1.1 Golden Section Search for the Mode of a Function

This project is not strongly related to any particular course.

1 Definition

A function f is *unimodal* on $[a, b]$ with a maximum at $c \in [a, b]$ if f is strictly increasing on $[a, c]$ and strictly decreasing on $[c, b]$. A function f is unimodal on $[a, b]$ with a minimum at c if and only if $-f$ is unimodal with a maximum at c.

2 The golden search method

Let f be unimodal with a maximum and let x, y be points in $[a, b]$ with $a < x < y < b$. We wish to locate the mode (i.e. the unique maximum) of f while minimising the number of evaluations of f that are required; this will be useful in cases where it is very expensive or time-consuming to evaluate f. If we evaluate f at x and at y we can make use of the following implications:

(i) $f(x) \geqslant f(y) \quad \Rightarrow \quad$ mode lies in $[a, y]$

(ii) $f(x) \leqslant f(y) \quad \Rightarrow \quad$ mode lies in $[x, b]$

to narrow the region of search for the mode to a shorter interval within which we already have one evaluation of f. A second evaluation within the subinterval enables us to repeat this process. We terminate the algorithm when the size of the interval is smaller than twice the required precision.

A convenient way to do this, with an efficiency not much less than that of the theoretical optimal method, is to divide $[a, b]$ at points x, y such that

$$(y - a)/(b - a) = (\sqrt{5} - 1)/2 \tag{1}$$

and

$$b - x = y - a\,; \tag{2}$$

that is, the interval is divided from each end in "golden section".

> **Question 1** Prove that the subinterval in which the mode is deduced to lie is found to be already divided in golden section from one end by the point in its interior at which we already have a function evaluation.

3 Programming

> **Programming Task:** Write a program to implement the golden section search (to locate modes that are either maxima or minima). Your program should prompt the user to enter the interval's boundaries and the required precision. Ensure also that your program does not evaluate f more often than is necessary. In your write-up explain the following points.
>
> (i) How does your program deal with the possibility that $f(x) = f(y)$ on one or more iterative steps?

(ii) Is it preferable to use equation (1) or equation (2) to locate the point for the second function evaluation in each new subinterval, and why?

(iii) How would your program function if the mode's position were at an end-point of the original interval?

Question 2 As a check that you understand the method, first program it to find the position of the mode in $[0, 1]$ of the function

$$f(x) = 1 + x + x^2 - 4x^4$$

to some appropriate accuracy. Your output should include the mode, the number of iterations performed and an indication of how accurate your result is.

3.1 Computational cost

Consider the alternative (and more intuitively obvious) algorithm in which, instead of using (1) and (2), the subdivisions are defined by $x - a = \frac{1}{3}(b - a)$, $y - a = \frac{2}{3}(b - a)$.

Question 3 What is likely to be the most time-consuming part of either algorithm in a real-life problem. How would the number of numerical operations required for this alternative algorithm compare with that required for the golden section search algorithm. Give quantitative estimates if possible.

[Note that no additional computational work should be done to answer this question.]

4 Theoretical considerations

Question 4 What properties of the function $f(x)$ determine the numerical accuracy that is attainable?

Question 5 If the mode was located to some accuracy, what would be the corresponding accuracy in the height of the mode? How does your answer depend on the properties of $f(x)$?

5 Application: Optimization of turntable and tone-arm design

With the resurgence in sales of *vinyl records*, attention has returned to optimizing the design of record players and, in particular, the positioning of the turntable and tone-arm.

The playing region of a vinyl gramophone record is an annulus of inner radius r and outer radius R. The record spins about its centre S. Sound is picked up from grooves in the record by a *stylus*, which is mounted at the end P of a *tone-arm* QP of length l which is fixed at Q (a point beyond the outer perimeter of the record). The grooves run (almost) in concentric circles around S. The sound pick-up is optimal if the stylus points in the same direction as the groove at P. See Figure 1, a plan view.

The designer of the tone-arm has the following parameters under his control:

(i) the distance d of the tone-arm pivot Q from the centre S of the record;

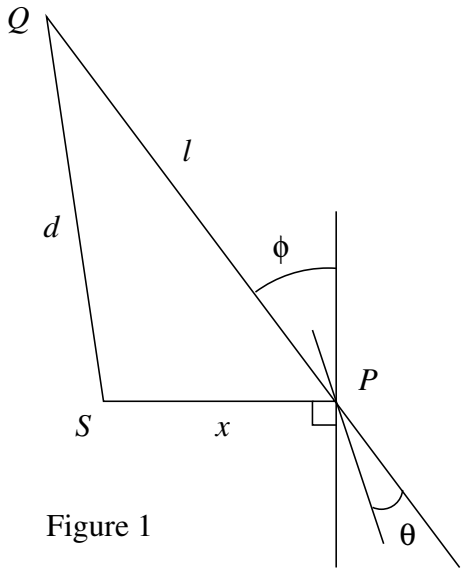

Figure 1

(ii) the toe-in angle θ between the tone-arm and the direction of the stylus.

The designer wishes the stylus to track so that its direction diverges as little as possible from alignment with the groove; that is, so that as the stylus moves from the outer edge of the record to the inner one, the maximum absolute value of $(\phi - \theta)$ is minimised, where ϕ is the angle between QP and the tangent to the groove at P.

Question 6 Taking $r = 6.5$, $R = 16$, $l = 24$ (all in cm), find the optimum values of d and θ using golden section search. Your program will need to carry out a double iteration. The inner iteration should find the variation of ϕ considered as a function of x (the distance SP); in other words, find $\Delta\phi = \max \phi(x) - \min \phi(x)$. The outer iteration then adjusts d to minimise $\Delta\phi$, and finally the optimum choice of θ can easily be made.

Your write-up should address the following points:

(i) What initial intervals did you use for the inner and outer iterations, and why?

(ii) Is it clear that the functions you were minimising/maximising are unimodal on the relevant intervals?

(iii) Approximately how many inner function evaluations are required to find d to one decimal place? How is the accuracy of your result affected by the accuracy with which you perform the inner and outer iterations?

Note that no extra marks are available for solving this part of the project analytically, though you may find that this is possible. No proofs are required at any stage: qualitative arguments supported by appropriate graphs will suffice.

1.1 Golden Section Search for the Mode of a Function

The Golden search methods depends on the following implications for a unimodal function with a maximum:

(i) $f(x) \geq f(y) \Rightarrow$ mode lies in $[a, y]$

(ii) $f(x) \leq f(y) \Rightarrow$ mode lies in $[x, b]$

Question 1

We are given:

$$\frac{y-a}{b-a} = \frac{\sqrt{5}-1}{2} \tag{1}$$

$$b - x = y - a \tag{2}$$

We will rearrange (1):

$$\begin{aligned} y &= a + (b-a)\frac{\sqrt{5}-1}{2} \\ &= \left(1 - \frac{\sqrt{5}-1}{2}\right)a + \left(\frac{\sqrt{5}-1}{2}\right)b \end{aligned}$$

Similarly:

$$\begin{aligned} x &= a + b - y \\ &= a + b - \left(1 - \frac{\sqrt{5}-1}{2}\right)a - \left(\frac{\sqrt{5}-1}{2}\right)b \\ &= \left(\frac{\sqrt{5}-1}{2}\right)a + \left(1 - \frac{\sqrt{5}-1}{2}\right)b \end{aligned}$$

Therefore:

$$y = \left(\frac{3-\sqrt{5}}{2}\right)a + \left(\frac{\sqrt{5}-1}{2}\right)b \tag{3}$$

$$x = \left(\frac{\sqrt{5}-1}{2}\right)a + \left(\frac{3-\sqrt{5}}{2}\right)b \tag{4}$$

Let us now assume $f(x) \geq f(y)$. Then by statement (i) we know that the mode lies in $[a, y]$. We now look for an x_2 and y_2 which divides the interval $[a, y]$ into golden sections.

By (3) and (4):

$$\begin{aligned} y_2 &= \left(\frac{3-\sqrt{5}}{2}\right)a + \left(\frac{\sqrt{5}-1}{2}\right)y \\ &= \left(\frac{3-\sqrt{5}}{2}\right)a + \left(\frac{\sqrt{5}-1}{2}\right)\left[\left(\frac{3-\sqrt{5}}{2}\right)a + \left(\frac{\sqrt{5}-1}{2}\right)b\right] \\ &= \left(\frac{3-\sqrt{5}}{2}\right)a + \left(\frac{4\sqrt{5}-8}{4}\right)a + \left(\frac{6-2\sqrt{5}}{4}\right)a \\ &= \left(\frac{\sqrt{5}-1}{2}\right)a + \left(\frac{3-\sqrt{5}}{2}\right)b \\ &= x \end{aligned} \tag{5}$$

Observe we have already evaluated the function at x.

Now let us now assume $f(x) \leq f(y)$. Then by statement (ii) we know that the mode lies in $[x, b]$. We now look for an x_2 and y_2 which divides the interval $[x, b]$ into golden sections.

By (4):

$$\begin{aligned} x_2 &= \left(\frac{\sqrt{5}-1}{2}\right)x + \left(\frac{3-\sqrt{5}}{2}\right)b \\ &= \left(\frac{\sqrt{5}-1}{2}\right)\left[\left(\frac{\sqrt{5}-1}{2}\right)a + \left(\frac{3-\sqrt{5}}{2}\right)b\right] + \left(\frac{3-\sqrt{5}}{2}\right)b \\ &= \left(\frac{3-\sqrt{5}}{2}\right)a + \left(\frac{\sqrt{5}-1}{2}\right)b \\ &= y \end{aligned} \tag{6}$$

Again we can observe that we have already evaluated the function at y.

So we have found the subinterval in which the mode is deduced to lie is found to be already divided in golden section from one end by a point in its interior at which we already have a function evaluation.

Programming Task 1

The script for this program is on pages 10-11.

(i) Let c be the mode. If $f(x) = f(y)$ occurs then both (i) and (ii) are true. We can deduce that the mode is in the subinterval $[x, y]$. There are two options how the program can proceed:

1. Apply only one of the two statements (WLOG statement (i)) and proceed as normal, i.e. the deduced subinterval is $[a, y]$.

   We will label the interior points of the subinterval x_2 and y_2. From equation (5) $y_2 = x$ and from equation (2) $x_2 = a + y - x$. By the definition of a unimodal, f is strictly increasing on $[a, c]$ and $a \leq x \leq c$ (because $c \in [x, y]$) so f is strictly increasing on $[a, x]$. Since $y_2 = x$ and $x_2 < y_2$ we can deduce $f(x_2) < f(y_2) = f(x)$ so by statement (ii) the mode lies in $[x_2, y]$.

   We will label the interior points of the new subinterval x_3 and y_3. From equation (6) we see that $x_3 = y_2 = x$ and equation (2) gives us $y_3 = a + 2(y - x)$. From here we will have one of the two following casesIf $f(x_3) \geq f(y_3)$ then our new interval is $[x_2, y_3]$ and the range of the interval is $y_3 - x_2 = a + 2(y - x) - (a + y - x) = y - x$. If $f(x_3) \leq f(y_3)$ then our new interval is $[x_3, y]$ and the range of the interval is $y - x_3 = y - x$.

2. Apply both statement (i) and statement (ii), i.e. the deduced subinterval is $[x, y]$ so the range of the deduced subinterval is already $y - x$. To proceed we will need to evaluate the function at two new point but of course this will cause the range of the interval to become smaller than $y - x$ (to satisfy curiosity we will find the new range to be $\frac{\sqrt{5}-1}{2}(y - x)$ via this method).

We see that the second method will have us find a smaller interval which the mode can be deduced to lie in with the same number of function evaluations. For this reason I choose to treat $f(x) = f(y)$ as a special case and evaluate my functions at two new internal points to continue the algorithm.

(ii) It is preferable to use equation (2) to locate the point for the second function evaluation. This is because MATLAB approximates $\sqrt{5}$. If we were to use equation (1) the error in evaluating $\sqrt{5}$ will lead to the each iteration increasing the error of around the location of our point.

(iii) If a functions mode is at $x = a$ the program we will always find $f(x) \geq f(y)$ as f is unimodal so must be strictly decreasing on $[a, b]$. So the program will always have implication (i) occur. Hence we will find an interval of the form $[a, a+\zeta]$ where ζ is less than twice the precision. A similar thing will occur if the mode is at $x = b$.

Question 2:

First some observations. The equation for $f(x)$ given is :

$$f(x) = 1 + x + x^2 - 4x^4 \tag{7}$$

Then:

$$\begin{aligned} f'(x) &= 1 + 2x - 16x^3 \\ &= (1 - 2x)(1 + 4x + 8x^2) \end{aligned} \tag{8}$$

Observe $1 + 4x + 8x^2$ is positive for $x \in [0, 1]$ so f'(x) is positive for $x \in [0, 1]$ if and only if $1 - 2x$ is positive, i.e. when $0 < x < \frac{1}{2}$. Also $f'(x)$ is negative whenever $\frac{1}{2} < x < 1$.

So $f(x)$ is strictly increasing on the interval $[0, 1/2]$ and strictly decreasing on the interval $[1/2, 1]$. So f is a unimodal function on $[0, 1]$ with maximum $\frac{1}{2}$.

We now test the algorithm with `e=0.0001`:

```
Input lower bound a = 0
Input upper bound b = 1
Input precision e = 0.0001
itteration        a                 b                 f(a)                f(b)
1                           0   1.000000000000000   1.000000000000000  -1.000000000000000
2                           0   0.618033988749895   1.000000000000000   1.416407864998738
3           0.236067977499790   0.618033988749895   1.279373587440064   1.416407864998738
4           0.381966011250105   0.618033988749895   1.442719099991588   1.416407864998738
5           0.381966011250105   0.527864045000420   1.442719099991588   1.495942493490441
6           0.437694101250946   0.527864045000420   1.482464578894537   1.495942493490441
7           0.472135954999580   0.527864045000420   1.496288634033866   1.495942493490441
8           0.472135954999580   0.506577808748213   1.496288634033866   1.499781377825735
9           0.485291572496006   0.506577808748213   1.498943579509001   1.499781377825735
10          0.493422191251787   0.506577808748213   1.499785931518325   1.499781377825735
11          0.493422191251787   0.501552810007567   1.499785931518325   1.499987913928827
12          0.496527811266921   0.501552810007567   1.499940053833899   1.499987913928827
13          0.498447189992433   0.501552810007567   1.499987973835465   1.499987913928827
14          0.498447189992433   0.500366568717945   1.499987973835465   1.499999327742749
15          0.499180327428324   0.500366568717945   1.499996645088233   1.499999327742749
16          0.499633431282055   0.500366568717945   1.499999328530857   1.499999327742749
17          0.499633431282055   0.500086535135786   1.499999328530857   1.499999962553167
```

```
18          0.499806501553627   0.500086535135786   1.499999812849710   1.499999962553167
19          0.499913464864214   0.500086535135786   1.499999962563535   1.499999962553167

itteration              x                   y                  f(x)                f(y)
1           0.381966011250105   0.618033988749895   1.442719099991588   1.416407864998738
2           0.236067977499790   0.381966011250105   1.279373587440064   1.442719099991588
3           0.381966011250105   0.472135954999580   1.442719099991588   1.496288634033866
4           0.472135954999580   0.527864045000420   1.496288634033866   1.495942493490441
5           0.437694101250946   0.472135954999580   1.482464578894537   1.496288634033866
6           0.472135954999580   0.493422191251787   1.496288634033866   1.499785931518325
7           0.493422191251787   0.506577808748213   1.499785931518325   1.499781377825735
8           0.485291572496006   0.493422191251787   1.498943579509001   1.499785931518325
9           0.493422191251787   0.498447189992433   1.499785931518325   1.499987973835465
10          0.498447189992433   0.501552810007567   1.499987973835465   1.499987913928827
11          0.496527811266921   0.498447189992433   1.499940053833899   1.499987973835465
12          0.498447189992433   0.499633431282055   1.499987973835465   1.499999328530857
13          0.499633431282055   0.500366568717945   1.499999328530857   1.499999327742749
14          0.499180327428324   0.499633431282055   1.499996645088233   1.499999328530857
15          0.499633431282055   0.499913464864214   1.499999328530857   1.499999962563535
16          0.499913464864214   0.500086535135786   1.499999962563535   1.499999962553167
17          0.499806501553627   0.499913464864214   1.499999812849710   1.499999962563535
18          0.499913464864214   0.499979571825198   1.499999962563535   1.499999997913517
19          0.499979571825198   0.500020428174802   1.499999997913517   1.499999997913380


interval =

0.499913464864214   0.500086535135786


maximum =

0.500000000000000
```

It appears as if the algorithm works! The output includes the mode and number of iterations preformed and we see that the answer given to us agrees with our analytic solution so the result is very accurate!

Question 3

In the real world the most time consuming task is likely the number of function evaluations. Using the alternative method suggested in the project manual will lead to roughly twice the number of function evaluations as each iteration requires two new function evaluations whereas the golden search only requires one.

Question 4

The property of $f(x)$ which determines the numerical accuracy that is attained is how smooth the function is.

Question 5

The accuracy of the mode depends greatly on how smooth the function is. Particularly on how smooth the function is at the mode. One can imagine a unimodal function with a maximum at c for which we have found via the golden search to be in the interval $[c-3\epsilon, c+\epsilon]$, i.e. this interval is less than twice the

required precision. This means the method is telling us the mode is at $x = c - \epsilon$. However if the function isn't very smooth around the mode we could see that f rapidly increases in $(c-\epsilon, c)$ and decreases rapidly in $(c, c+\epsilon)$. For such a function the accuracy of the height of the mode will not be very good.

If we assume now that the function is able to be differentiated twice and we have found the mode to be at $c+h$ (h is not the precision and c is the mode found analytically), we may use Taylor's series to write the the value f the function evaluated at this point as $f(c+h) = f(c) + h^2 \dfrac{f''(c)}{2} + o(h^2)$ (as we are at a turning point). Therefore the accuracy of the height of a mode for a twice differentiable function is dependent on the second derivative.

Question 6

(i) Before creating an algorithm I need to first find the function I am trying to minimise and interval I should use. To aid me I have included a sketch (fingure 1.1) on page 6.

First we will apply cosine rule on triangle SPQ:

$$\begin{aligned} d^2 &= l^2 + x^2 - 2lx\cos\phi^c \\ &= l^2 + x^2 - 2lx\sin\phi \end{aligned}$$

Therefore:

$$\phi(x,d) = \arcsin\left(\frac{d^2 - l^2 - x^2}{48x}\right) \tag{9}$$

Now we have a function we need to find the interval of d and x.The interval of x is simply $r \leq x \leq R$ as the point P must move from the outer to the inner radius so as to cover the whole playing region. As for our range in d we must ensure that the path of P covers the whole playing area. This means that the circle on which is the path of p (red in diagram) must intersect with both the inner and outer radius. We can see (on figure 1.1) it is impossible or a circle centred at a point outside the playing region to cross the inner boundary without crossing the outer boundary. So we look for intersections of the circles:

$$t^2 + y^2 = r^2 \tag{10}$$

$$(d-t)^2 + y^2 = l^2 \tag{11}$$

So we have $d^2 - 2td + r^2 = l^2$ if equations intersect, hence:

$$t = \frac{d^2 + r^2 - l^2}{2d} \tag{12}$$

Putting this back into equation (9):

$$\begin{aligned} &\left(\frac{d^2 + r^2 - l^2}{2d}\right)^2 + y^2 = r^2 \\ &y^2 = r^2 - \left(\frac{d^2 + r^2 - l^2}{2d}\right)^2 \geq 0 \\ &4d^2r^2 \geq d^4 + r^4 + l^4 - 2d^2l^2 - 2l^2r^2 + 2d^2r^2 \end{aligned}$$

So:

$$0 \geq \big(d - (l+r)\big)\big(d + (l+r)\big)\big(d - (l-r)\big)\big(d + (l-r)\big) \tag{13}$$

Now as in question $l = 24$, $R = 16$ and $r = 6.5$. We are taking $d > 0$ so we obtain our (for the outer iteration) range to be:

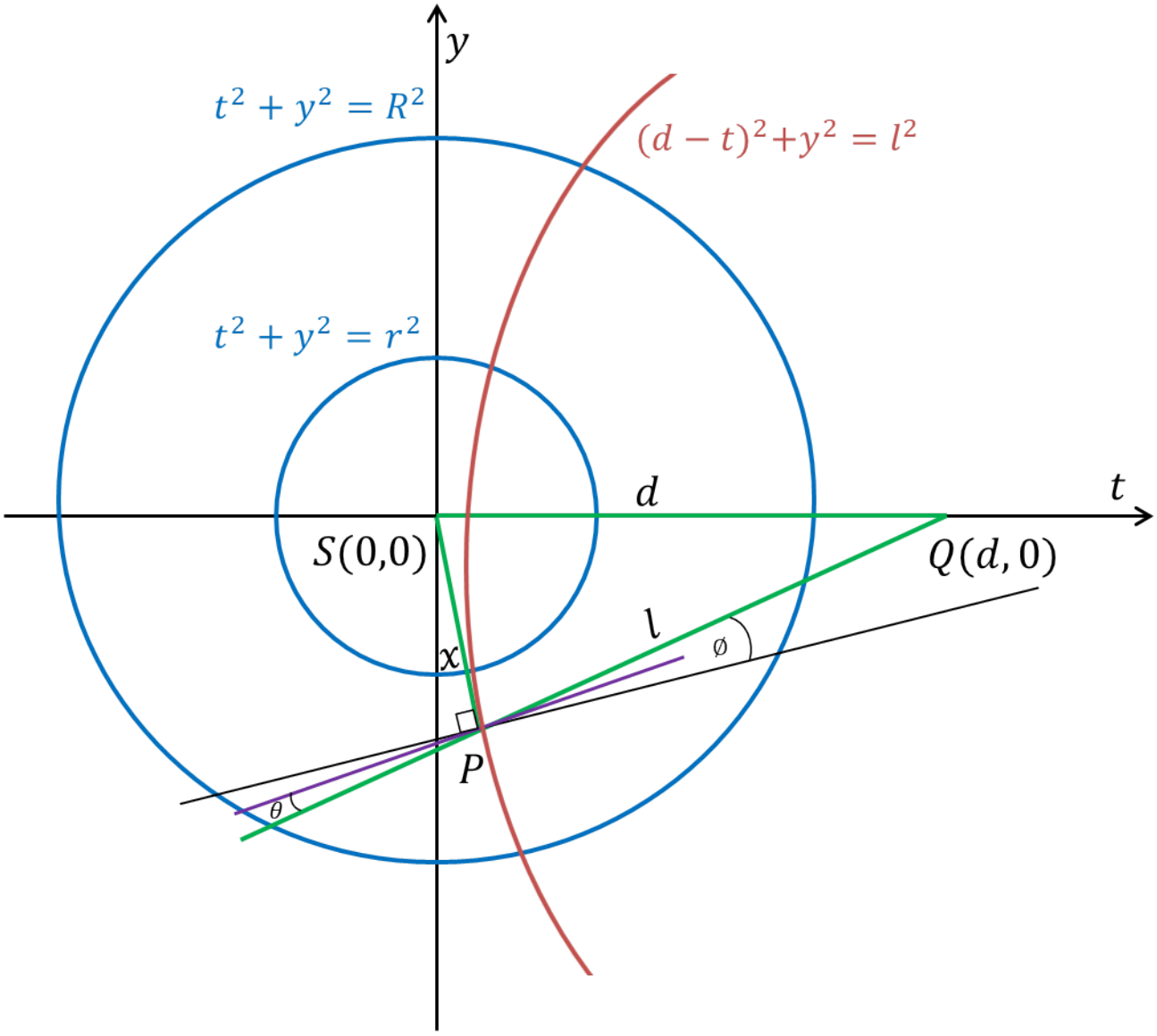

Figure 1.1: Blue circles represent the boundaries of the playing region, the red curve shows the path of P (extended beyond the region it will need to move though), green triangle is SPQ and the purple line is the stylus. Created in Microsoft Powerpoint 2010.

$$17.5 \leq d \leq 30.5 \tag{14}$$

As stated previously $r \leq x \leq R$ so (the interval for the inner range is):

$$6.5 \leq x \leq 16 \tag{15}$$

(ii) We must check now $\phi(x, d)$ is a unimodal function in the given interval.

First at $d = 24$ we see that $\phi(x, 24) = \arcsin(-\frac{x}{48})$ which is strictly monotonic decreasing function. So $\phi(x, 24)$ is unimodal with a maximum and unimodal with a minimum.

now for $d \neq 24$ we have:

$$\phi(x, d) = \arcsin\left(\frac{d^2 - 24^2}{48x} - \frac{x}{48}\right) \tag{16}$$

Hence:

$$\left.\frac{\partial \phi}{\partial x}\right|_d = \frac{24^2 - d^2 - x^2}{48x^2\sqrt{1 - \left(\frac{d^2 - 24^2}{48x} - \frac{x}{48}\right)^2}} \tag{17}$$

For the given value x may take the denominator of (17) is always positive. Therefore the gradientis postive if and only if $24^2 - d^2 - x^2 \geq 0$.

Since $x \geq 6.5$ we can deduce that $\frac{d\phi}{dx} \leq 0$ for all x whenever $d \geq \sqrt{24^2 - 6.5^2}$ or for $d \geq \frac{\sqrt{2135}}{2} \approx 23.1$ so for such d, we find ϕ is strictly monotonic decreasing. So the maximum is at a and the minimum at b, i.e. $\Delta\phi = \phi(6.5, d) - \phi(16, d)$.

For values of $d < \frac{\sqrt{2135}}{2}$ the gradient is positive for small x and is negative or large x so we have a unimodal function with a maximum. For a unimodal function on an interval $[a, b]$ with a maximum c observe that the functions minimum must be either at a or at b since the function is strictly increasing on $[a, c]$ and strictly decreasing on $[c, b]$. We may then compare endpoints to find the minimum. We will do this, first through a plot generated on MATLAB to see that the whole function is unimodal and secondly in our program.

Figure 1.2 shows $\Delta\phi$ against d. For $d \geq \frac{\sqrt{2135}}{2}$ we see one line which is found simply the $\phi(6.5, d) - \phi(16, d)$. For $d > \frac{\sqrt{2135}}{2}$ know that there is a turning point at the maximum. By (17) that turning point is at $x_t = \sqrt{24^2 - d^2}$. The minimum may be at $x = 6.5$ or $x = 16$. As we don't know which it is I have plot $\phi(x_t, d) - \phi(6.5, d)$ and $\phi(x_t, d) - \phi(16, d)$. Of course the curve we are concerned with the the parth further from the x-axis.

The graph shows us we have a unimodal function.

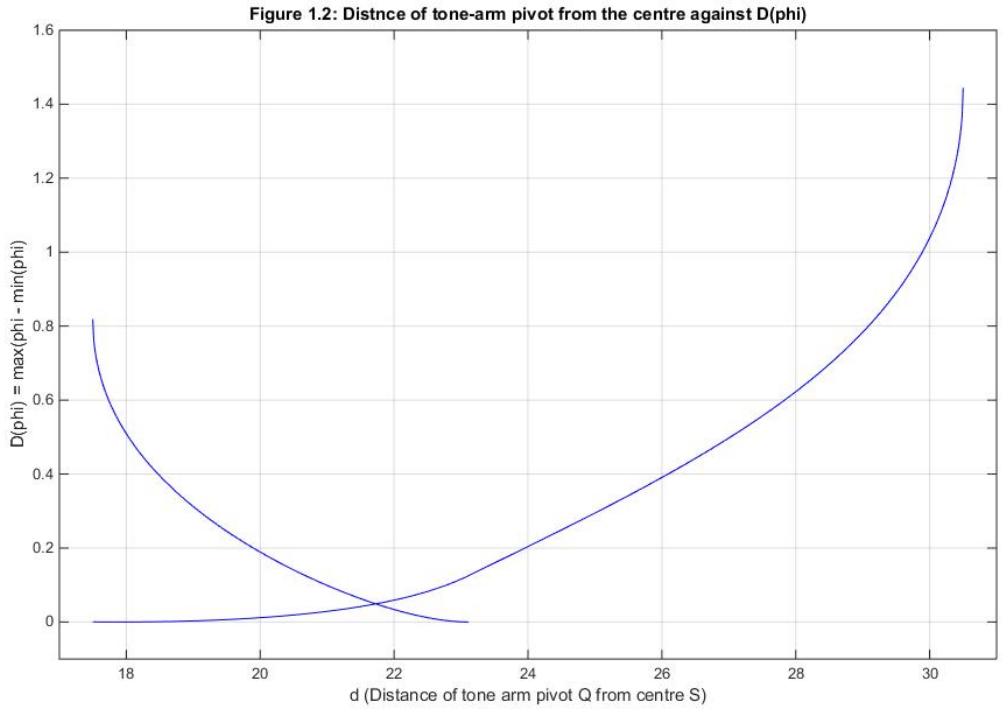

Figure 1.2: Horizontal Axis labelled "d" and vertical axis "$\Delta\phi$" takes is path further from the x-axis. Plot created on MATLAB using program see page 12 for the code used to generate it.

(iii) Now let us first count the number of inner processes needed to find d to one decimal place. This means the process terminates when $d_{max} - d_{min} < 0.2$. So we look for the solution to $(30.5 - 17.5)\left(\frac{sqrt(5)-1}{2}\right)^n < 0.2$. The first solution is $n = 9$. Next we need to pick the error we allow in the inner process so that we still have d accurate to one decimal place. This means that the inner error must be less than $\frac{1}{180}$, so say it is $1/200$. By similar reasoning the process will terminate after 16 iterations. The first iteration requires an extra function evaulation in both cases. This means we require 160 inner function evaluations!

Now I will finally use my program to find the optimal distance and choice of θ (code can be found on pages 13-19):

```
Input precision e_d = 0.1
Input precision e_r = 5*10^(-3)
itteration                d_min                d_max                d_x
1.000000000000000   17.500000000000000   30.500000000000000   22.465558146251368
2.000000000000000   22.465558146251368   30.500000000000000   25.534441853748632
3.000000000000000   25.534441853748632   30.500000000000000   27.431116292502736
4.000000000000000   27.431116292502736   30.500000000000000   28.603325561245896
5.000000000000000   28.603325561245896   30.500000000000000   29.327790731256840
6.000000000000000   29.327790731256840   30.500000000000000   29.775534829989056
7.000000000000000   29.775534829989056   30.500000000000000   30.052255901267785
8.000000000000000   30.052255901267785   30.500000000000000   30.223278928721271
9.000000000000000   30.223278928721271   30.500000000000000   30.328976972546513
10.000000000000000   30.328976972546513   30.500000000000000   30.394301956174758

itteration                d_y                g(d,x)                g(d,y)
1.000000000000000   25.534441853748632   8.673906191556574   6.395324728755531
2.000000000000000   27.431116292502736   8.673906191556574   6.059304217826806
3.000000000000000   28.603325561245896   8.673906191556574   5.808850642554686
```

```
4.000000000000000  29.327790731256840   8.673906191556574   5.617510976568427
5.000000000000000  29.775534829989056   8.673906191556574   5.469599483224715
6.000000000000000  30.052255901267785   8.673906191556574   5.354569687326388
7.000000000000000  30.223278928721271   8.673906191556574   5.264854456236592
8.000000000000000  30.328976972546513   8.673906191556574   5.194826383748490
9.000000000000000  30.394301956174758   8.673906191556574   5.140214977963780
10.000000000000000  30.434675016371756   8.673906191556574   5.097742024390837


interval =

30.328976972546513  30.500000000000000


d =

30.414488486273257
```

So from the program we find that $d = 30.4$ and $\theta = 1.8$ (both to ne decimal place).

Appendix: Programming Task 1

This is labelled `Programming_Task_1` in folders.

```
clc;

clear;

format long;
syms t X Y;

%Inputs
a = input('Input lower bound a = ');
b = input('Input upper bound b = ');
e = input('Input precision e = ');

f = @(t) 1 + t + t^2 -4*t^4;
%f = @(t) (t - 0.5*(sqrt(5)-1))*(0.5*(3-sqrt(5))-t) + 1;
%f = @(t) 1-t;
%f = @(t) -4 + 5*t - t^2

iter = 1;
%table = [];
%Initial process
if b - a - 2*e > 0;
f_a = f(a);
f_b = f(b);
y = a + (sqrt(5) - 1)*(b-a)/2;
x = a + b - y;
f_x = f(x);
f_y = f(y);
%table=[iter a b f_a f_b x y f_x f_y];
table1= [iter a b f_a f_b];
table2= [iter x y f_x f_y];
end

%itteration
%v= diff - 2*e
while b - a - 2*e > 0

iter = iter + 1;

if f_x > f_y;
% a = a;
b = y;
f_b = f_y;
y = x;
f_y = f_x;
x = a + b - y ;
%x = b - (sqrt(5) - 1)*(b-a)/2;
f_x = f(x);
%display('1logic');

elseif f_y< f_x;
a = x;
f_a = f_x;
% b = b;
x = y;
```

```
f_x = f_y;
y = a + b - x;
%y = a + (sqrt(5) - 1)*(b-a)/2;
f_y = f(y);
%display('2logic');

else
a = x;
f_a = f_x;
b = y;
f_b = f_y;
y = a + (sqrt(5) - 1)*(b-a)/2;
x = a + b - y;
f_x = f(x);
f_y = f(y);
end
%display('logic3')
%table=[table; iter a b f_a f_b x y f_x f_y];
table1= [table1; iter a b f_a f_b];
table2= [table2; iter x y f_x f_y];
%v= b -a  - 2*e

end

%disp('       itteration                a                b
f(a)               f(b)                   x                y
                f(x)               f(y)');
%disp(table);

disp('       itteration               a                b
f(a)               f(b)')
disp(table1);

disp('       itteration               x                y
 f(x)               f(y)')
disp(table2);

interval =[a b];
display(interval);

maximum = 0.5*(a+b);
display(maximum);
```

Figure 1.2

This is labelled `Figure` in folders and used to generate figure1.2.

```
clc;

clear;

format long;
syms d l x b a d1 d2;


fplot(@(x) asin((1/(48*sqrt((24)^2 - (x)^2)))*2*((x)^2-(24)^2)) -
asin((1/(48*16))*((x)^2-(24)^2-(16)^2)),[17.5 0.5*sqrt(2135)],'b')
hold on
title('Figure 1.2: Distnce of tone-arm pivot from the centre against D(phi)')
xlabel('d (Distance of tone arm pivot Q from centre S)')
ylabel('D(phi) = max(phi - min(phi)')
axis([17 31 -0.1 1.6])
fplot(@(x) asin((1/(48*6.5))*((x)^2-(24)^2-(6.5)^2)) - asin((1/(48*16))*((x)^2-(24)^2-(16)^2)),
[0.5*sqrt(2135) 30.5],'b')
fplot(@(x) asin((1/(48*sqrt((24)^2 - (x)^2)))*2*((x)^2-(24)^2)) - asin((1/(48*6.5))*
((x)^2-(24)^2-(6.5)^2)),[17.5 0.5*sqrt(2135)],'b')
hold off
grid on

axis([17 31 -0.1 1.6])
```

Question 6

This is labelled `Question_6` in folders.

```
clc;

clear;

format long;
syms d l x b a;

f = @(d,l,x) asin(((d^2)-(l^2)-(x^2))/(48*x));
g = @(b,a) b - a;

%Inputs
x_min=6.5;
x_max=16;
l=24;
e_d = input('Input precision e_d = ');
e_x = input('Input precision e_r = ');

d_min = 17.5;
d_max = 30.5;

iter = 1;

%Initial process (outer)
if d_max - d_min - 2*e_d > 0;

d_y = d_min + (sqrt(5) - 1)*(d_max-d_min)/2;
d_x = d_min + d_max - d_y;

%At d_x:
%Initial Process (inner) at d_x:
x_min=6.5;
x_max=16;
if x_max - x_min - 2*e_x > 0;
x_y = x_min + (sqrt(5) - 1)*(x_max-x_min)/2;
x_x = x_min + x_max - x_y;
f_dx_xx = f(d_x,l,x_x);
f_dx_xy = f(d_x,l,x_y);
end

% itteration at d_x to find max:
x_min=6.5;
x_max=16;
while x_max - x_min - 2*e_x > 0


if f_dx_xx > f_dx_xy;
x_max = x_y;
x_y = x_x;
f_dx_xy = f_dx_xx;
x_x = x_min + x_max - x_y ;
f_dx_xx = f(d_x,l,x_x);

elseif f_dx_xy< f_dx_xx;
x_min = x_x;
```

```
x_x = x_y;
f_dx_xx = f_dx_xy;
x_y = x_min + x_max - x_x;
f_dx_xy = f(d_x,l,x_y);

else
x_min = x_x;
x_max = x_y;
x_y = x_min + (sqrt(5) - 1)*(x_max-x_min)/2;
x_x = x_min + x_max - x_y;
f_dx_xx = f(d_x,l,x_x);
f_dx_xy = f(d_x,l,x_y);
end
end

dx_max = 0.5*(x_x+x_y);

%check endpoints to find min at d_x:
f_dx_xmin = f(d_x,l,x_x);
f_dx_xmax = f(d_x,l,x_y);

if f_dx_xmin < f_dx_xmax;
dx_min = f_dx_xmin;
else
dx_min = f_dx_xmax;
end

g_dx = g(dx_max,dx_min);

%At d_y:
x_min=6.5;
x_max=16;
%Initial Process (inner) at d_y:
if x_max - x_min - 2*e_x > 0;
x_y = x_min + (sqrt(5) - 1)*(x_max-x_min)/2;
x_x = x_min + x_max - x_y;
f_dy_xx = f(d_y,l,x_x);
f_dy_xy = f(d_y,l,x_y);
end

% itteration at d_y to find max:
while x_max - x_min - 2*e_x > 0

if f_dy_xx > f_dy_xy;
x_max = x_y;
x_y = x_x;
f_dy_xy = f_dy_xx;
x_x = x_min + x_max - x_y ;
f_dy_xx = f(d_y,l,x_x);

elseif f_dy_xy< f_dy_xx;
x_min = x_x;
x_x = x_y;
f_dy_xx = f_dy_xy;
x_y = x_min + x_max - x_x;
f_dy_xy = f(d_y,l,x_y);
```

```
else
x_min = x_x;
x_max = x_y;
x_y = x_min + (sqrt(5) - 1)*(x_max-x_min)/2;
x_x = x_min + x_max - x_y;
f_dy_xx = f(d_y,l,x_x);
f_dy_xy = f(d_y,l,x_y);
end

end

dy_max = 0.5*(x_x+x_y);

%check endpoints to find min at d_y:
f_dy_xmin = f(d_y,l,x_x);
f_dy_xmax = f(d_y,l,x_y);

if f_dy_xmin < f_dy_xmax;
dy_min = f_dy_xmin;
else
dy_min = f_dy_xmax;
end

g_dy = g(dy_max,dy_min);

table1 = [iter d_min d_max d_x];
table2= [iter d_y g_dx g_dy];

end

%itteration outer

while d_max - d_min - 2*e_d > 0;

x_min = 6.5;
x_max = 16;
iter = iter + 1;

if g_dx < g_dy;
d_max = d_y;
d_y = d_x;
g_dy = g_dx;
d_x = d_min +d_max - d_y;

%inner process to fimd f_dx;
%At d_x:
%Initial Process (inner) at d_x:
if x_max - x_min - 2*e_x > 0;
x_y = x_min + (sqrt(5) - 1)*(x_max-x_min)/2;
x_x = x_min + x_max - x_y;
f_dx_xx = f(d_x,l,x_x);
f_dx_xy = f(d_x,l,x_y);
end

% itteration at d_x to find max:
while x_max - x_min - 2*e_x > 0
```

```
if f_dx_xx > f_dx_xy;
x_max = x_y;
x_y = x_x;
f_dx_xy = f_dx_xx;
x_x = x_min + x_max - x_y ;
f_dx_xx = f(d_x,l,x_x);

elseif f_dx_xy< f_dx_xx;
x_min = x_x;
x_x = x_y;
f_dx_xx = f_dx_xy;
x_y = x_min + x_max - x_x;
f_dx_xy = f(d_x,l,x_y);

else
x_min = x_x;
x_max = x_y;
x_y = x_min + (sqrt(5) - 1)*(x_max-x_min)/2;
x_x = x_min + x_max - x_y;
f_dx_xx = f(d_x,l,x_x);
f_dx_xy = f(d_x,l,x_y);
end
end

dx_max = 0.5*(x_x+x_y);

%check endpoints to find min at d_x:
f_dx_xmin = f(d_x,l,x_x);
f_dx_xmax = f(d_x,l,x_y);

if f_dx_xmin < f_dx_xmax;
dx_min = f_dx_xmin;
else
dx_min = f_dx_xmax;
end

g_dx = g(dx_max,dx_min);


elseif g_dx > g_dy;
d_min = d_x;
d_x = d_y;
d_y = d_min + d_max - d_y;

%inner process to find f_dy;
%dy
%Initial Process (inner) at d_y:
if x_max - x_min - 2*e_x > 0;
x_y = x_min + (sqrt(5) - 1)*(x_max-x_min)/2;
x_x = x_min + x_max - x_y;
f_dy_xx = f(d_y,l,x_x);
f_dy_xy = f(d_y,l,x_y);
end

% itteration at d_y to find max:
while x_max - x_min - 2*e_x > 0
```

```
if f_dy_xx > f_dy_xy;
x_max = x_y;
x_y = x_x;
f_dy_xy = f_dy_xx;
x_x = x_min + x_max - x_y ;
f_dy_xx = f(d_y,l,x_x);

elseif f_dy_xy< f_dy_xx;
x_min = x_x;
x_x = x_y;
f_dy_xx = f_dy_xy;
x_y = x_min + x_max - x_x;
f_dy_xy = f(d_y,l,x_y);

else
x_min = x_x;
x_max = x_y;
x_y = x_min + (sqrt(5) - 1)*(x_max-x_min)/2;
x_x = x_min + x_max - x_y;
f_dy_xx = f(d_y,l,x_x);
f_dy_xy = f(d_y,l,x_y);
end

end

dy_max = 0.5*(x_x+x_y);

%check endpoints to find min at d_y:
f_dy_xmin = f(d_y,l,x_x);
f_dy_xmax = f(d_y,l,x_y);

if f_dy_xmin < f_dy_xmax;
dy_min = f_dy_xmin;
else
dy_min = f_dy_xmax;
end

g_dy = g(dy_max,dy_min);

else
d_min = d_x;
d_max = d_y;
d_y = d_min + (sqrt(5) - 1)*(d_max-d_min)/2;
d_x = d_min + d_max - d_y;

%At d_x:
%Initial Process (inner) at d_x:


if x_max - x_min - 2*e_x > 0;
x_y = x_min + (sqrt(5) - 1)*(x_max-x_min)/2;
x_x = x_min + x_max - x_y;
f_dx_xx = f(d_x,l,x_x);
f_dx_xy = f(d_x,l,x_y);
%disp('done');
end
```

```
% itteration at d_x to find max:

while x_max - x_min - 2*e_x > 0

if f_dx_xx > f_dx_xy;
x_max = x_y;
x_y = x_x;
f_dx_xy = f_dx_xx;
x_x = x_min + x_max - x_y ;
f_dx_xx = f(d_x,l,x_x);

elseif f_dx_xy< f_dx_xx;
x_min = x_x;
x_x = x_y;
f_dx_xx = f_dx_xy;
x_y = x_min + x_max - x_x;
f_dx_xy = f(d_x,l,x_y);

else
x_min = x_x;
x_max = x_y;
x_y = x_min + (sqrt(5) - 1)*(x_max-x_min)/2;
x_x = x_min + x_max - x_y;
f_dx_xx = f(d_x,l,x_x);
f_dx_xy = f(d_x,l,x_y);
end
end

dx_max = 0.5*(x_x+x_y);

%check endpoints to find min at d_x:
f_dx_xmin = f(d_x,l,x_x);
f_dx_xmax = f(d_x,l,x_y);

if f_dx_xmin < f_dx_xmax;
dx_min = f_dx_xmin;
else
dx_min = f_dx_xmax;
end

g_dx = g(dx_max,dx_min);

%At d_y:
x_min=6.5;
x_max=16;
%Initial Process (inner) at d_y:
if x_max - x_min - 2*e_x > 0;
x_y = x_min + (sqrt(5) - 1)*(x_max-x_min)/2;
x_x = x_min + x_max - x_y;
f_dy_xx = f(d_y,l,x_x);
f_dy_xy = f(d_y,l,x_y);
end

% itteration at d_y to find max:
while x_max - x_min - 2*e_x > 0
```

```
if f_dy_xx > f_dy_xy;
x_max = x_y;
x_y = x_x;
f_dy_xy = f_dy_xx;
x_x = x_min + x_max - x_y ;
f_dy_xx = f(d_y,l,x_x);

elseif f_dy_xy< f_dy_xx;
x_min = x_x;
x_x = x_y;
f_dy_xx = f_dy_xy;
x_y = x_min + x_max - x_x;
f_dy_xy = f(d_y,l,x_y);

else
x_min = x_x;
x_max = x_y;
x_y = x_min + (sqrt(5) - 1)*(x_max-x_min)/2;
x_x = x_min + x_max - x_y;
f_dy_xx = f(d_y,l,x_x);
f_dy_xy = f(d_y,l,x_y);
end

end

dy_max = 0.5*(x_x+x_y);

%check endpoints to find min at d_y:
f_dy_xmin = f(d_y,l,x_x);
f_dy_xmax = f(d_y,l,x_y);

if f_dy_xmin < f_dy_xmax;
dy_min = f_dy_xmin;
else
dy_min = f_dy_xmax;
end

g_dy = g(dy_max,dy_min);

end

table1 = [table1; iter d_min d_max d_x];
table2= [table2; iter d_y g_dx g_dy];

end

disp('       itteration               d_min              d_max               d_x')
disp(table1);
disp('       itteration                d_y                 g(d,x)             g(d,y)')
disp(table2);

interval = [d_min d_max];
display(interval);

d = 0.5*(d_max +d_min);
display(d);
```