Week 15 – Monday

Mathematical Modeling (Math 420)

Paul J. Hurtado

4 Dec, 2017

Example 1.1 (MMM[1])

Q: When should the pig be sold *to maximize profit*?

Profit P is revenue (R) minus cost (C), therefore

$$\begin{aligned} P(t) &= R(t) - C(t) \\ &= p \cdot w - c \cdot t \\ &= (0.65 - 0.01\ t)(200 + 5\ t) - 0.45\ t \end{aligned}$$

[1] The optional textbook for the course.

Example 1.1: Generalized Model

Profit Equation:

$$P(t) = pw - ct = \overbrace{\underbrace{(p_0 - rt)}_{p(t)}\underbrace{(w_0 + gt)}_{w(t)}}^{\text{Revenue}} - \underbrace{ct}_{\text{Cost}}$$

Variables (time dependent):

p - pig price per pound

w - weight (lbs)

t - time (days)

Parameters (constants):

$p0$ - initial pig price per pound

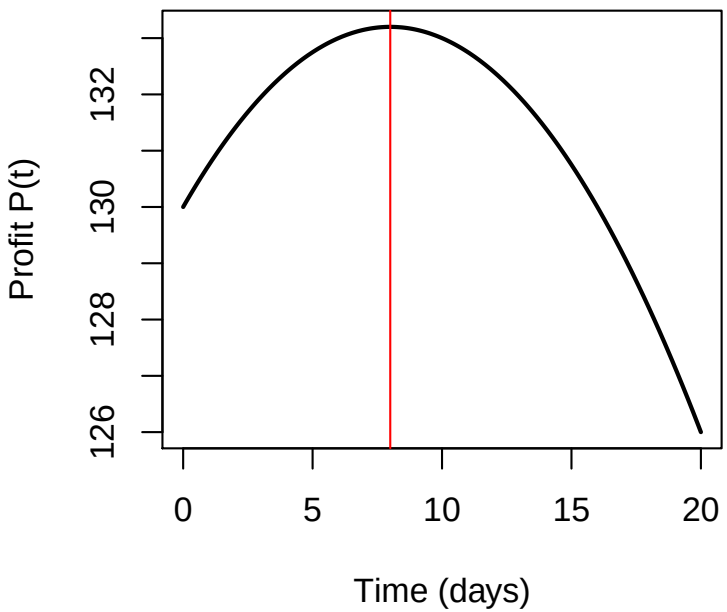
r - daily decrease in price p

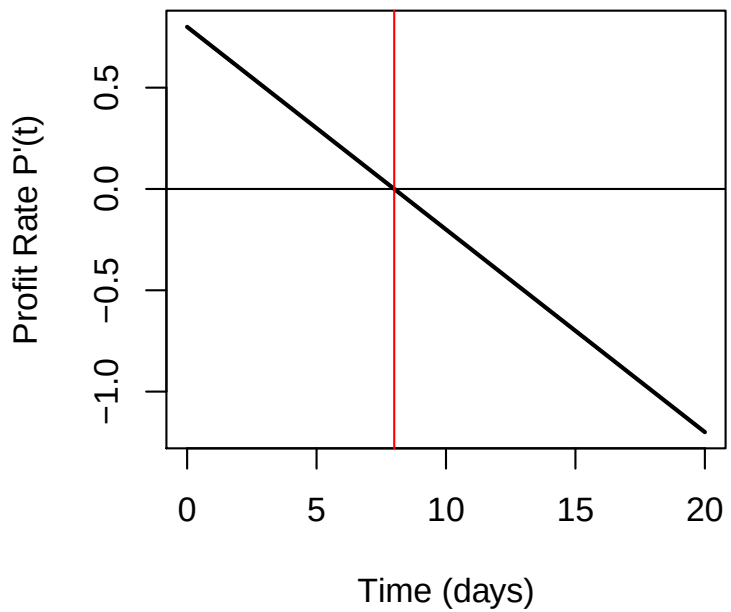
$w0$ - initial weight of pig

g - daily growth rate of pig

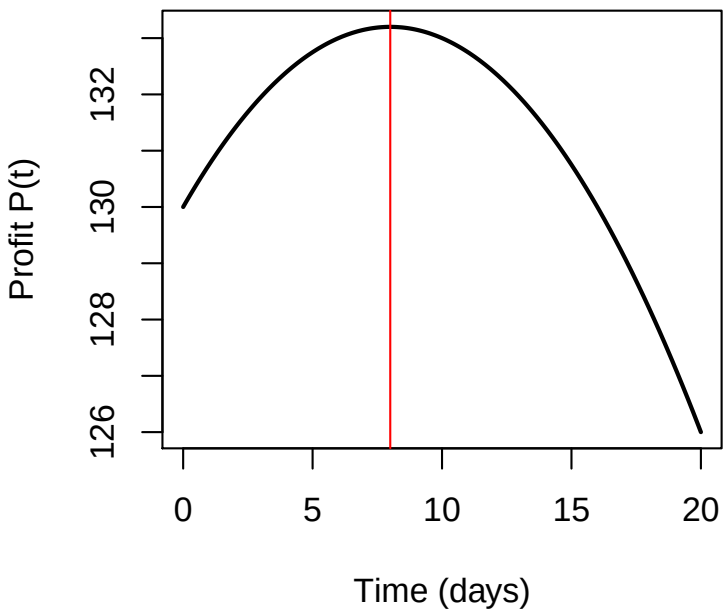
c - cost to keep pig ($/day)

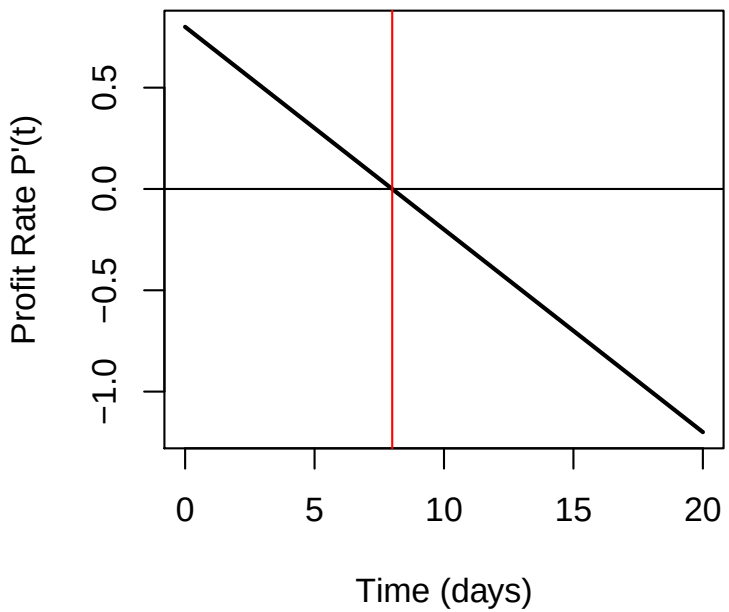
```r
## R code to plot P(t) and P'(t)
x=seq(0,20,length=200)
Pt=expression((0.65-0.01*x)*(200+5*x)-0.45*x,'x')
dPt=D(Pt,'x')
plot(x, (0.65-0.01*x)*(200+5*x)-0.45*x, type="l", lwd=2,
        ylab="Profit P(t)", xlab="Time (days)"); abline(v=8, col="red")
plot(x,eval(dPt),type="l",lwd=2,ylab="Profit Rate P'(t)",xlab="Time (days)")
xOpt = 8;
abline(h=0); abline(v=xOpt, col="red")
```



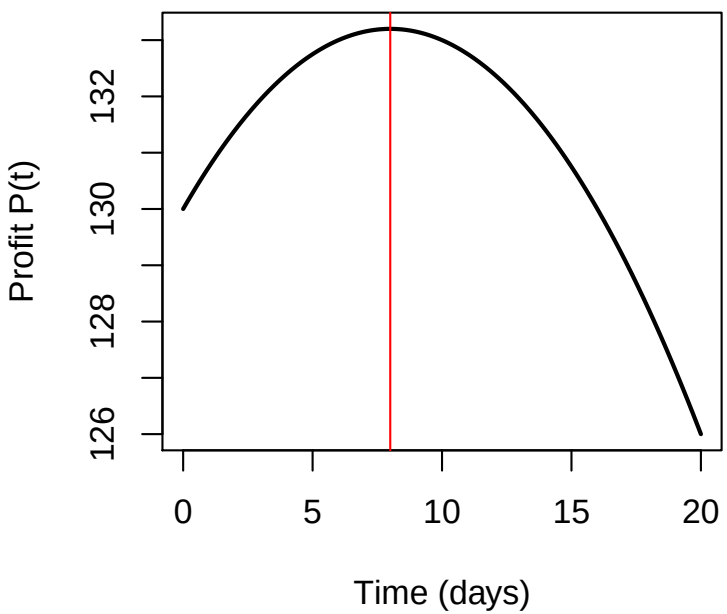


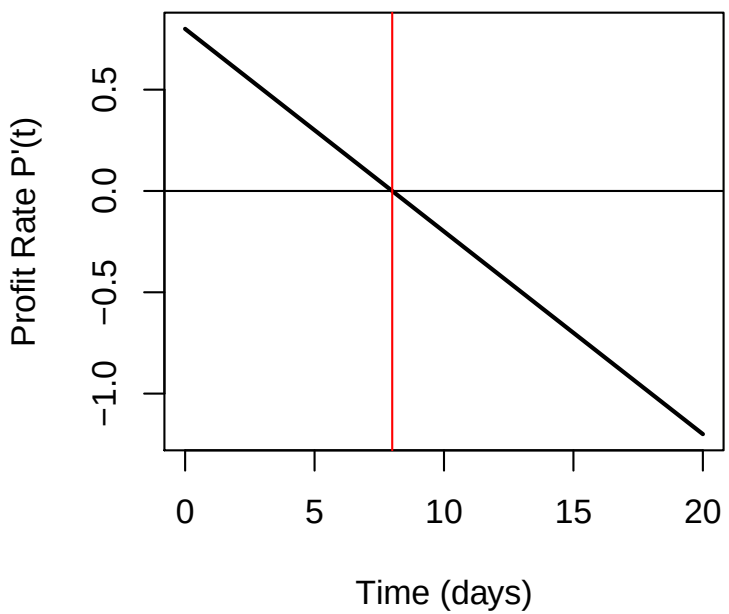
```r
## R code to plot P(t) and P'(t)
x=seq(0,20,length=200)
P=function(x) { (0.65-0.01*x)*(200+5*x)-0.45*x };
dP=function(x){ (8-x)/10 }; # Use Maxima!
plot(x, P(x), type="l", lwd=2,
        ylab="Profit P(t)", xlab="Time (days)"); abline(v=8, col="red")
plot(x, dP(x),type="l",lwd=2,ylab="Profit Rate P'(t)",xlab="Time (days)")
xOpt = 8;
abline(h=0); abline(v=xOpt, col="red")
```





```r
## R code to plot P(t) and P'(t)
x=seq(0,20,length=200)
P=function(x) { (0.65-0.01*x)*(200+5*x)-0.45*x };
dP=function(x){ (8-x)/10 }; # Use Maxima!
plot(x, P(x), type="l", lwd=2,
        ylab="Profit P(t)", xlab="Time (days)"); abline(v=8, col="red")
plot(x, dP(x),type="l",lwd=2,ylab="Profit Rate P'(t)",xlab="Time (days)")
xOpt = optimize(P,c(0,20),maximum=TRUE) # 1D optimization in R
abline(h=0); abline(v=xOpt, col="red")
```





Multivariable Optimization: Ex 2.1 (Pg 21)

Q: How many 19- & 21-inch TVs maximize profit?

Profit Equation:

$$P(x_{19}, x_{21}) = p\,x_{19} + q\,x_{21} - C = (p_0 - p_{19}\,x_{19} - p_{21}\,x_{21})x_{19} + (q_0 - q_{19}\,x_{19} - q_{21}\,x_{21})x_{21} - (c_0 + c_{19}\,x_{19} + c_{21}\,x_{21})$$

Variables (time dependent):

- x_{19} - number of 19in TVs sold
- x_{21} - number of 21in TVs sold
- p - 19in TV selling price ($)
- q - 21in TV selling price ($)

Parameters (constants):

- $p0$ - Retail price of 19-inch TV
- $q0$ - Retail price of 19-inch TV
- p_{19} - 19in discount / 19in TV sold
- p_{21} - 19in discount / 21in TV sold
- q_{19} - 21in discount / 19in TV sold
- ...

Multivariable Optimization: Ex 2.1 (Pg 21)

Approach #1: Find "interior" optima by solving $\nabla P = 0$.

Multivariable Optimization: Ex 2.1 (Pg 21)

Approach #1: Find "interior" optima by solving $\nabla P = 0$.

Using Maxima, we can find and solve

$$\frac{\partial P}{\partial x_{19}} = p_0 - 2p_{19}x_{19} - (q_{19} + p_{21})x_{21} - c_{19} = 0$$
$$\frac{\partial P}{\partial x_{21}} = q_0 - (q_{19} + p_{21})x_{19} - 2q_{21}x_{21} - c_{21} = 0$$

Multivariable Optimization: Ex 2.1 (Pg 21)

Approach #1: Find "interior" optima by solving $\nabla P = 0$.

Using Maxima, we can find and solve

$$\frac{\partial P}{\partial x_{19}} = p_0 - 2p_{19}x_{19} - (q_{19} + p_{21})x_{21} - c_{19} = 0$$
$$\frac{\partial P}{\partial x_{21}} = q_0 - (q_{19} + p_{21})x_{19} - 2q_{21}x_{21} - c_{21} = 0$$

Solving and using the given parameter values yields

$$\mathbf{x_{19}} = \mathbf{4735.04}\ldots \qquad \mathbf{x_{21}} = \mathbf{7042.74}\ldots$$

Multivariable Optimization: Ex 2.1 (Pg 21)

Approach #2: Use generic optimization routines to computationally maximize $P(x_{19}, x_{21})$ over $x_{19}, x_{21} > 0$.

See optimization.R

Constraints

Do we assume **global** or **local** optima?

Ex: Minimum of $(x - r)^2$ over $x \in \mathbb{R}$
Ex: Maximum of $\sin^2(x)$ over $x \in [0, 2\pi]$.

Are there domain constraints?

Box Constraints are the simplest domains restrictions:
Ex: Real-world constraint on sensible parameter values.
Ex: Minimize $f(\vec{x})$ given $l_i \leq x_i \leq u_i$
Ex: (TV Example) Recall we required $x_{19},\ x_{21} > 0$

Remember to check the boundary values!

Constraints

Equality Constraints:

Ex: Maximize $f(x, y)$ with constraints $g_i(x, y) = c_i$
Solution: Continuous functions? Use LaGrange Multipliers.

Inequality Constraints:

Ex: Maximize **linear** $f(x, y)$; linear constraints $g_i(x, y) \leq c_i$
Ex: Maximize **quadric** $f(x, y)$; linear constraints $g_i(x, y) \leq c_i$
Solution: Linear and Quadratic Programming, respectively.

LaGrange Multipliers

LaGrange Multipliers arise from necessary conditions for optimizing objective function $f(x)$ with **equality constraints** $g_i(x) = c_i$.

Theorem

If f and all g_i are **continuously differentiable**, *then:*

(a) *To maximize f, input x must satisfy*

$$\nabla f(x) = \sum_j \lambda_j \nabla g_j(x).$$

(b) *To minimize f, input x must satisfy*

$$-\nabla f(x) = \sum_j \lambda_j \nabla g_j(x).$$

Generalizing LaGrange Multipliers

Karush-Kuhn-Tucker (KKT) Multipliers arise from necessary conditions for optimizing objective function $f(x)$ w/ **equality** $(g_i(x) = c_i)$ and **ineq. constraints** $(h_j(x) \leq d_j)$.

Theorem

If f and all g_i and h_j are **continuously differentiable**, *then:*

(a) *To maximize f, input x must satisfy*

$$\nabla f(x) = \sum_i \mu_i \nabla h_i(x) + \sum_j \lambda_j \nabla g_j(x).$$

(b) *To minimize f, input x must satisfy*

$$-\nabla f(x) = \sum_i \mu_i \nabla h_i(x) + \sum_j \lambda_j \nabla g_j(x).$$

Constrained Optimization

Ex: Maximize general $f(x)$ with general constraints. (**Hard!**)
Solution: Equivalent, unconstrained objective function?

Smooth Constraints:

Ex: Maximize continuous $f(x, y)$ w/ constraints $g_i(x, y) = c_i$
Solution: Use LaGrange (or KKT) Multipliers.

Constrained Optimization

Ex: Maximize general $f(x)$ with general constraints. (**Hard!**)
Solution: Equivalent, unconstrained objective function?

Smooth Constraints:

Ex: Maximize continuous $f(x, y)$ w/ constraints $g_i(x, y) = c_i$
Solution: Use LaGrange (or KKT) Multipliers.

Linear Constraints:

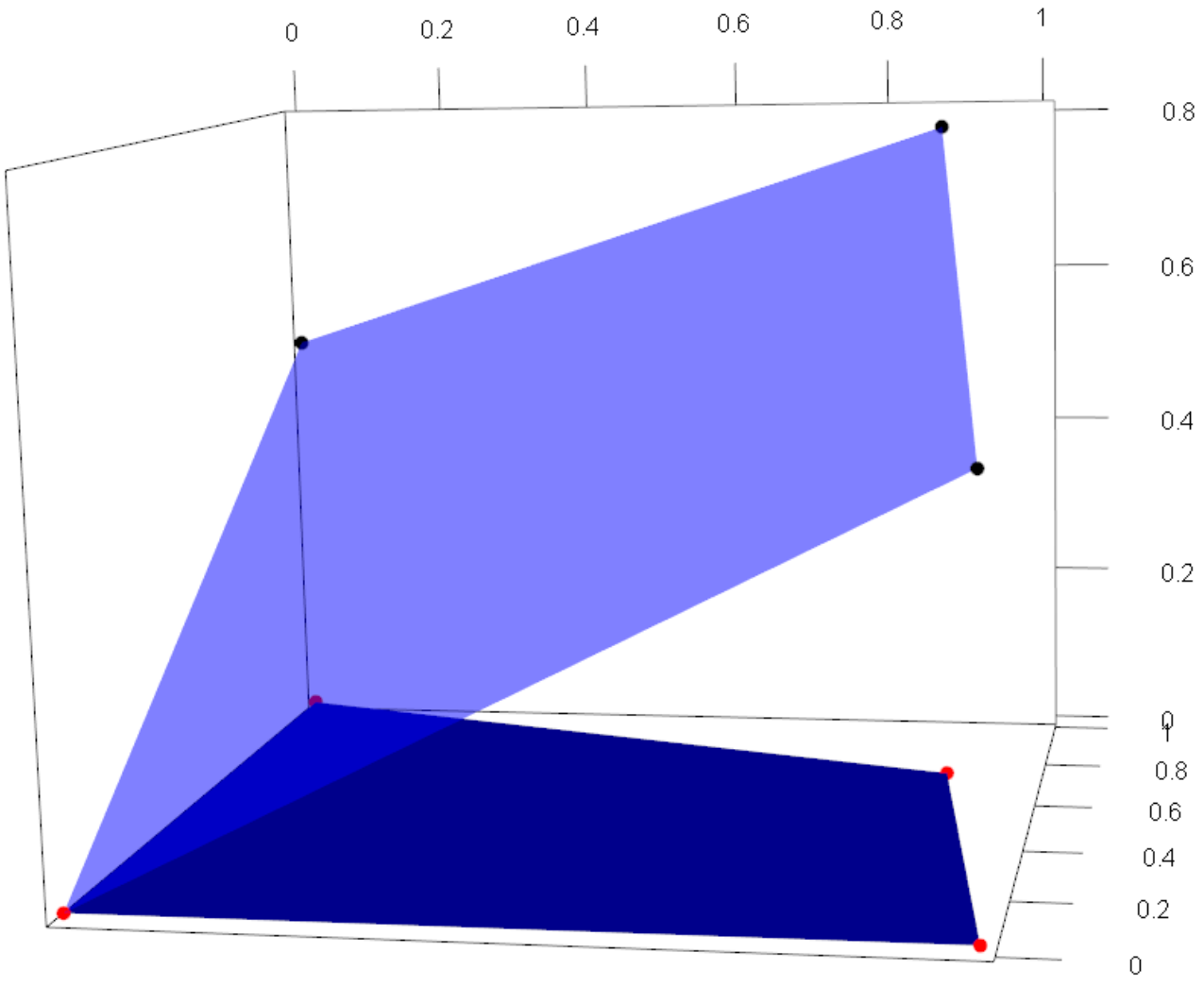
Ex: Maximize **linear** $f(x)$; linear constraints $g_i(x) \leq c_i$
Ex: Maximize **quadric** $f(x)$; linear constraints $g_i(x) \leq c_i$
Solution: Linear & Quadratic Programming, respectively

Example: Linear Programming

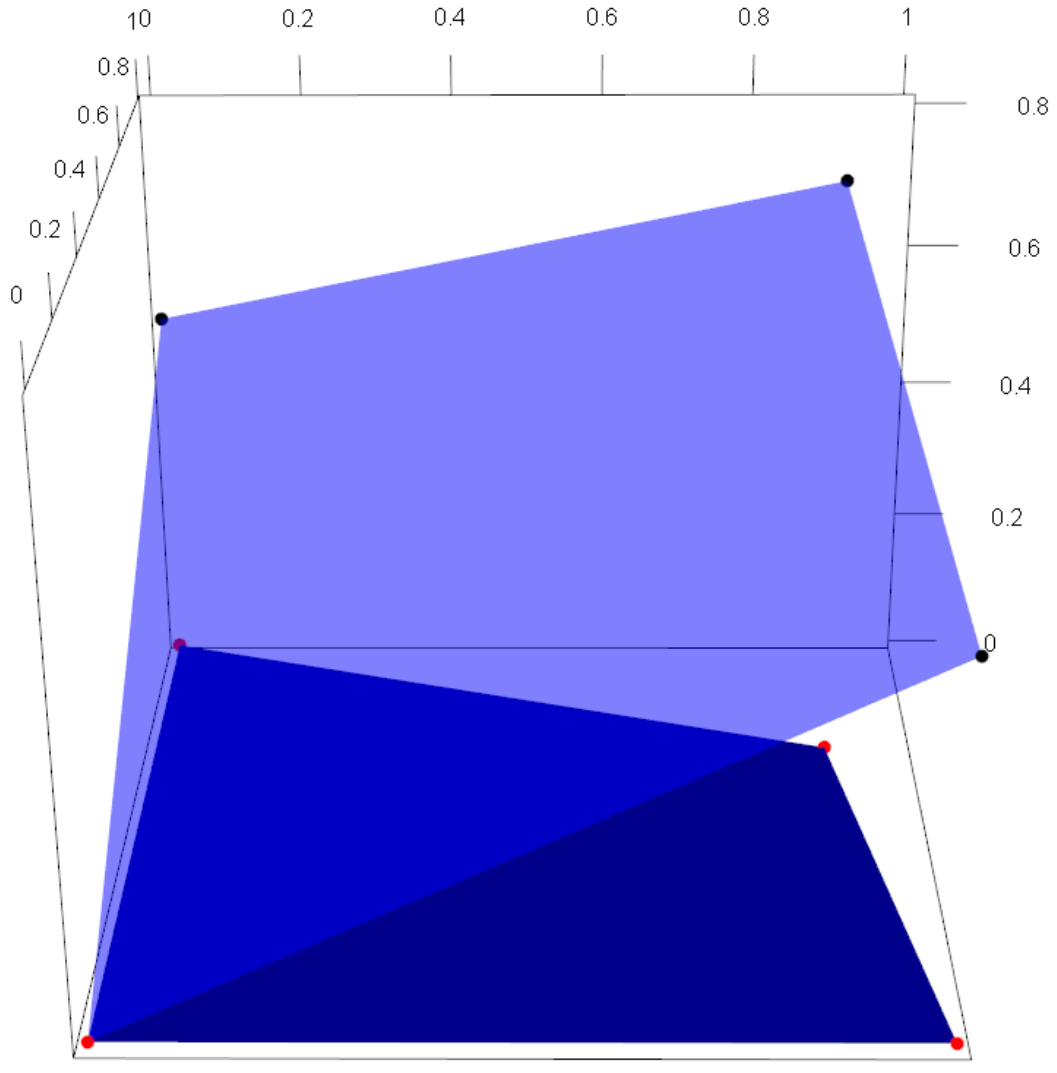
Code to plot example linear objective with linear constraints:

```
## Example of a linear objective with linear constraints
library(rgl)
x=c(0,  1, 0.9,  0,0)
y=c(0,  0, 0.7,  1,0)
z=c(0,0.5, 0.8,0.5,0)
polygon3d(x,y,z,lit=FALSE,col="blue",alpha=0.5,
          xlim=0:1, ylim=0:1, zlim=0:1)
polygon3d(x,y,z*0,lit=FALSE,col="darkblue")
points3d(x,y,rep(0,4),col="red",size=10)
points3d(x,y,z,col="black",size=10)
bbox3d(col="gray",alpha=0.5)
axes3d()
```

Example: Linear Programming



Example: Linear Programming



Gradient Methods

Examples: Broyden-Fletcher-Goldfarb-Shanno (BFGS) in R.

Step 1: Compute an approximate, or use-provided, Gradient (vector of partials).

Step 2: Compute an approximate, or use-provided, Hessian (matrix of 2nd order partials).

Step 3: Search along the line of steepest descent for a minimum.

Step 4: Choose a new point, and repeat.

Non-gradient Methods

Examples: Nelder-Mead in R.

Step 1: For a function of n variables, choose $n + 1$ nearby points to form a *simplex*.

Step 2: Reflect the "worst" point through the centroid of the remaining n points.

Step 3: Stretch in that direction if it's better, contract if worse.

Step 4: Repeat steps 2-3 for the new simplex.

Nelder-Mead Simplex Algorithm

http://en.wikipedia.org/wiki/Nelder%E2%80%93Mead_method#/media/File:Nelder_Mead2.gif

Which Method?

Analytical (general) results or Computational (limited) results?

How much do you know about your chosen objective surface?

How "smooth" is it?

How hard is it to compute objective function values?

How much computing power do you have?

Global or Local optimization?

See http://cran.r-project.org/web/views/Optimization.html

Good rule of thumb: Methods like Nelder-Mead, that assume little about the objective surface, tend to work well on a broad range of problems!