Christopher Lum
lum@uw.edu

Lecture 07b
Converting Constrained Optimization to Unconstrained Optimization Using the Penalty Method

The YouTube video entitled 'Converting Constrained Optimization to Unconstrained Optimization Using the Penalty Method' that covers this lecture is located at https://youtu.be/RTEpONXUJyE.

Outline

-References
-Unconstrained and Constrained Optimization
-Penalizing Equality Constraints
    -Convert to Unconstrained Optimization Problem
    -Extension to Higher Dimensions and Multiple Constraints
-Penalizing Inequality Constraints
-Barrier Method

References

Jensen, Paul A. and Bard, Jonathan F., "Operations Research Models and Methods," John Wiley and Sons, 2003. Ch. 10 Nonlinear Programming Methods (https://www.me.utexas.edu/~jensen/ORMM/supplements/pdf_supplements.html)

Unconstrained and Constrained Optimization

In general, consider an unconstrained optimization problem of

$(\wp_{UC})$ minimize $f_0(x)$ ***(Eq.1)***

Note: x is the decision vector but for sake of brevity, we use the simple notation of x rather than explicitly marking it as a vector, $\overline{x}$. These types of problems are desirable because we can use unconstrained

optimization algorithms (such as Matlab's fminsearch) to solve them.

Inequality and equality constraints can be added to ($\wp_{UC}$) to create a constrained optimization problem

($\wp_C$) minimize $f_0(x)$ ***(Eq.2)***

such that $f_i(x) \le 0$ $i = 1, 2, \ldots, n$ (n inequality constraints)
$f_i(x) = 0$ $i = n+1, n+2, \ldots, n+m$ (m equality constraints)

In general, the cost function, $f(x)$, and the constraint functions, $f_i(x)$, are nonlinear.

This makes this type of general, nonlinear, constrained optimization problem very difficult to solve.

Penalizing Equality Constraints

To keep the problem simple for now, we first consider a simpler case of ($\wp_C$) but with only equality constraints.

($\wp_{EO}$) minimize $f_0(x)$ ***(Eq.3)***

such that $f_i(x) = 0$ $i = 1, 2, \ldots, n$ (n equality constraints)

This can be solved using various optimization algorithms or procedures (see example below)

Example: 2D Quadratic

Let us examine a cost function of the form

$$f_0 = \frac{1}{2} x^T H x + g^T x + r \qquad \textbf{(Eq.4)}$$

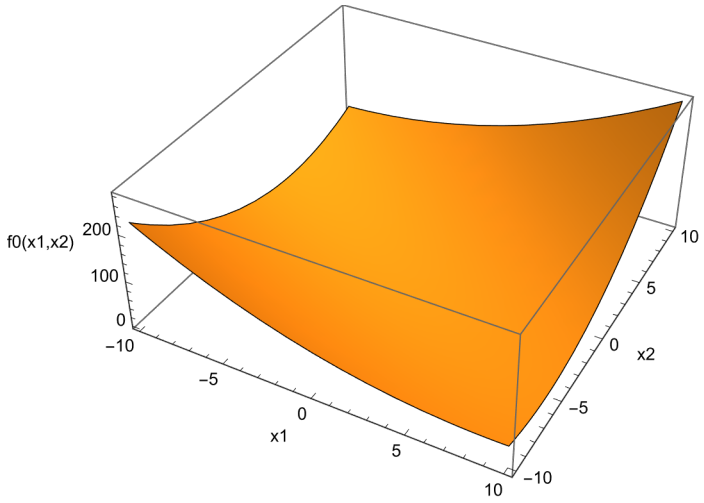
where $H = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$

$g = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$

$r = 2$

```
(*Define cost function*)
H = {{1, 1}, {1, 2}}; g = {{1}, {2}}; r = 2; x = {{x1}, {x2}};
temp = 1/2 Transpose[x].H.x + Transpose[g].x + r;
f0[x1_, x2_] = temp[[1, 1]]

(*Visualize cost function*)
Plot3D[f0[x1, x2], {x1, -10, 10}, {x2, -10, 10},
 AxesLabel → {"x1", "x2", "f0(x1,x2)"}, Mesh → False]
```

$$2 + x1 + 2\,x2 + \frac{1}{2}\,(x1\,(x1 + x2) + x2\,(x1 + 2\,x2))$$



So the constrained optimization problem with equality constraints only is

$$(\wp_{EO}) \qquad \text{minimize } f_0(x) \qquad \textbf{\textit{(Eq.5)}}$$

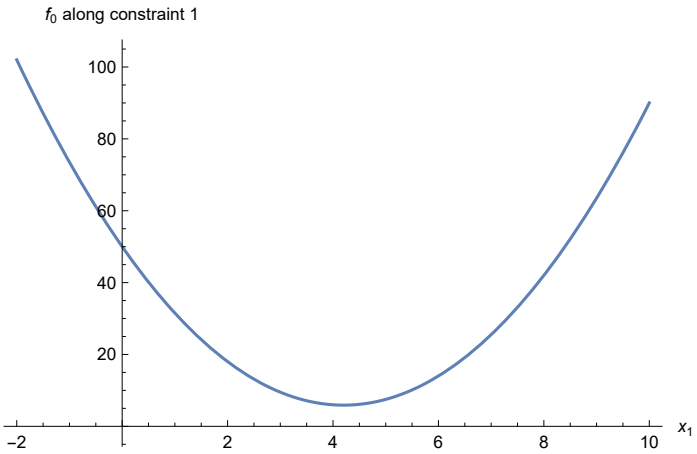
$$\text{such that } f_1(x) = x_2 - x_1 + 8 = 0$$

```
f1[x1_, x2_] = x2 - x1 + 8;
```

We can find the optimal solution analytically since the problem is so simple. We first lower the dimensionality of the problem by applying one of the constraints (solutions must satisfy $x_2 - x_1 + 8 = 0 \Longleftrightarrow x_2 = x_1 - 8$, or equivalently we can only consider the cost function at $f_0(x_1, x_1 - 8)$

```
f0scalar[x1_] = f0[x1, x1 - 8] // Simplify
Plot[f0scalar[x1], {x1, -2, 10}, AxesLabel → {"x1", "f0 along constraint 1"}]
```

$$50 - 21\,x1 + \frac{5\,x1^2}{2}$$



This is a 1D problem so we can simply find the minimum.

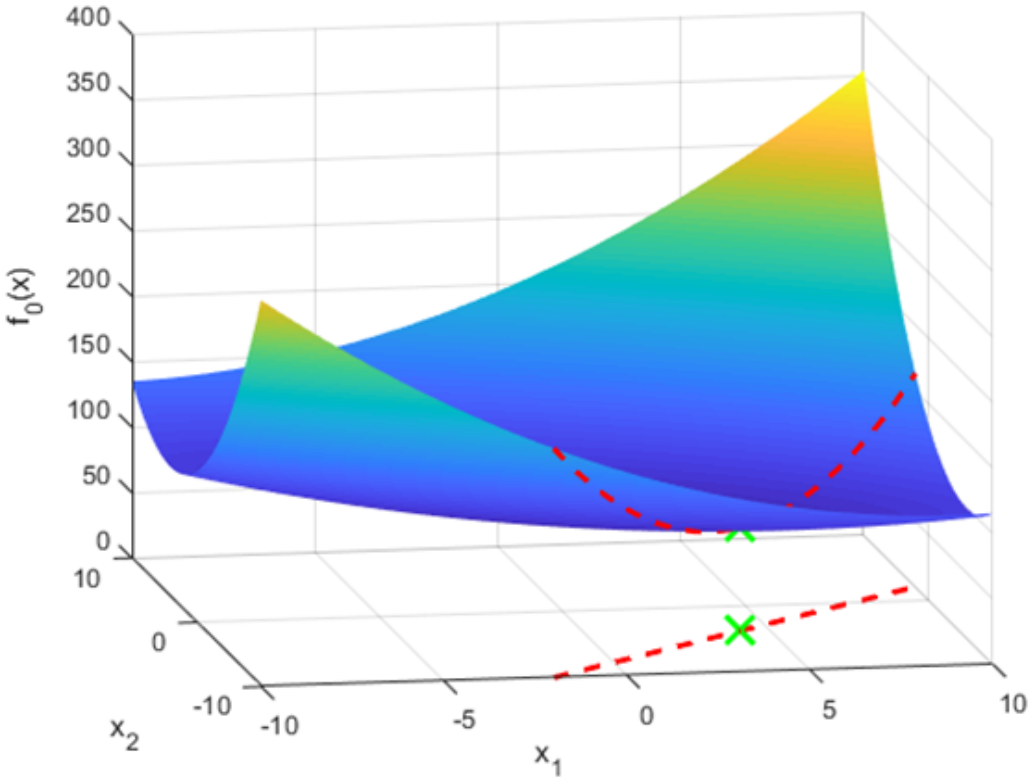
```
temp = Solve[D[f0scalar[x1], x1] == 0, x1];
x1star = x1 /. temp[[1]]
x2star = x1star - 8

x1star // N
x2star // N
```

$$\frac{21}{5}$$

$$-\frac{19}{5}$$

```
4.2
```

```
-3.8
```



We can verify that this solution satisfies the constraints

```
f1[x1star, x2star]
```

```
0
```

Convert to Unconstrained Optimization Problem

Let us step back and think about what is the meaning of the aforementioned equality constraint. The constraint is enforcing a condition that must be satisfied when solving the problem. It is akin to saying "if you step off this line, you die". But could we find the same solution if we only strongly penalized stepping off the line?

In that spirit, let us try to solve the constrained problem by instead generating an approximate unconstrained optimization by shuffling the constraint into the cost function

$$(\wp_{\text{approx}}) \quad \underset{x\in\mathbb{R}^2}{\text{minimize}}\ \hat{f}_0(x) = f_0(x) + \alpha f_1(x)^2 \qquad \textbf{(Eq.6)}$$

where $\alpha \geq 0$ is the **penalty parameter**
$f_1(x)^2$ is the **penalty function**

We make several observations of $(\wp_{\text{approx}})$:

1. There are no explicit constraints, therefore $\left(\wp_{\text{approx}}\right)$ is an unconstrained optimization problem.

2. The term $\alpha f_1(x)^2$ is greater than 0 whenever $f_1(x) \neq 0$ and has a minimum when $f_1(x) = 0$ ($f_1(x) = 0 \Longleftrightarrow$ constraint 1 is satisfied.)

3. The composite cost function $\hat{f}_0(x)$ has a minimum when the sum $f_0(x) + \alpha f_1(x)^2$ is minimal. This may not be a location which yields $f_1(x) = 0$ (as we will see shortly in an example). However, as α becomes large, it becomes more and more important to chose an x that yields $f_1(x) \approx 0$ and if there are degrees of freedom to spare, use those degrees to attempt to minimize $f_0(x)$. This is effectively a tuning knob that dictates how important satisfying this constraint is.

So the combination of the penalty parameter and penalty function are used to nudge the solution towards satisfying the constraint $f_1(x) = 0$

Example: 2D Quadratic (Continued)

Consider the quadratic problem we investigated earlier. The unconstrained optimization cost function, $\hat{f}_0(x)$ is given by Eq.6

```
f0hat[x1_, x2_] = f0[x1, x2] + α f1[x1, x2]^2 // Expand
```

$$2 + x1 + \frac{x1^2}{2} + 2\,x2 + x1\,x2 + x2^2 + 64\,\alpha - 16\,x1\,\alpha + x1^2\,\alpha + 16\,x2\,\alpha - 2\,x1\,x2\,\alpha + x2^2\,\alpha$$

So we see that this is simply a modified quadratic. Again, since the problem is so simple, we can analytically find the solution by finding where the gradient is zero

$$\hat{x}^* = \arg\left(\frac{\partial \hat{f}_0(x)}{\partial x} = 0\right) \qquad \textit{(Eq.7)}$$

```
(*Compute gradient*)
Print["∂f̂₀/∂x"]
df0hatdx1[x1_, x2_] = D[f0hat[x1, x2], x1]
df0hatdx2[x1_, x2_] = D[f0hat[x1, x2], x2]
Print[""]

(*Find where gradient is zero*)
Print["Solve ∂f̂₀/∂x=0"]
temp = Solve[{df0hatdx1[x1, x2] == 0, df0hatdx2[x1, x2] == 0}, {x1, x2}];
x1hatstar = x1 /. temp[[1]]
x2hatstar = x2 /. temp[[1]]
```

$\partial \hat{f}_0 / \partial x$

$1 + x1 + x2 - 16\,\alpha + 2\,x1\,\alpha - 2\,x2\,\alpha$

$2 + x1 + 2\,x2 + 16\,\alpha - 2\,x1\,\alpha + 2\,x2\,\alpha$

```
Solve ∂f̂₀/∂x=0
```

$$\frac{42\,\alpha}{1+10\,\alpha}$$

$$-\frac{1+38\,\alpha}{1+10\,\alpha}$$

As expected, the solution to $(\wp_{approx})$ depends on the value of α.

If $\alpha = 0$, the problem reverts to the unconstrained problem $(\wp_{UC})$ (see Eq.1).

```
x1hatstar /. {α → 0}
x2hatstar /. {α → 0}
```

```
0
```

```
-1
```

Obviously, this is very far from the constrained optimal of $x^* = \begin{pmatrix} 4.2 \\ -3.8 \end{pmatrix}$. However, if we increase $\alpha = 1$, we obtain

```
x1hatstar /. {α → 1} // N
x2hatstar /. {α → 1} // N
```

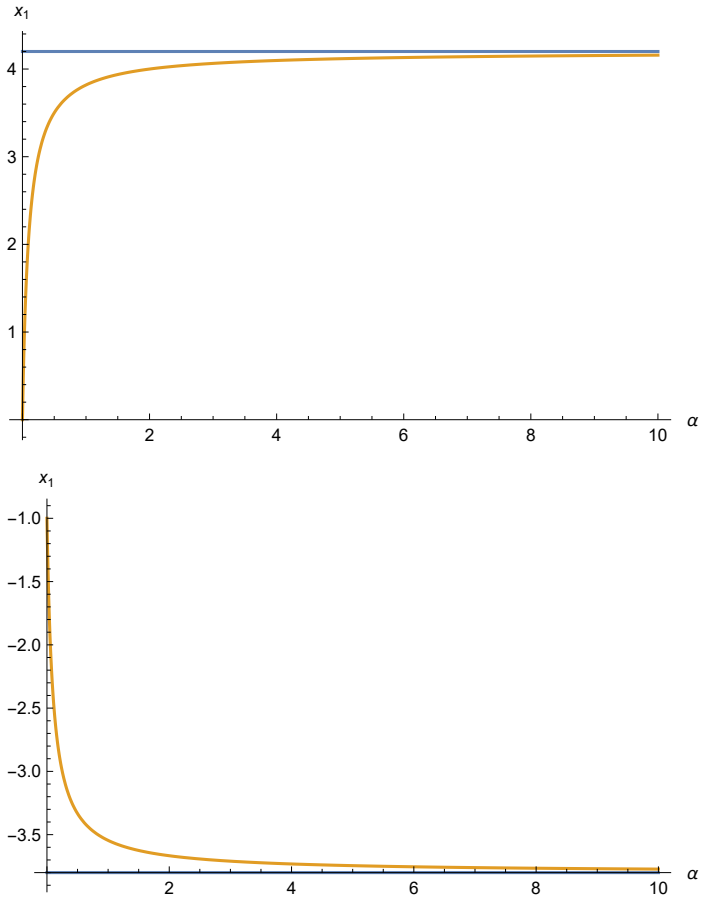
```
3.81818
```

```
-3.54545
```

Which is close but still different than the true $x^* = \begin{pmatrix} 4.2 \\ -3.8 \end{pmatrix}$

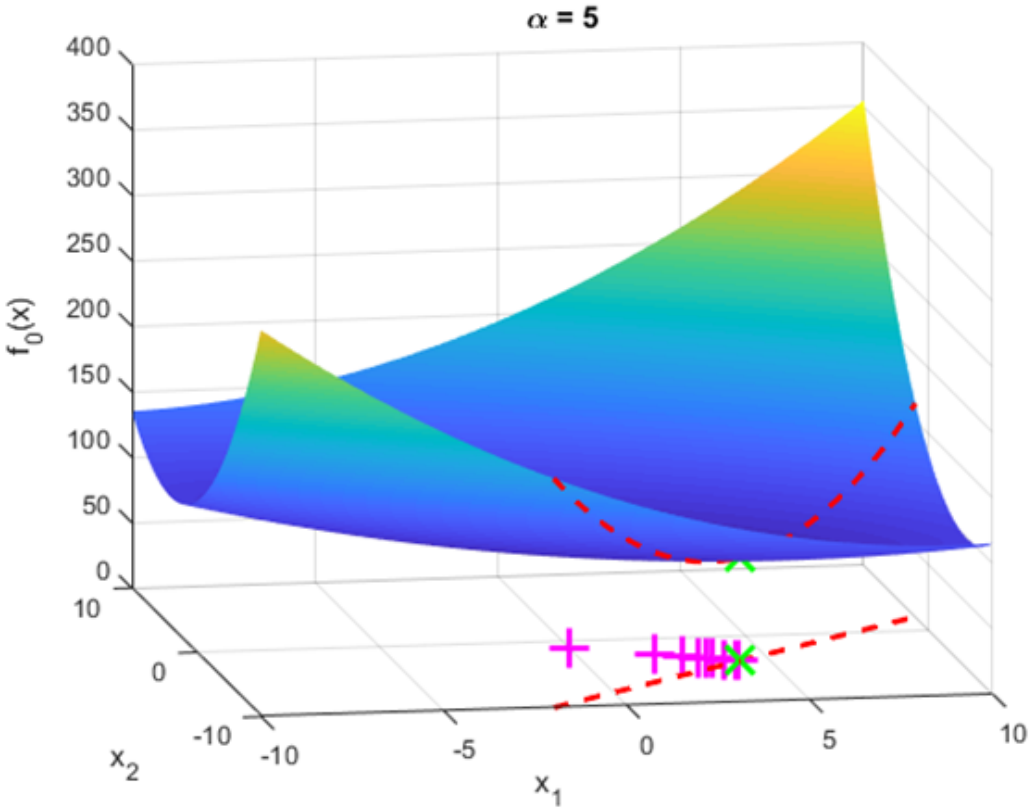
If we increase α, the solution of $\hat{x}^* \to x^*$

```
Plot[{x1star, x1hatstar}, {α, 0, 10}, AxesLabel → {"α", "x1"}, PlotRange → All]
Plot[{x2star, x2hatstar}, {α, 0, 10}, AxesLabel → {"α", "x1"}, PlotRange → All]
```



The power of this approach is that we do not need to analytically solve $(\wp_{approx})$ analytically (as we did in Eq.7). Instead, we can employ any unconstrained optimization algorithm/solver/routine to solve $(\wp_{approx})$.

For example, we can use Matlab's 'fminsearch' (look at help page then go to Matlab example). By increasing α, we penalize not meeting the constraint more and more (the magenta + signs show increasing α starts to meet the constrained problem).



An additional benefit of this approach is that we can use the general 'fminsearch' instead of more complex optimization algorithms (we do not need to purchase/depend on Matlab's Optimization Toolbox)

In essence, we are formulating the problem as an unconstrained problem by translating these hard constraints into "soft constraints" by moving them into the cost function. This is known as the **penalty method** first developed by Fiacco and McCormick in 1968. An excellent reference for this is Jensen and Bard, "Operations Research Models and Methods" (see references at bottom of notebook for full citation). In this case we have

$f_1(x)^2 = P(x)$ = penalty function
α = penalty parameter (non – negative)

Extension to Higher Dimensions and Multiple Constraints

We can easily extend this to p dimensions and m equality constraints

$$(\wp_{\text{approx}}) \quad \underset{x \in \mathbb{R}^p}{\text{minimize}}\ \hat{f}_0(x) = f_0(x) + \alpha_1 f_1(x)^2 + \alpha_1 f_2(x)^2 + \ldots + \alpha_m f_m(x)^2 \qquad \textbf{(Eq.8)}$$

Penalizing Inequality Constraints

In general, the penalty function does not need to take the form of

$$P(x) = f_i(x)^2$$

The key characteristics that we require of $P(x)$ are

i. $P(x)$ is continuous
ii. $P(x) \geq 0 \quad \forall\, x \in \Re^p$
iii. $P(x) = 0$ iff $f_i(x) = 0$ (minimum only when constraint is satisfied)

We can use these concepts (particularly iii) to extend this idea to apply to inequality constraints as well.

Recall the constrained problem (Eq.2, repeated here for convenience)

$$(\wp_C) \text{ minimize } f_0(x) \qquad \textbf{(Eq.2)}$$

such that $f_i(x) \leq 0 \qquad i = 1, 2, \ldots, n \qquad (n \text{ inequality constraints})$
$f_i(x) = 0 \qquad i = n+1, n+2, \ldots, n+m \quad (m \text{ equality constraints})$

In the case of an inequality constraint, a popular penalty function is

$$P(x) = \max[0, f_i(x)]^2 \qquad \textbf{(Eq.9)}$$

Note: Sometimes you will see a $\frac{1}{2}$ in front of the penalty function (especially when it is squared as it makes gradients easier).

We note several items about this penalty function

1. The square is outside the max function.

2. $P(x) \geq 0 \qquad \forall\, x \in \Re^p$

3. $f_i(x)$ is satisfied $\Longleftrightarrow f_i(x) \leq 0 \quad \Rightarrow \quad \max[0, f_i(x)]^2 = 0 \quad \Rightarrow \quad P(x) = 0$ when $f_i(x)$ is satisfied

4. $P(x) > 0$ when $f_i(x) > 0 \iff P(x) > 0$ when inequality constraint is not satisfied.

We now consider a problem with both equality and inequality constraints. For simplicity, we consider the same quadratic problem we examined earlier but add an inequality constraint of $f_2(x)$ as shown below

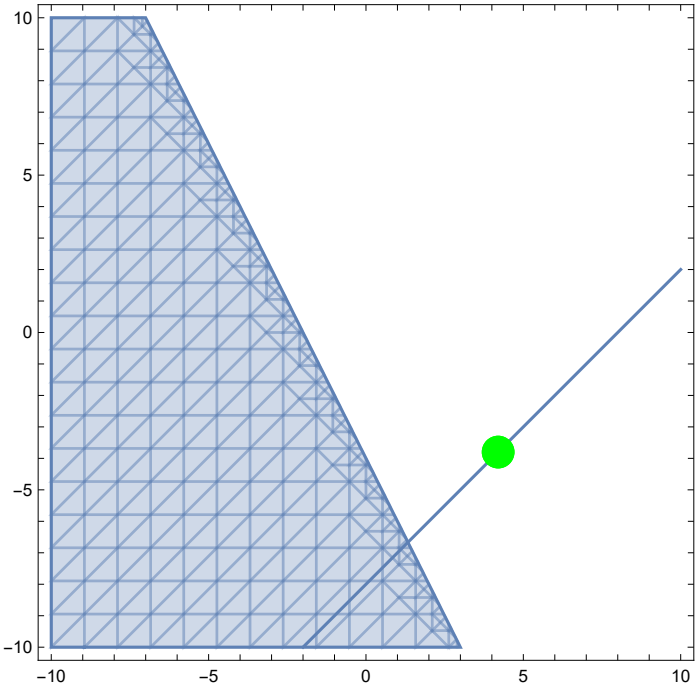
$$(\wp_C) \text{ minimize } f_0(x) \qquad \textbf{(Eq.10)}$$

such that $f_1(x) = x_2 - x_1 + 8 = 0$
$f_2(x) = 2x_1 + x_2 + 4 \leq 0$

$2x_1 + x_2 + 4 = 0$

```
f2[x1_, x2_] = 2 x1 + x2 + 4

(*Visualize the constraints*)
xstarf1only = {x1star, x2star}; (*Previous solution when only f1 was active*)

x1Min = -10;
x1Max = 10;
x2Min = -10;
x2Max = 10;
p1 = RegionPlot[f2[x1, x2] ≤ 0, {x1, x1Min, x1Max}, {x2, x2Min, x2Max}];
p2 = ContourPlot[f1[x1, x2] == 0, {x1, x1Min, x1Max}, {x2, x2Min, x2Max}];
p3 = ListPlot[{xstarf1only},
   PlotStyle → {Green, PointSize[0.05]}];
Show[p1, p2, p3,
 AxesLabel → {"x1", "x2"}]
```

```
4 + 2 x1 + x2
```



To apply the penalty method, we develop an approximate problem of

$$(\wp_{\text{approx}}) \quad \underset{x\in\mathbb{R}^2}{\text{minimize}}\ \hat{f}_0(x) = f_0(x) + \alpha_1 P_1(x) + \alpha_2 P_2(x) \qquad \textbf{(Eq.10)}$$

where $P_1(x) = f_1(x)^2$ (equality penalty function, same penalty function as before)

$P_2(x) = \max[0, f_2(x)]^2$ (inequality penalty function)

Solving Eq.10 via fminsearch with $\alpha_1 = \alpha_2 = 5$ yields

$$\hat{x}^* = \begin{pmatrix} 1.3187 \\ -6.2048 \end{pmatrix}$$

The solution should occur where the two constraints intersect

```
Solve[{f1[x1, x2] == 0, f2[x1, x2] == 0}, {x1, x2}] // N
```

```
{{x1 → 1.33333, x2 → -6.66667}}
```

So we have a very good agreement.

Barrier Method

This is related to the **barrier method** which we will cover in another video.