Christopher Lum
lum@uw.edu

Lecture 10f
Numerical Algorithms for Unconstrained Optimization: Step Size Via Line Minimization

Outline

-Introduction
-Choosing a Step Size
    -Line Minimization

Introduction

Recall that we are examining algorithms of the form

$$x^{k+1} = x^k + \alpha^k d^k \qquad k = 0, 1, \ldots \qquad \textbf{(Eq.1.5)}$$

where α^k = step size (positive scalar)
d^k = direction (vector)

Recall that for our analysis purposes, we consider d^k to be a unit vector.

In a previous lecture we examined the following two algorithms for choosing the step size.

1. Constant step size
2. Diminishing step size

As we saw previously, there are many pitfalls and things to watch out for when executing this numerical algorithm using constant or diminishing step size. The major deficiency in this algorithm is its lack of convergence which stems from the overly simplistic way we chose the step size. using a constant step size can lead to poor convergence. In addition, there is a danger of overshooting (going too far and then yielding $f(x^{k+1}) > f(x^k)$). This motivates our next method for choosing step size.

Step Size By Line Minimization

Choosing a step size based on the **line minimization rule** (aka **minimization rule, limited minimization rule,** or **line search**) can yield better results. In this method, α^k is such that the cost function is minimized along a line in the direction d^k starting from x^k. In other words α^k satisfies

$$(\wp_1)\quad \alpha^k = \arg\min_{\alpha \geq 0} f\left(x^k + \alpha\, d^k\right) \qquad \textbf{(Eq.1.10b)}$$

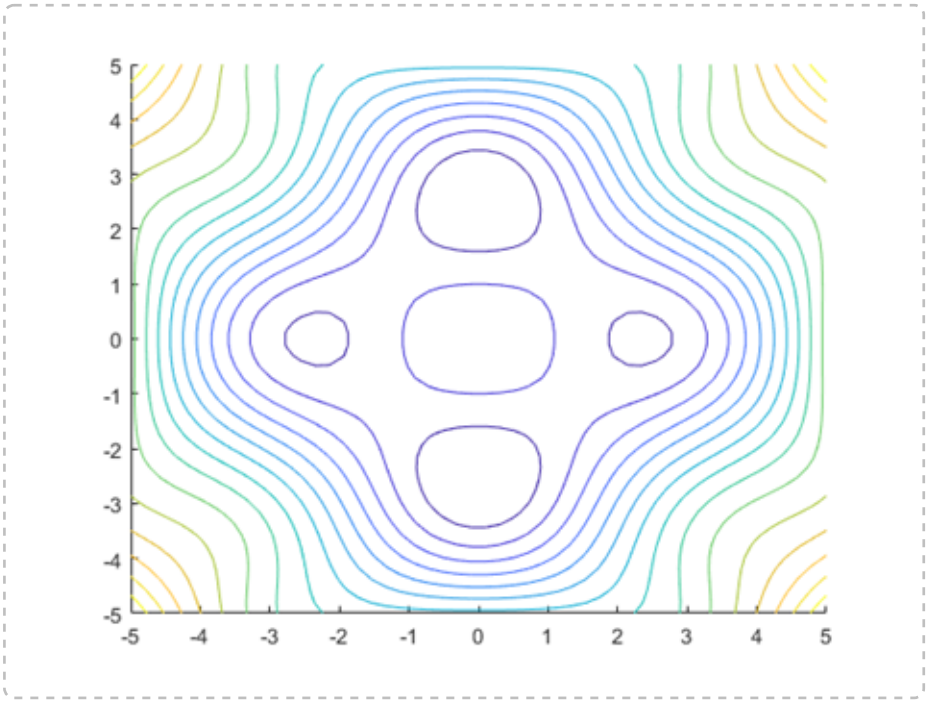
The benefit of $(\wp_1)$ is that it is a 1 dimensional optimization problem. One only needs to search in the direction of positive α to find the minimizer.

Example: Gradient Descent with Line Minimization

Consider the 2 dimensional cost function of the form

$$f_C(x) = 3\,x_1^2 + 2\,x_2^2 + 20\cos(x_2)\cos(x_1) + 40$$

A contour plot of this cost function was shown in the previous lecture and is repeated here for convenience



We can compute the gradient of this function easily

In[•]:=
```
(*Define function*)
fC[x1_, x2_] = 3 x1^2 + 2 x2^2 + 20 Cos[x2] Cos[x1] + 40;

(*Compute and display gradient*)
gradFC[x1_, x2_] = {{D[fC[x1, x2], x1]}, {D[fC[x1, x2], x2]}};
gradFC[x1, x2] // MatrixForm
```

Out[•]//MatrixForm=

$$\begin{pmatrix} 6\,x1 - 20\,\text{Cos}[x2]\,\text{Sin}[x1] \\ 4\,x2 - 20\,\text{Cos}[x1]\,\text{Sin}[x2] \end{pmatrix}$$

Suppose that we are at the point

$$x^k = \begin{pmatrix} -4 \\ 3 \end{pmatrix}$$

We can compute the gradient at this point and take the negative of the gradient as the descent direction (normalizing to a unit vector).

In[•]:=
```
(*Define x^k and find the function value at this point*)
Print["x^k"]
xk = {{-4}, {3}};
xk // MatrixForm
Print[" "]

(*Compute the gradient at this point*)
gradFCxk = gradFC[xk[[1, 1]], xk[[2, 1]]];

(*Choose the negative gradient as the descent direction (normalize to unit vector)*)
normGradFCxk = (gradFCxk[[1, 1]]^2 + gradFCxk[[2, 1]]^2)^(1/2);
dk = -gradFCxk * (1/normGradFCxk);

Print["d^k"]
dk // MatrixForm
Print[" "]

Print["d^k (numerical)"]
dk // MatrixForm // N
Print[" "]
```

x^k

Out[•]//MatrixForm=

$$\begin{pmatrix} -4 \\ 3 \end{pmatrix}$$

d^k

Out[•]//MatrixForm=

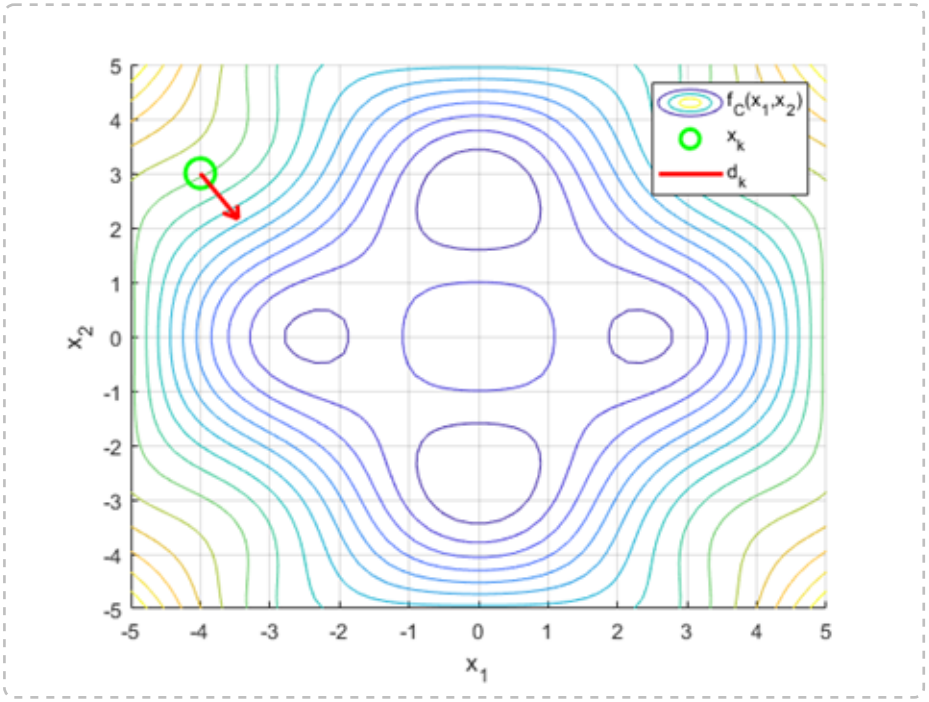
$$\begin{pmatrix} \frac{24-20\,\text{Cos}[3]\,\text{Sin}[4]}{\sqrt{(12-20\,\text{Cos}[4]\,\text{Sin}[3])^2+(-24+20\,\text{Cos}[3]\,\text{Sin}[4])^2}} \\ \frac{-12+20\,\text{Cos}[4]\,\text{Sin}[3]}{\sqrt{(12-20\,\text{Cos}[4]\,\text{Sin}[3])^2+(-24+20\,\text{Cos}[3]\,\text{Sin}[4])^2}} \end{pmatrix}$$

```
d^k (numerical)
```

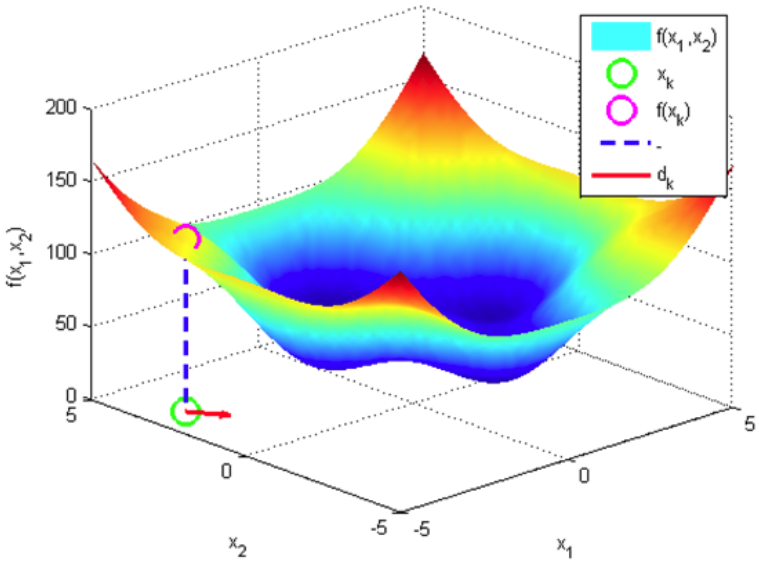
Out[•]//MatrixForm=

$$\begin{pmatrix} 0.545681 \\ -0.837993 \end{pmatrix}$$

We can now visualize the scenario



Or on the full surface plot



We now define the line at the point x^k and moving in the direction of d^k. The distance along this line is parameterized by the variable α.

$$r(\alpha) = x^k + \alpha\, d^k$$

$$= \begin{pmatrix} -4 \\ 3 \end{pmatrix} + \alpha \begin{pmatrix} 0.545681 \\ -0.837993 \end{pmatrix}$$

```
In[•]:= r[α] = xk + α dk;
        r[α] // MatrixForm // Simplify
```

Out[•]//MatrixForm=

$$\begin{pmatrix} -4 + \frac{\alpha\,(24-20\,\mathrm{Cos}[3]\,\mathrm{Sin}[4])}{\sqrt{(12-20\,\mathrm{Cos}[4]\,\mathrm{Sin}[3])^2+(-24+20\,\mathrm{Cos}[3]\,\mathrm{Sin}[4])^2}} \\ 3 + \frac{2\,\alpha\,(-3+5\,\mathrm{Cos}[4]\,\mathrm{Sin}[3])}{\sqrt{5\,(46-5\,\mathrm{Cos}[2]-5\,\mathrm{Cos}[14]-12\,\mathrm{Sin}[1]-36\,\mathrm{Sin}[7])}} \end{pmatrix}$$

Or if we want numerical values

```
In[•]:= r[α] // MatrixForm // N
```

Out[•]//MatrixForm=

$$\begin{pmatrix} -4. + 0.545681\,\alpha \\ 3. - 0.837993\,\alpha \end{pmatrix}$$

In other words, the line $r(\alpha)$ is given by x_1 and x_2 values that satisfy

$$r(\alpha) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -4 + 0.545681\,\alpha \\ 3 - 0.837993\,\alpha \end{pmatrix}$$

We can now parameterize the cost function along this line. In other words, we evaluate the cost function where x_1 and x_2 are specific values along the line specified by $r(\alpha)$.

$$f_C(r(\alpha)) = 3\,{x_1}^2 + 2\,{x_2}^2 + 20\cos(x_2)\cos(x_1) + 40$$

where $x_1 = -4 + \frac{\alpha(24-20\,\mathrm{Cos}[3]\,\mathrm{Sin}[4])}{\sqrt{(12-20\,\mathrm{Cos}[4]\,\mathrm{Sin}[3])^2+(24-20\,\mathrm{Cos}[3]\,\mathrm{Sin}[4])^2}} = -4 + 0.545681\,\alpha$

$$x_{2=}\, 3 + \frac{\alpha(-12+20\,\mathrm{Cos}[4]\,\mathrm{Sin}[3])}{\sqrt{(12-20\,\mathrm{Cos}[4]\,\mathrm{Sin}[3])^2+(24-20\,\mathrm{Cos}[3]\,\mathrm{Sin}[4])^2}} = 3 - 0.837993\,\alpha$$

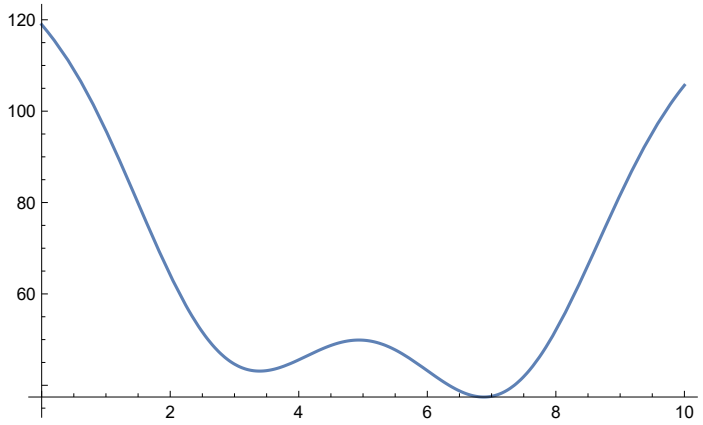
```
Print["Analytial Representation of 1D function"]
fCr[α_] = fC[r[α][[1, 1]], r[α][[2, 1]]] // Simplify
Print[" "]

Print["Numerical representation of 1D function"]
Expand[fCr[α]] // N

(*Plot the 1D function*)
Plot[fCr[α], {α, 0, 10}]
```

```
Analytial Representation of 1D function
```

$$40 + 20 \cos\left[3 + \frac{2\alpha\,(-3 + 5\cos[4]\sin[3])}{\sqrt{5 \times (46 - 5\cos[2] - 5\cos[14] - 12\sin[1] - 36\sin[7])}}\right] \cos\left[4 + \frac{2\alpha\,(-6 + 5\cos[3]\sin[4])}{\sqrt{5 \times (46 - 5\cos[2] - 5\cos[14] - 12\sin[1] - 36\sin[7])}}\right] + 3\left(-4 + \frac{\alpha\,(24 - 20\cos[3]\sin[4])}{\sqrt{(12 - 20\cos[4]\sin[3])^2 + (-24 + 20\cos[3]\sin[4])^2}}\right)^2 + 2\left(3 + \frac{2\alpha\,(-3 + 5\cos[4]\sin[3])}{\sqrt{5 \times (46 - 5\cos[2] - 5\cos[14] - 12\sin[1] - 36\sin[7])}}\right)^2$$

```
Numerical representation of 1D function
```

$$106. - 23.1523\,\alpha + 2.29777\,\alpha^2 + 20.\cos[3. - 0.837993\,\alpha]\cos[4. - 0.545681\,\alpha]$$



Although the result is somewhat complicated, we see that the benefit of this method is that it has reduced the cost function to a single dimension (it is only a function of α).

Therefore, solving ($\wp_1$) is significantly simpler, we only need to solve a 1D optimization problem

$$(\wp_1)\quad \alpha^k = \arg\min_{\alpha \geq 0} f_C\left(x^k + \alpha\, d^k\right)$$

or equivalently

$$(\wp_1)\ \ \alpha^k = \arg\min_{\alpha \geq 0} f_C(r(\alpha))$$

So we see that we can find the minimum by solving α which satisfies

$$\frac{d f_C(r(\alpha))}{d \alpha} = 0$$

```
Solve[D[fCr[α], α] == 0, α]
```

Solve: This system cannot be solved with the methods available to Solve.

$$\text{Solve}\left[\frac{6 \times (24 - 20 \cos[3] \sin[4]) \times \left(-4 + \frac{\alpha\,(24 - 20 \cos[3] \sin[4])}{\sqrt{(12 - 20 \cos[4] \sin[3])^2 + (-24 + 20 \cos[3] \sin[4])^2}}\right)}{\sqrt{(12 - 20 \cos[4] \sin[3])^2 + (-24 + 20 \cos[3] \sin[4])^2}} + \frac{8 \times (-3 + 5 \cos[4] \sin[3]) \times \left(3 + \frac{2\alpha\,(-3 + 5 \cos[4] \sin[3])}{\sqrt{5 \times (46 - 5 \cos[2] - 5 \cos[14] - 12 \sin[1] - 36 \sin[7])}}\right)}{\sqrt{5 \times (46 - 5 \cos[2] - 5 \cos[14] - 12 \sin[1] - 36 \sin[7])}} - 8 \cos\left[4 + \frac{2\alpha\,(-6 + 5 \cos[3] \sin[4])}{\sqrt{5 \times (46 - 5 \cos[2] - 5 \cos[14] - 12 \sin[1] - 36 \sin[7])}}\right] (-3 + 5 \cos[4] \sin[3]) \sqrt{\frac{5}{46 - 5 \cos[2] - 5 \cos[14] - 12 \sin[1] - 36 \sin[7]}} \sin\left[3 + \frac{2\alpha\,(-3 + 5 \cos[4] \sin[3])}{\sqrt{5 \times (46 - 5 \cos[2] - 5 \cos[14] - 12 \sin[1] - 36 \sin[7])}}\right] - 8 \cos\left[3 + \frac{2\alpha\,(-3 + 5 \cos[4] \sin[3])}{\sqrt{5 \times (46 - 5 \cos[2] - 5 \cos[14] - 12 \sin[1] - 36 \sin[7])}}\right] (-6 + 5 \cos[3] \sin[4]) \sqrt{\frac{5}{46 - 5 \cos[2] - 5 \cos[14] - 12 \sin[1] - 36 \sin[7]}} \sin\left[4 + \frac{2\alpha\,(-6 + 5 \cos[3] \sin[4])}{\sqrt{5 \times (46 - 5 \cos[2] - 5 \cos[14] - 12 \sin[1] - 36 \sin[7])}}\right] == 0, \alpha\right]$$

As can be seen, this is somewhat difficult to do analytically, so we numerically solve

```
temp = NSolve[D[fCr[α], α] == 0, α, Reals]
α1 = α /. temp[[1]];
α2 = α /. temp[[2]];
α3 = α /. temp[[3]];
```

```
{{α → 3.3906}, {α → 4.93559}, {α → 6.87888}}
```

We can then apply the 2nd derivative test to each of these points

```
D[fCr[α], {α, 2}] /. {α → α1}
D[fCr[α], {α, 2}] /. {α → α2}
D[fCr[α], {α, 2}] /. {α → α3}
```

```
17.8214
```

```
-13.6174
```

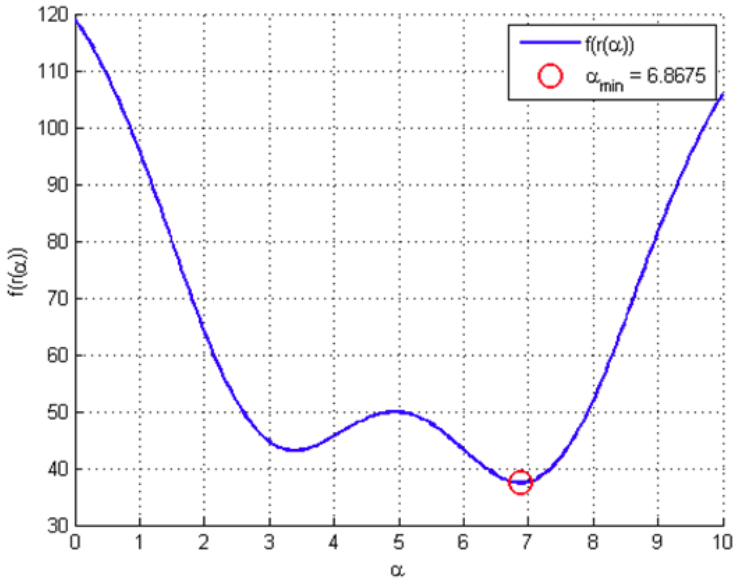
```
20.9862
```

So we see solutions 1 and 3 are local minima. We can evaluate which one has a lower cost function value

```
fCr[α1]
fCr[α3]
```

```
43.109
```

```
37.4333
```



So we see that the optimal step size is

$$\alpha^k \approx 6.8675$$

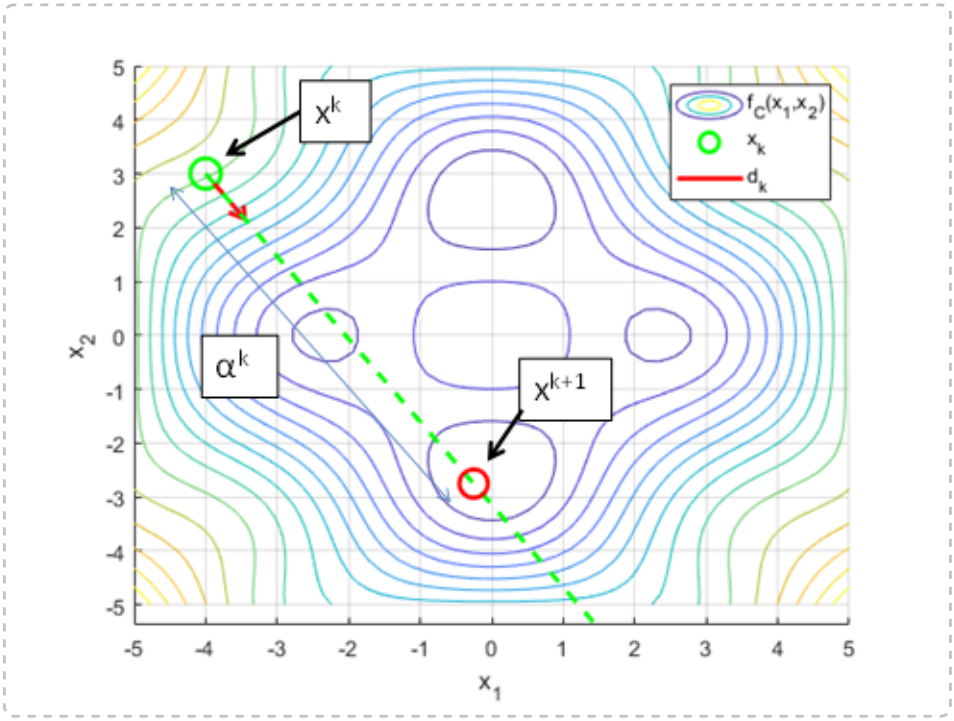
We can now choose the next point using Eq.1.5

$$x^{k+1} = x^k + \alpha^k d^k$$

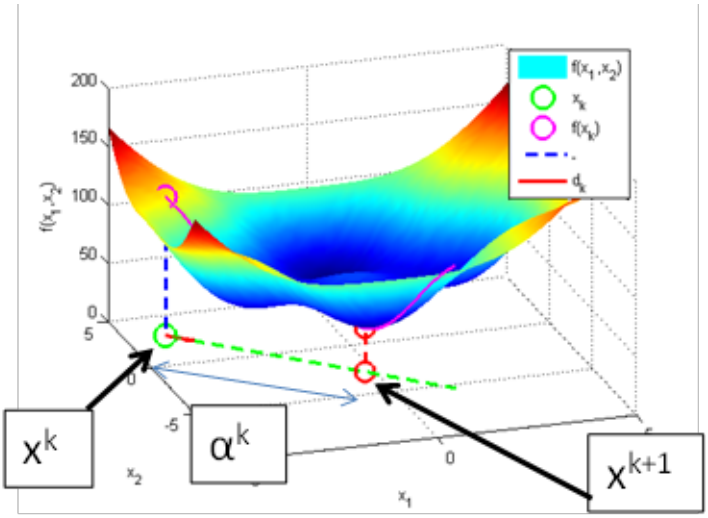
$$= \begin{pmatrix} -4 \\ 3 \end{pmatrix} + 6.8675 \begin{pmatrix} 0.5456 \\ -0.8379 \end{pmatrix}$$

$$x^{k+1} = \begin{pmatrix} -0.2526 \\ -2.7549 \end{pmatrix}$$

We can once again visualize this on the plot



Or on the full surface plot



We can check the norm of the gradient at this point

```
Norm[gradFC[-0.2526, -2.7549]]
```

```
7.18139
```

Since this is not zero (or within a specified tolerance), we repeat the process.

Generalize Line Minimization Process

To automate this process, we need to generalize some of this functionality.

```
Clear[xk, dk, r]
```

Step 1

```
r[α] = {{x1k}, {x2k}} + α {{d1k}, {d2k}};
r[α] // MatrixForm // Simplify
```

$$\begin{pmatrix} x1k + d1k\,\alpha \\ x2k + d2k\,\alpha \end{pmatrix}$$

Step 2

Convert *n* – dimensional function to 1D function

```
Print["Analytial Representation of 1D function"]
fCr[α_] = fC[r[α][[1, 1]], r[α][[2, 1]]] // Simplify
Print[" "]
```

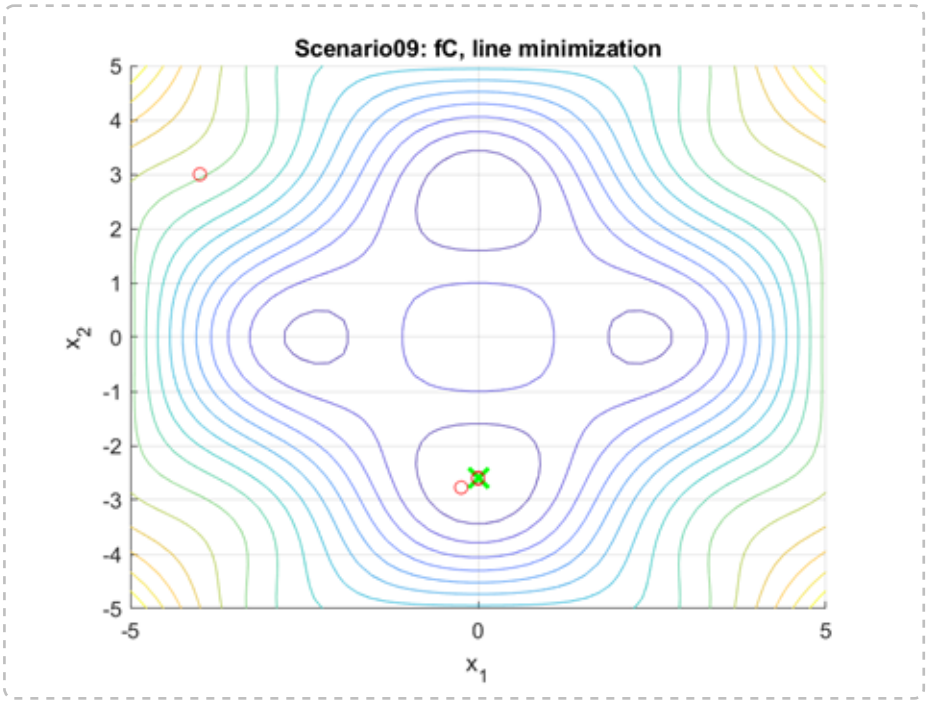
```
Analytial Representation of 1D function
```

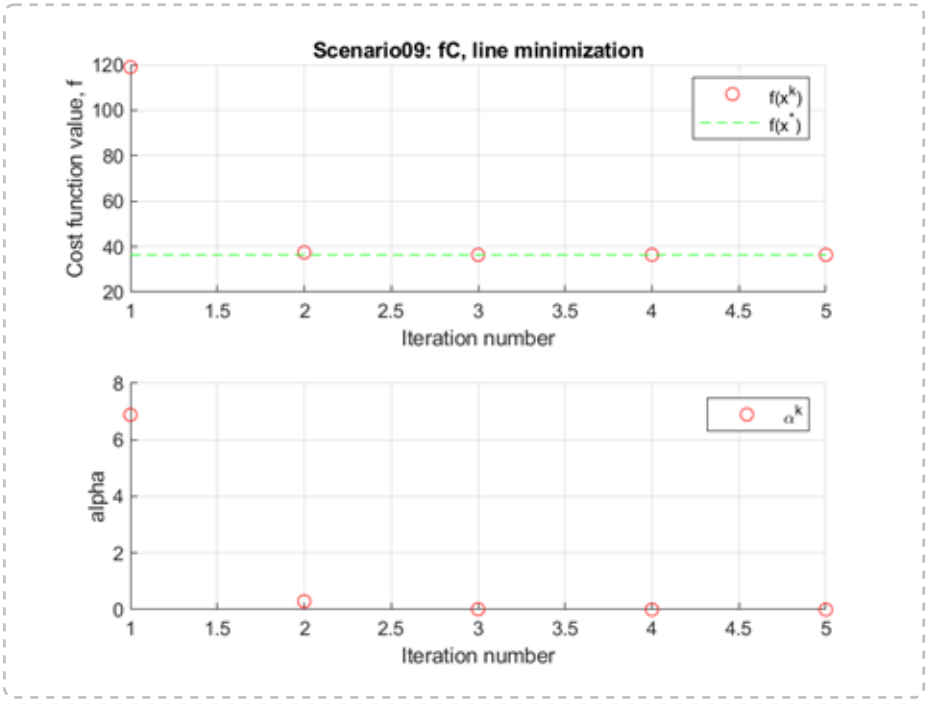
$$40 + 3\,(x1k + d1k\,\alpha)^2 + 2\,(x2k + d2k\,\alpha)^2 + 20\,\text{Cos}[x1k + d1k\,\alpha]\,\text{Cos}[x2k + d2k\,\alpha]$$

Step 3

Numerically minimize 1D function

The algorithm progress is shown below





As can be seen, this greatly outperforms the constant and diminishing step size. In fact, it appears to the find the minimum in only 4 steps.

Characterization of Cost Function Minima

From inspection and analysis, we can compute the stationary points. We see that one of the stationary points is at $x_1 = x_2 = 0$

```
In[•]:= Print["x* (point A)"]
        x1starA = 0
        x2starA = 0
        gradFC[x1starA, x2starA] // MatrixForm
        Print[" "]
```

```
x* (point A)
```

```
Out[•]= 0
```

```
Out[•]= 0
```

```
Out[•]//MatrixForm=
(0)
(0)
```

Two other stationary points are located along the line of $x_1 = 0$

```
In[•]:= temp = NSolve[gradFC[0, x2] == 0, x2, Reals]

        Print["x* (point B)"]
        x1starB = 0
        x2starB = x2 /. temp[[1]]
        gradFC[x1starB, x2starB] // MatrixForm // Chop
        Print[" "]

        Print["x* (point C)"]
        x1starC = 0
        x2starC = x2 /. temp[[3]]
        gradFC[x1starC, x2starC] // MatrixForm // Chop
        Print[" "]
```

```
Out[•]= {{x2 → -2.59574}, {x2 → 0.}, {x2 → 2.59574}}
```

```
x* (point B)
```

```
Out[•]= 0
```

```
Out[•]= -2.59574
```

```
Out[•]//MatrixForm=
(0)
(0)
```

```
x* (point C)
```

Out[•]= 0

Out[•]= 2.59574

Out[•]//MatrixForm=

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Two other stationary points are located along the line of $x_2 = 0$

In[•]:=
```
temp = NSolve[gradFC[x1, 0] == 0, x1, Reals]

Print["x* (point D)"]
x1starD = x1 /. temp[[1]]
x2starD = 0
gradFC[x1starD, x2starD] // MatrixForm // Chop
Print[" "]

Print["x* (point E)"]
x1starE = x1 /. temp[[3]]
x2starE = 0
gradFC[x1starE, x2starE] // MatrixForm // Chop
Print[" "]
```

Out[•]= {{x1 → −2.35644}, {x1 → 0.}, {x1 → 2.35644}}

```
x* (point D)
```

Out[•]= −2.35644

Out[•]= 0

Out[•]//MatrixForm=

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

```
x* (point E)
```

Out[•]= 2.35644

Out[•]= 0

Out[•]//MatrixForm=

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

We can characterize these stationary points

In[ ]:=
```
JA = fC[x1starA, x2starA]
JB = fC[x1starB, x2starB]
JC = fC[x1starC, x2starC]
JD = fC[x1starD, x2starD]
JE = fC[x1starE, x2starE]
```

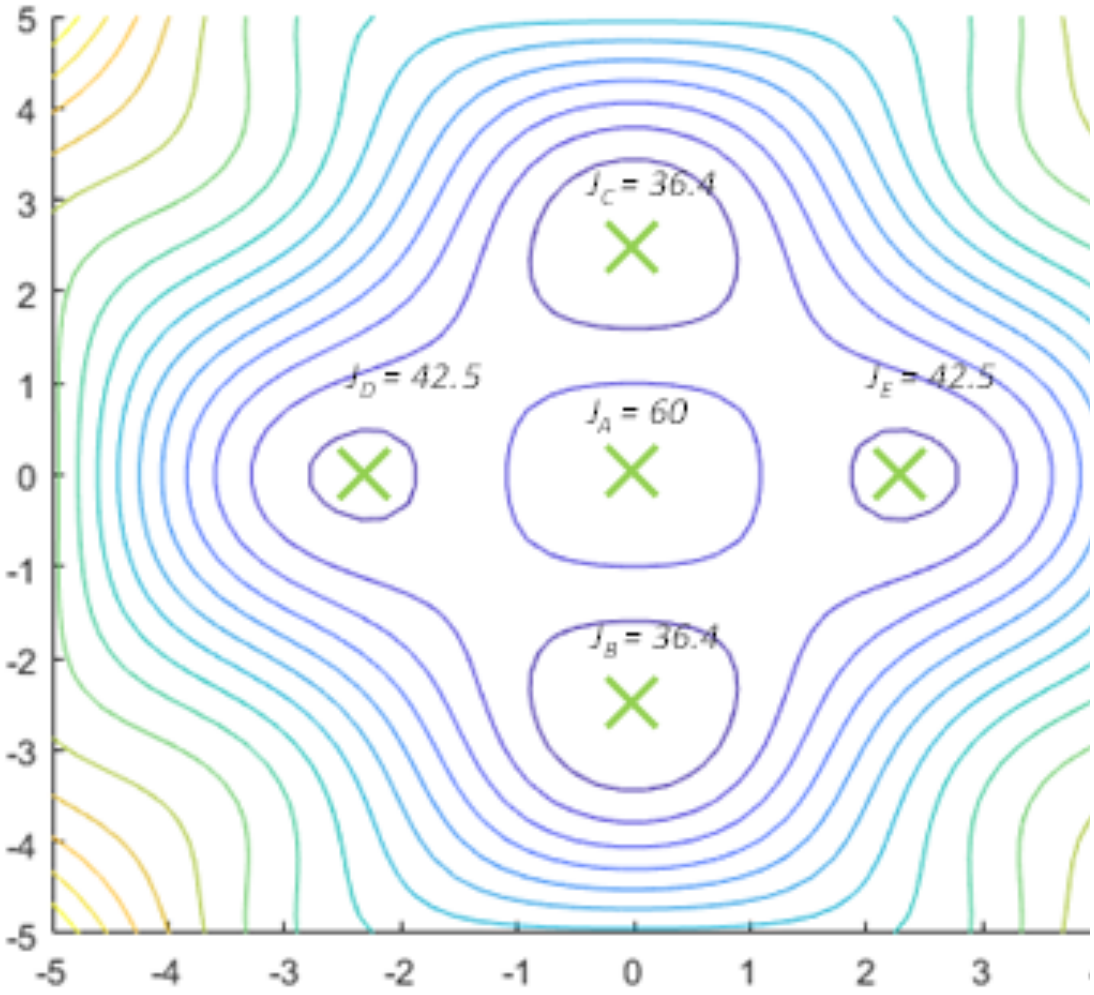
Out[ ]= 60

Out[ ]= 36.382

Out[ ]= 36.382

Out[ ]= 42.5128

Out[ ]= 42.5128



So as we can see, the algorithm finds the appropriate minimum.

Potential Issues

Note that this there are still some of the difficulties and limitations of this technique.

Typically Requires Analytical Representation of Cost Function

Requires an analytical representation of the cost function in order to have any chance of analytically solving ($\wp_1$).

($\wp_1$) May Not Be Analytically Solvable

Even with an analytical representation of cost function, ($\wp_1$) may not be solvable via the first and 2nd derivative technique.

How Can ($\wp_1$) Be Solved Confidently

If we need to resort to numerical techniques to solve ($\wp_1$), how we can ensure that we are searching all values of α appropriately (ie with appropriate $\Delta\alpha$ spacing). Because it may be impractical to numerically search over all $\alpha > 0$, sometimes ($\wp_1$) is modified to a limited line search of the form

$$(\wp_2) \quad \alpha^k = \arg\min_{\alpha \in [0,s]} f_C(r(\alpha))$$

where s = maximum search distance

Furthermore, if it is computationally expensive to evaluate the cost function, it may be difficult to solve ($\wp_2$). For example, searching over a large range of α values with small $\Delta\alpha$ will be computationally expensive.

These issues motivate other numerical step size techniques such as the Armijo Rule.

Multiple Minimia

The line minimization problem can still miss the global minima if the direction is not appropriate. For example, if the direction moves it towards a local rather than global min it can still get stuck in local minima ***<DRAW PICTURE SHOWING MUTIPLE MINIMIA AND IT GETS STUCK IN ONE>***