Root Finding and Optimization

Root finding, solving equations, and optimization are very closely related subjects, which occur often in practical applications.

$$\text{Root finding}: \quad f(x) = 0 \quad \text{Solve for } x$$

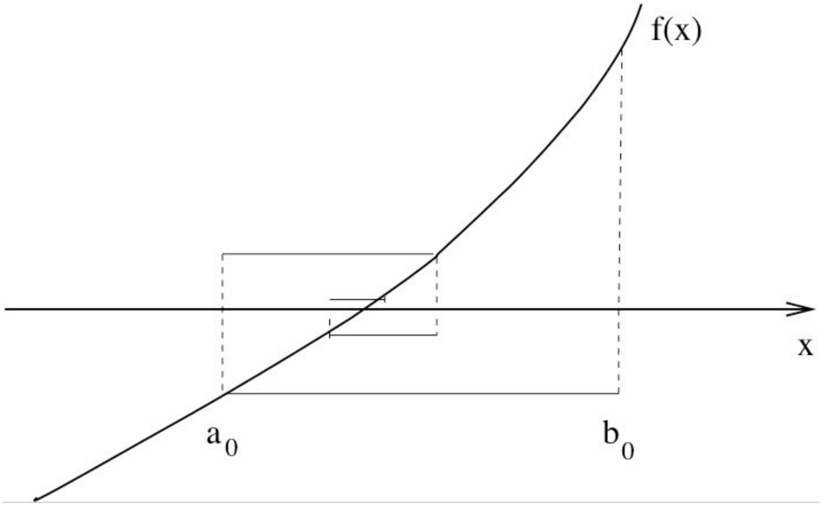
$$\text{Equation Solving}: f(x) = g(x), \quad \text{or } f(x) - g(x) = 0$$

$$\text{Optimization}: \quad \frac{dg(x)}{dx} = 0 \quad f(x) = \frac{dg(x)}{dx}$$

Start with one equation in one variable. Different methods have different strengths/weaknesses. However, all methods require a good starting value and some bounds on the possible values of the root(s).

Root Finding

Bisection



Need to find an interval where *f(x)* changes sign (implies a zero crossing). If no such interval, no root. Then, divide interval into

$$\left[a_0, a_0 + \frac{b_0 - a_0}{2}\right], \left[a_0 + \frac{b_0 - a_0}{2}, b_0\right]$$

Find interval where *f(x)* changes sign, and repeat until interval is small enough.

Root Finding

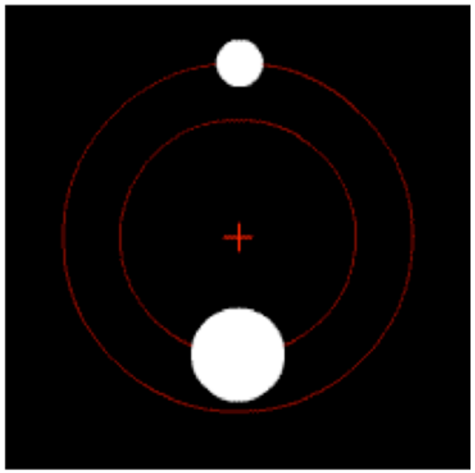
We will try this method on one of our old examples - planetary motion. Recall, for two masses m_1,m_2, we found:

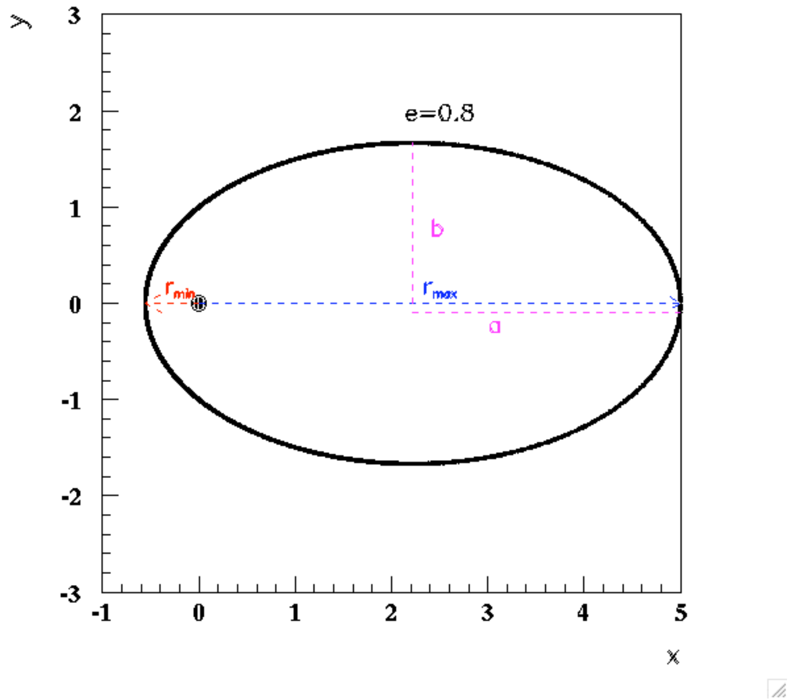
$$\frac{1}{r}=\left(\frac{\mu GMm}{L^2}\right)[1-e\cos(\theta+\theta_0)] \quad r=\frac{a(1-e^2)}{1-e\cos(\theta+\theta_0)} \quad \text{where} \quad \vec{r}=\vec{r}_1-\vec{r}_2$$

and $\quad \text{Reduced mass:} \quad \mu=\frac{m_1 m_2}{m_1+m_2}$

$$a=\frac{r_{\min}+r_{\max}}{2}=\left(\frac{L^2}{\mu GMm}\right)\left(\frac{1}{1-e^2}\right)$$

$$b=\left(\frac{L^2}{\mu GMm}\right)\left(\frac{1}{\sqrt{1-e^2}}\right)$$





2-body motion

Here, we are interested in plotting the orbit of individual masses as a function of time - i.e., taking equal time steps. The relationship between the angle and the time is:

$$t = \frac{T}{2\pi}(\xi - e\sin\xi)$$ where ξ is the angle from the center of the ellipse

The position of the individual masses is given by

$$x_1 = \frac{m_2}{m_1 + m_2} a(\cos\xi - e) \quad x_2 = -\frac{m_1}{m_1 + m_2} a(\cos\xi - e)$$

$$y_1 = \frac{m_2}{m_1 + m_2} a\sqrt{1 - e^2}\sin\xi \quad y_2 = -\frac{m_1}{m_1 + m_2} a\sqrt{1 - e^2}\sin\xi$$

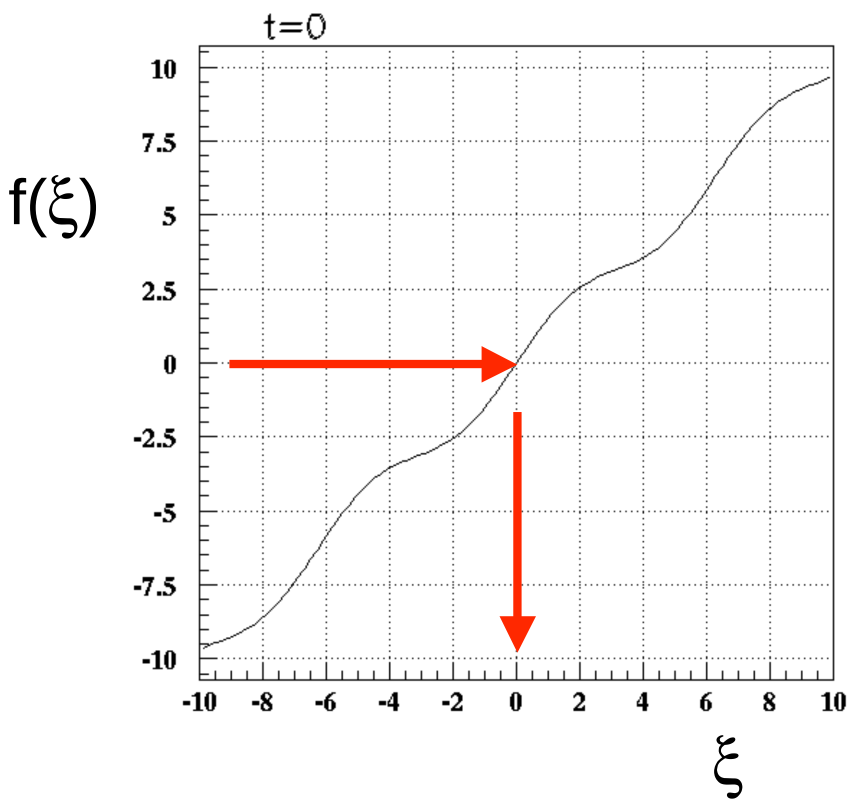
We first have to solve for $\xi(t)$, then for x,y. To solve for $\xi(t)$, need to find the root of

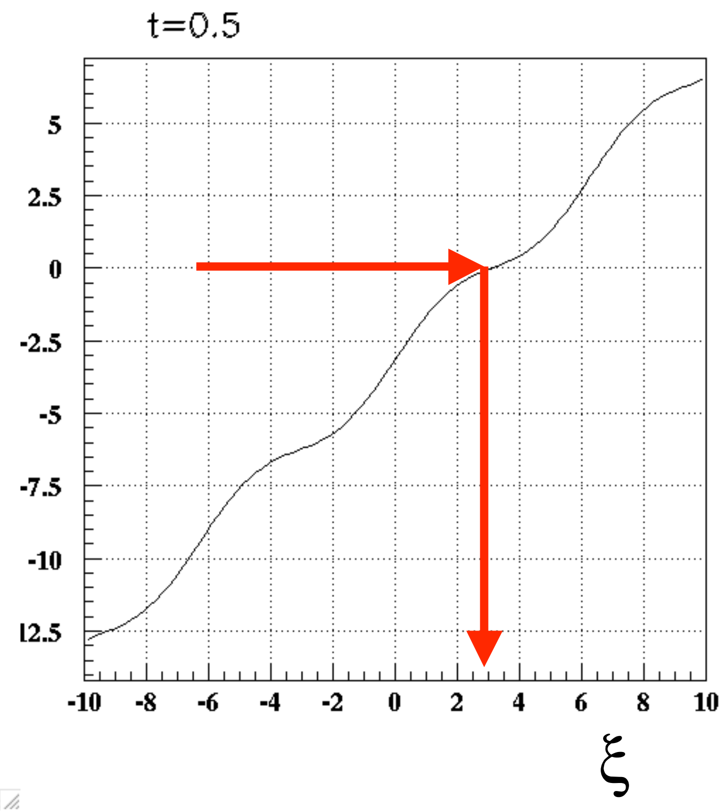
$$(\xi - e\sin\xi) - \frac{2\pi t}{T} = 0$$

for a given *t.*

2-body motion

$$(\xi - e\sin\xi) - \frac{2\pi t}{T} = 0$$





Need a starting interval for the bisection algorithm: We note that the maximum and minimum of the sin(ξ) term is 1, -1. So we can take as the starting range for ξ

$$\xi_a = \frac{2\pi t}{T} - e \qquad \xi_b = \frac{2\pi t}{T} + e$$

2-body motion

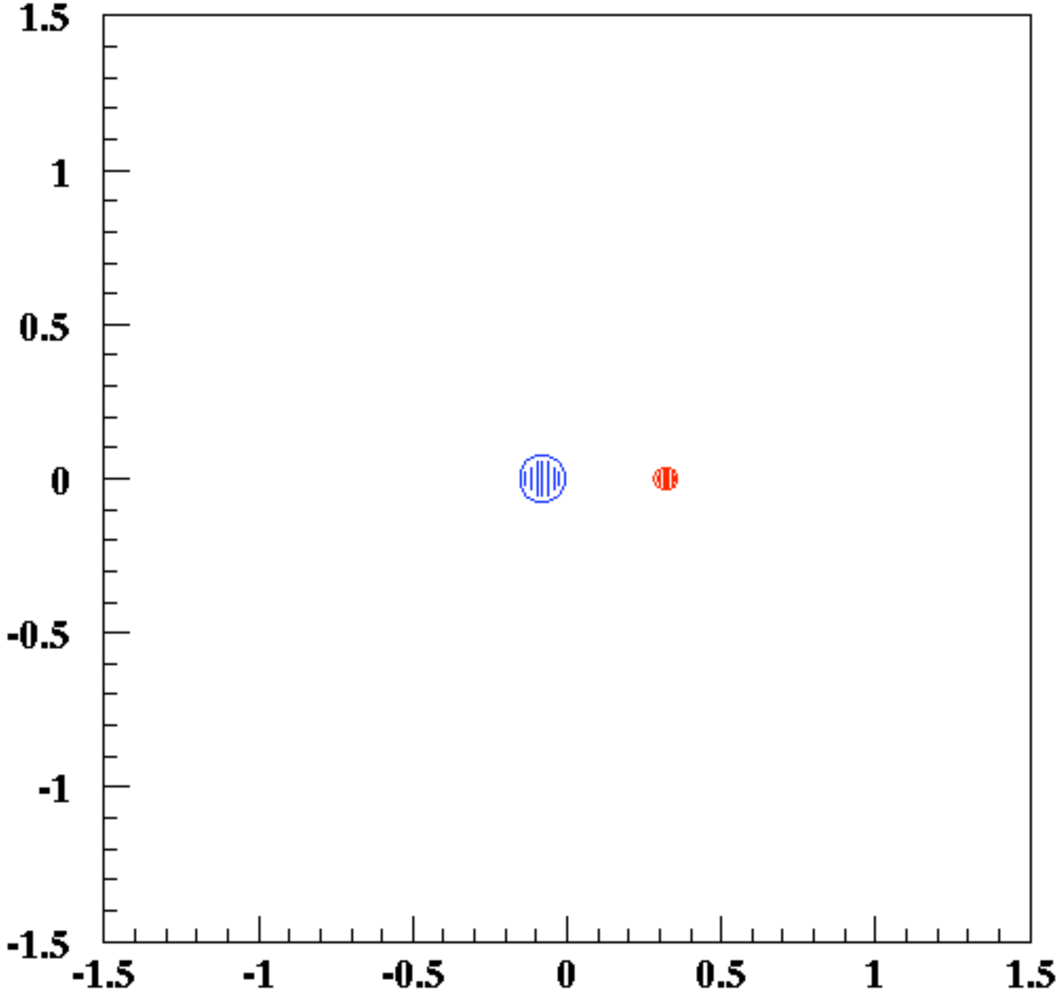
Let's take some random values for the parameters:
$T=1$, $a=1$, $e=0.6$, $m_2=4m_1$

tfunc= $(\xi - e\sin\xi) - \dfrac{2\pi t}{T}$

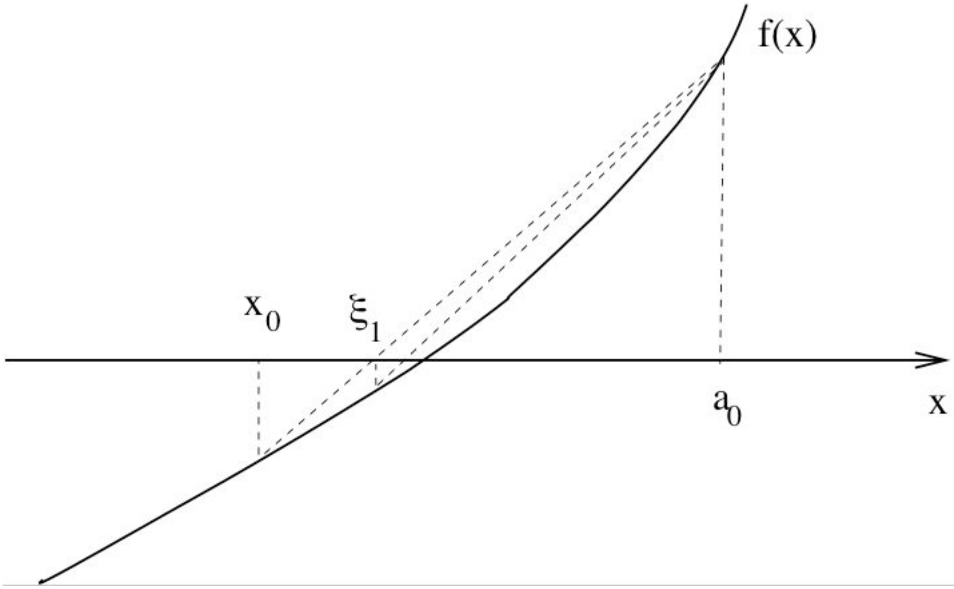
```
      accuracy=1.D-6
* Define range in which we search
      angle1=2*3.1415926*t-e
      angle2=angle1+2.*e
      try1=tfunc(e,period,t,angle1)
      try2=tfunc(e,period,t,angle2)
      If (try1*try2.gt.0) then
        print *,' Cannot find root - bad start parameters'
        return
      Endif
* Now update until within accuracy
 1    continue
      step=angle2-angle1
      angle2=angle1+step/2.
      try2=tfunc(e,period,t,angle2)
      If (try1*try2.lt.0.) goto 2   (root in this interval)
      If (try1.eq.0.) then
        angle=angle1
        return
      Elseif (try2.eq.0.) then         (check for exact landing)
        angle=angle2
        return
      Endif
      angle1=angle2
      try1=try2
      angle2=angle2+step/2.
      try2=tfunc(e,period,t,angle2)
      If (try1*try2.lt.0.) goto 2
      If (try1.eq.0.) then
        angle=angle1
        return
      Elseif (try2.eq.0.) then
        angle=angle2
        return
      Endif
 2    continue
      If ((angle2-angle1).gt.accuracy) goto 1
      angle=angle1+(angle2-angle1)/2.
```

Note: accuracy$\propto 2^{-(\#\ \text{iterations})}$.
For 10^{-6}, need 21 iterations

2-body motion



Regula Falsi



Similar to bisection, but use linear interpolation to speed up the convergence. Start with the interval $[x_0, a_0]$ with function values $f(x_0), f(a_0)$ such that $f(x_0)f(a_0) < 0$. Use linear interpolation to guess the zero crossing.

$$p(x) = f(x_0) + (x - x_0)\frac{f(a_0) - f(x_0)}{a_0 - x_0} \qquad p(x) = 0 \text{ gives } \xi_1 = \frac{a_0 f(x_0) - x_0 f(a_0)}{f(x_0) - f(a_0)}$$

Regula Falsi

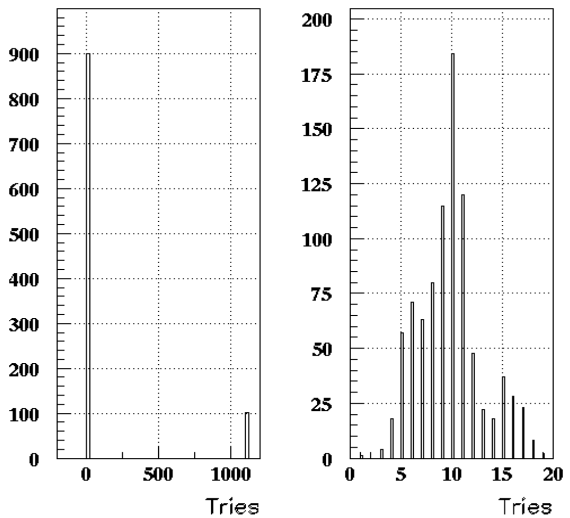
Now calculate $f(\xi_1)$

If $\quad f(x_0)f(\xi_1) < 0 \quad$ choose new interval $[x_0, \xi_1]$

else $\quad f(x_0)f(\xi_1) > 0 \quad$ choose new interval $[\xi_1, a_0]$

Iterate until interval sufficiently small

With our previous example (2-body motion), most of the time we are faster at converging that the bisection method, but we find that we sometimes need many iterations to reach the accuracy of 10^{-6}.



Problem occurs when either $f(x_0)$ or $f(a_0)$ very close to zero, then ξ_1 is very close to x_0 or a_0 and convergence is extremely slow (or not converging because of machine rounding)

$$p(x) = 0 \text{ gives } \xi_1 = \frac{a_0 f(x_0) - x_0 f(a_0)}{f(x_0) - f(a_0)}$$

Regula Falsi

So we add the extra condition that the interval has to shrink by at least the level of accuracy we are trying to reach. The logic is:

If $|a_0 - x_0| < accuracy$, Converged

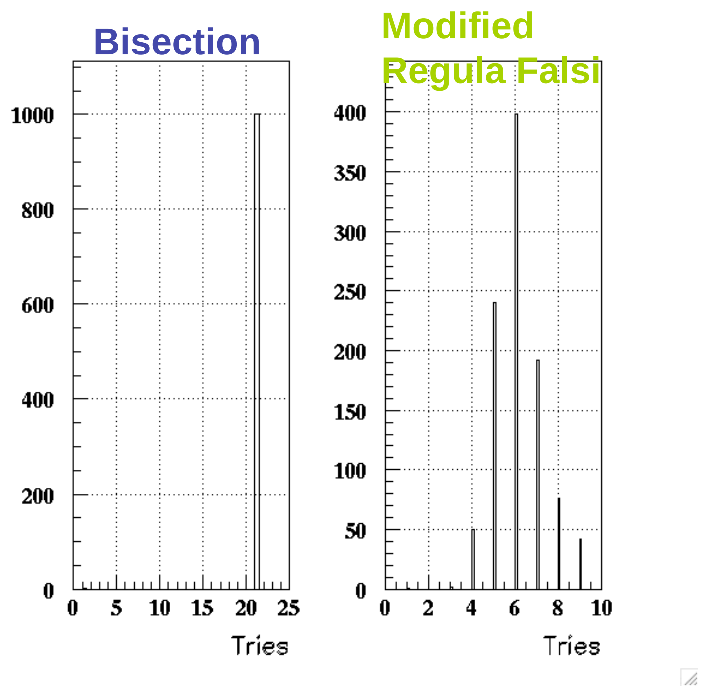
If $|\xi_1 - x_0| < accuracy/2$, $\xi_1 = x_0 + accuracy/2$

If $|\xi_1 - a_0| < accuracy/2$, $\xi_1 = a_0 - accuracy/2$

Then continue with the usual

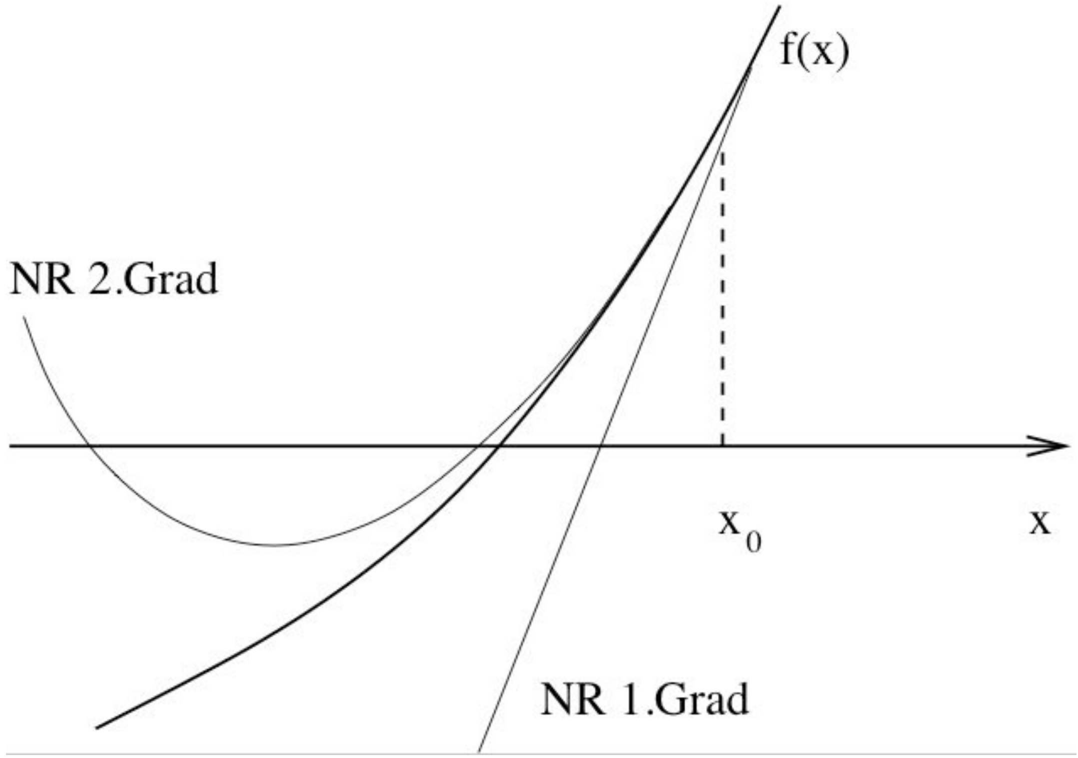
$f(x_0)f(\xi_1) < 0$ choose new interval $[x_0, \xi_1]$.

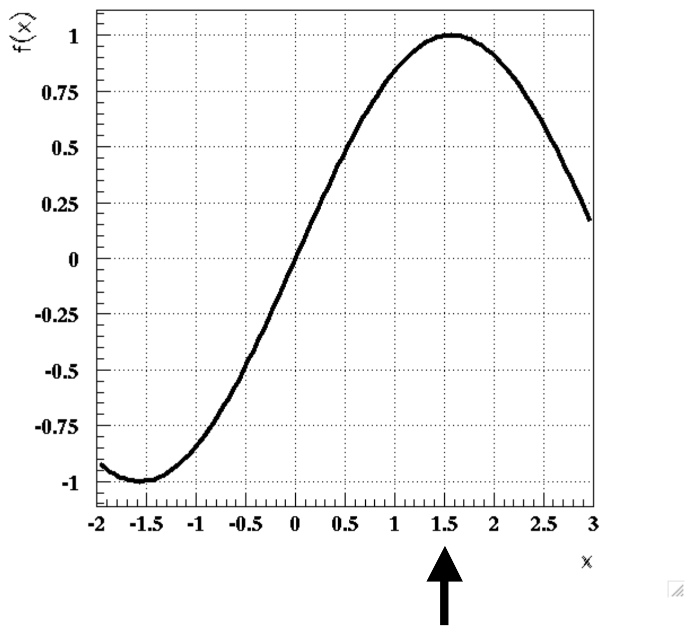
$f(x_0)f(\xi_1) > 0$ choose new interval $[\xi_1, a_0]$



Newton-Raphson Method

Here we use the slope (or also 2^{nd} derivative) at a guess position to extrapolate to the zero crossing. This method is the most powerful of the ones we consider today, since we can easily generalize to many parameters and many equations. However, it also has its drawbacks as we will see.





What if this is our guess ?

Newton-Raphson Method

The algorithm is: $$x_{i+1} = x_i - \frac{f(x)}{f'(x)}$$ 1^{st} order

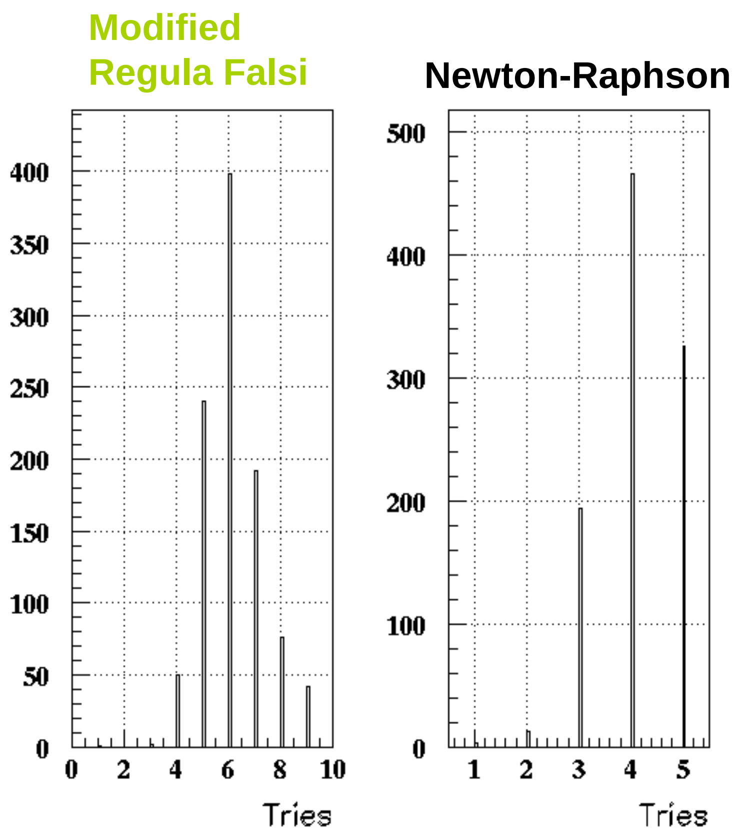
```
* make a starting guess for the angle
        angle=2.*3.1415926*t
        try=tfunc(e,period,t,angle)
* Now update until angular change within accuracy
        Do Itry=1,40
          slope=tfuncp(e,period,t,angle)
          angle1=angle-try/slope
          try1=tfunc(e,period,t,angle1)
          If (abs(angle1-angle).lt.accuracy) goto 1
          angle=angle1
          try=try1

        Enddo
1       continue
```

Analytic derivative needed for NR method

$$(\xi - e\sin\xi) - \frac{2\pi t}{T} = tfunc(e, period, t, angle)$$

$$(1 - e\cos\xi) = tfuncp(e, period, t, angle)$$

Newton-Raphson Method



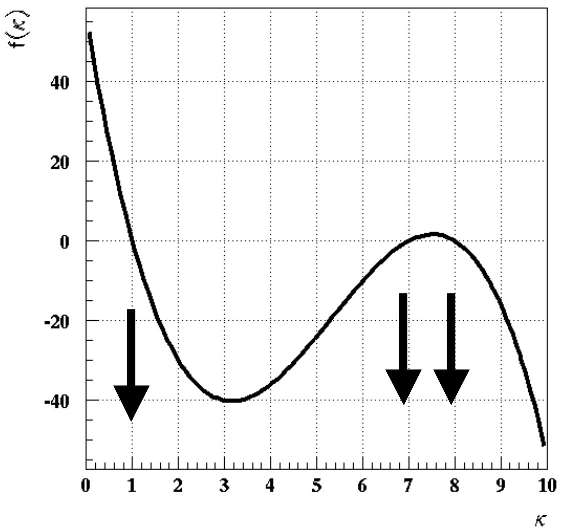
This is the fastest of the methods we have tried so far.

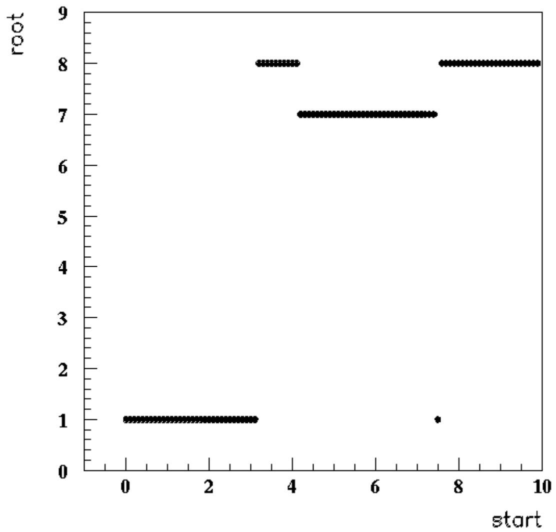
Note: if no analytic derivatives available, then calculate them numerically: **secant method**

Several Roots

Suppose we have a function which as several roots; e.g. the function we found a couple lectures ago when dealing with eigenvalues (principal axis problem):

$$0 = \left[-\kappa^3 + 8\kappa^2(a^2+b^2) - \kappa(16a^4 + 39a^2b^2 + 16b^4) + 28a^2b^2(a^2+b^2)\right]$$





It is important to analyze the problem and set the bounds correctly. For the NR method, if you are not sure where the root is, then need to give several different starting values and see what happens.

Several variable

The Newton-Raphson method is easily generalized to several functions of several variables:

$$f(x) = \begin{pmatrix} f_1(x_1, \cdots, x_n) \\ \vdots \\ f_n(x_1, \cdots, x_n) \end{pmatrix} = 0$$

We form the derivative matrix:

$$Df = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

If the matrix is not singular, we solve the SLE:

$$0 = f(\vec{x}^{(1)}) + Df(\vec{x}^{(0)})(\vec{x}^{(1)} - \vec{x}^{(0)})$$

The iteration is $\vec{x}^{(r+1)} = \vec{x}^{(r)} - \left(Df(\vec{x}^{(r)})\right)^{-1} f(\vec{x}^{(r)})$

Optimization

We now move to the closely related problem of optimization (finding the zeroes of a derivative). This is a very widespread problem in physics (e.g., finding the minimum χ^2, the maximum likelihood, the lowest energy state, …). Instead of looking for zeroes of a function, we look for extrema.

Finding global extrema is a very important and very difficult problem, particularly in the case of several variables. Many techniques have been invented, and we look at a few here. The most powerful techniques (using Monte Carlo methods) will be reserved for next semester.

Here, look for the minimum of $h(\vec{x})$. For a maximum, consider the minimum of -$h(\vec{x})$. We assume the function is at least twice differentiable.

Optimization

First and second derivatives:

$$\vec{g}^{\,t}(\vec{x}) = \left(\frac{\partial h}{\partial x_1}, \cdots, \frac{\partial h}{\partial x_n} \right) \qquad \text{a vector}$$

$$\mathrm{H}(\vec{x}) = \begin{pmatrix} \frac{\partial^2 h}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 h}{\partial x_1 \partial x_n} \\ & \ddots & \\ \frac{\partial^2 h}{\partial x_n \partial x_1} & & \frac{\partial^2 h}{\partial x_n \partial x_n} \end{pmatrix} \qquad \text{the Hessian matrix}$$

General technique:

1. Start with initial guess $\vec{x}^{(0)}$
2. Determine a direction, $\vec{s}$, and a step size λ
3. Iterate $\vec{x}$ until $\left|\vec{g}^{\,t}\right| < \varepsilon$, or cannot find smaller h $\qquad \vec{x}^{(r+1)} = \vec{x}^{(r)} + \lambda_r \vec{s}_r$

Steepest Descent

Reasonable try: steepest descent

$$\vec{s}_r = -\vec{g}_r \quad \text{step length from} \quad 0 = \frac{\partial h(\vec{x}_r - \lambda \vec{g}_r)}{\partial \lambda}$$

Note that consecutive steps are in orthogonal directions.

As an example, we will come back to the data smoothing example from Lecture 3:

$$\chi^2 = \sum_{i=0}^{n} \frac{(y_i - f(x_i;\vec{\lambda}))^2}{w_i^2}$$

$\vec{\lambda}$ are the parameters of the function to be fit

y_i are the measured points at values x_i

w_i is the weight given to point i

In our example: $f(x; A, \vartheta) = A\cos(x + \vartheta)$

and $w_i = 1 \quad \forall i$

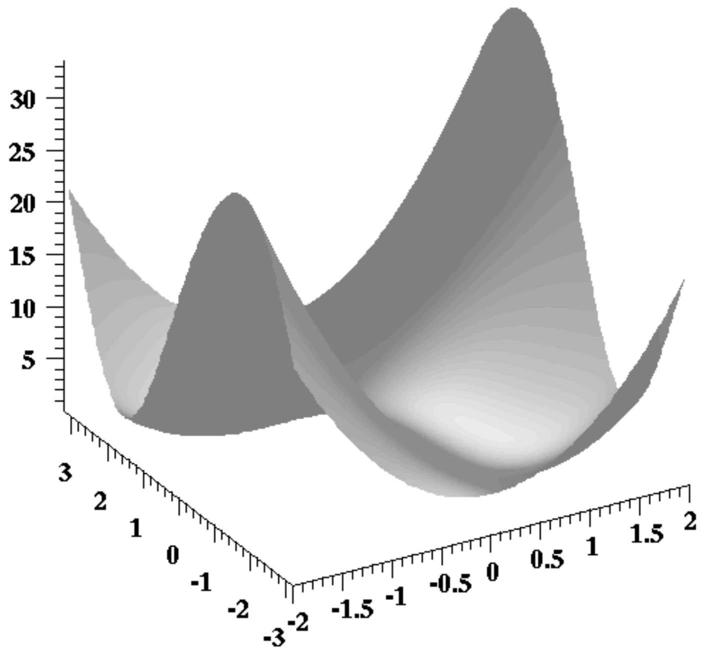
we want to minimize χ^2 as a function of *A* and φ

Steepest Descent

x	y
0.	0.
1.26	0.95
2.51	0.59
3.77	-0.59
5.03	-0.95
6.28	0.
7.54	0.95
8.80	0.59

$$\chi^2 = \sum_{i=0}^{n} \frac{(y_i - f(x_i;\vec{\lambda}))^2}{w_i^2}$$

$$h(A,\vartheta) = \chi^2 = \sum_{i=1}^{8} (y_i - A\cos(x_i + \vartheta))^2$$



Steepest Descent

To use our method, need to have the derivatives:

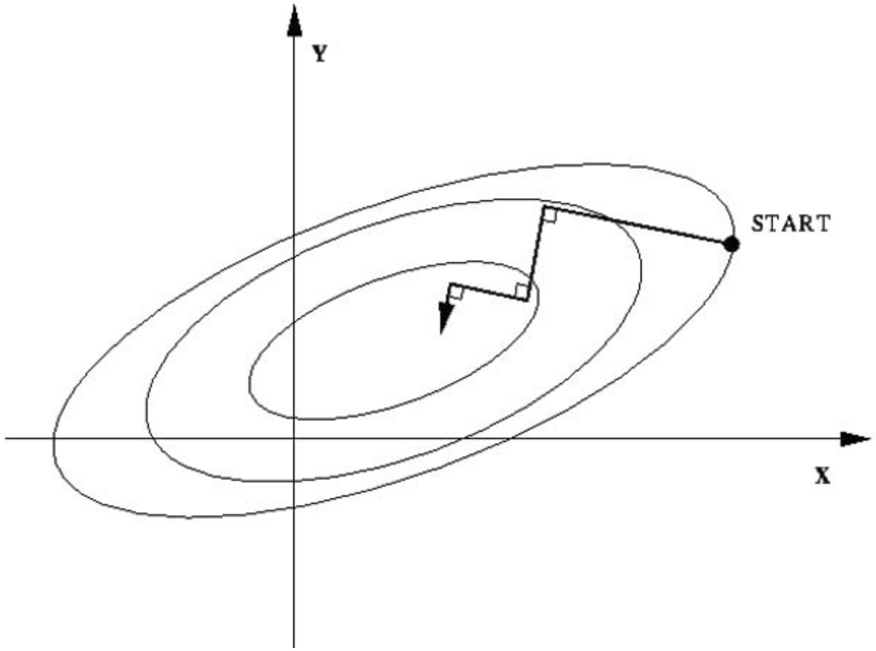
$$\vec{g}(A,\vartheta)=\begin{pmatrix}\sum_{i=1}^{8}2(y_i-A\cos(x_i+\vartheta))(-\cos(x_i+\vartheta))\\ \sum_{i=1}^{8}2(y_i-A\cos(x_i+\vartheta))(A\sin(x_i+\vartheta))\end{pmatrix}$$

recall step length from $0=\dfrac{dh(\vec{x}_r-\lambda\vec{g}_r)}{d\lambda_r}=\dfrac{d}{d\lambda_r}h(\vec{x}_{r+1})$

$$\frac{d}{d\lambda_r}h(\vec{x}_{r+1})=\nabla h(\vec{x}_{r+1})^T\cdot\frac{d}{d\lambda_r}\vec{x}_{r+1}=-\nabla h(\vec{x}_{r+1})^T\vec{g}_r$$

Setting to zero we see that the step length is chosen so as to make the next step orthogonal. We proceed in a zig-zag pattern.

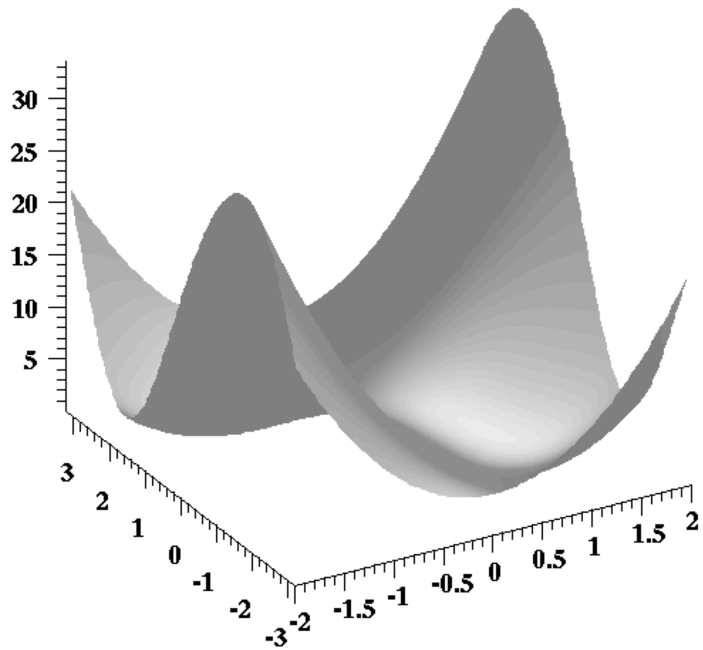
Steepest Descent

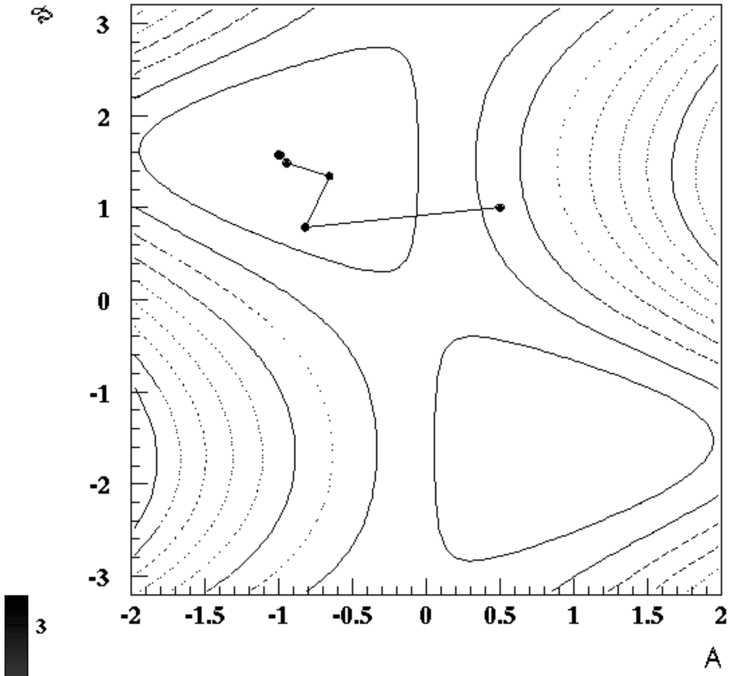


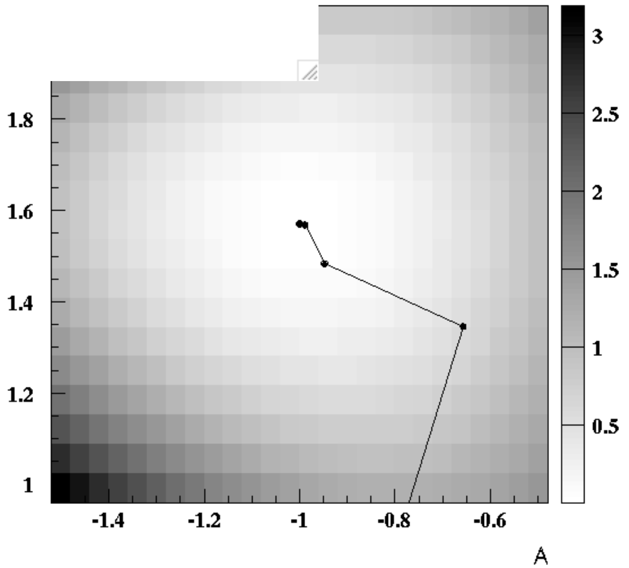
Determining the step size through the orthogonality is often difficult. Easier to do it by trial and error:

```
* Starting guesses for parameters
      A=0.5
      phase=1.
      step=1.
* Evaluate derivatives of function
      gA=dchisqdA(A,phase)
      gp=dchisqdp(A,phase)
*
      h=chisq(A,phase)
*
      Itry=0
 1    continue
      Itry=Itry+1
* update parameters for given step size
      A1=A-step*gA
      phase1=phase-step*gp
* reevaluate the chisquared
      h1=chisq(A1,phase1)
* change step size if chi squared increased
      If (h1.gt.h) then
        step=step/2.
        goto 1
      Endif
* Chi squared decreased, keep this update
      A=A1
      phase=phase1
      gA=dchisqdA(A,phase)
      gp=dchisqdp(A,phase)
```

Steepest Descent







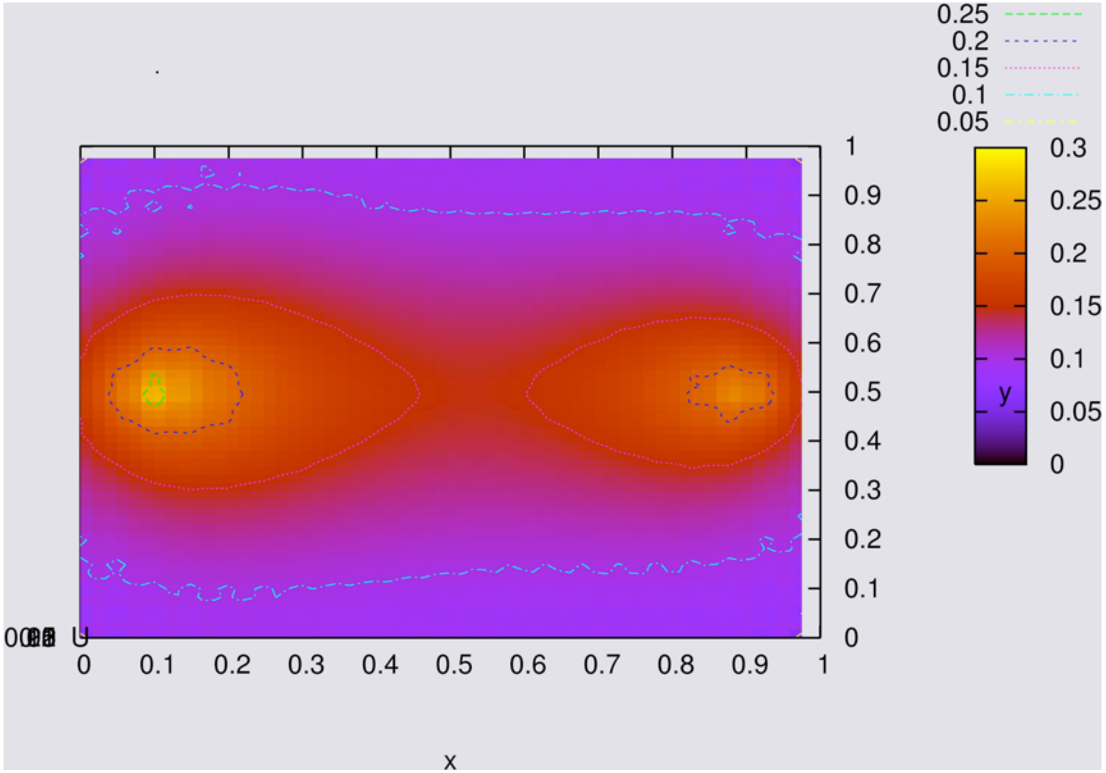
* Starting guesses for parameters
A=0.5
phase=1.
step=1.

Minimum found is A=-1, phase=π/2.
Previously, A=1, phase=- π/2

Other Techniques

- Conjugate Gradient
- Newton-Raphson
- Simulated annealing (Metropolis)
- Constrained optimization

Discuss briefly next time



You want to learn how to make these nice pictures ? Attend today's recitation. Special presentation from Fred on GNU.

Exercizes

1. Write a code for the 2-body motion studied in the lecture, but using numerically calculated derivatives (secant method). Compare the speed of convergence to that found for the NR method.

2. Code the χ^2 minimization problem. Find suitable starting conditions so that the second minimum is reached. Display graphically.