Lecture 14: Array Tricks

Ways to create an array:

Create the array "by hand"

```
A = [2 5 -3 6 1];
```

2	5	-3	6	1

Create and fill with zeros

```
A = zeros(1,5);
```

0	0	0	0	0

Use the colon notation

```
A = 5:9;
```

5	6	7	8	9

Create and fill with ones

```
A = ones(1,5);
```

1	1	1	1	1

5 evenly-spaced numbers between 0 and 1 inclusive

```
A = linspace(0,1,5);
```

0	0.25	0.5	0.75	1

Fill with random numbers

```
A = rand(1,5);
```

0.098	0.270	0.547	0.958	0.965

Ways to reference array elements:

Given: `A = 5:9;`

5	6	7	8	9

`A(1)`

5

`A(3);`

7

`A(end);`

9

`A(end-1);`

8

`A(2:4)`

6	7	8

`A(end:-1:1)`

9	8	7	6	5

`A(1:2:end);`

5	7	9

`A([1 2])`

5	6

Ways to create subsets of an array:

Given: `A =`

5	5	-2	8	-4	0

Returns all array elements greater than zero

`A(A > 0)`

5	5	8

Returns all array elements less than 5 and greater than -3

`A(A<5 & A>-3)`

-2	0

Returns all array elements equal to 5

`A(A == 5)`

5	5

Using one array to create a subset from another

Suppose you have two arrays that store the position and velocity of an object:

x =

0	5	8	7	4	2

v =

4	2	0	-1	-2	-2

The following commands create a subset of this data that only includes the data points where $v \geq 0$.

```
x1 = x(v>=0)
```

0	5	8

```
v1 = v(v>=0)
```

4	2	0

The Find() Command

The find() command returns the indices of an array that satisfy some condition. For example, the previous example could be done with the find command:

```
indx = find(v>=0)
```

1	2	3

The array indices of the data points of interest are now stored in the array indx. This array can be used to create the subset of x and v values:

```
x1 = x(indx)
```

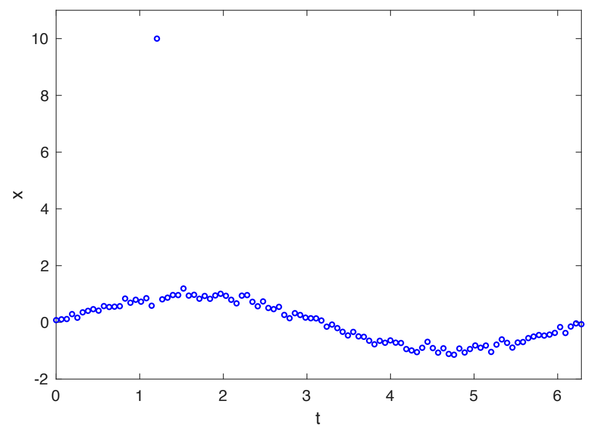
0	5	8

```
v1 = v(indx)
```

4	2	0

Example: Rejecting “bad” data points

Consider the following data set:



The position measurements generally appear to vary sinusoidally with time. There’s some noise in the data, but it seems to be less than the periodic variation. One data point, however, seems to be “off.” Let’s suppose you have some good reason to reject it (like your lab partner admits that he bumped the apparatus during that measurement). We want to reject this point.

Example: Rejecting “bad” data points

Here’s a simple way of creating a subset of your data that excludes the “bad” data point. We observe that the “good” data have $x < 4$.

`x` = array containing positions
`t` = array containing times

Create an array named `good` that stores the array indices of all the “good” data:

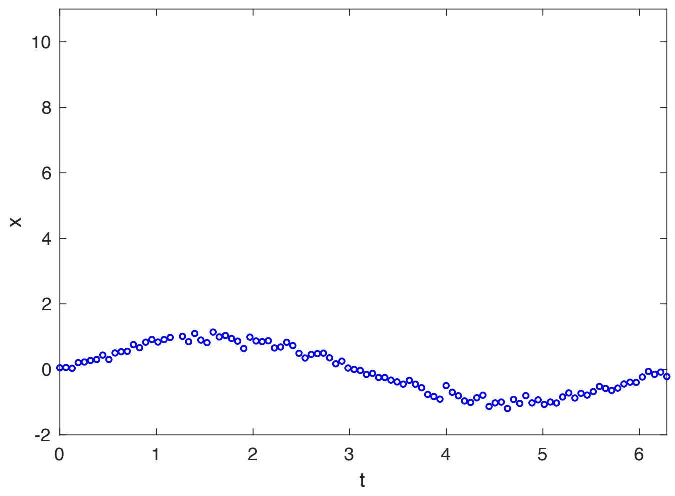
```
good = find(x < 4);
```

Now, create new arrays containing only the good data:

```
x_good = x(good);
t_good = t(good);
```

Example: Rejecting “bad” data points

Plotting these new arrays shows that we have, in fact, removed the bad data point:

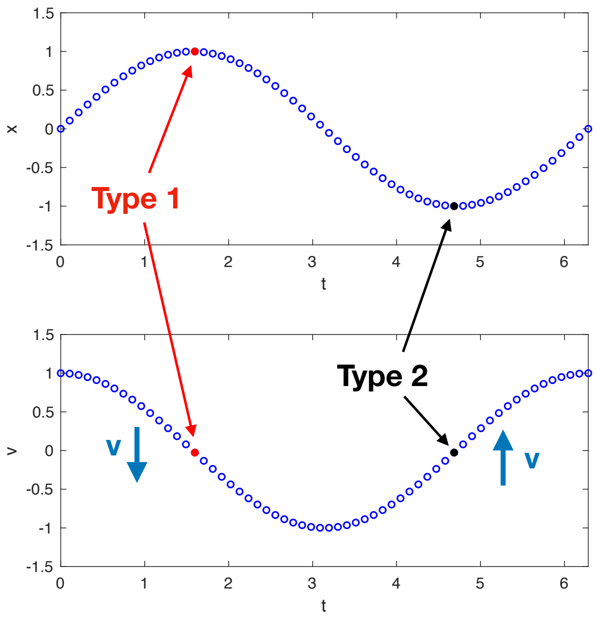


Example: Multiple Event Handling

Consider the following example that uses event handling when solving an ODE. The example defines two types of events:

Event **Type 1**:
Local maxima defined when the velocity v crosses zero from positive to negative

Event **Type 2**:
Local minima defined when the velocity v crosses zero from negative to positive



Event function `example_events.m`

Here's the event function to locate local min and local max:

```
function [trigger,is_terminal,direction] = example_events(t,w)

x = w(1);                % w(1) stores x
v = w(2);                % w(2) stores v

trigger = [v v];         % Event 1 is triggered when v = 0
                         % Event 2 is triggered when v = 0

is_terminal = [0 0];     % Event 1 doesn't stop the integration
                         % Event 2 doesn't stop the integration

direction = [-1 1];      % Event 1 detects local max
                         % Event 2 detects local min
```

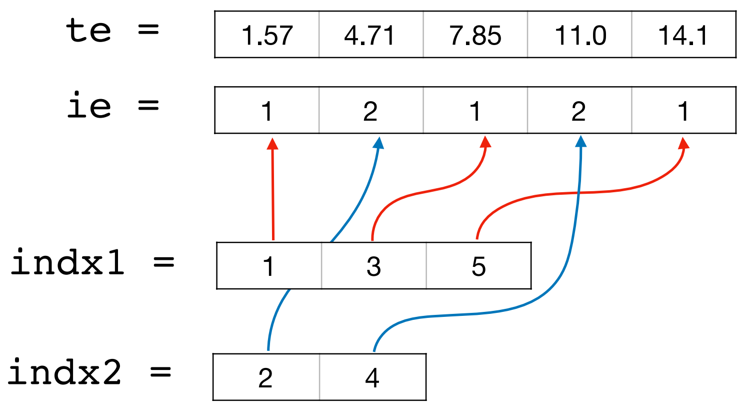
Main Code Snippet:

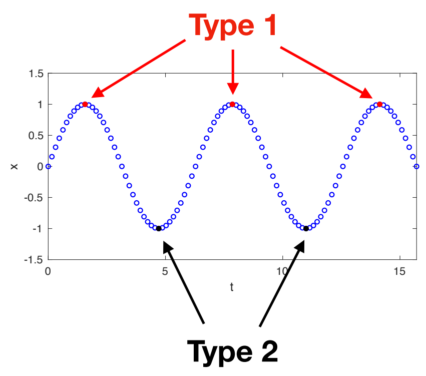
Here's how we can extract type1 and type2 events from the matrices returned by ode45():

```
[t,w,te,we,ie] = ode45(@ode04_derivs, [tBegin tEnd], ...
     [x0 v0], options);

xe = we(:,1);               % position for each event
ve = we(:,2);               % velocity for each event

% find event indices of type 1 and type 2 events
indx1 = find(ie == 1);      % local max
indx2 = find(ie == 2);      % local min

te1 = te(indx1);            % times of local max
te2 = te(indx2);            % times of local min

xe1 = xe(indx1);            % positions of local max
xe2 = xe(indx2);            % positions of local min
```

How could you calculate the period of the oscillation using the event arrays `te1` or `te2`?

Here’s how you can picture these arrays:





`te1 = te(indx1) =` | 1.57 | 7.85 | 14.1 |

`te2 = te(indx2) =` | 4.71 | 11.0 |