Solving ill-conditioned problems via Proximal Point method

- Suppose you have an objective which has a singular Hessian at the minimum (or maximum).
  - Economic examples: Flat top of likelihood hill, flat bottom to a moments criterion minimum
- Newton’s method may not properly converge for such problems
  - Round-off errors could cause convergence far from true solution
  - Any convergence will be slow.

Simple example

In[647]:=
```
a = 5; wgt =.; xold =.; yold =.
```

Suppose your objective is

In[648]:=
```
obj = (x + y - a)^4
```

Out[648]= $(-5 + x + y)^4$

There are multiple minima: any (x,y) such that x+y=5.
You can identify x+y but not (x,y)

In[649]:=
```
FindMinimum[obj, {x, 2}, {y, 2}]
```

Out[649]= $\{1. \times 10^{-16}, \{x \to 2.49995, y \to 2.49995\}\}$

This problem is so trivial and FindMinimum good enough that we get a solution. We stay with simple case to show basic idea.
So, suppose things did not go well.

Proximal Point method

Construct a penalty function

(xold, yold) is most recent guess

the penalty function is a quadratic penalty for choosing (x,y) different from (xold, yold)

In[650]:=
```
pen = (x - xold)^2 + (y - yold)^2
```
Out[650]= $(x - \text{xold})^2 + (y - \text{yold})^2$

Create a new objective function

In[651]:=
```
objProx = obj + wgt pen
```
Out[651]= $(-5 + x + y)^4 + \text{wgt}\left((x - \text{xold})^2 + (y - \text{yold})^2\right)$

objProx wants to minimize obj but imposes a cost for straying from (xold, yold)

We need to set the weight, and initial values for (xold, yold)

In[655]:=
```
wgt = 0.1;
xold = yold = 10;
```

In[657]:=
```
objProx
```
Out[657]= $0.1\left((-10 + x)^2 + (-10 + y)^2\right) + (-5 + x + y)^4$

Solve

In[658]:=
```
FindMinimum[objProx, {x, 2}, {y, 2}][[2]]
```
Out[658]= {x → 2.85478, y → 2.85478}

We get a solution. Let’s reset (xold, yold) and try again.

In[659]:=
```
{xold, yold} = {x, y} /. %
```
Out[659]= {2.85478, 2.85478}

In[660]:=
```
FindMinimum[objProx, {x, 2}, {y, 2}][[2]]
```
Out[660]= {x → 2.61451, y → 2.61451}

Repeat

In[661]:=
```
{xold, yold} = {x, y} /. %
```
Out[661]= {2.61451, 2.61451}

In[662]:=
```
FindMinimum[objProx, {x, 2}, {y, 2}][[2]]
```
Out[662]= {x → 2.56681, y → 2.56681}

In[663]:=
```
{xold, yold} = {x, y} /. %
```
Out[663]= {2.56681, 2.56681}

In[664]:=
```
FindMinimum[objProx, {x, 2}, {y, 2}][[2]]
```
Out[664]= {x → 2.54853, y → 2.54853}

In[665]:=
```
{xold, yold} = {x, y} /. %
```
Out[665]= {2.54853, 2.54853}

We now seemed to have become stuck. Remember that the weight is 0.1.
Let's reduce the weight on the penalty

In[666]:=
```
wgt = 0.001;
```

In[667]:=
```
FindMinimum[objProx, {x, 2}, {y, 2}][[2]]
```

Out[667]=
```
{x → 2.51304, y → 2.51304}
```

Progress! Let's repeat this a few times

In[668]:=
```
{xold, yold} = {x, y} /. %
```

Out[668]=
```
{2.51304, 2.51304}
```

In[669]:=
```
FindMinimum[objProx, {x, 2}, {y, 2}][[2]]
```

Out[669]=
```
{x → 2.50716, y → 2.50716}
```

In[670]:=
```
{xold, yold} = {x, y} /. %
```

Out[670]=
```
{2.50716, 2.50716}
```

In[671]:=
```
FindMinimum[objProx, {x, 2}, {y, 2}][[2]]
```

Out[671]=
```
{x → 2.50507, y → 2.50507}
```

In[672]:=
```
{xold, yold} = {x, y} /. %
```

Out[672]=
```
{2.50507, 2.50507}
```

We could reduce the penalty weight further and get closer to some (x, y) such that x+y=5, but let’s stop here.

What was the benefit of doing this?

Each step in the optimization problem was well-conditioned

Each step will converge quadratically to the solution of the penalized objective

You get arbitrarily close to some solution

You still cannot identify (x, y) but you can find a point that solves the problem

Identification

Economists are obsessed with identification

Why? No good reason.

My opinion: write down the model you think is valid and then let the computer tell you if you have identification.