Nonlinear Problems with General Constraints

- We now need to deal with inequality constraints
  - We can’t convert all inequality constraints to equality constraints
  - We still have inequalities with the slacks and surpluses
- So, we will look at a more general problem:

$$\begin{aligned}
&\text{Problem NLP}: \\
&\text{max or min } \; f(x_1, x_2, \ldots x_n) \\
&\text{subject to} \\
&g_1(x_1, x_2, \ldots x_n) \le b_1 \\
&g_2(x_1, x_2, \ldots x_n) \le b_2 \\
&\vdots \\
&g_{\mathrm{m}}(x_1, x_2, \ldots x_n) \le b_m
\end{aligned}$$

OR

$$\begin{aligned}
&\text{Problem NLP}: \\
&\text{max or min } \; f(\mathbf{x}) \\
&\text{subject to} \\
&g(\mathbf{x}) \le \mathbf{b}
\end{aligned}$$

(vector form)

- Here, we convert all equalities to a pair of inequalities

The Karush-Kuhn-Tucker Conditions

- These are necessary and sufficient conditions for an optimal solution
  - Published by Kuhn and Tucker in 1951
  - Subsequently discovered that Karush had derived the conditions in his Master’s thesis in 1939
  - Winston disenfranchises Karush, but *I’m* asserting his contribution
  - Consequently, we’ll call these the *KKT* conditions
- Necessary versus sufficient
  - Necessary: have to meet these conditions, but meeting the conditions is not enough
  - Sufficient: if you meet these conditions, you’ve got it

KKT Necessary Conditions

- Warning: these are written differently in different texts
  - Winston does a good job, we'll stick with him
  - He also covers the variations
- If Problem NLP is a maximization, then the necessary conditions for a feasible ***x**** to be optimal are:

$$\nabla f(\mathbf{x}^*) - \sum_i \lambda_i \nabla g_i(\mathbf{x}^*) = \mathbf{0}$$

(vector form, n equations, n unknowns)

$$\lambda_i \left[b_i - g_i(\mathbf{x}^*)\right] = 0 \text{ for all } i$$

$$\lambda_i \geq 0 \text{ for all } i$$

- If NLP is a minimization, then:

$$\nabla f(\mathbf{x}^*) + \sum_i \lambda_i \nabla g_i(\mathbf{x}^*) = \mathbf{0}$$

(vector form)

$$\lambda_i \left[b_i - g_i(\mathbf{x}^*)\right] = 0 \text{ for all } i$$

$$\lambda_i \geq 0 \text{ for all } i$$

Example

- Try this one:

$$\min \; z = 2x_1^{\;2} + 2x_1x_2 + x_2^{\;2} - 10x_1 - 10x_2$$

subject to

$$x_1^{\;2} + x_2^{\;2} \le 5$$

$$3x_1 + x_2 \le 6$$

- Necessary conditions:

$$\overbrace{\boxed{\begin{matrix}4x_1 + 2x_2 - 10\\ 2x_1 + 2x_2 - 10\end{matrix}}}^{\nabla f(\mathbf{x})} \begin{matrix}+\\+\end{matrix} \overbrace{\boxed{\begin{matrix}2\lambda_1 x_1 + 3\lambda_2\\ 2\lambda_1 x_2 + \lambda_2\end{matrix}}}^{\sum_i \lambda_i \nabla g_i(\mathbf{x}^*)} \begin{matrix}=0\\=0\end{matrix}$$

$$\lambda_1\left(x_1^{\;2} + x_2^{\;2} - 5\right) = 0$$

$$\lambda_2\left(3x_1 + x_2 - 6\right) = 0$$

$$\lambda_1, \lambda_2 \ge 0$$

How Might We Solve This?

- Assume one of the constraints isn't binding
  - Try it with the second constraint
  - This means that $\lambda_2 = 0$
  - It also means the first constraint *is* binding
- This reduces the equations to:

$$4x_1 + 2x_2 - 10 + 2\lambda_1 x_1 = 0$$

$$2x_1 + 2x_2 - 10 + 2\lambda_1 x_2 = 0$$

$$x_1^2 + x_2^2 = 5$$

$$\lambda_1 \geq 0$$

- After much tedious algebra: $x_1 = 1$, $x_2 = 2$, $\lambda_1 = 1$
- How do we know if this is optimal?

KKT Sufficient Conditions

- These are Winston’s Theorem 11 and 11’ (p. 674)
  - If Problem NLP is a *maximization*, and ***f*** is concave, and all the ***g***’s are convex, and ***x**** satisfies the necessary conditions, ***x**** is optimal
  - If Problem NLP is a *minimization*, and ***f*** is convex, and all the ***g***’s are convex, and ***x**** satisfies the necessary conditions, ***x**** is optimal
- Example objective function:
  - $$H_f(x) = \begin{bmatrix} 4 & 2 \\ 2 & 4 \end{bmatrix}$$
  - Determinants of principal minors are 4, 4, 16-4 = 12, all > 0
  - Objective function is convex

Example Sufficient Conditions

- Testing the first constraint:
  - $$H_{g_1}(x) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$
  - Principal minors here are all > 0, so it's convex
- Third constraint is linear, so it is convex and concave
- Result: the point is optimal

Does Any of This Look Familiar?

- Suppose we have a typical LP (and its dual):

$$\begin{aligned} &\text{Problem LP}: \max cx \\ &\qquad \text{subject to} \\ &\qquad Ax \le b \; (w) \\ &\qquad -x \le 0 \; (u) \end{aligned} \qquad\qquad \begin{aligned} &\text{Problem LPD}: \min wb \\ &\qquad \text{subject to} \\ &\qquad wA - u \ge c \; (x) \\ &\qquad w, u \ge 0 \end{aligned}$$

- Let’s write the KKT necessary conditions for this:

$$\begin{aligned} c - wA + u &= 0 && \textbf{DUAL FEASIBILITY} \\ \left.\begin{aligned} w(b - Ax) &= 0 \\ ux &= 0 \end{aligned}\right\} & && \textbf{COMPLEMENTARY SLACKNESS} \\ w, u &\ge 0 \end{aligned}$$

Moral(s)

- KKT necessary conditions are identical to LP optimality conditions
- Since an LP has a linear objective and constraints, it meets the sufficient conditions as well
- Consequently, LP is just a subset of NLP
- The multipliers in the KKT conditions (and Lagrangian methods) correspond to dual variables in LP
- Most NLP solution techniques exploit dual variables in some way

A Couple More Warnings About MPL

- MPL dislikes parenthetical expressions with variables
  - Example of an equation MPL refuses to parse:

$$\max z = \sqrt{S\ (S - x_1)(S - x_2)(S - x_3)}$$

  - To get around this first recognize maximizing this is equivalent:

$$\max z' = S\ (S - x_1)(S - x_2)(S - x_3)$$

  - Since S is a positive constant, we can get rid the first one:

$$\max z'' = (S - x_1)(S - x_2)(S - x_3)$$

  - Finally, multiply out all the terms:

$$\max z'' = S^3 - S^2 x_1 - S^2 x_2 - S^2 x_3 + S x_1 x_2 + S x_1 x_3 + S x_2 x_3 + x_1 x_2 x_3$$

Final Proviso

- Note that the previous argument applied to objectives
- For constraints, you must be much more careful about determining equivalent forms
- In general, MPL forces you to multiply out nonlinear terms
- Other algebraic modeling language are NOT this restrictive

MPL/CONOPT Is Useful, Though

- Here’s the MPL code for the example:

```
OPTIONS
  MODELTYPE=nonlinear;

VARIABLES
  x1, x2;

MODEL
  min z = 2*(x1^2)+2*x1*x2+x2^2-10x1-10x2

SUBJECT TO

  x1^2 + x2^2 < 5;

  3*x1 + x2 < 6;

END
```

- In this case, CONOPT gets the optimal answer directly

Quadratic Programming

- One group of nonlinear optimizations is straightforward to solve and has useful applications
  - Quadratic objective function
  - Linear constraints
  - Can be solved by simplex method (with some modifications)
- The general model:

$$\min \quad z = cx + \frac{1}{2} xHx$$

$$\text{subject to}$$

$$Ax \leq b$$

$$x \geq 0$$

  - *H* must be positive semidefinite (negative semidefinite for a max problem)

Markowitz Mean-Variance Model

- This is a popular portfolio model
  - Have a collection of investments
  - Know their average historical return
  - Also know the variance and covariance of their returns
  - Objective is to minimize some combination of risk and return
- General model
  - Indices: i,j = possible investments
  - Data:
    - ***BUDGET*** = amount to invest
    - ***RETURN*** = desired average return at end of time horizon
    - RET_i = average return of investment i
    - COV_{ij} = covariance of return for asset i and j
  - Variables: x_i = amount to invest in asset i

Mean/Variance Model (cont'd)

- Formulation:

$$\min \quad z = \sum_{ij} COV_{ij} * x_i * x_j$$

$$\text{subject to}$$

$$\sum_i x_i = BUDGET$$

$$\sum_i RET_i * x_i \geq RETURN$$

$$x_i \geq 0 \text{ for all } i$$

- Some comments:
  - This minimizes variance for a specified return
  - A variance-covariance matrix is *always* positive-semidefinite, so you don't need to worry about that
  - Note that this minimizes variance in both directions (upside and downside)

Example

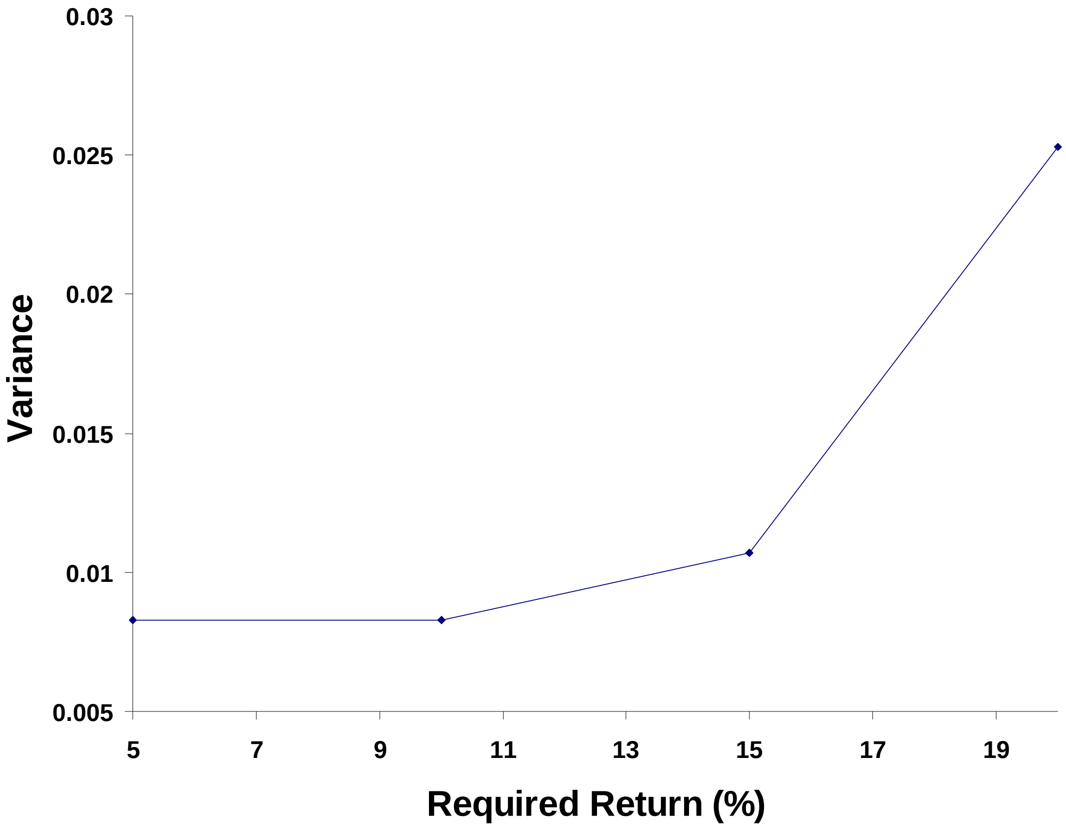
- From Markowitz (1959)
  - 3 stocks: ATT, GMC, USX
  - Average returns are 8.9%, 21.4%, and 23.5% respectively
  - Desired return: 15%
  - Covariance matrix:

	ATT	**GMC**	**USX**
ATT	0.01081	0.01241	0.01308
GMC	0.01241	0.05839	0.05543
USX	0.01308	0.05543	0.09423

- Let’s see what happens with various settings of desired return

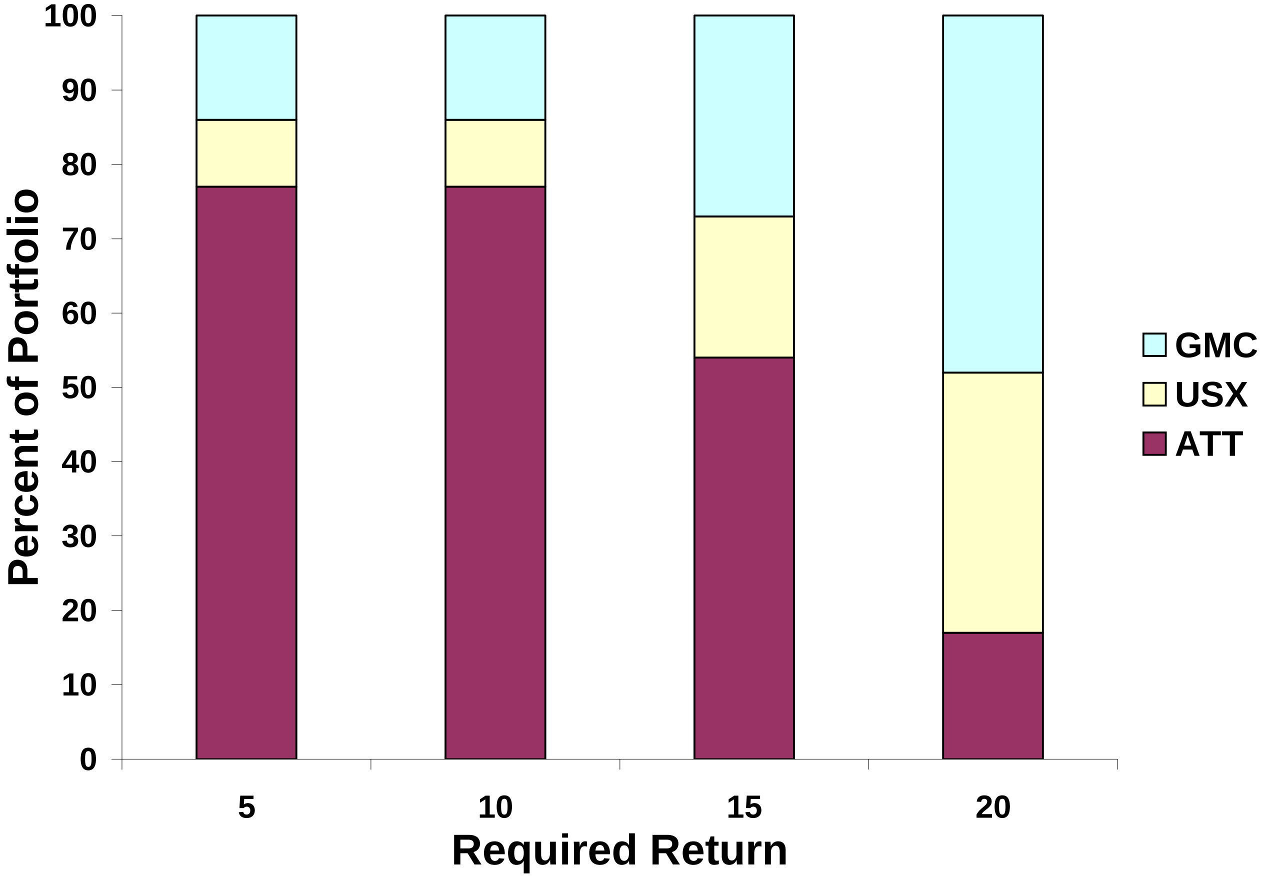
Results

- Chart below shows the trade between return and variation:



Results (cont'd)

- Here's how the mix changes:



Constrained Regression

- This is another application of quadratic programming
- In a normal linear regression problem:
  - You have a set of responses (***Y_i's***) and a set of ***j*** predictors (***X_{ij}'s***) for each ***Y_i***
  - You speculate the relationship is of the form:

$$Y_i = \beta_0 X_{i0} + \beta_1 X_{i1} + \cdots + \beta_n X_{in} + e_i$$

  - However, you have to estimate the ***β's*** (***e_i*** is random error)
- The classical statistical approach is to minimize the sum of the squared differences:

$$\min z = \sum_i \left(Y_i - b_0 X_{i0} - b_1 X_{i1} - \cdots - b_n X_{in}\right)^2$$
$$= \sum_i Y_i^2 - 2\sum_{ij} Y_i X_{ij} b_j + \sum_{ijk} X_{ij} X_{ik} b_j b_k$$

Constrained Regression (cont'd)

- Note the ***b's*** are the variables; ***X's*** and ***Y's*** are *data*
- Note also that it's equivalent to minimize:

$$\min z = -2\sum_{ij} Y_i X_{ij} b_j + \sum_{ijk} X_{ij} X_{ik} b_j b_k$$

- So why bother?
  - All spreadsheets and statistical packages do regression
  - The problem itself is an unconstrained quadratic optimization
  - We can solve it directly by differentiation
- This issue is that sometimes there may be constraints on the ***b's*** - for example:
  - Some must be nonnegative
  - Some must add to 1

Example

- Consider the following table:

Observation	***Xi0***	***Xi1***	***Xi2***	***Yi***
1	1	2	6	8
2	1	3	9	14
3	1	5	7	12
4	1	7	8	17
5	1	8	10	18

- We think the model is:

$$Y_i = \beta_0 X_{i0} + \beta_1 X_{i1} + \beta_2 X_{i2} + e_i$$

- Classical regression solves this and gets:
  - b_0 = -1.175
  - b_1 = 0.875
  - b_2 = 1.325

Now, Add a Constraint

- Suppose that the b's must add to > 1.5
- Can't use classical regression anymore
- However, we can just add a constraint to a quadratic optimization:

$$\min z = -2\sum_{ij} Y_i X_{ij} b_j + \sum_{ijk} X_{ij} X_{ik} b_j b_k$$

$$\text{subject to}$$

$$\sum_i b_j \geq 1.5$$

- In this case, the answer changes to:
  - $\boldsymbol{b_0}$ = -0.641
  - $\boldsymbol{b_1}$ = 0.891
  - $\boldsymbol{b_2}$ = 1.251

Solving Quadratic Programs in MPL

- Most LP solvers (like CPLEX) will solve quadratic programs
- Here’s the code for the Markowitz problem:

```
INDEX
  i := (att,gmc,usx);
  j := i;

OPTIONS
  modeltype = quadratic;

DATA
  RETURN = 1.20;
  RET[i] := (1.089,1.214,1.235);
  COV[i,j] := (0.01081,0.01241,0.01308,
               0.01241,0.05839,0.05543,
               0.01308,0.05543,0.09423);

VARIABLES
  x[i];
```

```
MODEL
  MIN variance =
      SUM(i,j: COV[i,j]*x[i]*x[i=j]);

SUBJECT TO

  budgetcon:
    SUM(i: x[i]) = 1;

  retcon:
    SUM(i: RET[i]*x[i]) > RETURN;

END
```