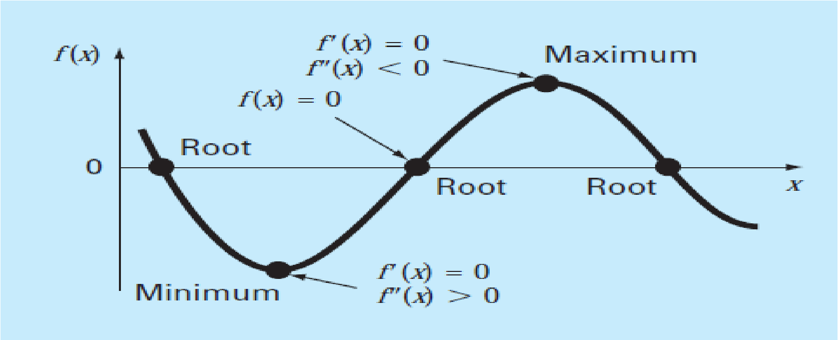
Optimisation: Finding Minima and Maxima

1. Mathematically, the optimal point on a function is that point where its gradient is flat.



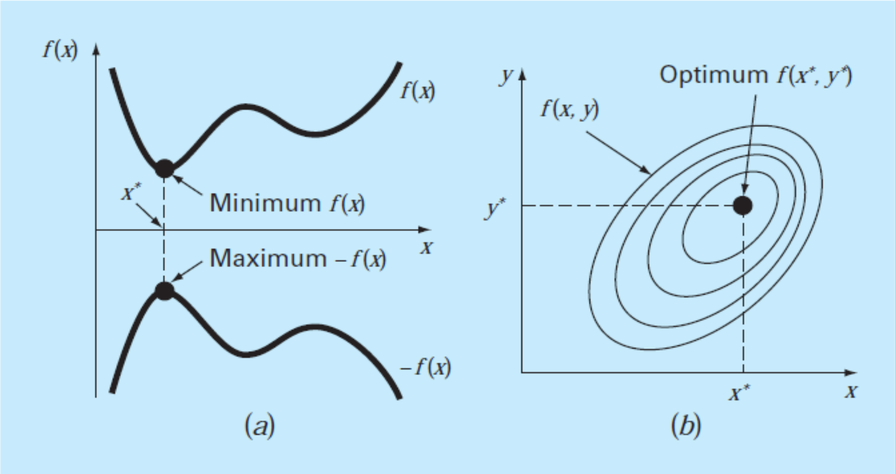
2. That is, the optimum point is usually at a maximum or at a minimum point of a curve.
3. Imagine that you are asked to plot a cost function for an operation that is dependent on a certain variable, and this function has a clear minimum point.
4. Computationally, a computer algorithm must identify this point systematically using the mathematical fact that the instantaneous gradient or derivative at the minimum point has a value of zero.

Optimisation: Finding Minima and Maxima

5. There are two main types of optimisation:

A. **One dimensional optimisation** (a curve in a plane (2 dimensional)

B. **Two dimensional optimisation** (a three dimensional functional surface).



6. Most optimisations are formulated as follows: Find x which maximises/minimises f(x) subject to the following constraints:

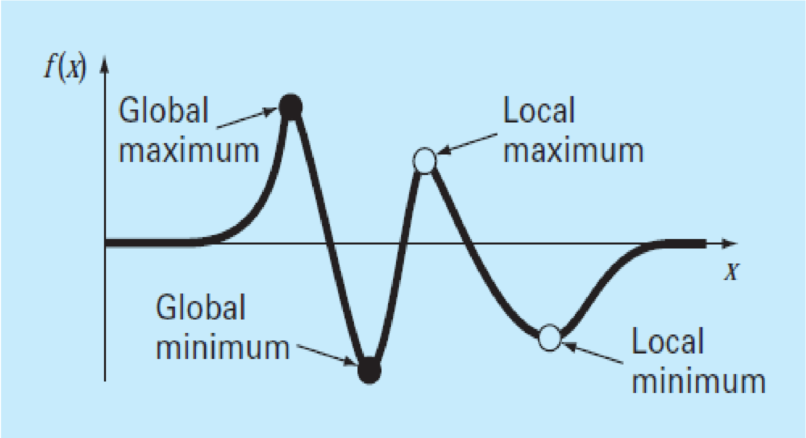
$$d_i(\mathbf{x}) \le a_i \qquad i = 1, 2, \ldots, m$$
$$e_i(\mathbf{x}) = b_i \qquad i = 1, 2, \ldots, p$$

7. d(x) are inequality constraints, e(x) are equality constraints

Optimisation: Finding Minima and Maxima

1. If both the function and the constraints are linear functions then the optimisation is an example of ***linear*** programming.
2. If f(x) is quadratic but its constraints are linear we have ***quadratic*** programming.
3. If both f(x) and the constraints are non-linear we have ***non-linear*** programming.
4. The degrees of freedom in an optimisation problem is calculated by the term n-p-m.
5. n = the number of dimensions in the x vector; p = the number of equality constraints; m is the number of inequality constraints.
6. To obtain a solution m+p < n. If either m or p or a combination (m+p) > n, the optimisation is said to be *overconstrained, i.e.,* the optimisation can't proceed.
7. **Always important to pre-assess if a problem is over-constrained** prior to committing further futile modelling effort.

Optimisation: Finding Minima and Maxima



8. In an unconstrained optimisation, one still has to provide for complications such as the distinction between global and local maxima/minima.
9. The most basic unconstrained one-dimensional search technique is the **golden search technique.**
10. This is closely modelled on the bisection method we used to find a function root, except this time we are targeting a function minimum/maximum.

Computational Methods and Modelling

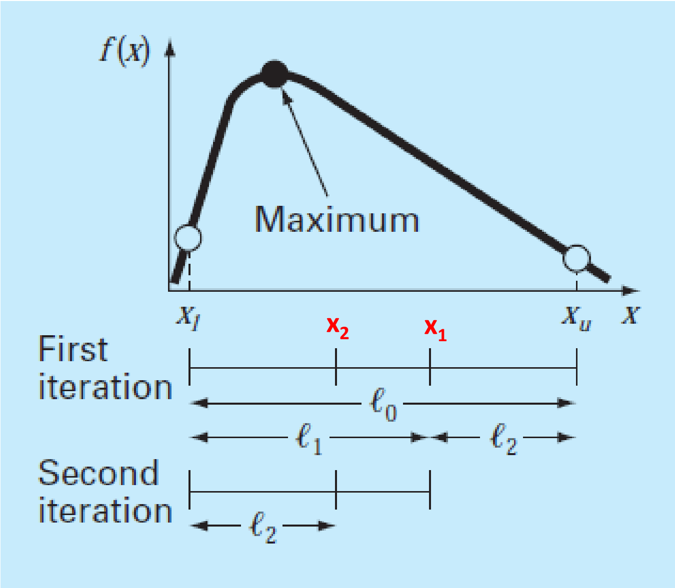
Lecture 10: Dr. Edward McCarthy

Topic 1: Optimisation of Functions.

THE UNIVERSITY *of* EDINBURGH
School of Engineering

Unidimensional Opt: Golden Search Technique

1. Instead of bisecting a suspected root, GS relies on selecting two estimate points either side of a maximum or minimum point (graph).



2. An effective strategy for selection of estimation points is achieved by firstly deriving a specific number called the Golden Ratio:

$$\ell_0 = \ell_1 + \ell_2$$

$$\frac{\ell_1}{\ell_0} = \frac{\ell_2}{\ell_1}$$

THE UNIVERSITY of EDINBURGH
School of Engineering

Unidimensional Opt: Golden Search Technique

6. By substitution of one equation into the other the following condition emerges

$$\frac{\ell_1}{\ell_1 + \ell_2} = \frac{\ell_2}{\ell_1}$$

7. The reciprocal of both sides is taken for R = l_2/l_1 giving a quadratic expression in R.

$$1 + R = \frac{1}{R} \qquad R^2 + R - 1 = 0$$

6. Solving for R we obtain the Golden Ratio, one of the most significant numbers in all of mathematics for centuries

$$R = \frac{-1 + \sqrt{1 - 4(-1)}}{2} = \frac{\sqrt{5} - 1}{2} = 0.61803\ldots$$

7. Two guesses x_u and x_l plus the Golden Ratio are now used to select new intermediate points:

$$d = \frac{\sqrt{5} - 1}{2}(x_u - x_l)$$
$$x_1 = x_l + d$$
$$x_2 = x_u - d$$

Unidimensional Opt: Golden Search Technique

1. After the initial bracketing interval has been selected, the function is evaluated at the two points.
2. Then two situations can arise:

A. $f(x_1) > f(x_2)$, then all points left of x_2 can be eliminated as no maximum can occur here. x_2 becomes the new x_l or leftmost point of the new region of interest.

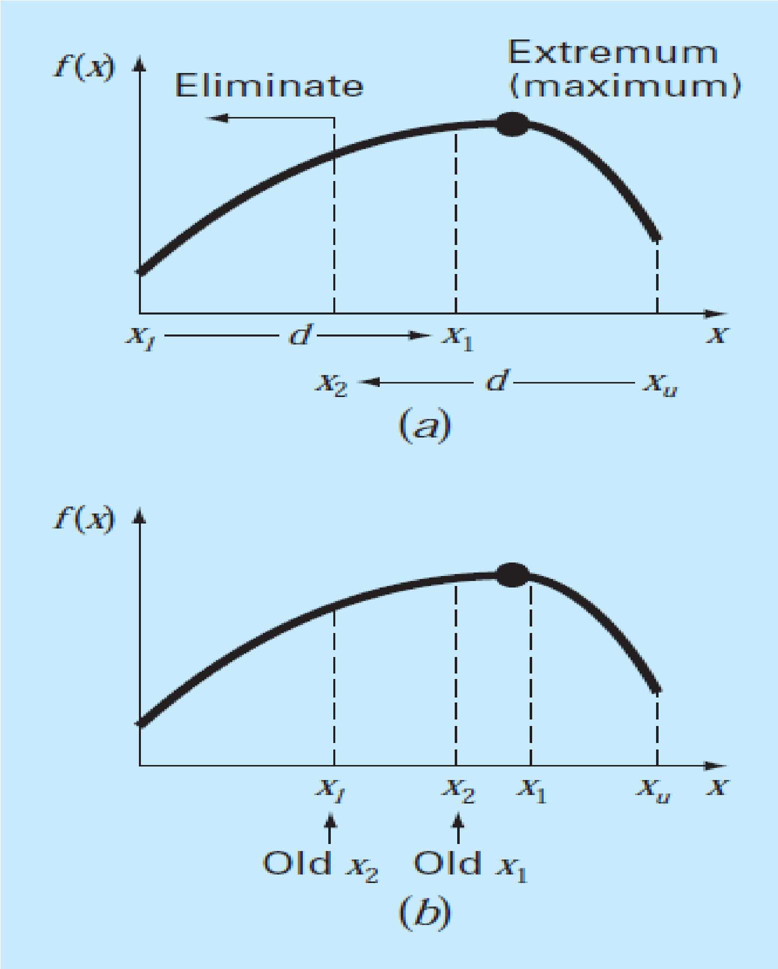
B. $f(x_1) < f(x_2)$, then all points right of x_1 can be eliminated from the region of interest. x_l becomes the new x_u for the next iteration.

3. The next step is to calculate the new x1 for the new region of interest. (Because we are using the Golden Ratio, we do not need to recalculate a new x_2 since it has been re-assigned to the old value of x_1.

$$x_1 = x_l + \frac{\sqrt{5}-1}{2}(x_u - x_l)$$

THE UNIVERSITY of EDINBURGH
School of Engineering

Unidimensional Opt: Golden Search Technique



4. The Golden Ratio has halved the number of necessary function evaluations needed to complete the algorithm.
5. The algorithm continues until x_1 and x_2 converge on the maximum point.
6. Convergence is guaranteed, though the rate of convergence is finite.

Unidimensional Opt: Golden Search Technique

1. Maximise the function using the GS routine:

$$f(x) = 2\sin x - \frac{x^2}{10}$$

3. The limits for this optimisation are: $x_l = 0$ and $x_u = 4$.
4. We use the golden ratio to create two new internal points in the domain of interest:

$$d = \frac{\sqrt{5}-1}{2}(4-0) = 2.472$$

$$x_1 = 0 + 2.472 = 2.472$$

$$x_2 = 4 - 2.472 = 1.528$$

5. Evaluate the function at the two interior points:

$$f(x_2) = f(1.528) = 2\sin(1.528) - \frac{1.528^2}{10} = 1.765$$

$$f(x_1) = f(2.472) = 0.63$$

6. The function value $f(x_2)$ is greater than $f(x_1)$; therefore no maximum exists in this interval.

THE UNIVERSITY *of* EDINBURGH
School of Engineering

Unidimensional Opt: Golden Search Technique

7. This means we can narrow the search domain by setting x_1 to be the new upper bound, $x_{u,new}$ thus disregarding the interval $x_1 < x < x_{u,old}$ from further searching. ($x_l = 0$; $x_{u,new} = 2.472$).
8. Also $x_{2,old}$ now becomes $x_{1,i+1}$.
9. Now ***only the new value of x_2*** needs to be computed using the GS as follows:

$$d = \frac{\sqrt{5}-1}{2}(2.472 - 0) = 1.528$$

$$x_2 = 2.4721 - 1.528 = 0.944$$

10. This iteration is more efficient as the number of computations has been reduced from three to two (and so on for all future iterations).
11. Since $f(x_{2,new}) = 1.5310, < f(x_{1,new}) = 1.765$, we know that the maximum of the function exists in the interval defined by $x_{2,new}$, $x_{1,new}$, and x.

Unidimensional Opt: Golden Search Technique

Table of Iterations for GS Optimisation

i	x_l	$f(x_l)$	x_2	$f(x_2)$	x_1	$f(x_1)$	x_u	$f(x_u)$	d
1	0	0	1.5279	1.7647	2.4721	0.6300	4.0000	−3.1136	2.4721
2	0	0	0.9443	1.5310	1.5279	1.7647	2.4721	0.6300	1.5279
3	0.9443	1.5310	1.5279	1.7647	1.8885	1.5432	2.4721	0.6300	0.9443
4	0.9443	1.5310	1.3050	1.7595	1.5279	1.7647	1.8885	1.5432	0.5836
5	1.3050	1.7595	1.5279	1.7647	1.6656	1.7136	1.8885	1.5432	0.3607
6	1.3050	1.7595	1.4427	1.7755	1.5279	1.7647	1.6656	1.7136	0.2229
7	1.3050	1.7595	1.3901	1.7742	1.4427	1.7755	1.5279	1.7647	0.1378
8	1.3901	1.7742	1.4427	1.7755	1.4752	1.7732	1.5279	1.7647	0.0851

1. The optimisation proceeds through 8 iterations, and it is clear to see that the four function evaluations begin to converge on the ultimate value for the function maximum in the original domain $x_{l,1} < x < x_{u,1}$.
2. The approximation error for a given estimate of the optimal x value, x_{opt} in a given iteration is calculated as follows: (where R = golden ratio).

$$\varepsilon_a = (1 - R)\left|\frac{x_u - x_l}{x_{opt}}\right| 100\%$$

Computational Methods and Modelling

Lecture 10: Dr. Edward McCarthy

Topic 1: Optimisation of Functions.

THE UNIVERSITY *of* EDINBURGH
School of Engineering

Unidimensional Opt: Golden Search Technique

```
FUNCTION Gold (xlow, xhigh, maxit, es, fx)
R = (5^0.5 - 1)/2
xℓ = xlow: xu = xhigh
iter = 1
d  = R * (xu - xℓ)
x1 = xℓ + d: x2 = xu - d
f1 = f(x1)
f2 = f(x2)
IF f1 > f2 THEN                              IF f1 < f2 THEN
  xopt = x1
  fx = f1
ELSE
  xopt = x2
  fx = f2
END IF
DO
  d = R*d
  IF f1 > f2 THEN                            IF f1 < f2 THEN
     xℓ = x2
     x2 = x1
     x1 = xℓ+d
     f2 = f1
     f1 = f(x1)
  ELSE
     xu = x1
     x1 = x2
     x2 = xu-d
     f1 = f2
     f2 = f(x2)
  END IF
  iter = iter+1
  IF f1 > f2 THEN                            IF f1 < f2 THEN
     xopt = x1
     fx = f1
  ELSE
     xopt = x2
     fx = f2
  END IF
  IF xopt ≠ 0. THEN
     ea = (1.-R) *ABS((xu - xℓ)/xopt) * 100.
  END IF
  IF ea ≤ es OR iter ≥ maxit EXIT
END DO
Gold = xopt
END Gold
```

(*a*) **Maximization** (*b*) **Minimization**

Maximisation line **Minimisation line**

Define the Golden Ratio R, define lower and upper limits of initial search region. Calculate inner search points, x_1 and x_2 using GR.

Set x_1 or x_2 as the initial dummy optimisation value x_{opt} depending on whether f_1 > or < f_2

Reset d (the floating golden interval) to R times its original value.

Recompare f_1 and f_2 to decide which is greater. If f_1 is greater than f_2, set x_l to x_2, and x_2 now becomes x_1, while the old x_1 is reset to x_l +d. Similarly f_2 is reset to f_1 (f_1 is now explicitly set to $f(x_1)$).

Otherwise, if f_1 is less than f_2, set x_u to x_1, and x_l now becomes x_2, while the old x_2 is reset to x_u-d. Similarly f_1 is reset to f_2 (f_2 is now explicitly set to $f(x_2)$).

We now move to the next iteration and implement another test of the values of the inner function values f_1 and f_2

Set x_{opt} to be x_1 and f_x to be f_1; otherwise if f_2 > f_1, set x_2 to be x_{opt} and fx = f2

If x_{opt} is not zero, calculate the approximation error, ea. If it is less than acceptable error, es, xopt is accepted as the optimum x-value for maximisation. If not DO repeats.

THE UNIVERSITY of EDINBURGH
School of Engineering

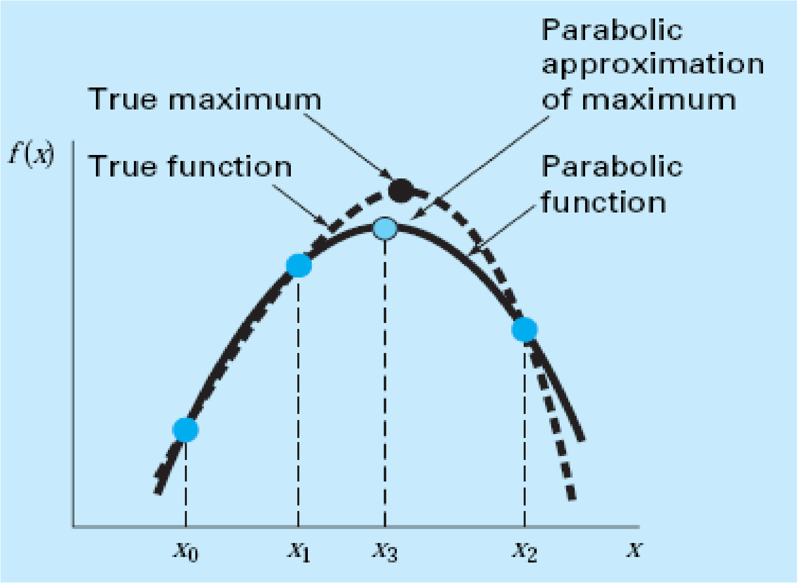
Parabolic Interpolation

Basis of Method

1. Parabolic Interpolation is used in many optimisation routines, since a parabola or quadratic curve can often provide a very good approximation to the test function being evaluated.
2. From theory, a unique parabola joins any three points in two-dimensional space.
3. Therefore if we suspect that an optimum point lies in the interval of any three points on a curve, we can fit those three points with a unique parabola.
4. We then differentiate the parabola to find a maximum point that is then an estimate of the true parabola of the test function.
5. The following expression can be used to do this using finite evaluations of points and their corresponding function values:

Parabolic Interpolation

Parabolic Function v. True Function



$$x_3 = \frac{f(x_0)(x_1^2 - x_2^2) + f(x_1)(x_2^2 - x_0^2) + f(x_2)(x_0^2 - x_1^2)}{2\,f(x_0)(x_1 - x_2) + 2\,f(x_1)(x_2 - x_0) + 2\,f(x_2)(x_0 - x_1)}$$

6. x_0, x_1, and x_2 are the initial guesses, and x_3 is the calculated approximation to the point, x_{opt}, that delivers a maximum of f(x).
7. The new points for the next iteration can be done by reassigning sequentially, i.e., $x_{0,new}=x_1$; $x_{1,new}=x_2$; $x_{2,new}=x_3$ etc.

Parabolic Interpolation: Example

1. Take a function such as:

$$f(x) = 2\sin x - \frac{x^2}{10}$$

2. We want to find the maximum value of this function using parabolic interpolation.
3. We take initial guesses of x_0, x_1 and x_2 = 0, 1 and 4, respectively.
4. We can evaluate the function at all three values as follows:

$$\begin{aligned} x_0 &= 0 \qquad & f(x_0) &= 0 \\ x_1 &= 1 \qquad & f(x_1) &= 1.5829 \\ x_2 &= 4 \qquad & f(x_2) &= -3.1136 \end{aligned}$$

5. Knowing these values we can now deploy an expression to calculate the next estimate of the maximum point x_3 as follows:

$$x_3 = \frac{f(x_0)\left(x_1^2 - x_2^2\right) + f(x_1)\left(x_2^2 - x_0^2\right) + f(x_2)\left(x_0^2 - x_1^2\right)}{2\,f(x_0)(x_1 - x_2) + 2\,f(x_1)(x_2 - x_0) + 2\,f(x_2)(x_0 - x_1)}$$

6. Populating this we get:

$$x_3 = \frac{0(1^2 - 4^2) + 1.5829(4^2 - 0^2) + (-3.1136)(0^2 - 1^2)}{2(0)(1-4) + 2(1.5829)(4-0) + 2(-3.1136)(0-1)} = 1.5055$$

Parabolic Interpolation: Example

6. The function at this point has a value of:

$$f(1.5055) = 1.7691.$$

7. Next we compare this value of the function at the new trial point with the three existing function values in 4 (left).
8. If f(1.5055) is greater than any of the function values in 4, we replace the closest of the old values (i.e., here f(x1)) with f(1.5055)
9. Then we repeat the calculation using Eqn. 1 opposite to evaluate the next point.

$$x_3 = \frac{1.5829(1.5055^2 - 4^2) + 1.7691(4^2 - 1^2) + (-3.1136)(1^2 - 1.5055^2)}{2(1.5829)(1.5055 - 4) + 2(1.7691)(4 - 1) + 2(-3.1136)(1 - 1.5055)}$$
$$= 1.4903$$

10. f(1.4903) = 1.7714. so evidence of convergence begins to appear

i	x_0	$f(x_0)$	x_1	$f(x_1)$	x_2	$f(x_2)$	x_3	$f(x_3)$
1	0.0000	0.0000	1.0000	1.5829	4.0000	−3.1136	1.5055	1.7691
2	1.0000	1.5829	1.5055	1.7691	4.0000	−3.1136	1.4903	1.7714
3	1.0000	1.5829	1.4903	1.7714	1.5055	1.7691	1.4256	1.7757
4	1.0000	1.5829	1.4256	1.7757	1.4903	1.7714	1.4266	1.7757
5	1.4256	1.7757	1.4266	1.7757	1.4903	1.7714	1.4275	1.7757

Parabolic Interpolation: Example

1. The table shows the progress of the algorithm towards common values of 1.7757 for each of $f(x_0)$, $f(x_1)$ and $f(x_3)$, with $f(x_2)$ deviating only slightly from this value.
2. The fact that **x_3 does not change through the last three iterations of the algorithm** means that **the routine has converged on a maximum function point.**

i	x_0	$f(x_0)$	x_1	$f(x_1)$	x_2	$f(x_2)$	x_3	$f(x_3)$
1	0.0000	0.0000	1.0000	1.5829	4.0000	−3.1136	1.5055	1.7691
2	1.0000	1.5829	1.5055	1.7691	4.0000	−3.1136	1.4903	1.7714
3	1.0000	1.5829	1.4903	1.7714	1.5055	1.7691	1.4256	1.7757
4	1.0000	1.5829	1.4256	1.7757	1.4903	1.7714	1.4266	1.7757
5	1.4256	1.7757	1.4266	1.7757	1.4903	1.7714	1.4275	1.7757

3. A similar technique could be used to estimate the minimum of a function, where it exists.
4. As will all numerical techniques, you need to graph the function to understand its shape and properties prior to selecting a routine.

Newton's Method (Optimisation)

5. Yet again, we meet Newton's method in this course, but this time in the context of optimisation.
6. Remember that for finding a root approximation, the following form of Newton's equation was used

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

7. For an optimisation problem the first insight is that **the optimum value (i.e. the maximum or minimum value of f(x) is the root of the gradient function f'(x)**
8. Therefore optimisation of f(x) is simply the finding of the root of f'(x), for which we rewrite the equation above as follows:

$$x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}$$

9. We know how to solve this equation because we've done it before…

THE UNIVERSITY of EDINBURGH
School of Engineering

Newton's Method (Optimisation)

$$x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}$$

1. Take the following example: (Ex. 13.3, Chapra)

$$f(x) = 2\sin x - \frac{x^2}{10}$$

2. Find the maximum of this function using Newton's Method: take an initial guess, x_0 = 2.5
3. Firstly, we need to evaluate the first and second derivatives of the function as follows:

$$f'(x) = 2\cos x - \frac{x}{5}$$

$$f''(x) = -2\sin x - \frac{1}{5}$$

4. Substituting these into Newton's Optimisation Algorithm we get:

$$x_{i+1} = x_i - \frac{2\cos x_i - x_i/5}{-2\sin x_i - 1/5}$$

THE UNIVERSITY of EDINBURGH
School of Engineering

Newton's Method (Optimisation)

5. For x_0 = 2.5 we get the following:

$$x_1 = 2.5 - \frac{2\cos 2.5 - 2.5/5}{-2\sin 2.5 - 1/5} = 0.99508$$

6. When f(x) is evaluated we get 1.57859.

7. A second iteration using x_1 gives:

$$x_1 = 0.995 - \frac{2\cos 0.995 - 0.995/5}{-2\sin 0.995 - 1/5} = 1.46901$$

8. F(x) for this value gives 1.77385. Because this value is significantly different from the last, it is clear that convergence has not been reached so we continue the method.

9. The following table summarises the steps taken and the data obtained for the rest of the technique:

i	x	$f(x)$	$f'(x)$	$f''(x)$
0	2.5	0.57194	−2.10229	−1.39694
1	0.99508	1.57859	0.88985	−1.87761
2	1.46901	1.77385	−0.09058	−2.18965
3	1.42764	1.77573	−0.00020	−2.17954
4	1.42755	1.77573	0.00000	−2.17952

THE UNIVERSITY *of* EDINBURGH
School of Engineering

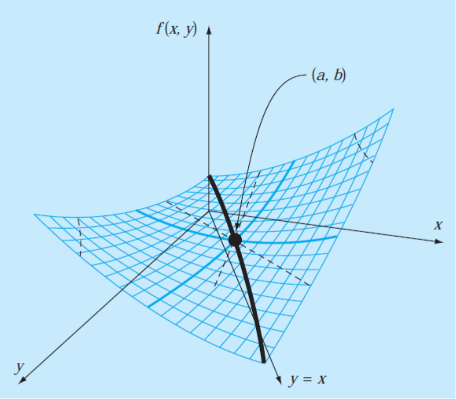
Newton's Method (Optimisation)

i	x	$f(x)$	$f'(x)$	$f''(x)$
0	2.5	0.57194	−2.10229	−1.39694
1	0.99508	1.57859	0.88985	−1.87761
2	1.46901	1.77385	−0.09058	−2.18965
3	1.42764	1.77573	−0.00020	−2.17954
4	1.42755	1.77573	0.00000	−2.17952

1. We can see that as the routine progresses, f(x) converges on a final value of 1.77573 (the last two iterations), and f'(x) is practically zero.
2. Note that the value of x at steps 3 and 4 changes slightly at the fourth decimal, although f(x) is static at the fifth decimal for both steps.
3. This shows why it is important that the algorithm progresses to f'(x) = 0 to sufficient decimal accuracy.
4. Generally, Newton's technique is accurate and efficient for cases where the derivatives can be evaluated easily.
5. However, if the initial guess is not sufficiently close, the technique can quickly diverge.

Newton’s Method (Two-Variable Optimisation)

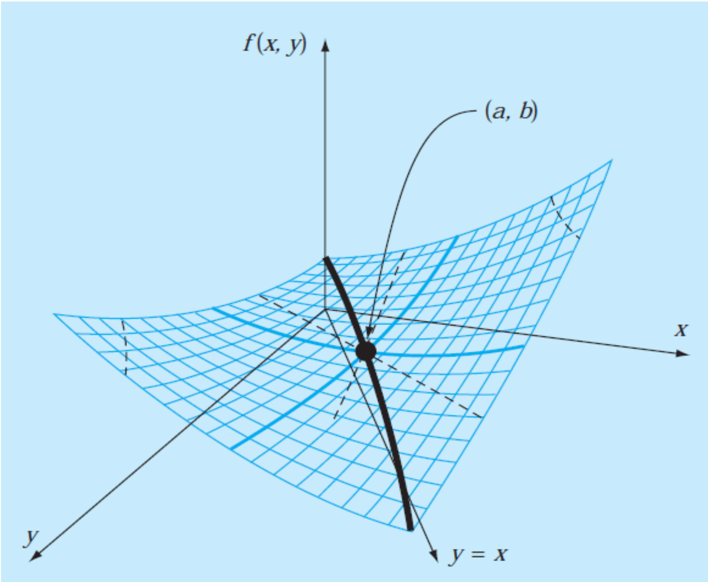
1. Newton’s Method is also used to perform optimisation of two variables on the same principle that underpins the use of Newton’s Method for one dimension.
2. Firstly, remember that for a one–dimensional optimisation problem, the first derivative of a function shows how steeply it is varying with the change of an independent variable, and whether a local maximum or minimum exists (when f’(x) = 0).



1. The second derivative shows whether the local stationary point of a function is a maximum (f’’(x) < 0) or minimum (f’’(x) > 0).
2. However, for a two-variable optimisation (i.e. a 3D surface function opposite), a further condition is needed to establish whether a maximum or minimum exists.
3. This is the second derivative of a function with respect to both x and y, the two independents.

THE UNIVERSITY *of* EDINBURGH
School of Engineering

Newton's Method (Two-Variable Optimisation)



6. The absolute value of the ***Hessian*** is a test point for either a maximum or minimum with the following test criteria.

$$|H| = \frac{\partial^2 f}{\partial x^2}\frac{\partial^2 f}{\partial y^2} - \left(\frac{\partial^2 f}{\partial x \partial y}\right)^2$$

If $|H| > 0$ and $\partial^2 f/\partial x^2 > 0$, then $f(x, y)$ has a local minimum.
If $|H| > 0$ and $\partial^2 f/\partial x^2 < 0$, then $f(x, y)$ has a local maximum.
If $|H| < 0$, then $f(x, y)$ has a saddle point.

THE UNIVERSITY *of* EDINBURGH
School of Engineering

Newton's Method (Two-Variable Optimisation)

1. The core equation of Newton's algorithm for optimising two dimensions as follows:

$$x_{i+1} = x_i - J.H^{-1} \qquad \textbf{Eqn A.}$$

2. Here J is the Jacobian Matrix which contains all the first derivatives of m functions with respect to n variables:

$$J = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial x_1} & \cdots & \frac{\partial F_m}{\partial x_n} \end{bmatrix}.$$

3. However, **for the case where only one function is being evaluated in multiple variables or dimensions, the Jacobian, J, reduces to just the first row of J above**.

6. The Hessian Matrix is one which contains all the coefficients involved in the Hessian expression, i.e.,

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

7. It is clear from Eqn. A that **the Newton technique will only calculate a meaningful new estimate if the term H^{-1} is non-zero.**

Newton's Method (Two-Variable Optimisation)

1. The conditions for maxima and minima are assessed **by taking the determinant of H** and testing its value. If it is zero no solution exists.
2. To calculate the second derivatives in MATLAB code you can use the ***diff*** function or you can use the expressions opposite to do the same.
3. The implementation of the Hessian matrix with a modified two-variable optimisation technique is possible to programme in Matlab using these principles.

$$\frac{\partial f}{\partial x} = \frac{f(x+\delta x, y) - f(x-\delta x, y)}{2\delta x}$$

$$\frac{\partial f}{\partial y} = \frac{f(x, y+\delta y) - f(x, y-\delta y)}{2\delta y}$$

$$\frac{\partial^2 f}{\partial x^2} = \frac{f(x+\delta x, y) - 2f(x, y) + f(x-\delta x, y)}{\delta x^2}$$

$$\frac{\partial^2 f}{\partial y^2} = \frac{f(x, y+\delta y) - 2f(x, y) + f(x, y-\delta y)}{\delta y^2}$$

THE UNIVERSITY of EDINBURGH School of Engineering

Newton's Method (Two-Variable Optimisation)

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{f(x+\delta x, y+\delta y) - f(x+\delta x, y-\delta y) - f(x-\delta x, y+\delta y) + f(x-\delta x, y-\delta y)}{4\delta x \delta y}$$

4. The equations above are examples of finite differences used by discrete programming processes to calculate derivatives.
5. The implementation of Newton's technique using the Jacobian and Hessian matrices is given in the next slide.
6. The determinant of the ***Hessian*** is a test point for either a maximum or minimum with the following test criteria.

$$|H| = \frac{\partial^2 f}{\partial x^2}\frac{\partial^2 f}{\partial y^2} - \left(\frac{\partial^2 f}{\partial x \partial y}\right)^2$$

If $|H| > 0$ and $\partial^2 f/\partial x^2 > 0$, then $f(x, y)$ has a local minimum.
If $|H| > 0$ and $\partial^2 f/\partial x^2 < 0$, then $f(x, y)$ has a local maximum.
If $|H| < 0$, then $f(x, y)$ has a saddle point.

Newton's Method (Two-Variable Optimisation)

```
% Newton's Method (Without Pre-Conditioner)
% Written by Soumitra Sitole
% Date: Mar 4, 2017
clc
clear
format long
% Function Definition (Enter your Function here):
syms X Y;
%f = X - Y + 2*X^2 + 2*X*Y + Y^2;
f=X^2+12*X*Y+Y^2+6*X
% Initial Guess (Choose Initial Guesses):
x(1) = 1;
y(1) = -5;
e = 10^(-8); % Convergence Criteria
i = 1; % Iteration Counter
% Gradient and Hessian Computation:
df_dx = diff(f, X);
df_dy = diff(f, Y);
J = [subs(df_dx,[X,Y], [x(1),y(1)]) subs(df_dy, [X,Y], [x(1),y(1)])]; % Gradient
ddf_ddx = diff(df_dx,X);
ddf_ddy = diff(df_dy,Y);
ddf_dxdy = diff(df_dx,Y);
ddf_ddx_1 = subs(ddf_ddx, [X,Y], [x(1),y(1)]);
ddf_ddy_1 = subs(ddf_ddy, [X,Y], [x(1),y(1)]);
ddf_dxdy_1 = subs(ddf_dxdy, [X,Y], [x(1),y(1)]);
H = [ddf_ddx_1, ddf_dxdy_1; ddf_dxdy_1, ddf_ddy_1]; % Hessian
S = inv(H); % Search Direction
end
```

Newton’s Method (Two-Variable Optimisation)

```
% Optimization Condition:
while norm(J) > e
    I = [x(i),y(i)]';
    x(i+1) = I(1)-S(1,:)*J';
    y(i+1) = I(2)-S(2,:)*J';
    i = i+1;
    J = [subs(df_dx,[X,Y], [x(i),y(i)]) subs(df_dy, [X,Y], [x(i),y(i)])]; % Updated Jacobian
    ddf_ddx_1 = subs(ddf_ddx, [X,Y], [x(i),y(i)]);
    ddf_ddy_1 = subs(ddf_ddy, [X,Y], [x(i),y(i)]);
    ddf_dxdy_1 = subs(ddf_dxdy, [X,Y], [x(i),y(i)]);
    H = [ddf_ddx_1, ddf_dxdy_1; ddf_dxdy_1, ddf_ddy_1]; % Updated Hessian
    S = inv(H); % New Search Direction
End
% Result Table:`
Iter = 1:i;
X_coordinate = x';
Y_coordinate = y';
Iterations = Iter';
T = table(Iterations,X_coordinate,Y_coordinate);
% Plots:
fcontour(f, 'Fill', 'On');
hold on;
plot(x,y,'*-r');
grid on;
% Output:
fprintf('Initial Objective Function Value: %d\n\n',subs(f,[X,Y], [x(1),y(1)]));
if (norm(J) < e)
    fprintf('Minimum succesfully obtained...\n\n');
end
fprintf('Number of Iterations for Convergence: %d\n\n', i);
fprintf('Point of Minima: [%d,%d]\n\n', x(i), y(i));
fprintf('Objective Function Minimum Value after Optimization: %f\n\n', subs(f,[X,Y], [x(i),y(i)]));
disp(T)
```