Introduction to Matlab

Jefferson Davis and Scott Michael
Research Analytics

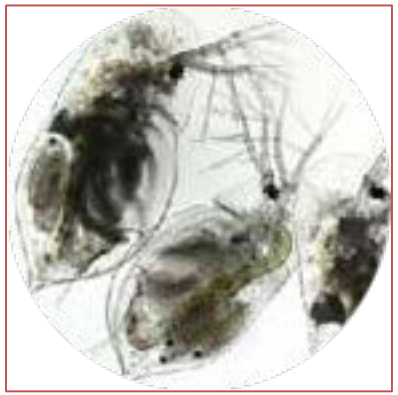
Matlab background

- Developed by Cleve Moler in the 1970s to give students easier access to numerical libraries for linear algebra
- MathWorks company founded in 1984 for commercial development
- The fundamental datatype is the matrix (array)

Matlab IU availability

- STC labs
- IUAnyware
- Karst, Big Red II, Carbonate, RED
- Free academic license via IUware with 46 toolboxes (depending on how you count Simulink)

Matlab research use

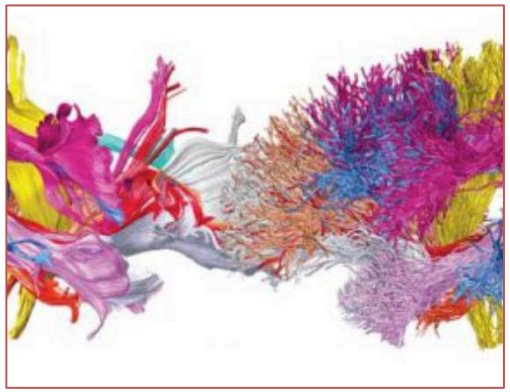


Dr. Spencer Hall,
Biology

infectious disease in food webs. Run simulations; solve systems; fit models; visualize

Dr. Shirin Hassan
School of Optometry

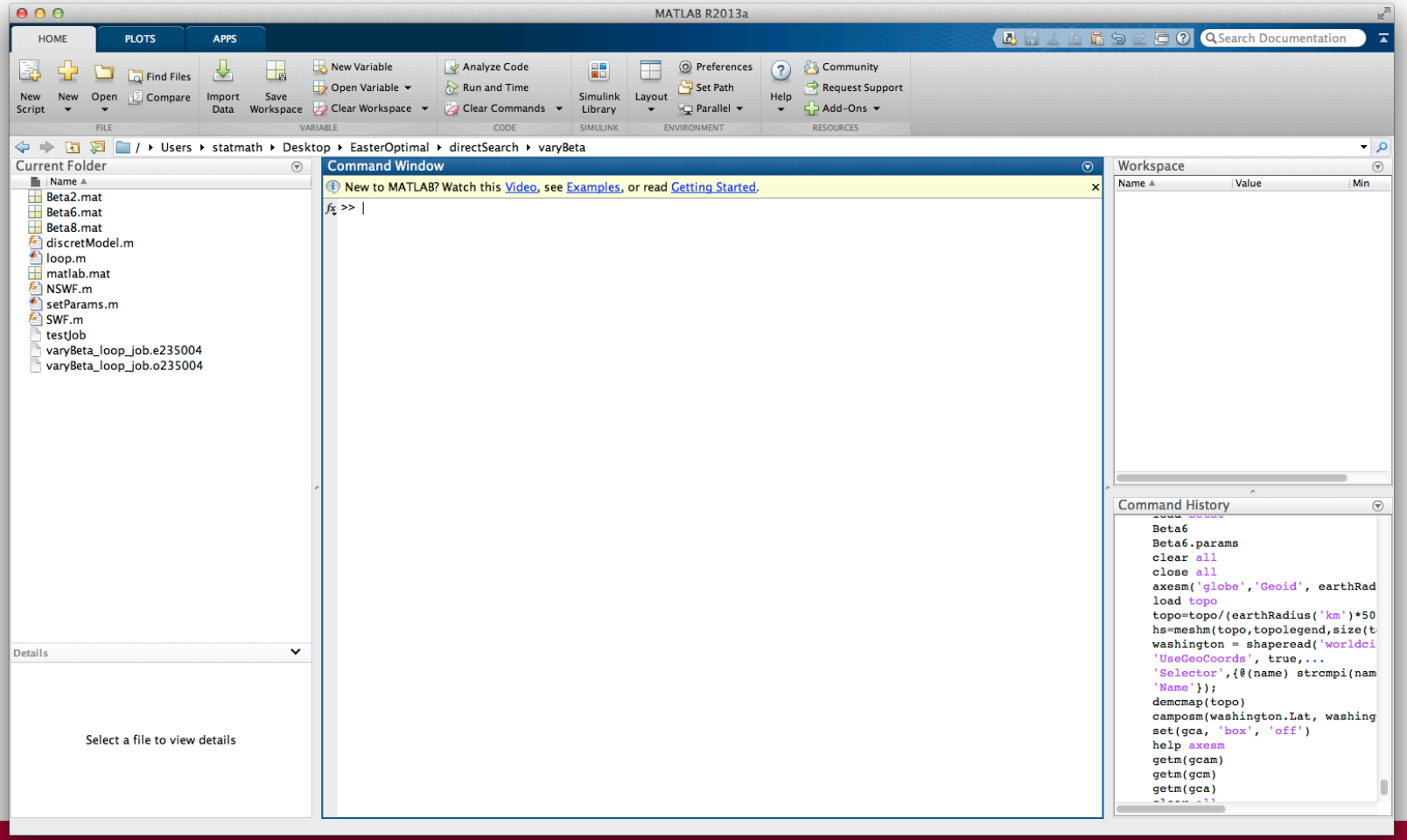
organize and analyze street crossing decisions of the visually impaired



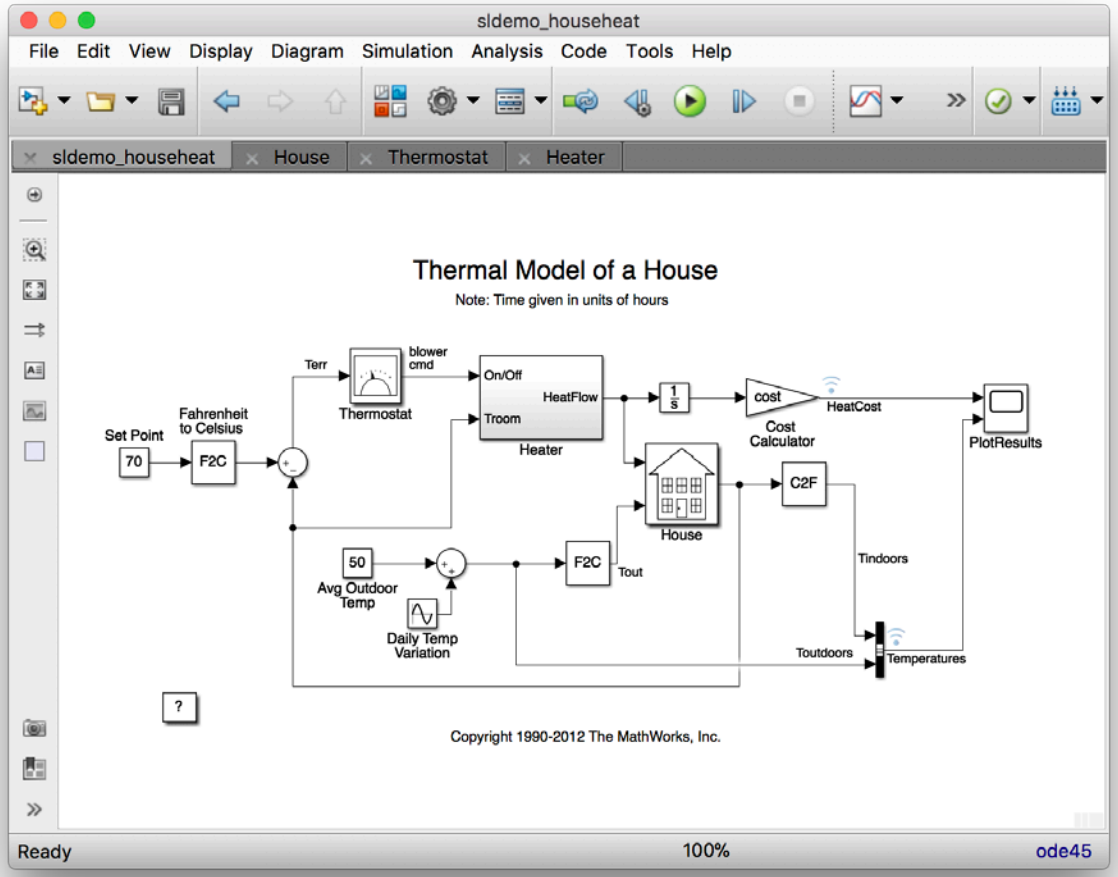
Dr. Franco Pestilli,
Psych. & Brain Sciences

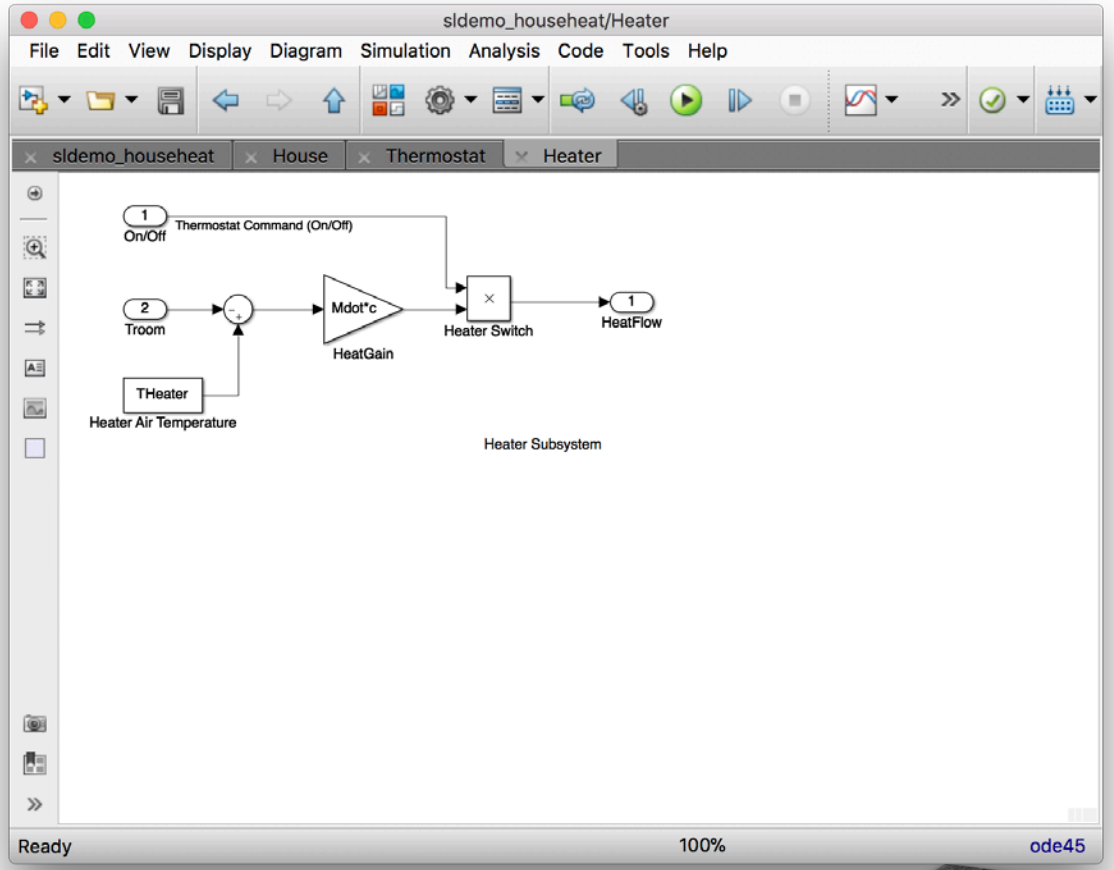
Study brain structure, function, and behavior

Taking a look at the interface



Taking a look at the interface





Arithmetic in Matlab

`2 + 3`	`ans = 5`
`a = 34 * 8`	`a = 272`
`pi`	`ans = 3.1416`
`i`	`ans = 0.0000 + 1.0000i`
`sin(pi)`	`ans = 1.2246e-16`
`eps`	`ans = 2.2204e-16`

Vectors in Matlab

`f = 1:6`	`f = 1 2 3 4 5 6`
`g = 0:2:6`	`g = 0 2 4 6`
`sin(g)`	`ans = 0 0.9093 -0.7568 -0.2794`
`g(3)`	`4`
`g(1:3)`	`0 2 4`
`g'`	`ans = 0` `2` `4` `6`

More Vectors

`g'`	`ans = 0 2 4 6`
`g + g`	`ans = 0 4 8 12`
`g * g'`	`ans = 120`
`g * g`	`Error using * Inner matrix dimensions must agree.`
`g .* g`	`ans = 0 4 16 36 64`

Matrices in Matlab

`h = [1 2 3; 4 5 6; 7 8 9]`	`h =` `1 2 3` `4 5 6` `7 8 9`
`h ^ 2`	`ans =` `30 36 42` `66 81 96` `102 126 150`
`h .^ 2`	`ans =` `1 4 9` `16 25 36` `49 64 81`

A few useful notes

- The help command will display a function's help text. The doc command brings up more information
  ```
  help sin
  doc sin
  ```
- The semi-colon (;) will suppress output
- The up arrow key will go back to previous commands
- Typing and then using the up arrow key goes back to previous commands that start with that text
- The exclamation point is used for shell commands
  ```
  ! rm matlab_crash_dump.*
  ```
- The percent sign is used for comments
  ```
  %This is a Matlab comment
  ```

Vectorized code

Matlab is optimized for operations on matrices and vectors.
Avoiding loops to access matrix values is a good idea.

Consider two ways to create a 1 x 10,000,000 vector
[1, 4, 9, 16,…,10000000^2]

`tic` `a=zeros(1,10000000);` `for i=1:100000000` `  a(i)=i^2;` `end` `toc`	`tic` `a=[1:10000000].^2;` `toc`
`Elapsed time is 0.110672 seconds.`	`Elapsed time is 0.064941 seconds.`

A note on initializing variables

Do it.

`tic` `a=zeros(1,10000000);` `for i=1:10000000` `  a(i)=i^2;` `end` `toc`	`tic` `for i=1:10000000` `  a(i)=i^2;` `end` `toc`
`Elapsed time is 0.107770 seconds.`	`Elapsed time is 0.983030 seconds. Yikes.`

Matlab: implicit parallelism in svd

Many functions will recognize the multicore environment and create an appropriate number of threads. A good example is singular value decomposition (SVD), rewriting a matrix as the product of "nice" matrices.

> matlab tic svd(rand(5000)) toc	> matlab -singleCompThread tic svd(rand(5000)) toc
Elapsed time is 15.343513 seconds.	Elapsed time is 157.930566 seconds.

Matlab: svd using the gpu

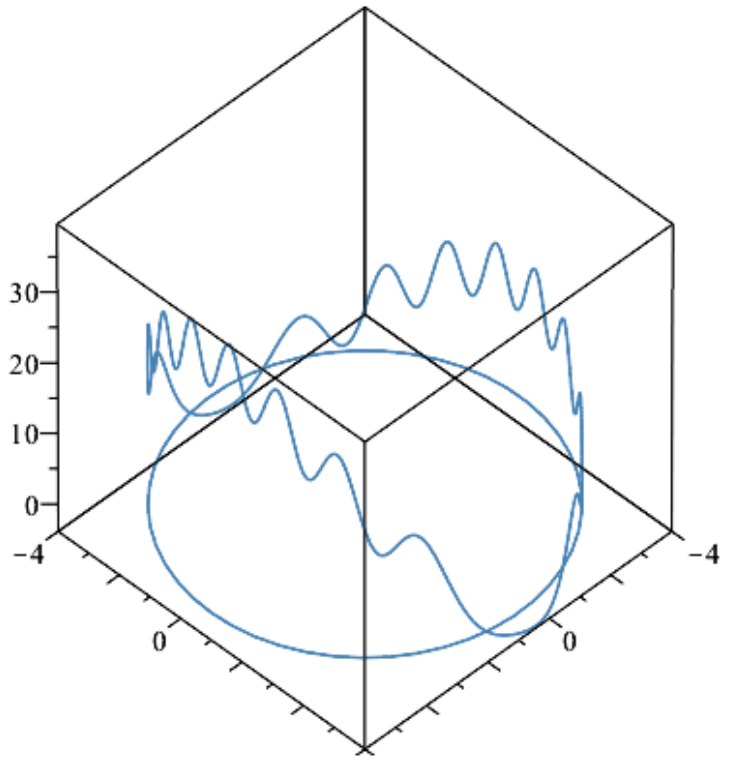
> qsub -I -q debug_gpu –l gres=ccm > ccmlogin > matlab tic svd(rand(5000)) toc	> qsub -I -q debug_gpu -lgres=ccm > ccmlogin > matlab tic svd(rand(5000,'gpuArray')); toc
Elapsed time is 62.310409 seconds.	Elapsed time is 16.616624 seconds.

Matlab: parallel-enabled functions

Sample problem: maximize the function $x_1^2 + 4*\sin(5*x_2)$ subject to the constraint $(x_1-1)^2 + (x_2-1)^2 = 25$

We first write a function to define the constraint mycon.m.

```
function [c,ceq] = mycon(x)
    c = (x(1)-1)^2 + (x(2)-1)^2 - 25;
    ceq = [];
```



Matlab: parallel-enabled functions

Then we set up the problem

```
opts = optimset('Algorithm','sqp');
problem = createOptimProblem('fmincon','objective', ...
  @(x) x(1)^2 + 4*sin(5*x(2)),'x0',[3 3],'lb',[-5-5], ...
  'ub',[5 5],'nonlcon',@mycon,'options',opts);
ms = MultiStart;
```

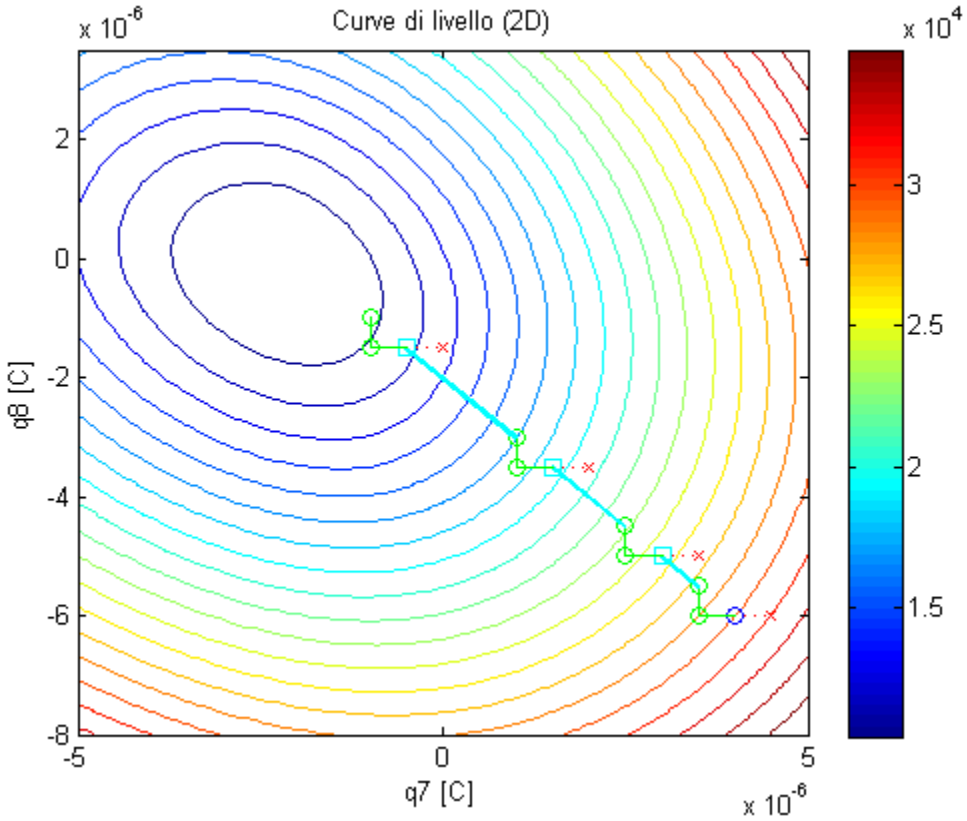
`ms.UseParallel = false;` `tic` `[x,f] =` `run(ms,problem,2000);` `toc`	`ms.UseParallel = true;` `tic` `[x,f] =` `run(ms,problem,2000);` `toc`
Elapsed time is 52.175676 seconds.	Elapsed time is 5.533175 seconds.

Matlab: parallel-enabled functions

The optimizer patternsearch is an example. At each step patternsearch checks the values of the objective function at near the current point. The first point with a lower value becomes the current point for the next step.

You can, however, use a pool of workers and check them in parallel. This means checking all the nearby points in the mesh.

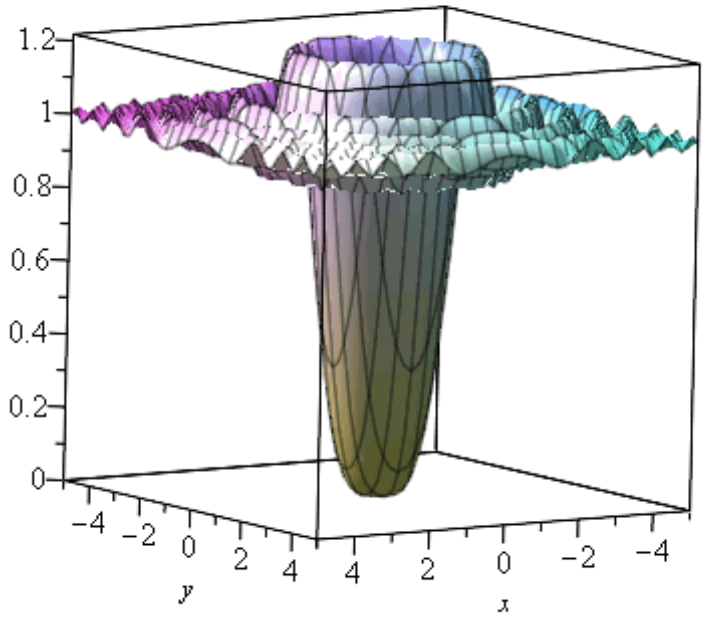
Is this a good idea?



Matlab: parallel-enabled functions

For example, say we want to minimize the function

$$1 - \frac{\sin\left(\sum_{i=1}^{100} x_i^2\right)}{\sum_{i=1}^{100} x_i^2}$$



We define a function in Matlab

```
function y = f(x)
y = 1 - sin(sum(x.^2)) ./ sum((x.^2));
```

Matlab: parallel-enabled functions

x0=ones(100,1)/100; tic patternsearch(@(x)f(x),x0); toc	parpool(15) options = psoptimset('UseParallel', true,... 'CompletePoll', 'on', 'Vectorized', 'off'); x0=ones(100,1)/100; tic patternsearch(@(x)f(x),... x0,[],[],[],[],[],[],[],... options); toc delete(gcp)
Elapsed time is 6.662042 seconds.	Hits iteration limit at 137.731795 seconds.

Matlab: parallel for loops

If you have a pool of parallel workers you can use them to run a for-loop with parfor.

`` tic for i=1:5000000 a(i)=max(rand(100,1)); end toc ```	``` parpool(15) tic parfor i=1:5000000 a(i)=max(rand(100,1)); end toc delete(gcp) ```
Elapsed time is 21.712727 seconds.	Elapsed time is 2.548504 seconds.

Matlab: parallel for loops

The are some restrictions on the loop, but the main one is that the order of evaluation can't matter. So the code below fails

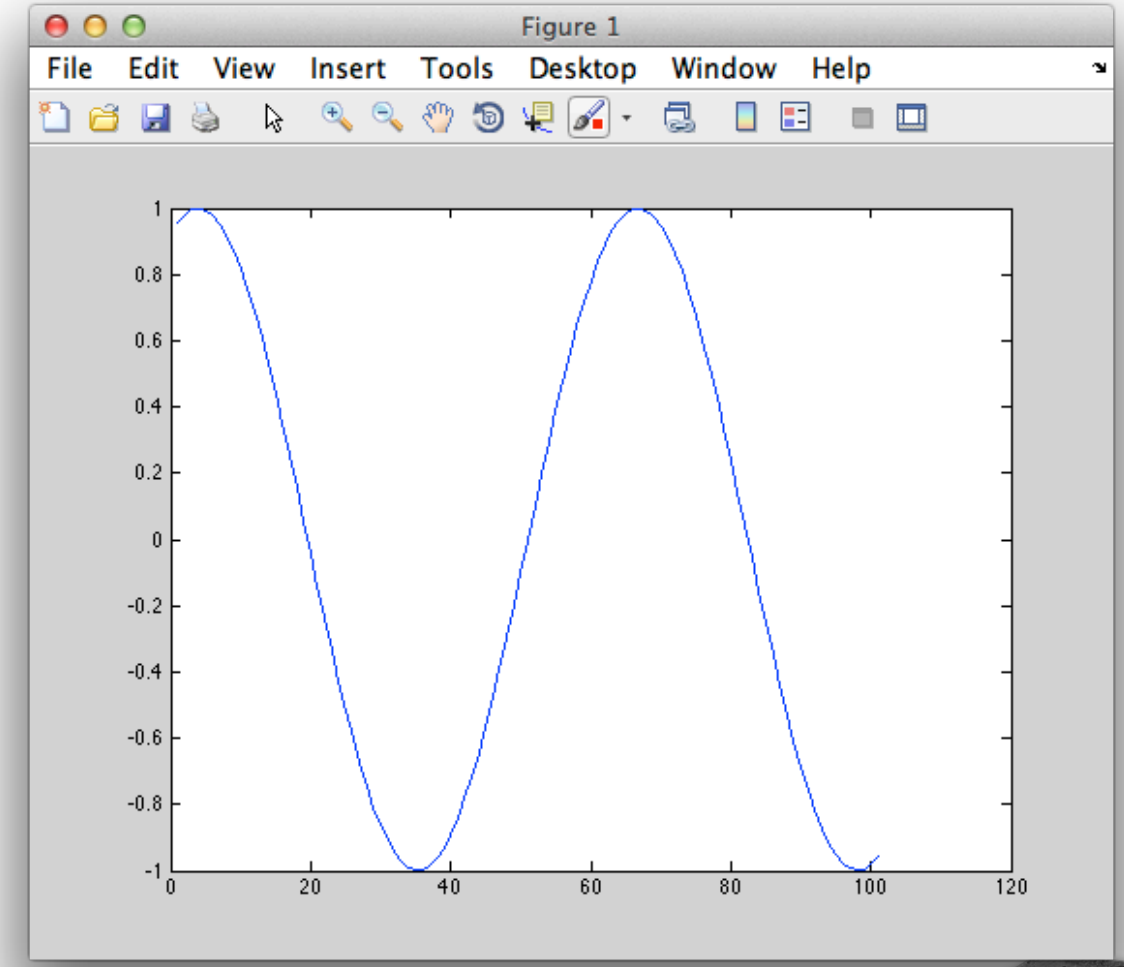
```
%Fibonacci failure
a(1)=1;a(2)=1;
parpool(2)
parfor i=3:100
   a(i)=a(i-1)+a(i-2);
end
delete(gcp)

%Should have used Binet's forumula
```

Plotting curves in Matlab

It's pretty straightforward!

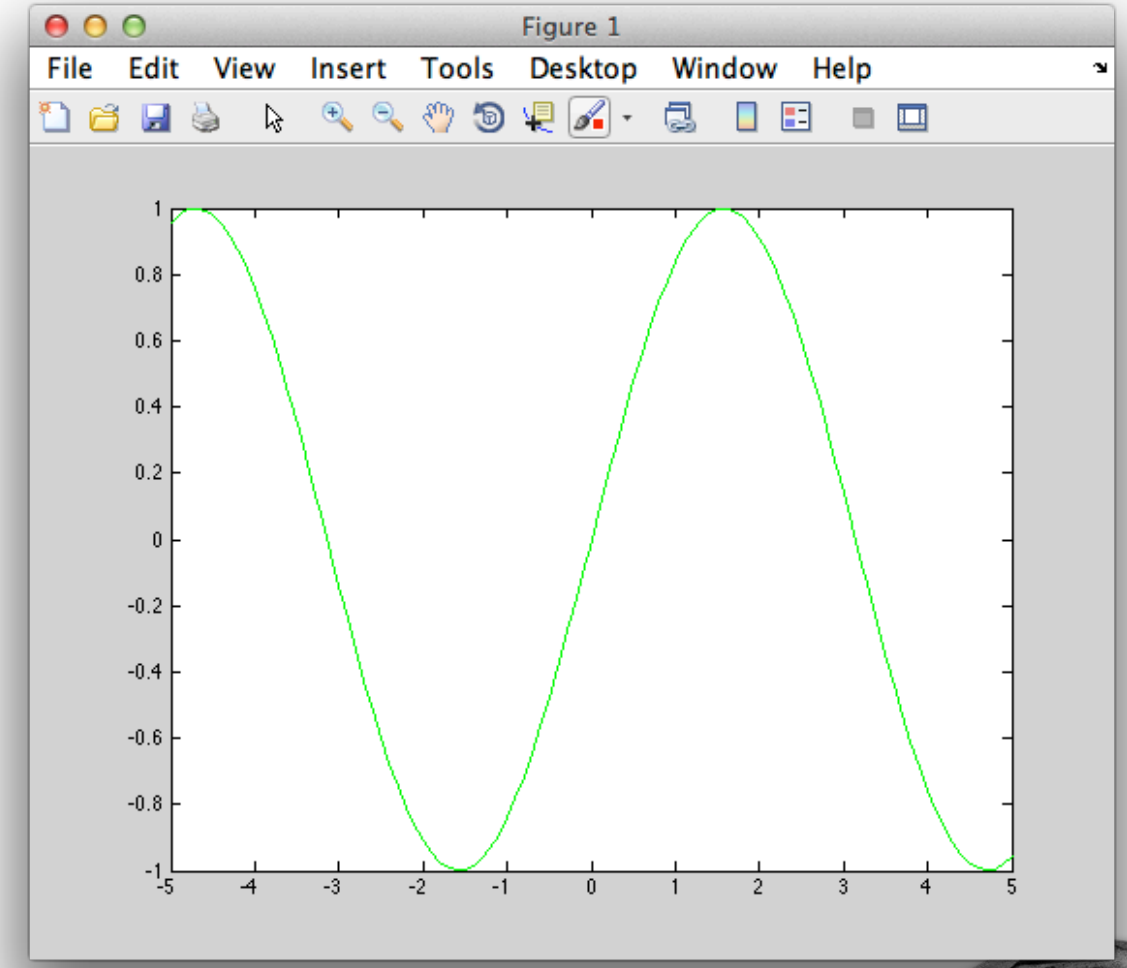
```
x=-5:.1:5;
plot(sin(x))
```

Note that both x and sin(x) are vectors of size 1 x 101.



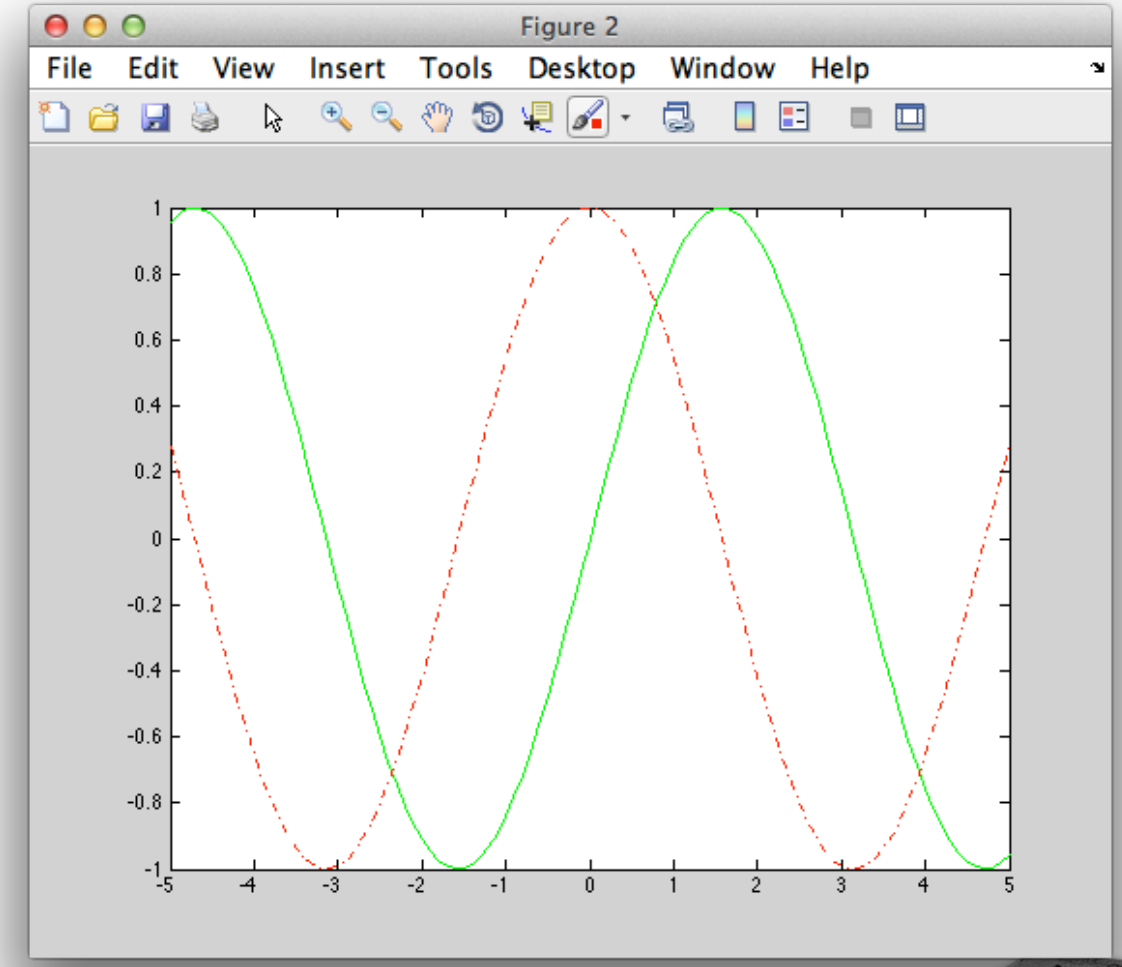
Plotting curves in Matlab

```
x=-5:.1:5;
plot(x,sin(x), 'g')
```



Plotting curves in Matlab

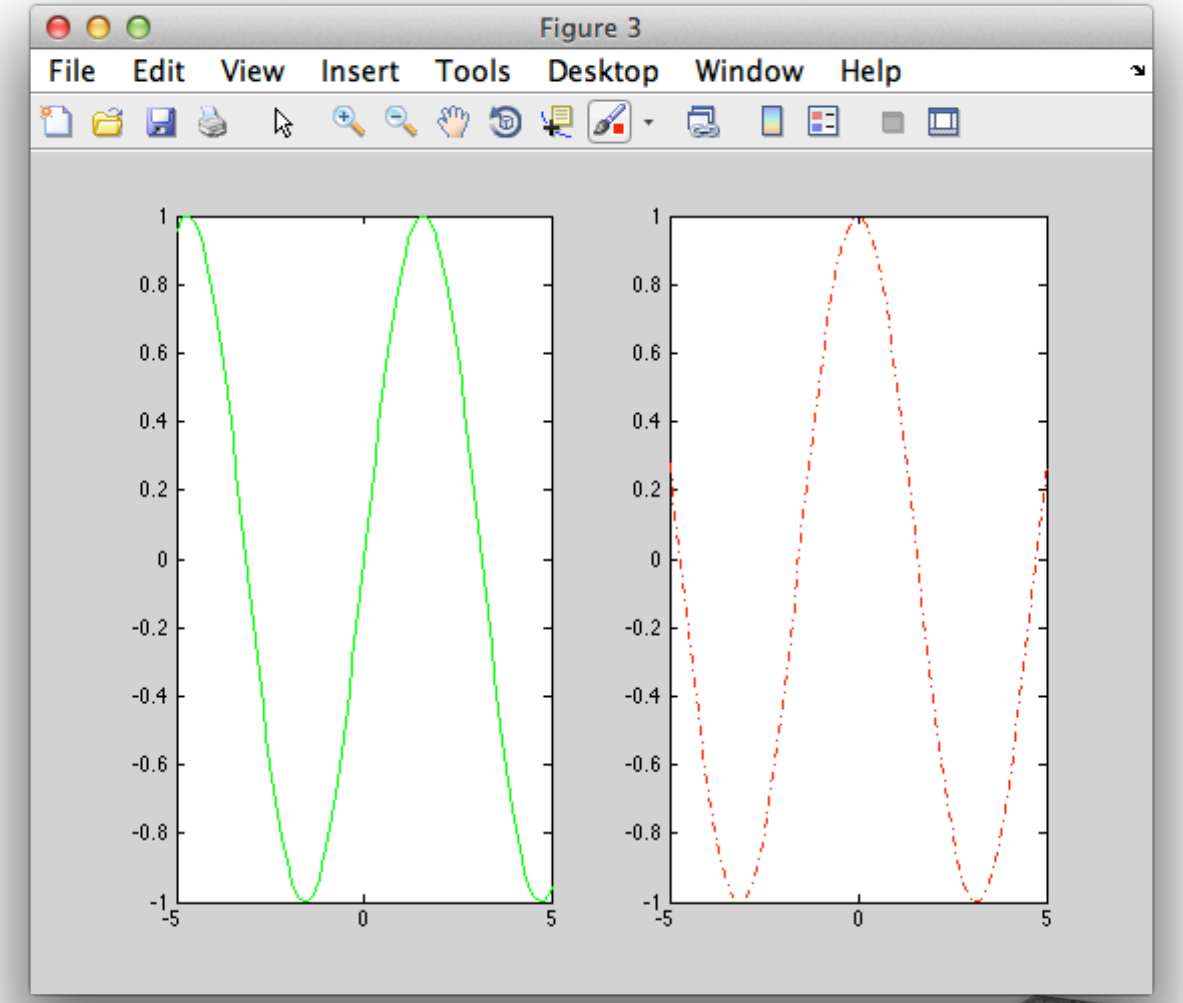
```
figure(2)
plot(x,sin(x),'g')
hold on
plot(x,cos(x),'r-.')
```



Plotting curves in Matlab

```
figure(3)
subplot(1,2,1)
plot(x,sin(x),'g')
subplot(1,2,2)
plot(x,cos(x),'r-.')
```

See help plot for more examples

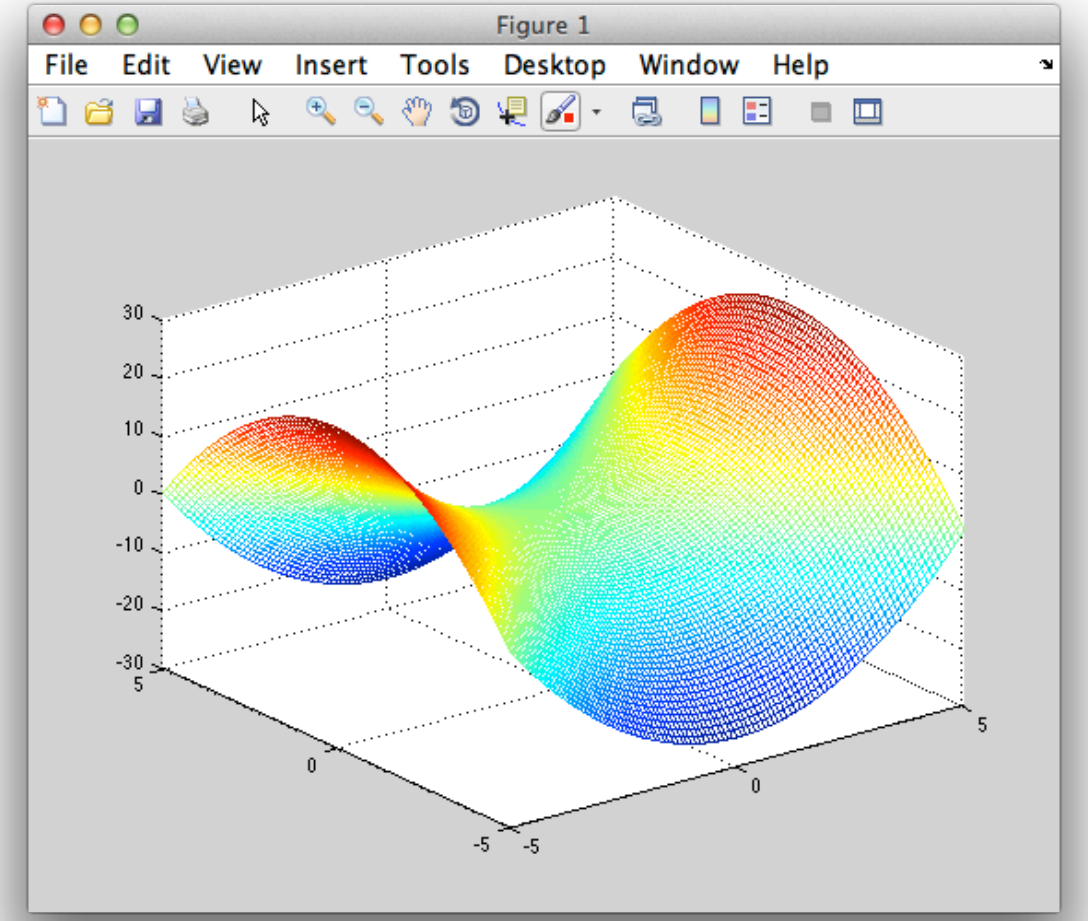


Plotting surfaces in Matlab

Plotting surfaces in Matlab is similar to plotting curves.

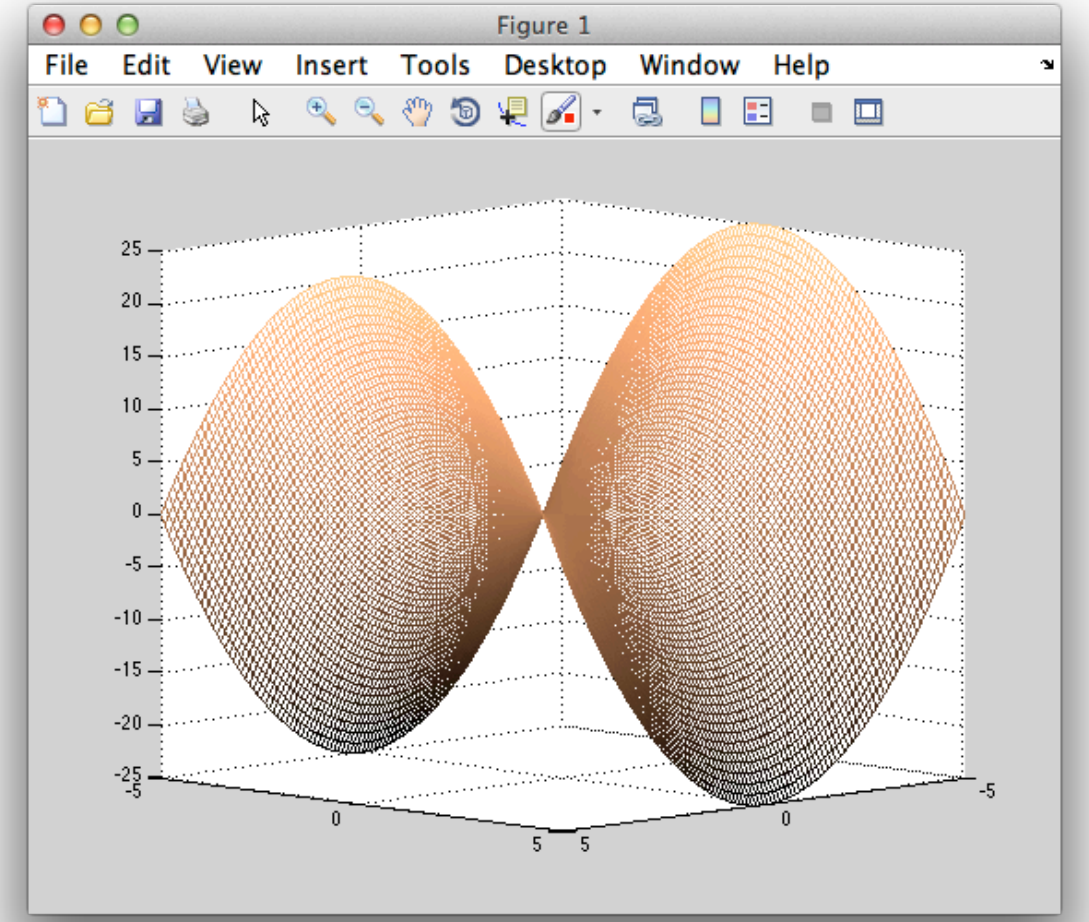
```
[x y] =
meshgrid(-5:.1:5, -5:.1:5);
z = x.^2 - y.^2;
mesh(x, y, z)
```

Here x, y, and z are all matrices of size 101 x 101.



Plotting surfaces in Matlab

```
colormap(copper)
set(gca, ...
'CameraPosition',[45 45 45])
```

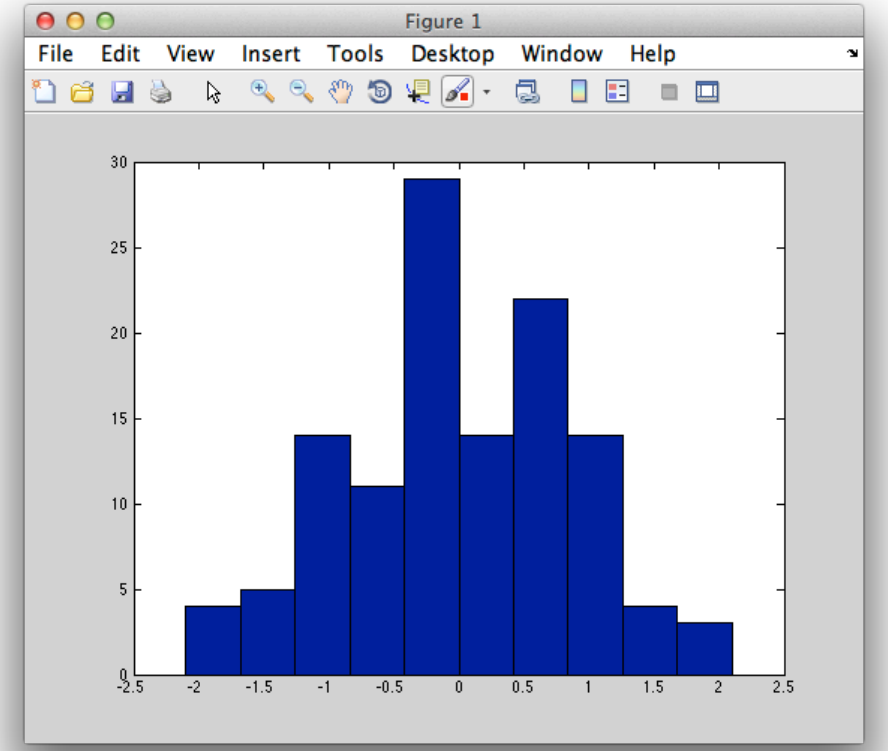


Statistical tests and statistical learning

The code below runs a Kolmogorov-Smirnov test on a column of data.

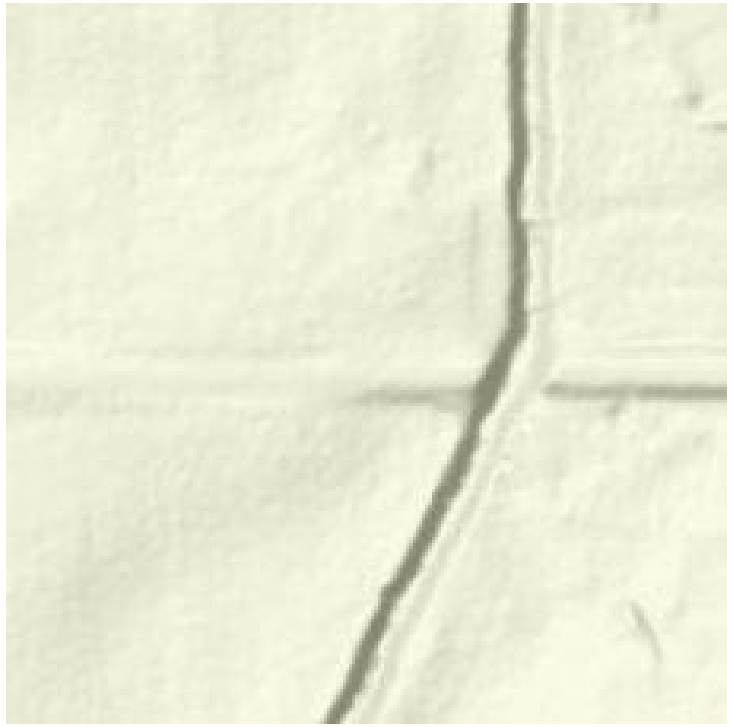
```
load examgrades;
hist(grades(:, 1))
test1 = grades(:,1);
x = (test1 - 75)/10;
[h p] = kstest(x)
h =
     0
p =
   0.561153346365393
hist(x)
```

The result h=0 shows kstest fails to reject the null hypothesis at the default 5% level.

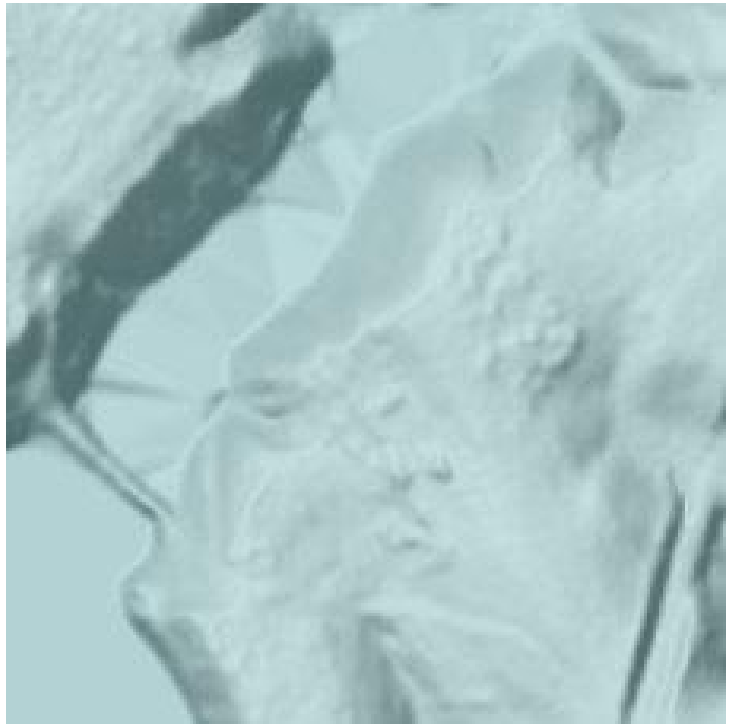


Statistical tests and statistical learning

For a final example we look at code to classify images. We can get a "decent" classifier with a little over a dozen lines of code.



There's a levee!



There's no levee.

Statistical tests and statistical learning

For a final example we look at code to classify images. We can get a "decent" classifier with a little over a dozen lines of code.

Menorah

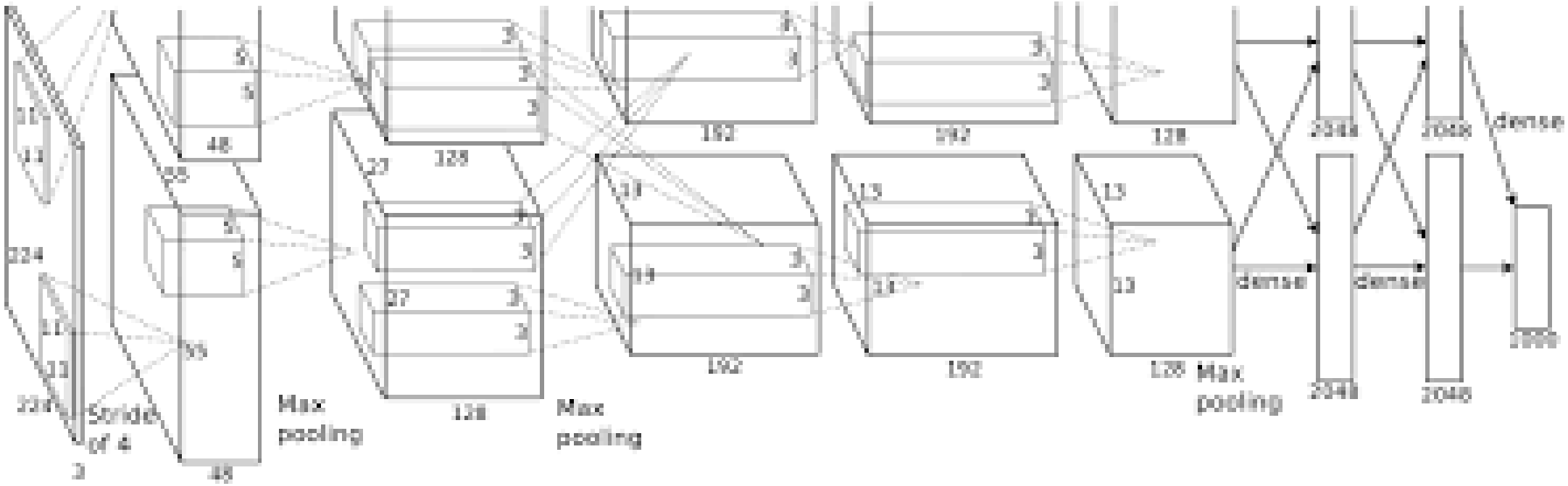
Sunflower

Statistical tests and statistical learning

```
%% Set up the imagestore
rootFolder = '.';
types = {'menorah', 'sunflower'};

imds = imageDatastore(fullfile(rootFolder, types), 'LabelSource',
'foldernames');
imds.ReadFcn = @(filename)readAndPreprocessImage(filename);

%There are 85 sunflowers and 87 menorahs. Let's equalize the numbers.
tbl = countEachLabel(imds);
minSetCount = min(tbl{:,2}); %This number is 85 here.
imds = splitEachLabel(imds, minSetCount, 'randomize');
```

Statistical tests and statistical learning

```
% Load pre-trained AlexNet
net = alexnet();
```

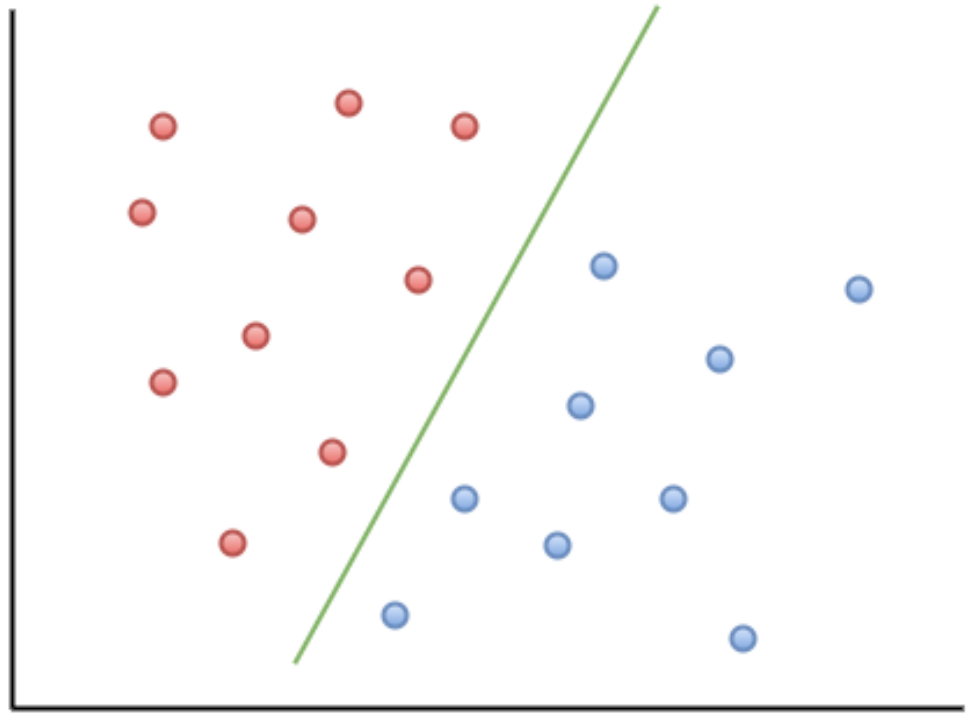


Statistical tests and statistical learning

```
%Spilt training and testing images in a 70/30 split
[trainingSet, testSet] = splitEachLabel(imds, 0.7, 'randomize');

featureLayer = 'fc7';
trainingFeatures = activations(net, trainingSet, featureLayer, ...
    'MiniBatchSize', 32, 'OutputAs', 'columns');

% Train multiclass SVM classifier using a fast linear solver.
classifier = fitcecoc(trainingFeatures, trainingSet.Labels, ...
    'Learners', 'Linear', 'Coding', 'onevsall', 'ObservationsIn', 'columns');
```

Statistical tests and statistical learning

Recall the a support vector machine give a linear classifier. This is the sort of thing that should be doable.



Statistical tests and statistical learning

```
testFeatures = activations(net, testSet, featureLayer);
predictedLabels = predict(classifier, testFeatures);
%The confusion matrix is diagonal. The classifier distinguishes sunflowerer
% and menorahs just fine.

confusionmat(testSet.Labels, predictedLabels)

    26     0
     0    26
```

Statistical tests and statistical learning

The classifier can't be improved. If we did need better we could swap out layers and retrain.

```
layersTransfer = net.Layers(1:end-3);
layers  = [
   layersTransfer
   fullyConnectedLayer(2)
   softmaxLayer
    classificationLayer];

netTransfer = trainNetwork(trainingSet,layers);
```