as the first element and 1 as the last, in increments of 0.001.

To square each element of x, use

```
>> y = x.^2;
```

Basic Operators and Functions

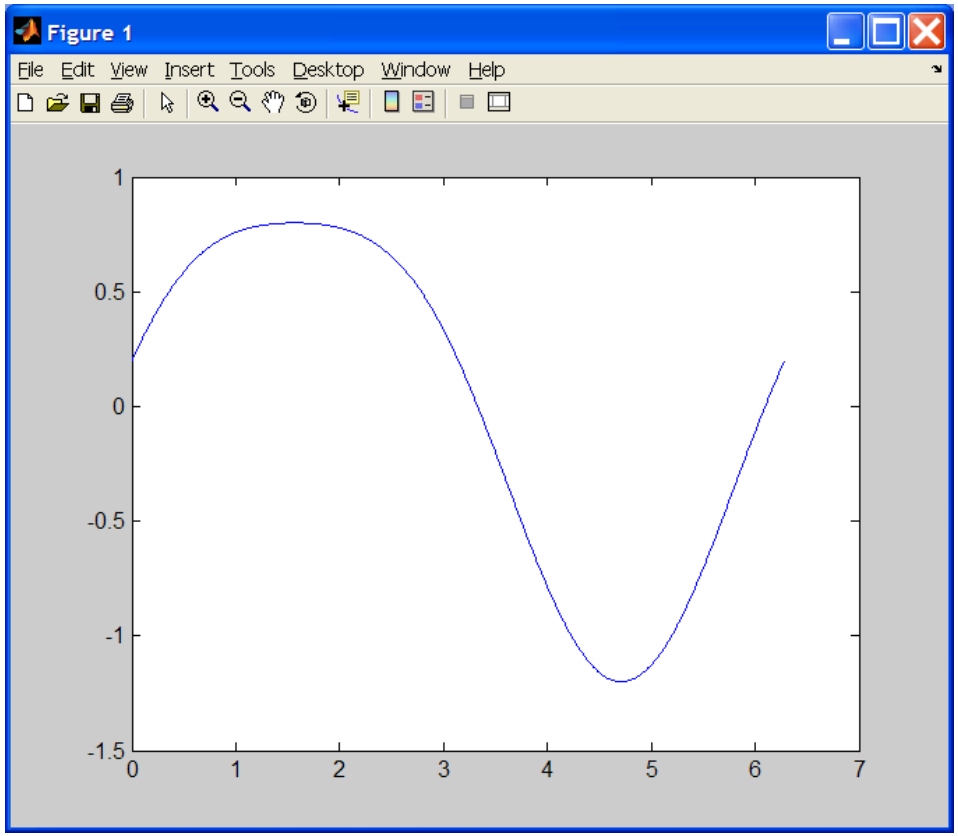
Operator	Description
`*,+,-,/`	Base arithmetic operators
`^`	Raise to power e.g. 3^2 (= 9)
`.*, ./, .^`	Element-element array operators
`mod(x,y)`	Modulus: i.e.remainder on division of x by y
`sin(x), cos(x), tan(x)`	Trigonometric functions
`exp(x)`	Exponential function e^x
`log(x)`	Natural logarithm (i.e. base e)
`\`	Array division
`x > y`	Returns 1 if x > y, or 0 otherwise
`>, <, >=, <=, ==`	Comparison operators
`plot(x,y)`	Plots array y against x

Working with .m files

Commands can also be entered and saved in .m files which can be executed from the command window as a script.

e.g. to generate a plot of y = sin(x) + 0.2cos(2x), save the following commands to a file named for example my_sine.m), and type `my_sine` at the command prompt:

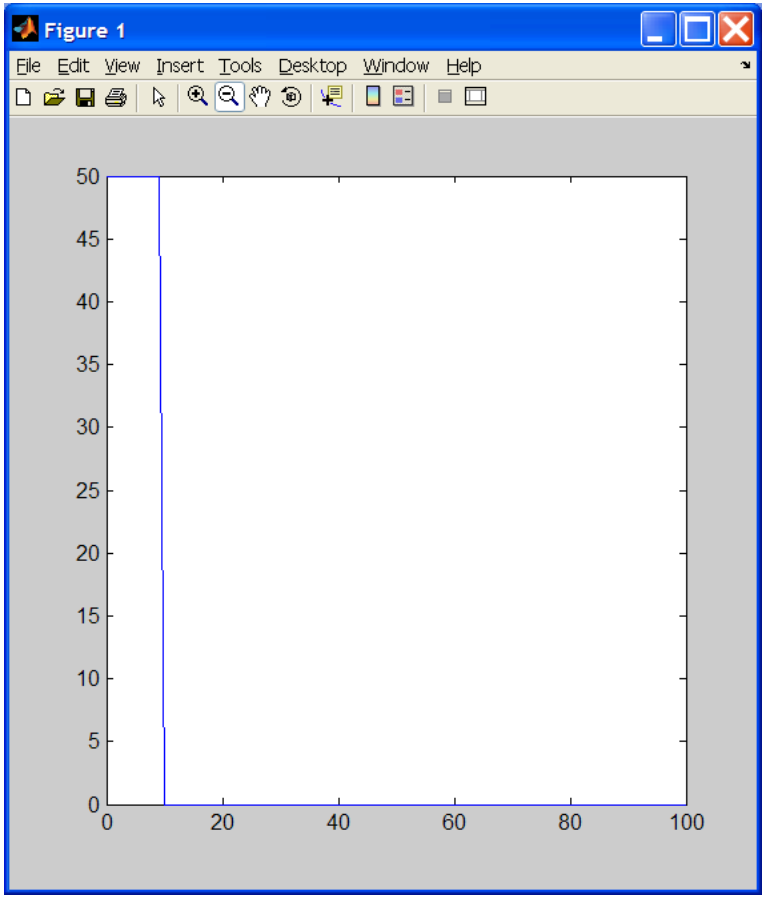
```
x = 0:2*pi/1000:2*pi;
y = sin(x)+0.2*cos(2*x);
plot(x,y);
```



if, else, end

`If… else` and `for` loop programming structures can be implemented as follows:

```
t = 0:1:100;
I = zeros(1,101);
for i = 1:101
    if (t(i) < 10)
        I(i) = 50;
    else
        I(i) = 0;
    end;
end;
plot(t,I);
```



MATLAB Documentation

Help on any command can be obtained by typing `help` followed by the command.

e.g. `>> help mod`

```
>> help mod
 MOD    Modulus after division.
    MOD(x,y) is x - n.*y where n = floor(x./y) if y ~= 0.  If y is not an
    integer and the quotient x./y is within roundoff error of an integer,
    then n is that integer.  The inputs x and y must be real arrays of the
    same size, or real scalars.
```

Typing doc will bring up a html manual:

e.g. `doc mod`

mod

Modulus after division

Syntax

```
M = mod(X,Y)
```

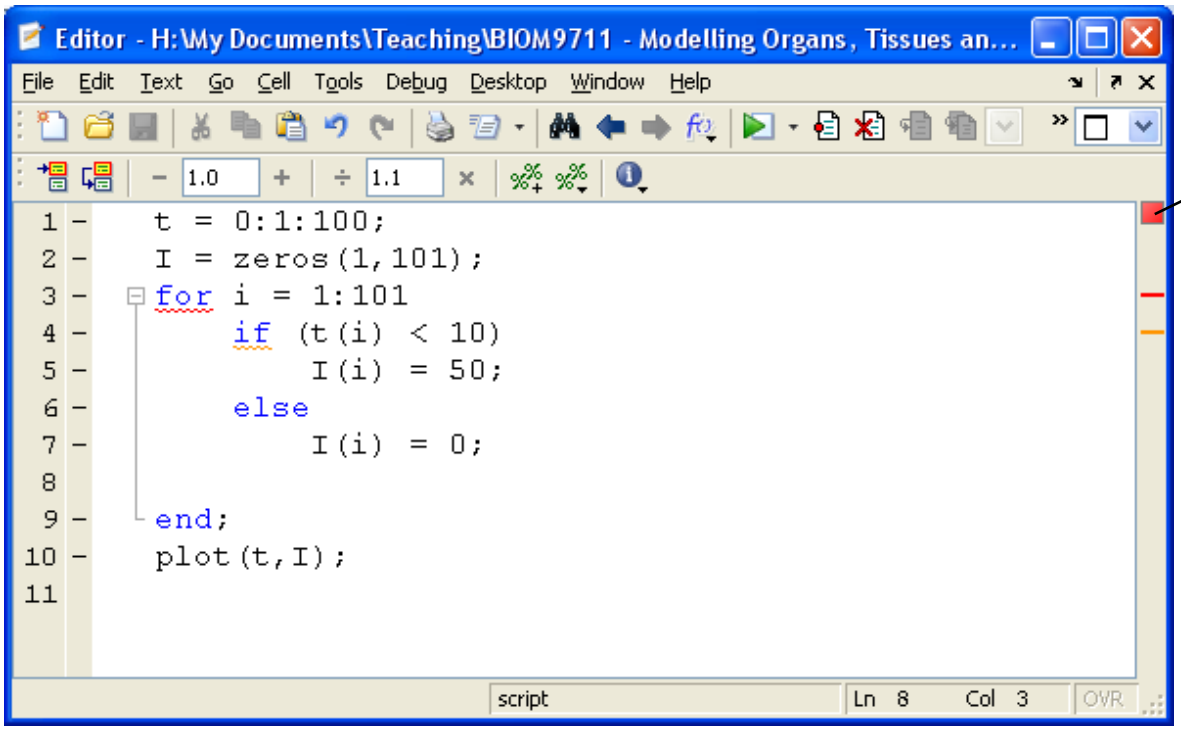
Description

`M = mod(X,Y)` if `Y ~= 0`, returns `X - n.*Y` where `n = floor(X./Y)`. If `Y` is not an integer and the quotient `X./Y` is within roundoff error of an integer, then `n` is that integer. The inputs `X` and `Y` must be real

Debugging Code in MATLAB

Matlab provides notifications of errors, warnings and recommendations for code using its M-Lint code analyser.

In the example below, an `end` statement is missing from the `if`, `else` loop:



Red indicator denotes syntax error

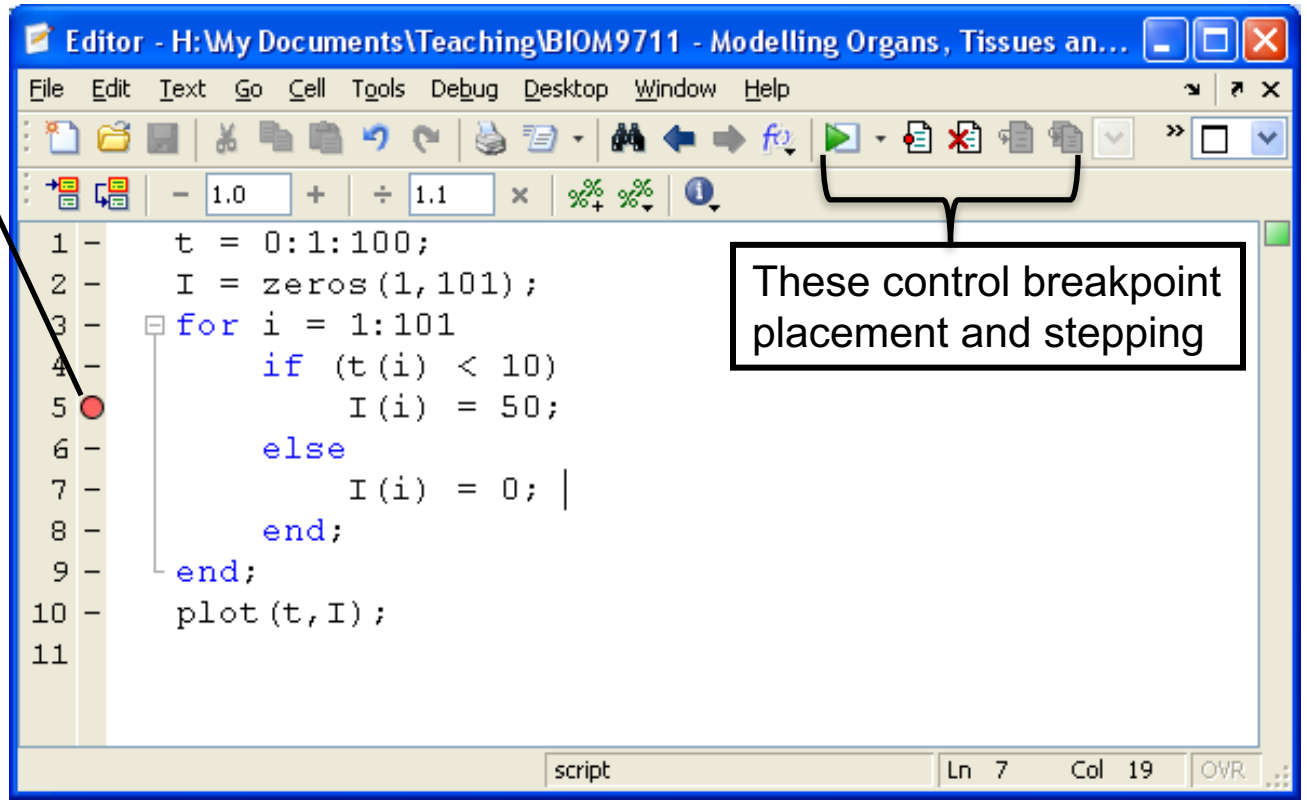
Orange indicator denotes warning

Green denotes no errors

Debugging Code in MATLAB

It is also possible to insert breakpoints into the code, and step through the code one line at a time. Variables can also be examined during stepping.

Left Clicking here will insert a breakpoint



These control breakpoint placement and stepping

Solving Linear Systems of Equations

Consider the following linear system of equations:

$$2x + 3y - 4z = 7$$

$$x + 5y - z = 2$$

$$x + y = 1$$

This can be represented by the array equation:

$$\mathbf{Ax} = \mathbf{b}$$

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & -4 \\ 1 & 5 & -1 \\ 1 & 1 & 0 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 7 \\ 2 \\ 1 \end{bmatrix}$$

Solving Linear Systems of Equations

In MATLAB, this system can be solved using:

```
>> A = [2, 3, -4; 1, 5, -1; 1, 1, 0];
>> b = [7; 2; 1];
>> x = A\b

x =
    1.0667
   -0.0667
   -1.2667
```

Use of the backslash (\) operator is equivalent to

```
>> x = inv(A)*b
```

User-Defined Functions

User-defined functions in MATLAB are saved in special .m files whose first line contains the `function` reserved word.

For example, to create a function to solve the system of equations Ax = b, save the following code in a file named `solve_my_system.m`

```
function x = solve_my_system(A, b)
        x = A\b;
```

Then execute the following in the command window:

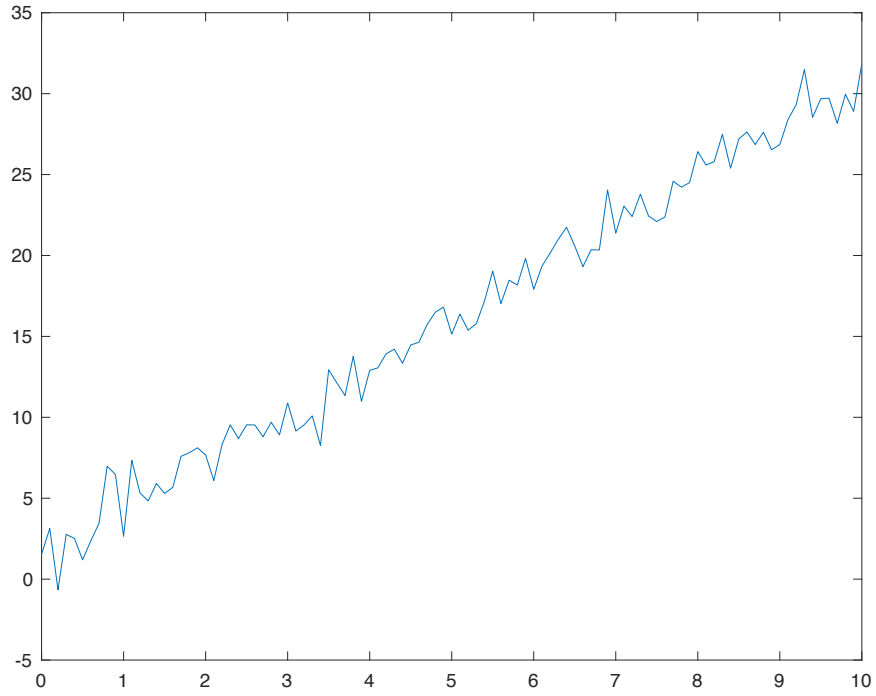
```
>> C = [2, 3; 1, 4];
>> d = [3; 8];
>> z = solve_my_system(C, d)
```

Curve Fitting

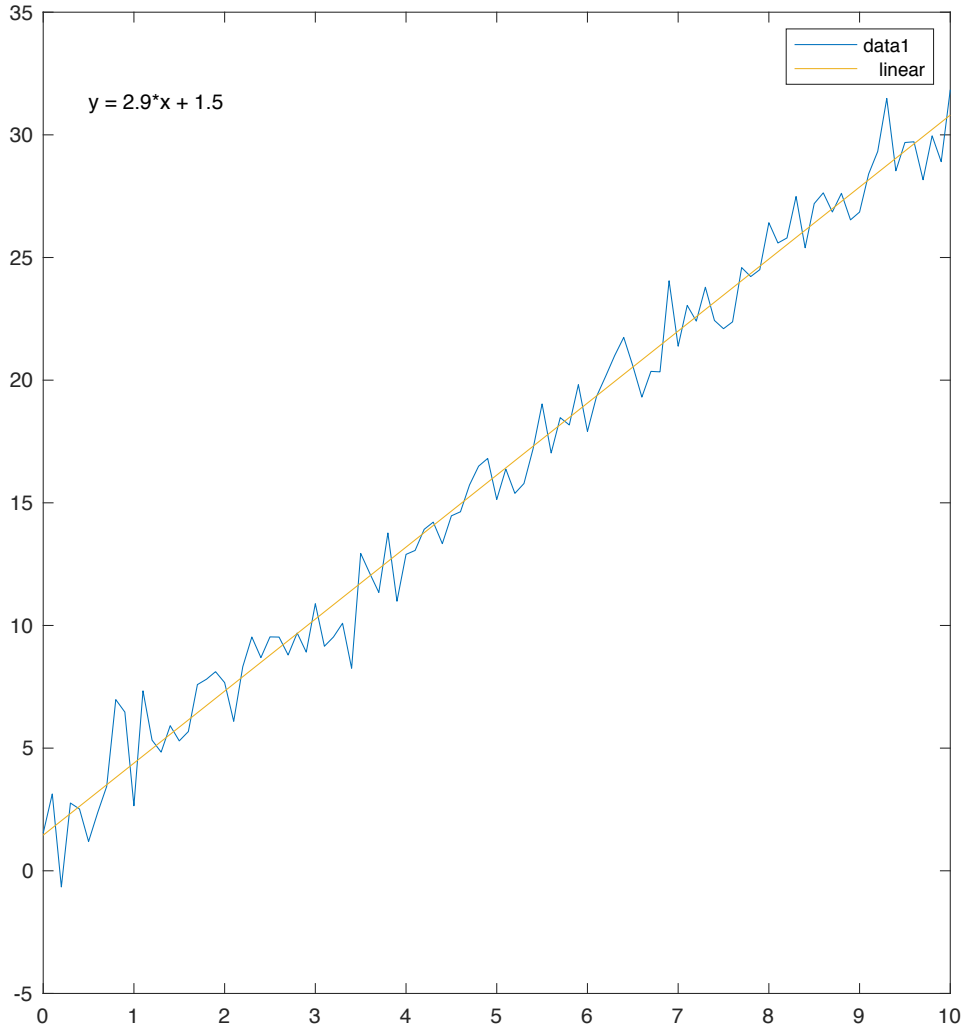
The following code generates and plots two 1D arrays with added random noise:

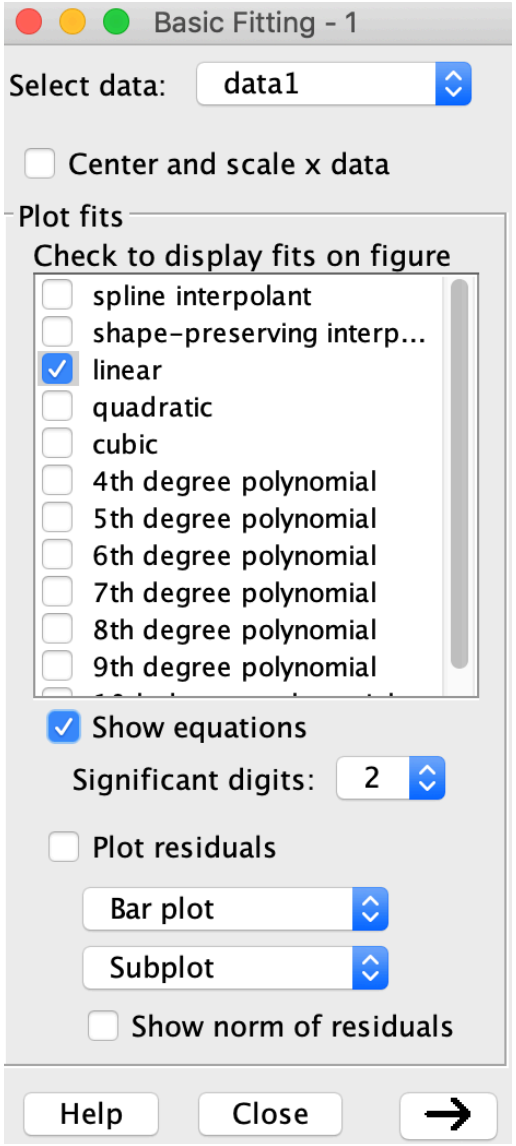
```
>> x = 0:0.1:10;

>> y = 3*x+1+randn(size(x));

>> plot(x,y)
```

On the Figure menu bar, go to
Tools → Basic Fitting
to bring up dialog window
of basic fitting tools…



Curve Fitting





Additional MATLAB Resources

- Matlab Moodle Module (for S2, 2018) at https://moodle.telt.unsw.edu.au/course/view.php?id=37082 containing YouTube videos describing basics of the MATLAB programming environment, quizzes to test your knowledge, and other tutorial material.
- Online tutorials from the MathWorks at https://au.mathworks.com/support/learn-with-matlab-tutorials.html
- Useful textbooks: