Numerical Differentiation

The derivative of a function $f(x)$ is defined as

$$f'(x) = \frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{(x + \Delta x) - x}$$

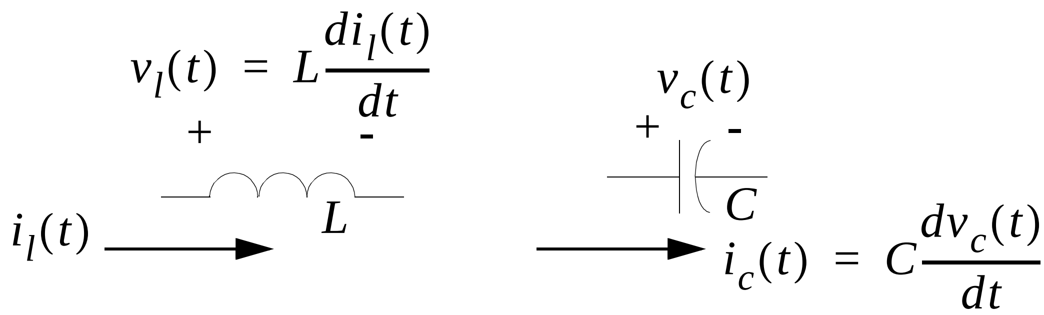
if the limit exists

- Physical examples of the derivative in action are:
  - Given $x(t)$ is the position in meters of an object at time *t*, the first derivative with respect to *t*, $x'(t)$, is the velocity in meters/second (note: The integral of velocity is position to within a constant)
  - Given $x(t)$ is the velocity in meters/second of an object at time *t*, the first derivative with respect to *t*, $x'(t)$, is the acceleration in meters/second squared
  - In an electrical circuit with time varying voltages and currents, the current flowing through a capacitor is given by

$$i_c(t) = C\frac{dv_c(t)}{dt}$$

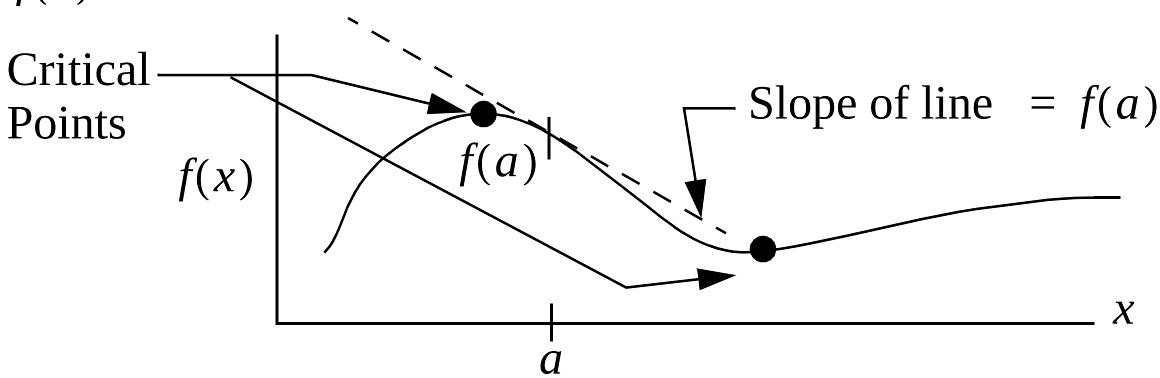
  - In an electrical circuit with time varying voltages and currents, the voltage across an inductor is given by

$$v_l(t) = L\frac{di_l(t)}{dt}$$



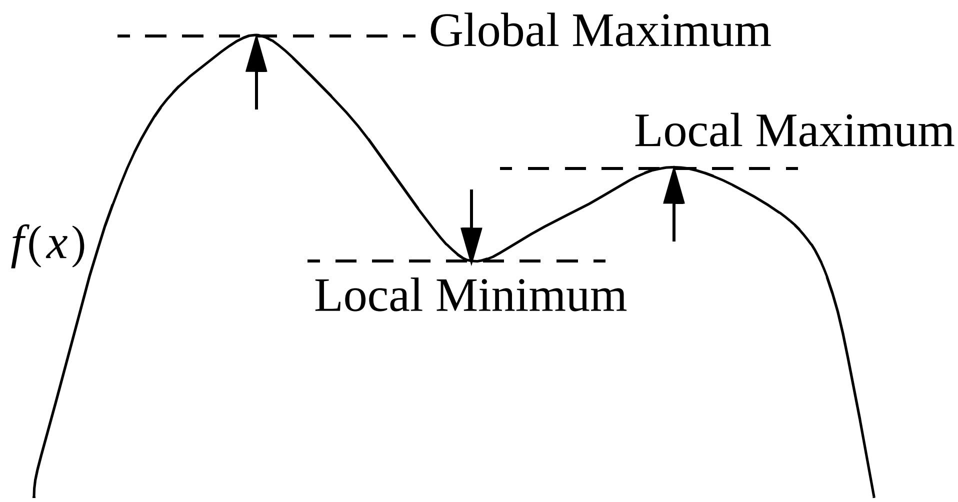
The Derivative and the Slope

- The derivative of $f(x)$ at a is the slope of the line tangent to $f(x)$ at a



- Points where the derivative of $f(x)$ is equal to zero are known as *critical points*
- The function may be horizontal in this region or may have reached a so-called *extrema* point, a point where $f(x)$ is at a local (global) maximum or local (global) minimum
- In calculus you learn that the second derivative can be used to determine if an extrema is a maximum or minimum
  – If the second derivative of $f(x)$ is positive the extrema is a minimum

– If the second derivative of $f(x)$ is negative the extrema is a maximum

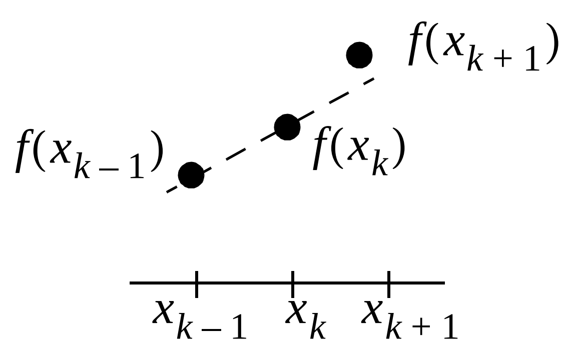


Derivative Approximations using Differences

- Numerical algorithms for computing the derivative of a function require the estimate of the slope of the function for some particular range of x values
- Three common approaches are the *backward difference*, *forward difference*, and the *central difference*

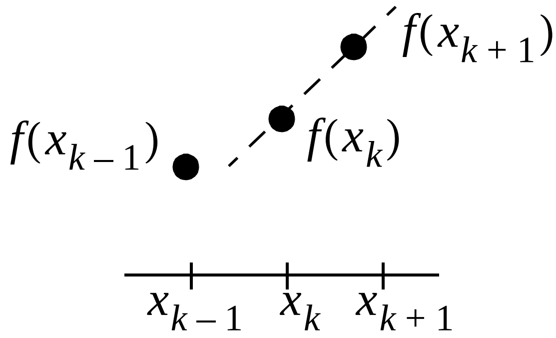
Backward Difference

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$



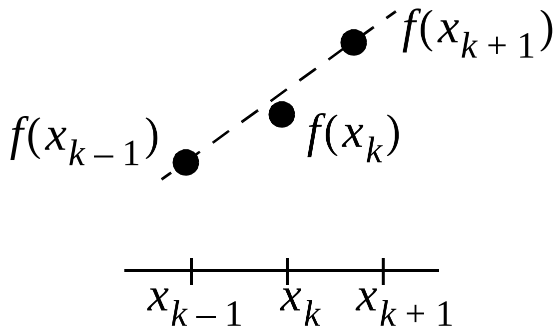
Forward Difference

$$f'(x_k) \approx \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k}$$



Central Difference

$$f'(x_k) \approx \frac{f(x_{k+1}) - f(x_{k-1})}{x_{k+1} - x_{k-1}}$$



- To estimate the second derivative we simple apply one of the above algorithms a second time, that is using the backward difference

$$f''(x) \approx \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}}$$

The MATLAB `diff` Function

- To make computing the numerical derivative a bit easier, MATLAB has the function `diff(x)` which computes the differences between adjacent values of the vector `x`
- The vector (matrix) that `diff(x)` returns contains one less element (row) than `x`

- If samples of $f(x)$ are contained in vector `y` and the corresponding x values in vector `x`, the derivative can be estimated using

  ```
  deriv_y = diff(y)./diff(x);
  ```

- The corresponding x values are obtained from the original `x` vector by trimming either the first or last value
  - Trimming the last value results in a forward difference estimate
  - Trimming the first value results in a backward difference estimate

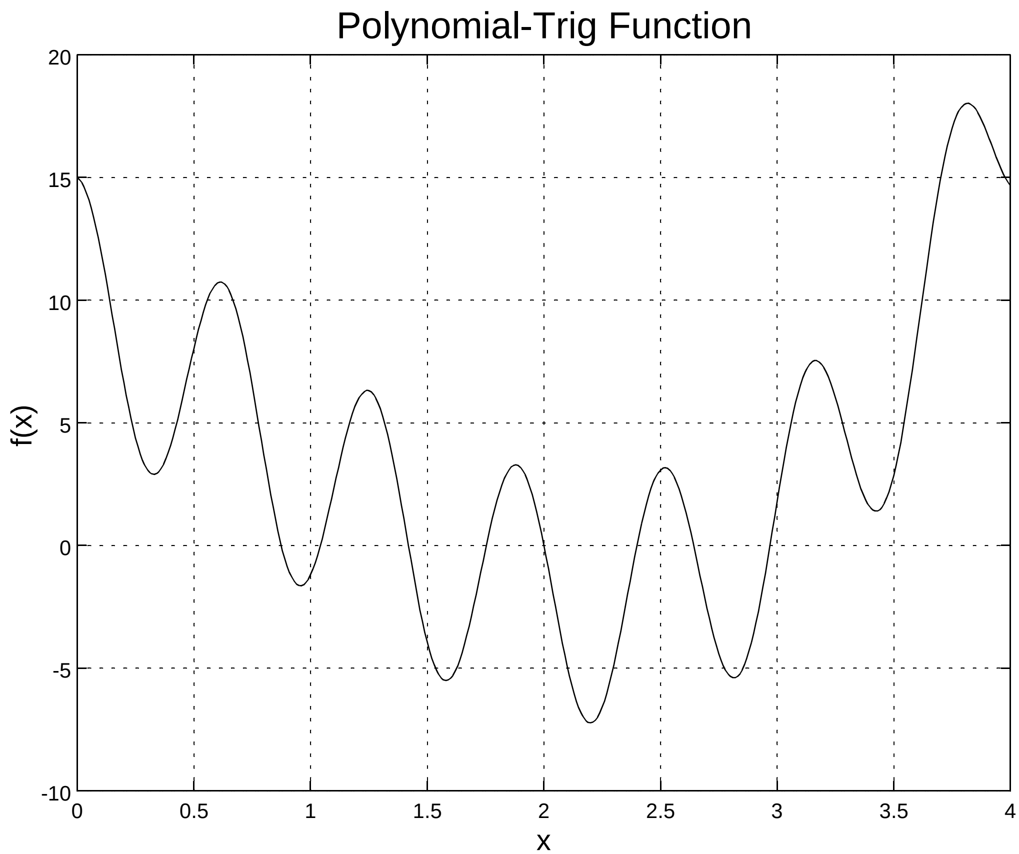
Example: Find the derivative of a function containing polynomial terms and a trig function

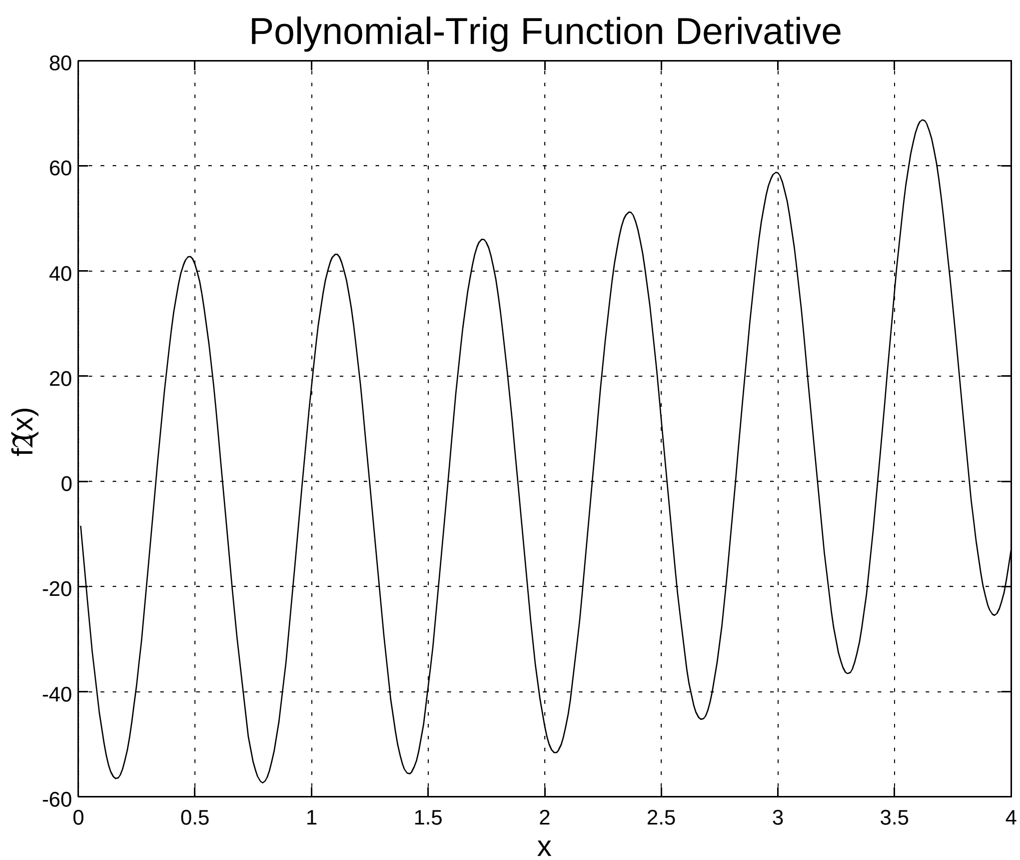
$$f(x) = 5\cos(10x) + x^3 - 2x^2 - 6x + 10$$

```
» x = 0:.01:4;
» y = 5*cos(10*x) + x.^3 - 2*x.^2 - 6*x +10;
» plot(x,y)
» deriv_y = diff(y)./diff(x);
» xd = x(2:length(x)-1);
» figure
» xd = x(2:length(x)); % Backward difference x values
» plot(xd,deriv_y)
```

- Direct analysis also yields

$$f'(x) = -50\sin(10x) + 3x^2 - 4x - 6$$


Polynomial-Trig Function
20
15
10
5
0
-5
-10
f(x)
0
0.5
1
1.5
2
2.5
3
3.5
4
x


Polynomial-Trig Function Derivative
80
60
40
20
0
-20
-40
-60
0
0.5
1
1.5
2
2.5
3
3.5
4
x

– Note: Here we have used a backward difference calculation

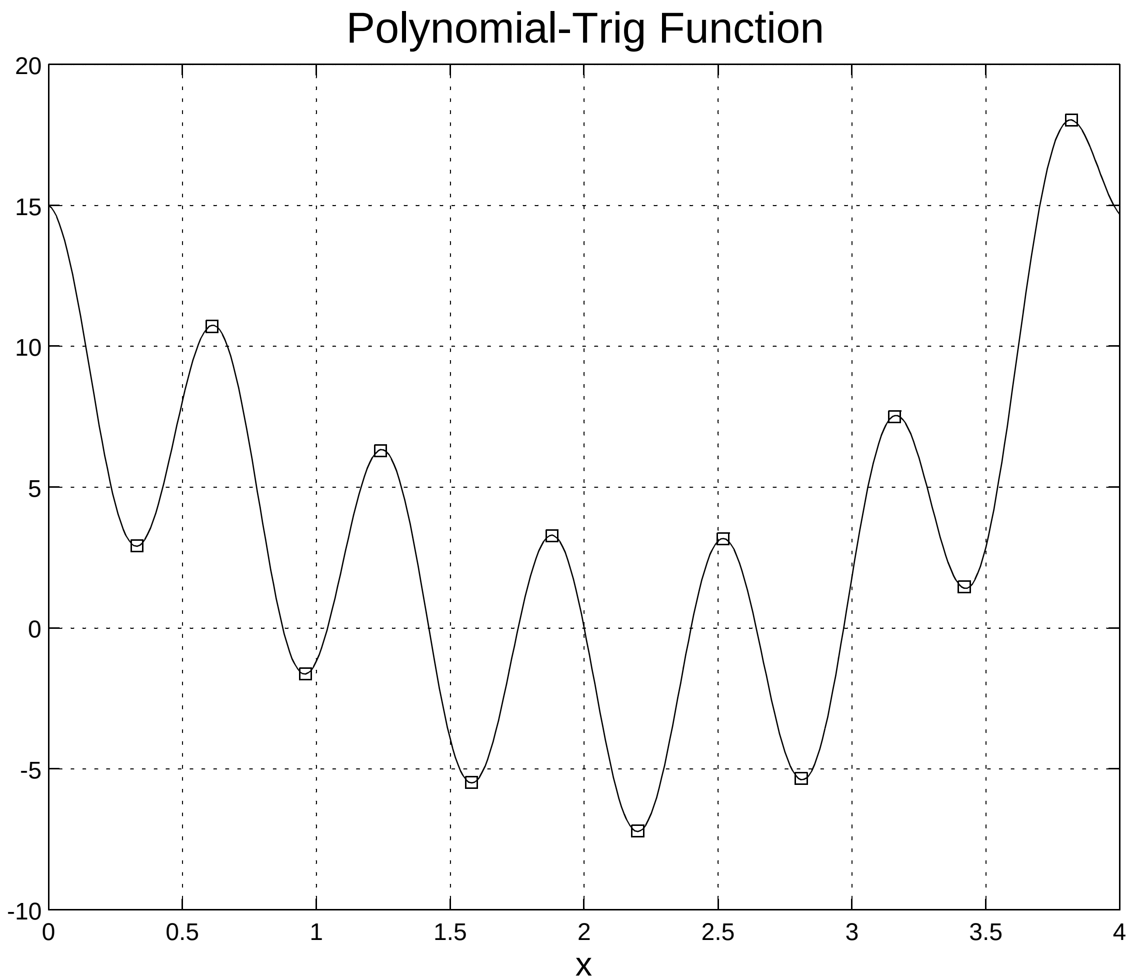
- To locate the critical points we can find the zero crossings of the derivative function

```
» delay_mult = deriv_y(1:length(deriv_y)-1)
    .*deriv_y(2:length(deriv_y));
» critical = xd(find(delay_mult < 0));
critical =

  Columns 1 through 5
    0.3300    0.6100    0.9600    1.2400    1.5800
  Columns 6 through 10
    1.8800    2.2000    2.5200    2.8100    3.1600
  Columns 11 through 12
    3.4200    3.8200
```

- We can plot these points over the original function to see if they are maxima or minima

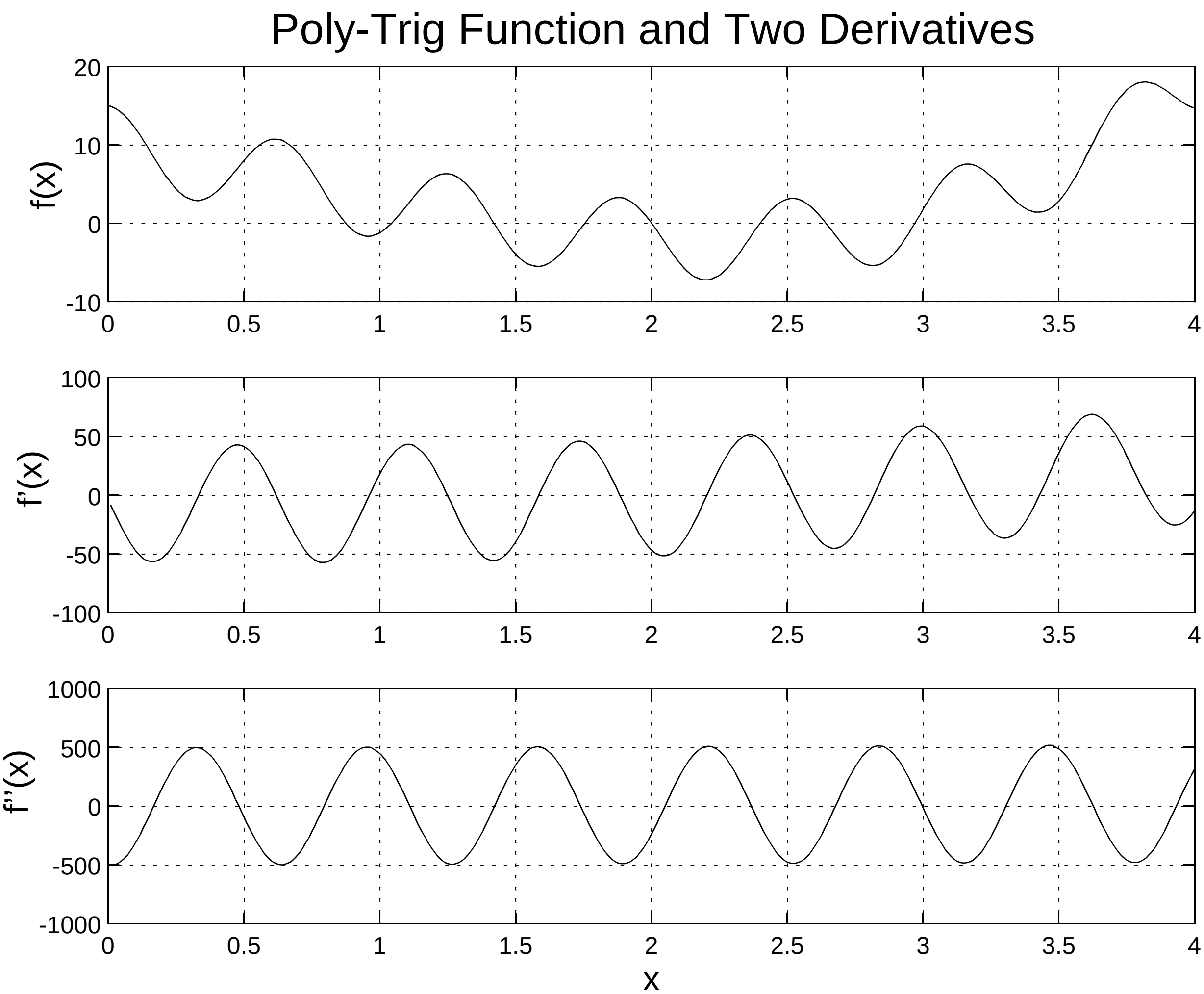
```
» plot(x,y)
» hold
Current plot held
» plot(xd(find(delay_mult < 0)),y(find(delay_mult <
0)),'s');
» title('Polynomial-Trig Function','fontsize',18)
» ylabel('f\''(x)','fontsize',14);
» xlabel('x','fontsize',14)
» grid
```



- To classify the extrema as local maxima or minima we can compute the second derivative and classify the samples of the second derivative at the extrema points

```
» dderiv_y = diff(deriv_y)./diff(xd);
» xdd = xd(2:length(xd));
» subplot(311)
» plot(x,y)
» grid
» ylabel('f(x)','fontsize',14)
» title('Poly-Trig Function and Two Derivatives',...
    'fontsize',18)
» subplot(312)
» plot(xd,deriv_y)
```

```
» grid
» ylabel('f\prime(x)','fontsize',14)
» subplot(313)
» plot(xdd,dderiv_y)
» grid
» xlabel('x','fontsize',14)
» ylabel('f\prime\prime(x)','fontsize',14)
```



- The extrema stored in `critical` are those indices of `x` where $f'(x)$ has zero crossings

- We can apply the same criteria, i.e., `find(delay_mult < 0)`, to determine the sign of the corresponding values critical values of the second derivative

```
» % Since the x values of dderiv_y are shifted to the
» % left by one we use the critical values indices
» % reduced by one.
» % Place the critical values of dderiv_y and xdd in
» % special arrays
» ddy_crit = dderiv_y(find(delay_mult < 0)-1)

ddy_crit =

  Columns 1 through 5
  496.6516 -480.0852  499.8707 -478.5845  504.9876
  Columns 6 through 10
 -486.7868  506.6496 -488.2557  493.7020 -482.9189
  Columns 11 through 12
  464.6619 -441.1139
» ddx_crit = xdd(find(delay_mult < 0)-1);
```

- We see that the second derivative at the critical values is either large negative or large positive, hence in this case we can easily decide if the extrema are maxima or minima

```
» % Now find the x values that correspond to maxima
» max_xval = ddx_crit(find(ddy_crit < 0))

max_xval =
    0.6100    1.2400    1.8800    2.5200    3.1600
    3.8200
```

```
» % Now find the x values  that correspond to minima
» min_xval = ddx_crit(find(ddy_crit > 0))

min_xval =
    0.3300    0.9600    1.5800    2.2000    2.8100
    3.4200
```