Nonlinear Programming and Process Optimization

Introduction

Unconstrained Optimization
- Algorithms
- Newton Methods
- Quasi-Newton Methods

Constrained Optimization
- Karush Kuhn-Tucker Conditions
- Special Classes of Optimization Problems
- Reduced Gradient Methods (GRG2, CONOPT, MINOS)
- Successive Quadratic Programming (SQP)
- Interior Point Methods (IPOPT)

Process Optimization
- Black Box Optimization
- Modular Flowsheet Optimization – Infeasible Path
- The Role of Exact Derivatives

Large-Scale Nonlinear Programming
- rSQP: Real-time Process Optimization
- IPOPT: Blending and Data Reconciliation

Further Applications
- Sensitivity Analysis for NLP Solutions
- Multi-Scenario Optimization Problems

Summary and Conclusions
Introduction

Optimization: given a system or process, find the best solution to this process within constraints.

Objective Function: indicator of "goodness" of solution, e.g., cost, yield, profit, etc.

Decision Variables: variables that influence process behavior and can be adjusted for optimization.

In many cases, this task is done by trial and error (through case study). Here, we are interested in a systematic approach to this task - and to make this task as efficient as possible.

Some related areas:
- Math programming
- Operations Research

Currently - Over 30 journals devoted to optimization with roughly 200 papers/month - a fast moving field!

Optimization Viewpoints

Mathematician - characterization of theoretical properties of optimization, convergence, existence, local convergence rates.

Numerical Analyst - implementation of optimization method for efficient and "practical" use. Concerned with ease of computations, numerical stability, performance.

Engineer - applies optimization method to real problems. Concerned with reliability, robustness, efficiency, diagnosis, and recovery from failure.
Optimization Literature

**Engineering**


**Numerical Analysis**


---

**Motivation**

**Scope of optimization**

Provide *systematic framework* for searching among a specified *space of alternatives* to identify an “optimal” design, i.e., as a *decision-making tool*

**Premise**

Conceptual formulation of optimal product and process design corresponds to a *mathematical programming problem*
### Optimization in Design, Operations and Control

<table>
<thead>
<tr>
<th>MILP</th>
<th>MINLP</th>
<th>Global</th>
<th>LP/QP</th>
<th>NLP</th>
<th>SA/GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>HENS</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>MENS</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Separations</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reactors</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Equipment Design</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Flowsheeting</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scheduling</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Supply Chain</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Real-time optimization</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Linear MPC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Nonlinear MPC</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hybrid</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Unconstrained Multivariable Optimization

**Problem:** \[ \text{Min } f(x) \quad (n \text{ variables}) \]

Equivalent to: \[ \text{Max } -f(x), \ x \in \mathbb{R}^n \]

**Nonsmooth Functions**
- Direct Search Methods
- Statistical/Random Methods

**Smooth Functions**
- 1st Order Methods
- *Newton Type Methods*
- Conjugate Gradients
Example: Optimal Vessel Dimensions

What is the optimal L/D ratio for a cylindrical vessel?

Constrained Problem

\[
\text{Min } \left\{ C_t \frac{\pi D^2}{2} + C_s \frac{4V}{D} \right\} = \text{cost}
\]

\[
\frac{d(\text{cost})}{dD} = C_t \pi D - \frac{4VC_s}{D^2} = 0
\]

\[
D = \left( \frac{4V}{\pi C_t} \right)^{1/3} \quad \Rightarrow \quad L = \left( \frac{4V}{\pi C_t} \right)^{1/3} \left( \frac{C_t}{C_s} \right)^{2/3}
\]

\[\Rightarrow \text{L/D} = \frac{C_t}{C_s}\]

Note:
- What if L cannot be eliminated in (1) explicitly? (strange shape)
- What if D cannot be extracted from (2)? (cost correlation implicit)

Two Dimensional Contours of F(x)

Convex Function    Nonconvex Function    Multimodal, Nonconvex

Discontinuous    Nondifferentiable (convex)
Local vs. Global Solutions

Convexity Definitions

• A set (region) X is convex, if and only if it satisfies:
  \[ \alpha y + (1-\alpha)z \in X \]
  for all \( \alpha, 0 \leq \alpha \leq 1 \), for all points y and z in X.

• \( f(x) \) is convex in domain X, if and only if it satisfies:
  \[ f(\alpha y + (1-\alpha)z) \leq \alpha f(y) + (1-\alpha)f(z) \]
  for any \( \alpha, 0 \leq \alpha \leq 1 \), at all points y and z in X.

• Find a local minimum point \( x^* \) for \( f(x) \) for feasible region defined by constraint functions:
  \[ f(x^*) \leq f(x) \text{ for all } x \text{ satisfying the constraints in some neighborhood around } x^* \]
  (not for all \( x \in X \))

• Sufficient condition for a local solution to the NLP to be a global is that \( f(x) \) is convex for \( x \in X \).

• Finding and verifying global solutions will not be considered here.

• Requires a more expensive search (e.g. spatial branch and bound).

Linear Algebra - Background

Some Definitions

• Scalars - Greek letters, \( \alpha, \beta, \gamma \)
• Vectors - Roman Letters, lower case
• Matrices - Roman Letters, upper case
• Matrix Multiplication:
  \[ C = AB \text{ if } A \in \mathbb{R}^{n \times m}, B \in \mathbb{R}^{m \times p} \text{ and } C \in \mathbb{R}^{n \times p}, C_{ij} = \sum_k A_{ik} B_{kj} \]
• Transpose - if \( A \in \mathbb{R}^{n \times m} \), interchange rows and columns --> \( A^T \in \mathbb{R}^{m \times n} \)
• Symmetric Matrix - \( A \in \mathbb{R}^{n \times n} \) (square matrix) and \( A = A^T \)
• Identity Matrix - \( I \), square matrix with ones on diagonal and zeroes elsewhere.
• Determinant: "Inverse Volume" measure of a square matrix
  \[ \det(A) = \sum_i (-1)^{i+j} A_{ij} A_{ji} \text{ for any } j, \text{ or} \]
  \[ \det(A) = \sum_j (-1)^{i+j} A_{ij} A_{ji} \text{ for any } i, \text{ where } A_{ij} \text{ is the determinant of an order } n-1 \text{ matrix with row } i \text{ and column } j \text{ removed.} \]
  \[ \det(I) = 1 \]
• Singular Matrix: \( \det(A) = 0 \)
### Linear Algebra - Background

**Gradient Vector** - $(\nabla f(x))$

$$\nabla f = \left[ \frac{\partial f}{\partial x_1} \right] \left[ \frac{\partial f}{\partial x_2} \right] \ldots \left[ \frac{\partial f}{\partial x_n} \right]$$

**Hessian Matrix** - $(\nabla^2 f(x))$ - Symmetric

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \ldots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \ldots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \ldots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Note: \(\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i}\)

---

### Linear Algebra - Background

- **Some Identities for Determinant**
  \(\text{det}(A \cdot B) = \text{det}(A) \cdot \text{det}(B); \quad \text{det}(A) = \text{det}(A^T)\)
  \(\text{det}(\alpha A) = \alpha^n \cdot \text{det}(A); \quad \text{det}(A) = \prod_i \lambda_i(A)\)

- **Eigenvalues**: \(\text{det}(A - \lambda I) = 0\), **Eigenvector**: \(Av = \lambda v\)
  Characteristic values and directions of a matrix.
  For nonsymmetric matrices eigenvalues can be complex, so we often use **singular values**, \(\sigma = \lambda(A^TA)^{1/2} \geq 0\)

- **Vector Norms**
  \(\|x\|_p = \left\{ \sum_i |x_i|^p \right\}^{1/p}\)
  (Most common are \(p = 1, p = 2\) (Euclidean) and \(p = \infty\) (max norm = max \(|x_i|\)))

- **Matrix Norms**
  \(\|A\| = \max \|Ax\|/\|x\|\) over \(x\) (for \(p\)-norms)
  \(\|A\|_\infty - \max \text{column sum of } A, \max_j (\sum_i |A_{ij}|)\)
  \(\|A\|_1 - \max \text{maximum row sum of } A, \max_i (\sum_j |A_{ij}|)\)
  \(\|A\|_2 = [\sigma_{\text{max}}(A)]\) (spectral radius)
  \(\|A\|_F = [\sum_i \sum_j (A_{ij})^2]^{1/2}\) (Frobenius norm)
  \(\kappa(A) = \|A\| \|A^{-1}\|\) (condition number) = \(\sigma_{\text{max}}/\sigma_{\text{min}}\) (using 2-norm)
Linear Algebra - Eigenvalues

Find $v$ and $\lambda$, where $Av_i = \lambda_i v_i$, $i = 1, n$

Note: $Av = \lambda v = (A - \lambda I) v = 0$ or $\det (A - \lambda I) = 0$

For this relation $\lambda$ is an eigenvalue and $v$ is an eigenvector of $A$.

If $A$ is symmetric, all $\lambda_i$ are real

- $\lambda_i > 0$, $i = 1, n$: $A$ is positive definite
- $\lambda_i < 0$, $i = 1, n$: $A$ is negative definite
- $\lambda_i = 0$, some $i$: $A$ is singular

Quadratic Form can be expressed in Canonical Form (Eigenvalue/Eigenvector)

$$x^T Ax \Rightarrow AV = V \Lambda$$

$V$ - eigenvector matrix $(n \times n)$

$\Lambda$ - eigenvalue (diagonal) matrix $= \text{diag}(\lambda_i)$

If $A$ is symmetric, all $\lambda_i$ are real and $V$ can be chosen orthonormal $(V^{-1} = V^T)$.

Thus, $A = V \Lambda V^{-1} = V \Lambda V^T$

For Quadratic Function: $Q(x) = a^T x + \frac{1}{2} x^T Ax$

Define: $z = V^T x$ and $Q(Vz) = (a^T V) z + \frac{1}{2} z^T (V^T AV) z$

$$= (a^T V) z + \frac{1}{2} z^T \Lambda z$$

Minimum occurs at (if $\lambda_i > 0$) $x = -A^{-1} a$ or $x = Vz = -V(\Lambda^{-1} V^T a)$

Positive (Negative) Curvature

Positive (Negative) Definite Hessian

Both eigenvalues are strictly positive (negative)

- $A$ is positive (negative) definite
- Stationary points are minima (maxima)
Zero Curvature
Singular Hessian

One eigenvalue is zero, the other is strictly positive or negative
- A is positive semidefinite or negative semidefinite
- There is a ridge of stationary points (minima or maxima)

Indefinite Curvature
Indefinite Hessian

One eigenvalue is positive, the other is negative
- Stationary point is a saddle point
- A is indefinite

Note: these can also be viewed as two dimensional projections for higher dimensional problems
Eigenvalue Example

\[
\begin{align*}
\text{Min } Q(x) &= \begin{bmatrix} 1^T \end{bmatrix}^T x^T + \frac{1}{2} x^T \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} x \\
AV &= VA \quad \text{with } \Lambda = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \\
V^T AV &= \Lambda = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \quad \text{with } V = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}
\end{align*}
\]

- All eigenvalues are positive
- Minimum occurs at \( z^* = -A^{-1}V^Ta \)

\[
\begin{align*}
z &= V^T x = \begin{bmatrix} (x_1 - x_2)/\sqrt{2} \\ (x_1 + x_2)/\sqrt{2} \end{bmatrix} \\
x &= Vz = \begin{bmatrix} (x_1 + x_2)/\sqrt{2} \\ (-x_1 + x_2)/\sqrt{2} \end{bmatrix} \\
z^* = \begin{bmatrix} 0 \\ -2/(3\sqrt{2}) \end{bmatrix} \\
x^* = \begin{bmatrix} -1/3 \\ -1/3 \end{bmatrix}
\end{align*}
\]

Comparison of Optimization Methods

1. Convergence Theory
   - Global Convergence - will it converge to a local optimum (or stationary point) from a poor starting point?
   - Local Convergence Rate - how fast will it converge close to this point?

2. Benchmarks on Large Class of Test Problems

Representative Problem (Hughes, 1981)

\[
\begin{align*}
\text{Min } f(x_1, x_2) &= \alpha \exp(-\beta) \\
u &= x_1 - 0.8 \\
v &= x_2 - (a_1 + a_2 u^2 (1 - u)^{1/2} - a_3 u) \\
\alpha &= -b_1 + b_2 u^2 (1 + u)^{1/2} + b_3 u \\
\beta &= c_1 v^2 (1 - c_2 v)/(1 + c_3 u^2)
\end{align*}
\]

\[
\begin{align*}
a &= [0.3, 0.6, 0.2] \\
b &= [5, 26, 3] \\
c &= [40, 1, 10] \\
x^* &= [0.7395, 0.3144] \quad f(x^*) = -5.0893
\end{align*}
\]
Regions where minimum eigenvalue is greater than: 
\[ [0, -10, -50, -100, -150, -200] \]

What conditions characterize an optimal solution?

**Unconstrained Local Minimum**

**Necessary Conditions**
\[ \nabla f(x^*) = 0 \]
\[ p^T\nabla^2 f(x^*) p \geq 0 \quad \text{for} \ p \in \mathbb{R}^n \]
(positive semi-definite)

**Sufficient Conditions**
\[ \nabla f(x^*) = 0 \]
\[ p^T\nabla^2 f(x^*) p > 0 \quad \text{for} \ p \in \mathbb{R}^n \]
(positive definite)

For smooth functions, why are contours around optimum elliptical?

**Taylor Series** in n dimensions about \( x^* \):
\[
f(x) = f(x^*) + \nabla f(x^*)^T (x - x^*) + \frac{1}{2} (x - x^*)^T \nabla^2 f(x^*)(x - x^*) + O\left(\|x - x^*\|^3\right)
\]

Since \( \nabla f(x^*) = 0 \), \( f(x) \) is purely quadratic for \( x \) close to \( x^* \)
Newton's Method

Taylor Series for $f(x)$ about $x^k$

Take derivative wrt $x$, set LHS $\approx 0$

$$0 \approx \nabla f(x) = \nabla f(x^k) + \nabla^2 f(x^k) (x - x^k) + O(||x - x^k||^2)$$

$$\Rightarrow (x - x^k) \equiv d = - (\nabla^2 f(x^k))^{-1} \nabla f(x^k)$$

- $f(x)$ is convex (concave) if for all $x \in \mathbb{R}^n$, $\nabla^2 f(x)$ is positive (negative) semidefinite
  - i.e. $\min \lambda_j \geq 0$ (max $\lambda_j \leq 0$)
- Method can fail if:
  - $x^0$ far from optimum
  - $\nabla^2 f$ is singular at any point
  - $f(x)$ is not smooth
- Search direction, $d$, requires solution of linear equations.
- Near solution:
  $$||x^{k+1} - x^*|| = O(||x^k - x^*||^2)$$

Basic Newton Algorithm - Line Search

0. Guess $x^0$, Evaluate $f(x^0)$.

1. At $x^k$, evaluate $\nabla f(x^k)$.

2. Evaluate $B^k = \nabla^2 f(x^k)$ or an approximation.

3. Solve: $B^k d = -\nabla f(x^k)$
   If convergence error is less than tolerance:
   e.g., $||\nabla f(x^k)|| \leq \varepsilon$ and $||d|| \leq \varepsilon$ STOP, else go to 4.

4. Find $\alpha$ so that $0 < \alpha \leq 1$ and $f(x^k + \alpha d) < f(x^k)$ sufficiently (Each trial requires evaluation of $f(x)$)

5. $x^{k+1} = x^k + \alpha d$. Set $k = k + 1$ Go to 1.
Newton's Method - Convergence Path

Starting Points

[0.8, 0.2] needs steepest descent steps w/ line search up to 'O', takes 7 iterations to $\|\nabla f(x^*)\| \leq 10^{-6}$

[0.35, 0.65] converges in four iterations with full steps to $\|\nabla f(x^*)\| \leq 10^{-6}$

Newton's Method - Notes

- Choice of $B^k$ determines method.
  - Steepest Descent: $B^k = \gamma I$
  - Newton: $B^k = \nabla^2 f(x)$
- With suitable $B^k$, performance may be good enough if $f(x^k + \alpha d)$ is sufficiently decreased (instead of minimized along line search direction).
- Trust region extensions to Newton's method provide very strong global convergence properties and very reliable algorithms.
- Local rate of convergence depends on choice of $B^k$.

\[
\text{Newton – Quadratic Rate : } \lim_{k \to \infty} \frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|^2} = K
\]

\[
\text{Steepest descent – Linear Rate : } \lim_{k \to \infty} \frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|} < 1
\]

\[
\text{Desired? – Superlinear Rate : } \lim_{k \to \infty} \frac{\|x^{k+1} - x^*\|}{\|x^k - x^*\|} = 0
\]
Quasi-Newton Methods

Motivation:
• Need $B^k$ to be positive definite.
• Avoid calculation of $\nabla^2 f$.
• Avoid solution of linear system for $d = -(B^k)^{-1} \nabla f(x^k)$

Strategy: Define matrix updating formulas that give $(B^k)$ symmetric, positive definite and satisfy:

$$(B^{k+1})(x^{k+1} - x^k) = (\nabla f^{k+1} - \nabla f^k) \quad \text{(Secant relation)}$$

DFP Formula: (Davidon, Fletcher, Powell, 1958, 1964)

$$B^{k+1} = B^k + \frac{(y - B^k s)y^T + y (y - B^k s)^T}{y^T s} - \frac{(y - B^k s)^T s y y^T}{y^T s (y^T s)}$$

$$(B^{k+1})^{-1} = H^{k+1} = H^k + \frac{ss^T}{s^T y} - \frac{H^k y y^T H^k}{y^T H^k y}$$

where:
$$s = x^{k+1} - x^k$$
$$y = \nabla f(x^{k+1}) - \nabla f(x^k)$$

BFGS Formula (Broyden, Fletcher, Goldfarb, Shanno, 1970-71)

$$B^{k+1} = B^k + \frac{yy^T}{s^T y} - \frac{B^k s s^T B^k}{s^T B^k s}$$

$$(B^{k+1})^{-1} = H^{k+1} = H^k + \frac{(s - H^k y)s^T + s (s - H^k y)^T}{y^T s} - \frac{(y - H^k s)^T y s y^T}{(y^T s) (y^T s)}$$

Notes:
1) Both formulas are derived under similar assumptions and have symmetry
2) Both have superlinear convergence and terminate in $n$ steps on quadratic functions. They are identical if $\alpha$ is minimized.
3) BFGS is more stable and performs better than DFP, in general.
4) For $n \leq 100$, these are the best methods for general purpose problems if second derivatives are not available.
Quasi-Newton Method - BFGS Convergence Path

Starting Point
[0.2, 0.8] starting from $B^0 = I$, converges in 9 iterations to $\|\nabla f(x^*)\| \leq 10^{-6}$

Sources For Unconstrained Software

- Harwell (HSL)
- IMSL
- NAg - Unconstrained Optimization Codes
- Netlib (www.netlib.org)
  - MINPACK
  - TOMS Algorithms, etc.

These sources contain various methods
  - Quasi-Newton
  - Gauss-Newton
  - Sparse Newton
  - Conjugate Gradient
Constrained Optimization
(Nonlinear Programming)

Problem: \[ \text{Min} \ f(x) \]
\[ \text{s.t.} \ g(x) \leq 0 \]
\[ h(x) = 0 \]

where:
- \( f(x) \) - scalar objective function
- \( x \) - \( n \) vector of variables
- \( g(x) \) - inequality constraints, \( m \) vector
- \( h(x) \) - \( meq \) equality constraints.

Sufficient Condition for Global Optimum
- \( f(x) \) must be convex, and
- feasible region must be convex,
  i.e. \( g(x) \) are all convex
  \( h(x) \) are all linear

Except in special cases, there is no guarantee that a local optimum is global
if sufficient conditions are violated.

Example: Minimize Packing Dimensions

What is the smallest box for three round objects?
Variables: \( A, B, (x_1, y_1), (x_2, y_2), (x_3, y_3) \)
Fixed Parameters: \( R_1, R_2, R_3 \)
Objective: Minimize Perimeter = \( 2(A+B) \)
Constraints: Circles remain in box, can't overlap
Decisions: Sides of box, centers of circles.

\[
\begin{align*}
  x_1, y_1 & \leq R_1 \\
  x_2, y_2 & \leq R_2 \\
  x_3, y_3 & \leq R_3 \\
  x_1 & \leq B - R_1, \ y_1 \leq A - R_1 \\
  x_2 & \leq B - R_2, \ y_2 \leq A - R_2 \\
  x_3 & \leq B - R_3, \ y_3 \leq A - R_3 \\
  (x_1 - x_2)^2 + (y_1 - y_2)^2 & \geq (R_1 + R_2)^2 \\
  (x_1 - x_3)^2 + (y_1 - y_3)^2 & \geq (R_1 + R_3)^2 \\
  (x_2 - x_3)^2 + (y_2 - y_3)^2 & \geq (R_2 + R_3)^2 \\
  \text{no overlaps}
\end{align*}
\]
Characterization of Constrained Optima

What conditions characterize an optimal solution?

Unconstrained Local Minimum
Necessary Conditions
\[ \forall f(x^*) = 0 \]
\[ p^T \nabla^2 f(x^*) p \geq 0 \quad \text{for} \quad p \in \mathbb{R}^n \]
(positive semi-definite)

Unconstrained Local Minimum
Sufficient Conditions
\[ \forall f(x^*) = 0 \]
\[ p^T \nabla^2 f(x^*) p > 0 \quad \text{for} \quad p \in \mathbb{R}^n \]
(positive definite)
Optimal solution for inequality constrained problem

\[ \text{Min} \quad f(x) \]
\[ \text{s.t.} \quad g(x) \leq 0 \]

Analogy: Ball rolling down valley pinned by fence
Note: Balance of forces (\( \nabla f, \nabla g \))

Optimal solution for general constrained problem

Problem: Min \( f(x) \)
\[ \text{s.t.} \quad g(x) \leq 0 \]
\[ h(x) = 0 \]

Analogy: Ball rolling on rail pinned by fences
Balance of forces: \( \nabla f, \nabla g_1, \nabla h \)
Necessary First Order Karush Kuhn - Tucker Conditions

\[ \nabla L(x^*, u, v) = \nabla f(x^*) + \nabla g(x^*) u + \nabla h(x^*) v = 0 \]

(Balance of Forces)

\[ u \geq 0 \] (Inequalities act in only one direction)

\[ g(x^*) \leq 0, \quad h(x^*) = 0 \] (Feasibility)

\[ u_i g_i(x^*) = 0 \] (Complementarity: either \( g_i(x^*) = 0 \) or \( u_i = 0 \))

\( u, v \) are "weights" for "forces," known as KKT multipliers, shadow prices, dual variables

“To guarantee that a local NLP solution satisfies KKT conditions, a constraint qualification is required. E.g., the Linear Independence Constraint Qualification (LICQ) requires active constraint gradients, \( [\nabla g_i(x^*) \nabla h(x^*)] \), to be linearly independent. Also, under LICQ, KKT multipliers are uniquely determined.”

Necessary (Sufficient) Second Order Conditions

- Positive curvature in "constraint" directions.
- \( p^T \nabla^2 L(x^*) p \geq 0 \) (\( p^T \nabla^2 L(x^*) p > 0 \))
  where \( p \) are the constrained directions: \( \nabla h(x^*)^T p = 0 \)
  for \( g_i(x^*)=0, \, \nabla g_i(x^*)^T p = 0, \) for \( u_i > 0, \, \nabla g_i(x^*)^T p \leq 0, \) for \( u_i = 0 \)

Single Variable Example of KKT Conditions

Min \( (x)^2 \) s.t. \(-a \leq x \leq a, \, a > 0 \)

\( x^* = 0 \) is seen by inspection

Lagrange function :

\[ L(x, u) = x^2 + u_f(x-a) + u_2(-a-x) \]

First Order KKT conditions:

\[ \nabla L(x, u) = 2 x + u_f(x-a) = 0 \]

\[ u_f (x-a) = 0 \]

\[ u_2 (-a-x) = 0 \]

\[-a \leq x \leq a, \quad u_f, u_2 \geq 0 \]

Consider three cases:

- \( u_f \geq 0, \, u_2 = 0 \) Upper bound is active, \( x = a, \, u_f = -2a, \, u_2 = 0 \)
- \( u_f = 0, \, u_2 \geq 0 \) Lower bound is active, \( x = -a, \, u_f = -2a, \, u_2 = 0 \)
- \( u_f = u_2 = 0 \) Neither bound is active, \( u_f = 0, \, u_2 = 0, \, x = 0 \)

Second order conditions \( (x^*, u_f, u_2 =0) \)

\[ \nabla^2 L(x^*, u^*) = 2 \]

\[ p^T \nabla^2 L(x^*, u^*) p = 2 (\Delta x)^2 > 0 \]
Single Variable Example of KKT Conditions - Revisited

Min \(-x^2\) s.t. \(-a \leq x \leq a, a > 0\)

\(x^* = \pm a\) is seen by inspection

Lagrange function:

\[ L(x, u) = -x^2 + u_1(x-a) + u_2(-a-x) \]

First Order KKT conditions:

\[ \nabla L(x, u) = -2x + u_1 - u_2 = 0 \]
\[ u_1(x-a) = 0 \]
\[ u_2(-a-x) = 0 \]
\[-a \leq x \leq a \quad u_1, u_2 \geq 0 \]

Consider three cases:

- \(u_1 \geq 0, u_2 = 0\) Upper bound is active, \(x = a, u_1 = 2a, u_2 = 0\)
- \(u_1 = 0, u_2 \geq 0\) Lower bound is active, \(x = -a, u_2 = 2a, u_1 = 0\)
- \(u_1 = u_2 = 0\) Neither bound is active, \(u_1 = 0, u_2 = 0, x = 0\)

Second order conditions \((x^*, u_1, u_2 = 0)\)

\[ \nabla_x^2 L(x^*, u^*) = -2 \]
\[ p^T \nabla_x^2 L(x^*, u^*) p = -2(\Delta x)^2 < 0 \]

Interpretation of Second Order Conditions

For \(x = a\) or \(x = -a\), we require the allowable direction to satisfy the active constraints exactly. Here, any point along the allowable direction, \(x^*\) must remain at its bound.

For this problem, however, there are no nonzero allowable directions that satisfy this condition. Consequently the solution \(x^*\) is defined entirely by the active constraint. The condition:

\[ p^T \nabla_x^2 L(x^*, u^*, v^*) p > 0 \]

for the allowable directions, is vacuously satisfied - because there are no allowable directions that satisfy \(\nabla g(x^*)^T p = 0\). Hence, sufficient second order conditions are satisfied.

As we will see, sufficient second order conditions are satisfied by linear programs as well.
Role of KKT Multipliers

Also known as:
- Shadow Prices
- Dual Variables
- Lagrange Multipliers

Suppose $a$ in the constraint is increased to $a + \Delta a$

$$f(x^*) = -(a + \Delta a)^2$$

and

$$\frac{f(x^*, a + \Delta a) - f(x^*, a)}{\Delta a} = -2a - \Delta a$$

$$\frac{df(x^*)}{da} = -2a = -u_i$$

Another Example: Constraint Qualifications

$$\text{Min } x_1$$

s.t. $x_2 \geq 0$

$x_2 \leq (x_1)^3$

$x_1^* = x_2^* = 0$

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & -3(x_1)^2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \neq 0$$

$-x_2 \leq 0, u_1 \geq 0, u_1 x_2 = 0$

$x_2 - (x_1)^3 \leq 0, u_2 \geq 0, u_2 (x_2 - (x_1)^3) = 0$

**KKT conditions not satisfied at NLP solution**

*Because a CQ is not satisfied (e.g., LICQ)*
Linear Programming:
\[
\text{Min } c^T x \\
\text{s.t. } A x \leq b \\
\quad C x = d, \quad x \geq 0
\]
Functions are all convex \( \Rightarrow \) global min.
Because of Linearity, can prove solution will always lie at vertex of feasible region.

Simplex Method
- Start at vertex
- Move to adjacent vertex that offers most improvement
- Continue until no further improvement

Notes:
1) LP has wide uses in planning, blending and scheduling
2) Canned programs widely available.

Linear Programming Example

Simplex Method
\[
\text{Min } -2x_1 - 3x_2 \\
\text{s.t. } 2x_1 + x_2 \leq 5 \\
\quad x_1, x_2 \geq 0
\]
Now, define \( f = -2x_1 - 3x_2 \) \( \Rightarrow \) \( f + 2x_1 + 3x_2 = 0 \)
Set \( x_1, x_2 = 0, \ x_3 = 5 \) and form tableau

\[
\begin{array}{cccc|c}
 x_1 & x_2 & x_3 & f & b \\
 2 & 1 & 1 & 0 & 5 \\
 2 & 3 & 0 & 1 & 0 \\
\end{array}
\]

To decrease \( f \), increase \( x_2 \). How much? so \( x_3 \geq 0 \)

\[
\begin{array}{cccc|c}
 x_1 & x_2 & x_3 & f & b \\
 2 & 1 & 1 & 0 & 5 \\
 -4 & 0 & -3 & 1 & -15 \\
\end{array}
\]

\( f \) can no longer be decreased! \textit{Optimal}

Underlined terms are -(reduced gradients); nonbasic variables \( (x_1, x_3) \), basic variable \( x_2 \)
Quadratic Programming

Problem: \[ \text{Min} \quad a^T x + \frac{1}{2} x^T B x \]
\[ \text{s.t.} \quad A x \leq b \]
\[ \quad C x = d \]
1) Can be solved using LP-like techniques:
   (Wolfe, 1959)
   \[ \Rightarrow \quad \text{Min} \quad \sum_j (z_j^+ + z_j^-) \]
   \[ \text{s.t.} \quad a + B x + A^T u + C^T v = z^+ - z^- \]
   \[ \quad A x - b + s = 0 \]
   \[ \quad C x - d = 0 \]
   \[ \quad u, s, z^+, z^- \geq 0 \]
   \[ \{u, s, z^+ = 0 \} \]
   with complicating conditions.

2) If B is positive definite, QP solution is unique.
   If B is pos. semidefinite, optimum value is unique.

3) Other methods for solving QP's (faster)
   - Complementary Pivoting (Lemke)
   - Range, Null Space methods (Gill, Murray).

Portfolio Planning Problem

Definitions:
- \( x_i \) - fraction or amount invested in security i
- \( r_i(t) \) - (1 + rate of return) for investment i in year t.
- \( \mu_i \) - average \( r(t) \) over T years, i.e.
  \[ \mu_i = \frac{1}{T} \sum_{t=1}^{T} r_i(t) \]

\[ \text{Max} \quad \sum_i \mu_i x_i \]
\[ \text{s.t.} \quad \sum_i x_i = 1 \]
\[ x_i \geq 0, \text{ etc.} \]

Note: maximize average return, no accounting for risk.
Portfolio Planning Problem

Definition of Risk - fluctuation of \( r_i(t) \) over investment (or past) time period.
To minimize risk, minimize variance about portfolio mean (risk averse).

Variance/Covariance Matrix, \( S \)

\[
\{ S \}_{ij} = \sigma_{ij}^2 = \frac{1}{T} \sum_{t=1}^{T} \left( r_i(t) - \mu_i \right) \left( r_j(t) - \mu_j \right)
\]

\[
\text{Min } x^T S x
\]

s.t. \( \sum x_i = 1 \)

\[
\sum \mu_i x_i \geq R
\]

\( x_i \geq 0, \text{ etc.} \)

Example: 3 investments

1. IBM \( \mu_1 = 1.3 \)
2. GM \( \mu_2 = 1.2 \)
3. Gold \( \mu_3 = 1.08 \)

\( S = \begin{bmatrix} 3 & 1 & -0.5 \\ 1 & 2 & 0.4 \\ -0.5 & 0.4 & 1 \end{bmatrix} \)

Portfolio Planning Problem - GAMS

```plaintext
SIMPLE PORTFOLIO INVESTMENT PROBLEM (MARKOWITZ)

OPTION LIMROW=0;
OPTION LIMXOL=0;
VARIABLES IBM, GM, GOLD, OBJQP, OBJLP;
EQUATIONS E1,E2,QP,LP;
LP.. OBJLP =E= 1.3*IBM + 1.2*GM + 1.08*GOLD;
QP.. OBJQP =E= 3*IBM**2 + 2*IBM*GM - IBM*GOLD + 2*GM**2 - 0.8*GM*GOLD + GOLD**2;
E1..1.3*IBM + 1.2*GM + 1.08*GOLD =G= 1.15;
E2.. IBM + GM + GOLD =E= 1;
IBM.LO = 0.;
IBM.UP = 0.75;
GM.LO = 0.;
GM.UP = 0.75;
GOLD.LO = 0.;
GOLD.UP = 0.75;
MODEL PORTQP/QP,E1,E2;#
MODEL PORTLP/LP,E2;#
SOLVE PORTLP USING LP MAXIMIZING OBJLP;
SOLVE PORTQP USING NLP MINIMIZING OBJQP;
```

47
Portfolio Planning Problem - GAMS

SOLVE SUMMARY
***** MODEL STATUS = 1 OPTIMAL
***** OBJECTIVE VALUE = 1.2750
RESOURCE USAGE, LIMIT = 1.270 1000.000
ITERATION COUNT, LIMIT = 1 1000
BDM - LP VERSION 1.01
A. Brooke, A. Drud, and A. Meeraus,
Analytic Support Unit,
Development Research Department,
World Bank,
Washington D.C. 20433, U.S.A.

EXECUTION TIME           =         3.510 SECONDS
GENERATION TIME
CODE LENGTH
DERIVITIVE POOL
NON ZERO ELEMENTS
BLOCKS OF VARIABLES
BLOCKS OF EQUATIONS
MODEL STATISTICS

Model Statistics     SOLVE PORTQP USING NLP FROM LINE 34
SIMPLE PORTFOLIO INVESTMENT PROBLEM (MARKOWITZ)

**** REPORT SUMMARY  :
- - - -  V AR OBJLP
- - - -  V AR GM
- - - -  V AR IBM
- - - -  EQU E2
- - - -  EQU LP
EXIT - -  OPTIMAL SOLUTION FOUND.

Estimate work space needed - - 33 Kb
Work space allocated - - 231 Kb
EXIT - -  OPTIMAL SOLUTION FOUND.

- - - -  EQU LP
LOWER  LEVEL  UPPER  MARGINAL
1.000  .  1.000  1.200
- - - -  EQU E2
LOWER  LEVEL  UPPER  MARGINAL
1.000  .  1.000  1.200
- - - -  VAR IBM
LOWER  LEVEL  UPPER  MARGINAL
0.750  .  0.750  .
- - - -  VAR GM
LOWER  LEVEL  UPPER  MARGINAL
0.250  .  0.750  .
- - - -  VAR GOLD
LOWER  LEVEL  UPPER  MARGINAL
1.275  +INF  -0.120
- - - -  VAR OBJLP  INF
LOWER  LEVEL  UPPER  MARGINAL
1.275  +INF  -0.120

**** REPORT SUMMARY :  0 NONOPT
0 INFEASIBLE
0 UNBOUNDED

SIMPLE PORTFOLIO INVESTMENT PROBLEM (MARKOWITZ)
Model Statistics     SOLVE PORTQP USING NLP FROM LINE 34

MODEL STATISTICS
BLOCKS OF EQUATIONS = 3 SINGLE EQUATIONS
BLOCKS OF VARIABLES = 4 SINGLE VARIABLES
NON ZERO ELEMENTS  = 10 NON LINEAR N-Z
DERIVITIVE POOL    = 8 CONSTANT POOL
CODE LENGTH         = 95
GENERATION TIME     = 2.360 SECONDS
EXECUTION TIME      = 5.310 SECONDS

EXIT - -  OPTIMAL SOLUTION FOUND
MAJOR ITNS, LIMIT = 1
FUNOBJ, FUNCON CALLS = 3
SUPERBASICS       = 8
INTERPRETER USAGE = 23
NORM RG / NORM PI = 3.732E-17

- - - -  EQU OP
LOWER  LEVEL  UPPER  MARGINAL
1.000  .  1.000  1.216
- - - -  EQU E1
LOWER  LEVEL  UPPER  MARGINAL
1.150  .  1.150  +INF
- - - -  EQU E2
LOWER  LEVEL  UPPER  MARGINAL
1.000  .  1.000  0.556
- - - -  VAR IBM
LOWER  LEVEL  UPPER  MARGINAL
0.183  .  0.750  .
- - - -  VAR GM
LOWER  LEVEL  UPPER  MARGINAL
0.248  .  0.750  .
- - - -  VAR GOLD
LOWER  LEVEL  UPPER  MARGINAL
0.569  .  0.750  .
- - - -  VAR OBJLP  INF
LOWER  LEVEL  UPPER  MARGINAL
1.421  +INF  .

**** REPORT SUMMARY :  0 NONOPT
0 INFEASIBLE
0 UNBOUNDED
0 ERRORS

SIMPLE PORTFOLIO INVESTMENT PROBLEM (MARKOWITZ)
Model Statistics     SOLVE PORTQP USING NLP FROM LINE 34
EXECUTION TIME      = 1.090 SECONDS

Portfolio Planning Problem - GAMS

SOLVE SUMMARY
***** MODEL STATUS = 1 OPTIMAL
***** OBJECTIVE VALUE = 0.4210
RESOURCE USAGE, LIMIT = 3.129 1000.000
ITERATION COUNT, LIMIT = 3 1000
EVALUATION ERRORS = 0

MODEL PORTLP
TYPE LP
SOLVER MINOS5
**** SOLVER STATUS = 1 NORMAL COMPLETION
**** MODEL STATUS = 2 LOCALLY OPTIMAL
**** OBJECTIVE VALUE = 0.4210
RESOURCE USAGE, LIMIT = 3.129 1000.000
ITERATION COUNT, LIMIT = 3 1000
EVALUATION ERRORS = 0

M I N O S  5.3  (Nov. 1990)
B.A. Murtagh, University of New South Wales
and
P.J. Gill, W. Murray, M.A. Saunders and M.H. Wright
Systems Optimization Laboratory, Stanford University.

EXIT - -  OPTIMAL SOLUTION FOUND
MAJOR ITNS, LIMIT = 1
FUNOOBJ, FUNCON CALLS = 8
SUPERBASICS = 1
INTERPRETER USAGE = 23
NORM RG / NORM PI = 3.732E-17

- - - -  EQU OP
LOWER  LEVEL  UPPER  MARGINAL
1.000  .  1.000  1.216
- - - -  EQU E1
LOWER  LEVEL  UPPER  MARGINAL
1.150  .  1.150  +INF
- - - -  EQU E2
LOWER  LEVEL  UPPER  MARGINAL
1.000  .  1.000  0.556
- - - -  VAR IBM
LOWER  LEVEL  UPPER  MARGINAL
0.183  .  0.750  .
- - - -  VAR GM
LOWER  LEVEL  UPPER  MARGINAL
0.248  .  0.750  .
- - - -  VAR GOLD
LOWER  LEVEL  UPPER  MARGINAL
0.569  .  0.750  .
- - - -  VAR OBJLP  INF
LOWER  LEVEL  UPPER  MARGINAL
1.421  +INF  .

**** REPORT SUMMARY :  0 NONOPT
0 INFEASIBLE
0 UNBOUNDED
0 ERRORS

SIMPLE PORTFOLIO INVESTMENT PROBLEM (MARKOWITZ)
Model Statistics     SOLVE PORTQP USING NLP FROM LINE 34
EXECUTION TIME      = 1.090 SECONDS
Motivation: Build on unconstrained methods wherever possible.

Classification of Methods:

- **Reduced Gradient Methods** - (with Restoration) GRG2, CONOPT
- **Reduced Gradient Methods** - (without Restoration) MINOS
- **Successive Quadratic Programming** - generic implementations
- **Penalty Functions** - popular in 1970s, but fell into disfavor. Barrier Methods have been developed recently and are again popular.
- **Successive Linear Programming** - only useful for "mostly linear" problems

We will concentrate on algorithms for first four classes.

Evaluation: Compare performance on "typical problem," cite experience on process problems.

---

Representative Constrained Problem
(Hughes, 1981)

Min \( f(x_1, x_2) = \alpha \exp(-\beta) \)
\[ g_1 = (x_2 + 0.1)^2[x_1^2 + 2(1-x_2)(1-2x_2)] - 0.16 \leq 0 \]
\[ g_2 = (x_1 - 0.3)^2 + (x_2 - 0.3)^2 - 0.16 \leq 0 \]
\( x^* = [0.6335, 0.3465] \quad f(x^*) = -4.8380 \)
Reduced Gradient Method with Restoration

**GRG2/CONOPT**

\[ \text{Min } f(x) \quad \text{s.t. } g(x) + s = 0 \text{ (add slack variable)} \]

\[ h(x) = 0 \]

\[ a \leq x \leq b, \quad s \geq 0 \]

Partition variables into:
- \( z_B \) - dependent or basic variables
- \( z_N \) - nonbasic variables, fixed at a bound
- \( z_S \) - independent or superbasic variables

**Modified KKT Conditions**

\[
\nabla f(z) + \nabla c(z) \lambda - \nu_L + \nu_U = 0
\]

\[ c(z) = 0 \]

\[ z^{(i)} = z_U^{(i)} \quad \text{or} \quad z^{(i)} = z_L^{(i)}, \quad i \in N \]

\[ \nu_U^{(i)}, \nu_L^{(i)} = 0, \quad i \notin N \]

---

**Reduced Gradient Method with Restoration**

**GRG2/CONOPT**

\[ a) \quad \nabla_S f(z) + \nabla_S c(z) \lambda = 0 \]

\[ b) \quad \nabla_B f(z) + \nabla_B c(z) \lambda = 0 \]

\[ c) \quad \nabla_N f(z) + \nabla_N c(z) \lambda - \nu_L + \nu_U = 0 \]

\[ d) \quad z^{(i)} = z_U^{(i)} \quad \text{or} \quad z^{(i)} = z_L^{(i)}, \quad i \in N \]

\[ e) \quad c(z) = 0 \Rightarrow z_B = z_B(z_S) \]

- Solve bound constrained problem in space of superbasic variables
  (apply gradient projection algorithm)
- Solve (e) to eliminate \( z_B \)
- Use (a) and (b) to calculate reduced gradient wrt \( z_S \)
- Nonbasic variables \( z_N \) (temporarily) fixed (d)
- Repartition based on signs of \( \nu \), if \( z^{(i)} \) remain at bounds or if \( z_B \) violate bounds
Definition of Reduced Gradient

\[
\frac{df}{dz_s} = \frac{\partial f}{\partial z_s} + \frac{dz_B}{dz_s} \frac{\partial f}{\partial z_B}
\]

Because \( c(z) = 0 \), we have:

\[
dc = [\frac{\partial c}{\partial z_S}]^T dz_S + [\frac{\partial c}{\partial z_B}]^T dz_B = 0
\]

\[
\frac{dz_B}{dz_S} = \left[ \frac{\partial c}{\partial z_S} \right]^T \left[ \frac{\partial c}{\partial z_B} \right]^{-1} = -\nabla_{z_S} c \left[ \nabla_{z_B} c \right]^{-1}
\]

This leads to:

\[
\frac{df}{dz_S} = \nabla_S f(z) - \nabla_S c \left[ \nabla_{z_B} c \right]^{-1} \nabla_{z_B} c(z) \lambda
\]

• By remaining feasible always, \( c(z) = 0, a \leq z \leq b \), one can apply an unconstrained algorithm (quasi-Newton) using \((df/dz_s)\), using (b).

• Solve problem in reduced space of \( z_S \) variables, using (e).

Example of Reduced Gradient

\[
\text{Min } x_1^2 - 2x_2
\]

\[
\text{s.t. } 3x_1 + 4x_2 = 24
\]

\[
\nabla c^T = [3 \ 4], \quad \nabla f^T = [2x_1 \ -2]
\]

Let \( z_S = x_1, \ z_B = x_2 \)

\[
\frac{df}{dz_S} = \frac{\partial f}{\partial z_S} - \nabla_{z_S} c \left[ \nabla_{z_B} c \right]^{-1} \frac{\partial f}{\partial z_B}
\]

\[
\frac{df}{dx_1} = 2x_1 - 3[4]^{-1}(-2) = 2x_1 + 3/2
\]

If \( \nabla c^T \) is \( m \times n \); \( \nabla z_S c^T \) is \( m \times (n-m) \); \( \nabla z_B c^T \) is \( m \times m \)

\((df/dz_S)\) is the change in \( f \) along constraint direction per unit change in \( z_S \)
Gradient Projection Method
(superbasic → nonbasic variable partition)

Define the projection of an arbitrary point $x$ onto box feasible region.
The $i$th component is given by

$$P(x, l, u)_i = \begin{cases} 
  l_i & \text{if } x_i < l_i, \\
  x_i & \text{if } x_i \in [l_i, u_i], \\
  u_i & \text{if } x_i > u_i.
\end{cases}$$

Piecewise linear path $x(t)$ starting at the reference point $x_0$ and obtained by projecting steepest descent (or any search) direction at $x_0$ onto the box region is given by

$$x(t) = P(x^0 - tg, l, u),$$

where $g$ is the reduced gradient, $t$ is the stepsize.

Also, can adapt to (quasi-) Newton method.

Sketch of GRG Algorithm

1. Initialize problem and obtain a feasible point at $z^0$
2. At feasible point $z^k$, partition variables $z$ into $z_N$, $z_B$, $z_S$
3. Calculate reduced gradient, $(df/dz_S)$
4. Evaluate search direction for $z_S$, $d = B^{-1}(df/dz_S)$
5. Perform a line search.
   - Find $\alpha \in (0,1]$ with $z_S := z_S^k + \alpha d$
   - Solve for $c(z_S^k + \alpha d, z_B, z_N) = 0$
   - If $f(z_S^k + \alpha d, z_B, z_N) < f(z_S^k, z_B, z_N)$,
     set $z_S^{k+1} = z_S^k + \alpha d$, $k := k+1$
6. If $||(df/dz_S)|| < \varepsilon$, Stop. Else, go to 2.
Reduced Gradient Method with Restoration

Fails, due to singularity in basis matrix \( \frac{dc}{dz_B} \)
Reduced Gradient Method with Restoration

Possible remedy: repartition basic and superbasic variables to create nonsingular basis matrix \( \frac{dc}{dz_B} \)

GRG Algorithm Properties

1. GRG2 has been implemented on PC's as GINO and is very reliable and robust. It is also the optimization solver in MS EXCEL.
2. CONOPT is implemented in GAMS, AIMMS and AMPL
3. GRG2 uses Q-N for small problems but can switch to conjugate gradients if problem gets large. CONOPT uses exact second derivatives.
4. Convergence of \( c(z_S, z_B, z_N) = 0 \) can get very expensive because \( \nabla c(z) \) is calculated repeatedly.
5. Safeguards can be added so that restoration (step 5.) can be dropped and efficiency increases.

Representative Constrained Problem Starting Point \([0.8, 0.2]\)
- GINO Results - 14 iterations to \( \|\nabla f(x^*)\| \leq 10^{-6} \)
- CONOPT Results - 7 iterations to \( \|\nabla f(x^*)\| \leq 10^{-6} \) from feasible point.
Reduced Gradient Method without Restoration

Motivation: Efficient algorithms are available that solve linearly constrained optimization problems (MINOS):

\[ \text{Min } f(x) \]
\[ \text{s.t. } Ax \leq b \]
\[ Cx = d \]

Extend to nonlinear problems, through successive linearization

Develop major iterations (linearizations) and minor iterations (GRG solutions).

Strategy: (Robinson, Murtagh & Saunders)

1. Partition variables into basic, nonbasic variables and superbasic variables.
2. Linearize active constraints at \( z^k \)
   \[ D^k z = r^k \]
3. Let \( \psi = f(z) + \lambda^T c(z) + \beta (c(z)^T c(z)) \) (Augmented Lagrange),
4. Solve linearly constrained problem:
   \[ \text{Min } \psi(z) \]
   \[ \text{s.t. } Dz = r \]
   \[ a \leq z \leq b \]
   using reduced gradients to get \( z^{k+1} \)
5. Set \( k = k + 1 \), go to 2.
6. Algorithm terminates when no movement between steps 2) and 4).
MINOS/Augmented Notes

1. MINOS has been implemented very efficiently to take care of linearity. It becomes LP Simplex method if problem is totally linear. Also, very efficient matrix routines.

2. No restoration takes place, nonlinear constraints are reflected in \( \psi(z) \) during step 3). MINOS is more efficient than GRG.

3. Major iterations (steps 3) - 4)) converge at a quadratic rate.

4. Reduced gradient methods are complicated, monolithic codes: hard to integrate efficiently into modeling software.

**Representative Constrained Problem** – Starting Point \([0.8, 0.2]\)

MINOS Results: 4 major iterations, 11 function calls
to \( \|Vf(x^*)\| \leq 10^{-6} \)

Successive Quadratic Programming (SQP)

**Motivation:**
- Take KKT conditions, expand in Taylor series about current point.
- Take Newton step (QP) to determine next point.

**Derivation – KKT Conditions**
\[
\begin{align*}
\nabla_x L(x^*, u^*, v^*) &= \nabla f(x^*) + \nabla g_A(x^*) u^* + \nabla h(x^*) v^* = 0 \\
h(x^*) &= 0 \\
g_A(x^*) &= 0, \quad \text{where } g_A \text{ are the active constraints.}
\end{align*}
\]

**Newton - Step**
\[
\begin{bmatrix}
\nabla_{xx} L & \nabla g_A & \nabla h \\
\nabla g_A^T & 0 & 0 \\
\nabla h^T & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\Delta x \\
\Delta u \\
\Delta v
\end{bmatrix}
= 

\begin{bmatrix}
\nabla_x L \left( x^k, u^k, v^k \right) \\
g_A \left( x^k \right) \\
h(x^k)
\end{bmatrix}
\]

**Requirements:**
- \( \nabla_{xx} L \) must be calculated and should be ‘regular’
- correct active set \( g_A \)
- good estimates of \( u^k, v^k \)
**SQP Chronology**

1. Wilson (1963)
   - active set can be determined by solving QP:
     \[
     \begin{align*}
     \text{Min} & \quad \nabla f(x_k)^T d + \frac{1}{2} d^T \nabla^2 L(x_k, u_k, v_k) d \\
     \text{s.t.} & \quad g(x_k) + \nabla g(x_k)^T d \leq 0 \\
     & \quad h(x_k) + \nabla h(x_k)^T d = 0
     \end{align*}
     \]

2. Han (1976), (1977), Powell (1977), (1978)
   - approximate \( \nabla^2 L \) using a positive definite quasi-Newton update (BFGS)
   - use a line search to converge from poor starting points.

**Notes:**
- Similar methods were derived using penalty (not Lagrange) functions.
- Method converges quickly; very few function evaluations.
- Not well suited to large problems (full space update used).
  - For \( n > 100 \), say, use reduced space methods (e.g. MINOS).

**Elements of SQP – Hessian Approximation**

What about \( \nabla^2 L \)?
- need to get second derivatives for \( f(x), g(x), h(x) \).
- need to estimate multipliers, \( u_k, v_k; \) \( \nabla^2 L \) may not be positive semidefinite

\[ B^{k+1} = B^k + \frac{yy^T}{s^T y} - \frac{B^k s s^T B^k}{s B^k s} \]

**BFGS Formula**
\[ s = x^{k+1} - x^k \]
\[ y = \nabla L(x^{k+1}, u^{k+1}, v^{k+1}) - \nabla L(x^k, u^{k+1}, v^{k+1}) \]
- second derivatives approximated by change in gradients
- positive definite \( B^k \) ensures unique QP solution
Elements of SQP – Search Directions

How do we obtain search directions?
- Form QP and let QP determine constraint activity
- At each iteration, $k$, solve:

$$\begin{align*}
\text{Min} & \quad \nabla f(x^k)^T d + 1/2 \ d^T B^k d \\
\text{s.t.} & \quad g(x^k) + \nabla g(x^k)^T d \leq 0 \\
& \quad h(x^k) + \nabla h(x^k)^T d = 0
\end{align*}$$

Convergence from poor starting points
- As with Newton’s method, choose $\alpha$ (stepsize) to ensure progress toward optimum: $x^{k+1} = x^k + \alpha d$.
- $\alpha$ is chosen by making sure a merit function is decreased at each iteration.

**Exact Penalty Function**
$$
\psi(x) = f(x) + \mu \left[ \sum \max (0, g_j(x)) + \sum |h_j(x)| \right]
\mu > \max_j \{|u_j|, |v_j|\}
$$

**Augmented Lagrange Function**
$$
\psi(x) = f(x) + u^T g(x) + v^T h(x) + \eta/2 \left\{ \sum (h_j(x))^2 + \sum \max (0, g_j(x))^2 \right\}
$$

Newton-Like Properties for SQP

**Fast Local Convergence**
- $B = \nabla_{xx} L$ Quadratic
- $\nabla_{xx} L$ is p.d and $B$ is Q-N 1 step Superlinear
- $B$ is Q-N update, $\nabla_{xx} L$ not p.d 2 step Superlinear

**Enforce Global Convergence**
- Ensure decrease of merit function by taking $\alpha \leq 1$
- Trust region adaptations provide a stronger guarantee of global convergence - but harder to implement.
Basic SQP Algorithm

0. **Guess** $x^0$, Set $B^0 = I$ (Identity). Evaluate $f(x^0)$, $g(x^0)$ and $h(x^0)$.
1. At $x^k$, evaluate $Vf(x^k)$, $Vg(x^k)$, $Vh(x^k)$.
2. If $k > 0$, update $B^k$ using the BFGS Formula.
3. Solve: 
   \[
   \begin{aligned}
   & \text{Min}_d \quad Vf(x^k)^T d + 1/2 \ d^T B^k d \\
   \text{s.t.} & \quad g(x^k) + Vg(x^k)^T d \leq 0 \\
   & \quad h(x^k) + Vh(x^k)^T d = 0
   \end{aligned}
   \]
   If KKT error less than tolerance: $\|\nabla L(x^*)\| \leq \epsilon$, $\|h(x^*)\| \leq \epsilon$,
   $\|g(x^*)\| \leq \epsilon$. STOP, else go to 4.
4. **Find** $\alpha$ so that $0 < \alpha \leq 1$ and $\psi(x^k + \alpha d) < \psi(x^k)$ sufficiently
   (Each trial requires evaluation of $f(x)$, $g(x)$ and $h(x)$).
5. $x^{k+1} = x^k + \alpha d$. Set $k = k + 1$ Go to 2.

Problems with SQP

- Nonsmooth Functions - Reformulate
- Ill-conditioning - Proper scaling
- Poor Starting Points – Trust Regions can help
- Inconsistent Constraint Linearizations
- Can lead to infeasible QP's

\[
\begin{aligned}
\text{Min} \quad & x_2 \\
\text{s.t.} \quad & 1 + x_1 - (x_2)^2 \leq 0 \\
& 1 - x_1 - (x_2)^2 \leq 0 \\
& x_2 \geq -1/2
\end{aligned}
\]
**SQP Test Problem**

Min $x_2$

s.t. $-x_2 + 2 x_1^2 - x_1^3 \leq 0$

$-x_2 + 2 (1-x_1)^2 - (1-x_1)^3 \leq 0$

$x^* = [0.5, 0.375]$.

**SQP Test Problem – First Iteration**

Start from the origin $(x_0 = [0, 0]^T)$ with $B^0 = I$, form:

$Min \quad d_2 + 1/2 (d_1^2 + d_2^2)$

s.t. $d_2 \geq 0$

$d_1 + d_2 \geq 1$

$d = [1, 0]^T$, with $\mu_1 = 0$ and $\mu_2 = 1$. 
From $x_1 = [0.5, 0]^T$ with $B^l = I$
(no update from BFGS possible), form:

Min $d_2 + 1/2 (d_1^2 + d_2^2)$
s.t. $-1.25 d_1 - d_2 + 0.375 \leq 0$
     $1.25 d_1 - d_2 + 0.375 \leq 0$

$d = [0, 0.375]^T$ with $\mu_1 = 0.5$ and $\mu_2 = 0.5$

$x^* = [0.5, 0.375]^T$ is optimal

Starting Point [0.8, 0.2] - starting from $B^0 = I$ and staying in bounds
and linearized constraints; converges in 8 iterations to $||\nabla f(x^*)|| \leq 10^{-6}$
Barrier Methods for Large-Scale Nonlinear Programming

Original Formulation
\[
\min_{x \in \mathbb{R}^n} \quad f(x) \\
\text{s.t.} \quad c(x) = 0 \quad \text{Can generalize for} \quad a \leq x \leq b \\
\quad x \geq 0
\]

Barrier Approach
\[
\min_{x \in \mathbb{R}^n} \quad \varphi_\mu(x) = f(x) - \mu \sum_{i=1}^{n} \ln x_i \\
\text{s.t.} \quad c(x) = 0
\]
\[\Rightarrow \quad \text{As} \quad \mu \to 0, \quad x^*(\mu) \to x^* \quad \text{Fiacco and McCormick (1968)}
\]

Solution of the Barrier Problem

\[\Rightarrow \quad \text{Newton Directions (KKT System)} \]
\[\nabla f(x) + A(x)\lambda - \nu = 0 \]
\[X\nu - \mu e = 0 \]
\[e^T = [1, 1, \ldots], \quad X = \text{diag}(x) \]
\[A = \nabla c(x), \quad W = \nabla_{xx} L(x, \lambda, \nu) \]
\[c(x) = 0 \]

\[\Rightarrow \quad \text{Reducing the System} \]
\[
\begin{bmatrix}
W + \Sigma & A \\
A^T & 0
\end{bmatrix}
\begin{bmatrix}
d_x \\
\lambda^+
\end{bmatrix}
= -
\begin{bmatrix}
\nabla \varphi_\mu \\
c
\end{bmatrix}
\]
\[\Sigma = X^{-1}V \]

IPOPT Code – www.coin-or.org
Global Convergence of Newton-based Barrier Solvers

Merit Function

Exact Penalty: \( P(x, \eta) = f(x) + \eta \|c(x)\| \)

Augmented Lagrangian: \( L^*(x, \lambda, \eta) = f(x) + \lambda^T c(x) + \eta \|c(x)\|^2 \)

Assess Search Direction (e.g., from IPOPT)

**Line Search** – choose stepsize \( \alpha \) to give sufficient decrease of merit function using a ‘step to the boundary’ rule with \( \tau \sim 0.99. \)

\[
\begin{align*}
\text{for } \alpha \in (0, \alpha^*], \quad x_{k+1} &= x_k + \alpha d_x \\
x_k + \alpha d_x &\geq (1 - \tau)x_k > 0 \\
v_{k+1} &= v_k + \alpha d_v \geq (1 - \tau)v_k > 0 \\
\lambda_{k+1} &= \lambda_k + \alpha (\lambda_k - \lambda_k)
\end{align*}
\]

- How do we balance \( \phi(x) \) and \( c(x) \) with \( \eta \)?
- Is this approach globally convergent? Will it still be fast?

Global Convergence Failure

*(Wächter and B., 2000)*

\[
\begin{align*}
\text{Min } f(x) \\
\text{s.t. } x_1 - x_3 - \frac{1}{2} &= 0 \\
(x_1)^2 - x_2 - 1 &= 0 \\
x_2, x_3 &\geq 0
\end{align*}
\]

Newton-type line search ‘stalls’ even though descent directions exist

\[
A(x^k)^T d_x + c(x^k) = 0 \\
x^k + \alpha d_x > 0
\]

Remedies:
- Composite Step Trust Region (Byrd et al.)
- Filter Line Search Methods
Line Search Filter Method

Store ($\phi_k$, $\theta_k$) at allowed iterates

Allow progress if trial point is acceptable to filter with $\theta$ margin

If switching condition
\[ \alpha [-\nabla \phi_k^T d]^a \geq \delta[\theta_k]^b, a > 2b > 2 \]
is satisfied, only an Armijo line search is required on $\phi_k$.

If insufficient progress on stepsize, evoke restoration phase to reduce $\theta$.

Global convergence and superlinear local convergence proved (with second order correction)

Implementation Details

Modify KKT (full space) matrix if singular

\[
\begin{bmatrix}
W_k + \Sigma_k + \delta_1 & A_k \\
A_k^T & -\delta_2 I
\end{bmatrix}
\]

- $\delta_1$ - Correct inertia to guarantee descent direction
- $\delta_2$ - Deal with rank deficient $A_k$

KKT matrix factored by MA27

Feasibility restoration phase

\[
\min \| c(x) \|_1 + \| x - x_k \|_Q^2
\]

Apply Exact Penalty Formulation

Exploit same structure/algorithm to reduce infeasibility
IPOPT Algorithm – Features

Line Search Strategies for Globalization
- $\ell_2$ exact penalty merit function
- augmented Lagrangian merit function
- Filter method (adapted and extended from Fletcher and Leyffer)

Algorithmic Properties
Globally, superlinearly convergent (Wächter and B., 2005)
Easily tailored to different problem structures

Hessian Calculation
- BFGS (full/LM and reduced space)
- SR1 (full/LM and reduced space)
- Exact full Hessian (direct)
- Exact reduced Hessian (direct)
- Preconditioned CG

Freely Available
CPL License and COIN-OR distribution: http://www.coin-or.org
IPOPT 3.1 recently rewritten in C++
Solved on thousands of test problems and applications

IPOPT Comparison on 954 Test Problems

![Graph showing IPOPT comparison on 954 test problems.]

Percent solved within S (min. CPU time)

- IPOPT
- IPOPT (no scaling)
- Knitro 3.1
- Loco 6.06
Recommendations for Constrained Optimization

1. Best current algorithms
   - GRG 2/CONOPT
   - MINOS
   - SQP
   - IPOPT

2. GRG 2 (or CONOPT) is generally slower, but is robust. Use with highly nonlinear functions. Solver in Excel!

3. For small problems ($n \leq 100$) with nonlinear constraints, use SQP.

4. For large problems ($n \geq 100$) with mostly linear constraints, use MINOS.

   ==> Difficulty with many nonlinearities

---

**Available Software for Constrained Optimization**

**SQP Routines**
- HSL, NaG and IMSL (NLPQL) Routines
- NPSOL – Stanford Systems Optimization Lab
- SNOPT – Stanford Systems Optimization Lab (rSQP discussed later)
- IPOPT – http://www.coin-or.org

**GAMS Programs**
- CONOPT - Generalized Reduced Gradient method with restoration
- MINOS - Generalized Reduced Gradient method without restoration

*Available from the CACHE office. The cost for this package including Process Design Case Students, GAMS: A User's Guide, and GAMS - The Solver Manuals, and a CD-ROM is $65 per CACHE supporting departments, and $100 per non-CACHE supporting departments and individuals. To order please complete standard order form and fax or mail to CACHE Corporation. More information can be found on http://www.che.utexas.edu/cache/gams.html*

**MS Excel**
- Solver uses Generalized Reduced Gradient method with restoration
Rules for Formulating Nonlinear Programs

1) Avoid overflows and undefined terms, (do not divide, take logs, etc.)
   e.g. \( x + y - \ln z = 0 \rightarrow x + y - u = 0 \)
   \( \exp u - z = 0 \)

2) If constraints must always be enforced, make sure they are linear or bounds.
   e.g. \( v(xy - z^2)^{1/2} = 3 \rightarrow \quad vu = 3 \)
   \( u^2 - (xy - z^2) = 0, \ u \geq 0 \)

3) Exploit linear constraints as much as possible, e.g. mass balance
   \( x_i L + y_i V = F z_i \rightarrow \quad I_i + v_i = f_i \)
   \( L - \sum I_i = 0 \)

4) Use bounds and constraints to enforce characteristic solutions.
   e.g. \( a \leq x \leq b, \ g(x) \leq 0 \)
   to isolate correct root of \( h(x) = 0. \)

5) Exploit global properties when possibility exists. Convex (linear equations?)
   Linear Program? Quadratic Program? Geometric Program?  

6) Exploit problem structure when possible.
   e.g. \( \text{Min } (Tx - 3Ty) \)
   \( s.t. \quad xT + y - T^2 y = 5 \)
   \( 4x - 5Ty + Tx = 7 \)
   \( 0 \leq T \leq 1 \)
   (If \( T \) is fixed \( \Rightarrow \) solve LP) \( \Rightarrow \) put \( T \) in outer optimization loop.

Process Optimization
Problem Definition and Formulation

Mathematical Modeling and Algorithmic Solution
Hierarchy of Nonlinear Programming Formulations and Model Intrusion

- SAND Full Space Formulation
- Adjoint Sens & SAND Adjoint
- SAND Tailored
- Direct Sensitivities
- Multi-level Parallelism
- Black Box

Compute Efficiency

CLOSED

OPEN

Decision Variables

$10^0$  $10^1$  $10^2$  $10^3$

Large Scale NLP Algorithms

Motivation: Improvement of Successive Quadratic Programming as Cornerstone Algorithm

→ process optimization for design, control and operations

Evolution of NLP Solvers:

SQP → rSQP → IPOPT

rSQP++ → IPOPT 3.x

2000 - : Simultaneous dynamic optimization over 1 000 000 variables and constraints

Current: Tailor structure, architecture and problems
**Modular Simulation Mode**

Physical Relation to Process

- Intuitive to Process Engineer
- Unit equations solved internally
- tailor-made procedures.

Convergence Procedures - for simple flowsheets, often identified from flowsheet structure
- Convergence "mimics" startup.
- Debugging flowsheets on "physical" grounds

**Flowsheet Optimization Problems - Introduction**

Design Specifications

Specify # trays reflux ratio, but would like to specify overhead comp. ==> Control loop  -Solve Iteratively

Nested Recycles Hard to Handle

Best Convergence Procedure?

- Frequent block evaluation can be expensive
- Slow algorithms applied to flowsheet loops.
- NLP methods are good at breaking loops
Chronology in Process Optimization

1. Early Work - Black Box Approaches
   Friedman and Pinder (1972) 75-150
   Gaddy and co-workers (1977) 300

2. Transition - more accurate gradients
   Parker and Hughes (1981) 64
   Biegler and Hughes (1981) 13

3. Infeasible Path Strategy for Modular Simulators
   Biegler and Hughes (1982) <10
   Chen and Stadtherr (1985)
   Kaijaluoto et al. (1985)
   and many more

4. Equation Based Process Optimization
   Westerberg et al. (1983) <5
   Shewchuk (1985) 2
   DMO, NOVA, RTOPT, etc. (1990s) 1-2

Process optimization should be as cheap and easy as process simulation

Process Simulators with Optimization Capabilities (using SQP)

Aspen Custom Modeler (ACM)
Aspen/Plus
gProms
Hysim/Hysys
Massbal
Optisim
Pro/II
ProSim
ROMeo
VTPLAN
Simulation and Optimization of Flowsheets

\[ \text{Min } f(x), \text{ s.t. } g(x) \leq 0 \]

For single degree of freedom:

- work in space defined by curve below.
- requires repeated (expensive) recycle convergence

---

Expanded Region with Feasible Path

\[ f(x, y(x)) \]
**"Black Box" Optimization Approach**
- Vertical steps are expensive (flowsheet convergence)
- Generally no connection between x and y.
- Can have "noisy" derivatives for gradient optimization.

**SQP - Infeasible Path Approach**
- solve and optimize simultaneously in x and y
- extended Newton method
Optimization Capability for Modular Simulators
(FLOWTRAN, Aspen/Plus, Pro/II, HySys)

Architecture
- Replace convergence with optimization block
- Problem definition needed (in-line FORTRAN)
- Executive, preprocessor, modules intact.

Examples
1. Single Unit and Acyclic Optimization
   - Distillation columns & sequences

2. "Conventional" Process Optimization
   - Monochlorobenzene process
   - NH3 synthesis

3. Complicated Recycles & Control Loops
   - Cavett problem
   - Variations of above

Optimization of Monochlorobenzene Process

PHYSICAL PROPERTY OPTIONS
Cavett Vapor Pressure
Redlich-Kwong Vapor Fugacity
Corrected Liquid Fugacity
Ideal Solution Activity Coefficient

OPT (SCOPT) OPTIMIZER
Optimal Solution Found After 4 Iterations
Kuhn-Tucker Error 0.29616E-05
Allowable Kuhn-Tucker Error 0.19826E-04
Objective Function -0.98259

Optimization Variables
32.006 0.38578 200.00 120.00

Tear Variables
0.10601E-19 13.064 79.229 120.00 50.000

Tear Variable Errors (Calculated Minus Assumed)
-0.10601E-19 0.72209E-06
-0.36563E-04 0.00000E+00
0.00000E+00

-Results of infeasible path optimization
-Simultaneous optimization and convergence of tear streams.
Ammonia Process Optimization

Hydrogen and Nitrogen feed are mixed, compressed, and combined with a recycle stream and heated to reactor temperature. Reaction occurs in a multibed reactor (modeled here as an equilibrium reactor) to partially convert the stream to ammonia. The reactor effluent is cooled and product is separated using two flash tanks with intercooling. Liquid from the second stage is flashed at low pressure to yield high purity NH$_3$ product. Vapor from the two stage flash forms the recycle and is recompressed.

### Optimization Problem

Max \{Total Profit @ 15% over five years\)

s.t.

- 10$^5$ tons NH$_3$/yr.
- Pressure Balance
- No Liquid in Compressors
- 1.8 \leq H2/N2 \leq 3.5
- Treact \leq 1000\,^\circ\,F
- NH$_3$ purged \leq 4.5 lb mol/hr
- NH$_3$ Product Purity \geq 99.9%
- Tear Equations

### Performance Characteristics

- 5 SQP iterations.
- 2.2 base point simulations.
- Objective function improves by \$20.66 \times 10^6$ to $24.93 \times 10^6$.
- Difficult to converge flowsheet at starting point

<table>
<thead>
<tr>
<th>Item</th>
<th>Optimum</th>
<th>Starting point</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective Function($10^6$)</td>
<td>24.9286</td>
<td>20.659</td>
</tr>
<tr>
<td>1. Inlet temp. reactor (°F)</td>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>2. Inlet temp. 1st flash (°F)</td>
<td>65</td>
<td>65</td>
</tr>
<tr>
<td>3. Inlet temp. 2nd flash (°F)</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>4. Inlet temp. rec. comp. (°F)</td>
<td>80.52</td>
<td>107</td>
</tr>
<tr>
<td>5. Purge fraction (%)</td>
<td>0.0085</td>
<td>0.01</td>
</tr>
<tr>
<td>6. Reactor Press. (psia)</td>
<td>2163.5</td>
<td>2000</td>
</tr>
<tr>
<td>7. Feed 1 (lb mol/hr)</td>
<td>2629.7</td>
<td>2632.0</td>
</tr>
<tr>
<td>8. Feed 2 (lb mol/hr)</td>
<td>691.78</td>
<td>691.4</td>
</tr>
</tbody>
</table>
How accurate should gradients be for optimization?

Recognizing True Solution
• KKT conditions and Reduced Gradients determine true solution
• Derivative Errors will lead to wrong solutions!

Performance of Algorithms
Constrained NLP algorithms are gradient based
(SQP, Conopt, GRG2, MINOS, etc.)
Global and Superlinear convergence theory assumes accurate gradients

Worst Case Example (Carter, 1991)
Newton’s Method generates an ascent direction and fails for any $\varepsilon$!

$$\begin{align*}
\text{Min } f(x) &= x^T Ax \\
A &= \begin{bmatrix} \varepsilon + 1/\varepsilon & \varepsilon - 1/\varepsilon \\ \varepsilon - 1/\varepsilon & \varepsilon + 1/\varepsilon \end{bmatrix} \\
x_0 &= [1 \quad 1]^T \\
g(x_0) &= \nabla f(x_0) + O(\varepsilon) \\
d &= -A^{-1}g(x_0)
\end{align*}$$

Implementation of Analytic Derivatives

Automatic Differentiation Tools
JAKE-F, limited to a subset of FORTRAN (Hillstrom, 1982)
DAPRE, which has been developed for use with the NAG library (Pryce, Davis, 1987)
ADOL-C, implemented using operator overloading features of C++ (Griewank, 1990)
ADIFOR, (Bischof et al, 1992) uses source transformation approach FORTRAN code.
TAPENADE, web-based source transformation for FORTRAN code.

Relative effort needed to calculate gradients is not $n+1$ but about 3 to 5
(Wolfe, Griewank)
Large-Scale SQP

\[ \begin{align*}
\text{Min} & \quad f(z) \\
\text{s.t.} & \quad c(z) = 0 \\
& \quad z_L \leq z \leq z_U
\end{align*} \]

\[ \begin{align*}
\text{Min} & \quad \nabla f(z^k)^T d + \frac{1}{2} d^T W_k d \\
\text{s.t.} & \quad c(z^k) + (A^k)^T d = 0 \\
& \quad z_L \leq z^k + d \leq z_U
\end{align*} \]

**Characteristics**

- Many equations and variables (≥ 100 000)
- Many bounds and inequalities (≥ 100 000)

**Few degrees of freedom (10 - 100)**
- Steady state flowsheet optimization
- Real-time optimization
- Parameter estimation

**Many degrees of freedom (≥ 1000)**
- Dynamic optimization (optimal control, MPC)
- State estimation and data reconciliation
Few degrees of freedom => reduced space SQP (rSQP)

• Take advantage of sparsity of $A = \nabla c(x)$
• project $W$ into space of active (or equality constraints)
• curvature (second derivative) information only needed in space of degrees of freedom
• second derivatives can be applied or approximated with positive curvature (e.g., BFGS)
• use dual space QP solvers

+ easy to implement with existing sparse solvers, QP methods and line search techniques
+ exploits 'natural assignment' of dependent and decision variables (some decomposition steps are 'free')
+ does not require second derivatives

- reduced space matrices are dense
- may be dependent on variable partitioning
- can be very expensive for many degrees of freedom
- can be expensive if many QP bounds

Reduced space SQP (rSQP)
Range and Null Space Decomposition

Assume no active bounds, QP problem with $n$ variables and $m$ constraints becomes:

\[
\begin{bmatrix}
W^k & A^k \\
A^kT & 0
\end{bmatrix}
\begin{bmatrix}
d \\
\lambda_+
\end{bmatrix} =
-\begin{bmatrix}
\nabla f(x^k) \\
c(x^k)
\end{bmatrix}
\]

• Define reduced space basis, $Z^k \in \mathbb{R}^{n \times (n-m)}$ with $(A^k)^T Z^k = 0$
• Define basis for remaining space $Y^k \in \mathbb{R}^{n \times m}$, $[Y^k Z^k] \in \mathbb{R}^{n \times n}$ is nonsingular.
• Let $d = Y^k d_Y + Z^k d_Z$ to rewrite:

\[
\begin{bmatrix}
Y^k & Z^k \\
0 & I
\end{bmatrix}
\begin{bmatrix}
W^k & A^k \\
A^kT & 0
\end{bmatrix}
\begin{bmatrix}
Y^k & Z^k \\
0 & I
\end{bmatrix}
\begin{bmatrix}
d_Y \\
d_Z
\end{bmatrix} =
-\begin{bmatrix}
Y^k & Z^k \\
0 & I
\end{bmatrix}
\begin{bmatrix}
\nabla f(x^k) \\
c(x^k)
\end{bmatrix}
\]
Reduced space SQP (rSQP)
Range and Null Space Decomposition

\[
\begin{bmatrix}
Y^k A^k \\
W^k Y^k \\
Z^k W^k Z^k \\
A^k Y^k
\end{bmatrix}
\begin{bmatrix}
d_Y \\
d_Z \\
\lambda^*_+ \\
0
\end{bmatrix} =
\begin{bmatrix}
-Y^k \nabla f(x^k) \\
-Z^k \nabla f(x^k) \\
c(x^k)
\end{bmatrix}
\]

- \((ATY) d_Y = -c(x^k)\) is square, \(d_Y\) determined from bottom row.
- Cancel \(Y^T W Y\) and \(Z^T W Z\); (unimportant as \(d_Z, d_Y \rightarrow 0\))
- \((YA) \lambda = -Y^T \nabla f(x^k), \lambda\) can be determined by first order estimate
- Calculate or approximate \(w = Z^T W Y d_Y\), solve \(Z^T W Z d_Z = -Z^T \nabla f(x^k) - w\)
- Compute total step: \(d = Y d_Y + Z d_Z\)

**Reduced space SQP (rSQP) Interpretation**

**Range and Null Space Decomposition**

- SQP step (d) operates in a higher dimension
- Satisfy constraints using range space to get \(d_Y\)
- Solve small QP in null space to get \(d_Z\)
- In general, same convergence properties as SQP.
Choice of Decomposition Bases

1. Apply QR factorization to $A$. Leads to dense but well-conditioned $Y$ and $Z$.

$$ A = Q \begin{bmatrix} R \\ 0 \end{bmatrix} = \begin{bmatrix} Y \\ Z \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix} $$

2. Partition variables into decisions $u$ and dependents $v$. Create orthogonal $Y$ and $Z$ with embedded identity matrices ($A^T Z = 0, Y^T Z = 0$).

$$ A^T = \begin{bmatrix} \nabla_u c^T & \nabla_v c^T \end{bmatrix} = \begin{bmatrix} N & C \end{bmatrix} $$

$$ Z = \begin{bmatrix} I \\ -C^{-1} N \end{bmatrix}, \quad Y = \begin{bmatrix} N^T C^{-T} \\ I \end{bmatrix} $$

3. Coordinate Basis - same $Z$ as above, $Y^T = \begin{bmatrix} 0 & I \end{bmatrix}$

- Bases use gradient information already calculated.
- Adapt decomposition to QP step
- Theoretically same rate of convergence as original SQP.
- Coordinate basis can be sensitive to choice of $u$ and $v$. Orthogonal is not.
- Need consistent initial point and nonsingular $C$; automatic generation

rSQP Algorithm

1. Choose starting point $x^0$.
2. At iteration $k$, evaluate functions $f(x^k)$, $c(x^k)$ and their gradients.
3. Calculate bases $Y$ and $Z$.
4. Solve for step $d_Y$ in Range space from $(A^T Y) d_Y = -c(x^k)$
5. Update projected Hessian $B^k \approx Z^T W Z$ (e.g., with BFGS), $w_k$ (e.g., zero)

$$ \min \ (Z^T \nabla f(x^k) + w^k)^T d_Z + 1/2 d_Z^T B^k d_Z $$

subject to $x_L \leq x^k + Y d_Y + Z d_Z \leq x_U$

7. If error is less than tolerance stop. Else
8. Solve for multipliers using $(Y^T A) \lambda = -Y^T \nabla f(x^k)$
10. Find step size $\alpha$ and calculate new point, $x_{k+1} = x_k + \alpha d$
11. Continue from step 2 with $k = k + 1$. 
### rSQP Results: Computational Results for General Nonlinear Problems

Vasanharajan et al (1990)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Specifications</th>
<th>MINOS (5.2)</th>
<th>Reduced SQP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N</td>
<td>M</td>
<td>MEQ</td>
</tr>
<tr>
<td>Ramsey</td>
<td>34</td>
<td>23</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chenery</td>
<td>44</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Korceg</td>
<td>100</td>
<td>96</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cansig</td>
<td>280</td>
<td>243</td>
<td>243</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ganges</td>
<td>357</td>
<td>274</td>
<td>274</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* CPU Seconds - VAX 6320

### rSQP Results: Computational Results for Process Problems

Vasanharajan et al (1990)

<table>
<thead>
<tr>
<th>Prob.</th>
<th>Specifications</th>
<th>MINOS (5.2)</th>
<th>Reduced SQP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N</td>
<td>M</td>
<td>MEQ</td>
</tr>
<tr>
<td>Absorber</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(a)</td>
<td>50</td>
<td>42</td>
<td>42</td>
</tr>
<tr>
<td>(b)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distill’n</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ideal</td>
<td>228</td>
<td>227</td>
<td>227</td>
</tr>
<tr>
<td>(a)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(b)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(c)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distill’n</td>
<td>569</td>
<td>567</td>
<td>567</td>
</tr>
<tr>
<td>Nonideal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(a)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(b)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(c)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distill’n</td>
<td>977</td>
<td>975</td>
<td>975</td>
</tr>
<tr>
<td>Nonideal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(a)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(b)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(c)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(F) Failed</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* CPU Seconds - VAX 6320
+ MINOS (5.1)
Comparison of SQP and rSQP

Coupled Distillation Example - 5000 Equations
Decision Variables - boilup rate, reflux ratio

<table>
<thead>
<tr>
<th>Method</th>
<th>CPU Time</th>
<th>Annual Savings</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. SGP*</td>
<td>2 hr</td>
<td>negligible</td>
<td>Base Case</td>
</tr>
<tr>
<td>2. rSQP</td>
<td>15 min.</td>
<td>$ 42,000</td>
<td>Base Case</td>
</tr>
<tr>
<td>3. rSQP</td>
<td>15 min.</td>
<td>$ 84,000</td>
<td>Higher Feed Tray Location</td>
</tr>
<tr>
<td>4. rSQP</td>
<td>15 min.</td>
<td>$ 84,000</td>
<td>Column 2 Overhead to Storage</td>
</tr>
<tr>
<td>5. rSQP</td>
<td>15 min</td>
<td>$107,000</td>
<td>Cases 3 and 4 together</td>
</tr>
</tbody>
</table>

Real-time Optimization with rSQP
Sunoco Hydrocracker Fractionation Plant
(Bailey et al, 1993)

Existing process, optimization on-line at regular intervals: 17 hydrocarbon components, 8 heat exchangers, absorber/stripper (30 trays), debutanizer (20 trays), C3/C4 splitter (20 trays) and deisobutanizer (33 trays).

- square parameter case to fit the model to operating data.
- optimization to determine best operating conditions
Model consists of 2836 equality constraints and only ten independent variables. It is also reasonably sparse and contains 24123 nonzero Jacobian elements.

\[
P = \sum_{i \in \Omega} z_i C_i G_i + \sum_{i \in \Omega} z_i C_i E_i + \sum_{m=1}^{NP} z_i C_i P_n - U
\]

**Cases Considered:**
1. Normal Base Case Operation
2. Simulate fouling by reducing the heat exchange coefficients for the debutanizer
3. Simulate fouling by reducing the heat exchange coefficients for splitter feed/bottoms exchangers
4. Increase price for propane
5. Increase base price for gasoline together with an increase in the octane credit

<table>
<thead>
<tr>
<th>Heat Exchange Coefficient (TJ/d°C)</th>
<th>Case 0 Base Parameter</th>
<th>Case 1 Base Optimization</th>
<th>Case 2 Fouling 1</th>
<th>Case 3 Fouling 2</th>
<th>Case 4 Changing Market 1</th>
<th>Case 5 Changing Market 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debutanizer Feed/Bottoms</td>
<td>6.565x10^-4 1.030x10^-3</td>
<td>6.565x10^-4 1.030x10^-3</td>
<td>5.000x10^-4 2.000x10^-4</td>
<td>6.565x10^-4 1.030x10^-3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Splitter Feed/Bottoms</td>
<td>1.030x10^-3 1.030x10^-3</td>
<td>5.000x10^-4 2.000x10^-4</td>
<td>6.565x10^-4 1.030x10^-3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pricing</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Propane ($/m³)</td>
<td>180</td>
<td>180</td>
<td>180</td>
<td>300</td>
<td>300</td>
<td>300</td>
</tr>
<tr>
<td>Gasoline Base Price ($/m³)</td>
<td>300</td>
<td>300</td>
<td>300</td>
<td>300</td>
<td>300</td>
<td>350</td>
</tr>
<tr>
<td>Octane Credit ($/(RON m³))</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td>10</td>
</tr>
</tbody>
</table>

| Profit                            | 230968.96             | 239277.37               | 239265.57        | 236706.82        | 258913.28               | 370053.98               |
| Change from base case ($/d, %)    | -                     | 8308.41                 | 8298.61          | 5737.86          | 27944.32                | 139085.02               |
| (3.6%) clean                     | (3.6%) clean          | (3.6%) clean            | (2.5%) clean     | (12.1%) clean    | (60.2%) clean           |                        |

**Infeasible Initialization**

<table>
<thead>
<tr>
<th>MINOS</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>5 / 275</td>
<td>9 / 788</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>(Major/Minor)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU Time (s)</td>
<td>182</td>
<td>5768</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>sQP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iterations</td>
<td>5</td>
<td>20</td>
<td>12</td>
<td>24</td>
<td>17</td>
<td>12</td>
</tr>
<tr>
<td>CPU Time (s)</td>
<td>23.3</td>
<td>80.1</td>
<td>54.0</td>
<td>93.9</td>
<td>69.8</td>
<td>54.2</td>
</tr>
</tbody>
</table>

**Parameter Initialization**

<table>
<thead>
<tr>
<th>MINOS</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>n/a</td>
<td>12 / 132</td>
<td>14 / 120</td>
<td>16 / 156</td>
<td>11 / 166</td>
<td>11 / 76</td>
</tr>
<tr>
<td>(Major/Minor)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU Time (s)</td>
<td>n/a</td>
<td>462</td>
<td>408</td>
<td>1022</td>
<td>916</td>
<td>309</td>
</tr>
<tr>
<td>sQP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Iterations</td>
<td>n/a</td>
<td>13</td>
<td>8</td>
<td>18</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>CPU Time (s)</td>
<td>n/a</td>
<td>58.8</td>
<td>43.8</td>
<td>74.4</td>
<td>52.5</td>
<td>49.7</td>
</tr>
<tr>
<td>sQP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Time (%)                          | 12.8%                 | 12.7%                   | 10.7%            | 7.3%             | 5.8%                    | 16.1%                   |
| sQP                               |                       |                         |                  |                  |                         |                         |

| Time MINOS (%)                    |                       |                         |                  |                  |                         |                         |
Nonlinear Optimization Engines

Evolution of NLP Solvers:

\[ \text{process optimization for design, control and operations} \]

SQP \rightarrow \text{rSQP} \rightarrow \text{IPOPT}

’00s: Simultaneous dynamic optimization over 1 000 000 variables and constraints

Many degrees of freedom => full space IPOPT

\[
\begin{bmatrix}
W^k + \sum A^k \\
A^k T
\end{bmatrix}
\begin{bmatrix}
d \\
\lambda_+
\end{bmatrix}
= - \left[ \begin{array}{c}
\nabla \varphi(x^k) \\
\c(x^k)
\end{array} \right]
\]

- work in full space of all variables
- second derivatives useful for objective and constraints
- use specialized large-scale Newton solver

+ $W = \nabla_{xx} L(x, \lambda)$ and $A = \nabla c(x)$ sparse, often structured
+ fast if many degrees of freedom present
+ no variable partitioning required

- second derivatives strongly desired
- $W$ is indefinite, requires complex stabilization
- requires specialized large-scale linear algebra
Blending Problem & Model Formulation

\[ \text{max} \sum_{t} \left( \sum_{k} c_{k} f_{t,k} - \sum_{i} c_{i} f_{t,i} \right) \]

s.t. \[ \sum_{k} f_{t,j,k} - \sum_{i} f_{t,i,j} + v_{t+1,j} = v_{t,j} \]
\[ f_{t,k} - \sum_{j} f_{t,j,k} = 0 \]
\[ \sum_{k} q_{t,j,k} f_{t,j,k} - \sum_{i} q_{t,i,j} f_{t,i,j} + q_{t+1,j} v_{t+1,j} = q_{t,j} v_{t,j} \]
\[ q_{t,k} f_{t,k} - \sum_{j} q_{t,j,k} f_{t,j,k} = 0 \]
\[ \min \left\{ q_{t,k} \right\} \leq q_{t,k} \leq q_{k,\text{max}} \]
\[ \min \left\{ v_{j} \right\} \leq v_{t,j} \leq v_{j,\text{max}} \]

f & v ------ flowrates and tank volumes
q ------ tank qualities

Model Formulation in AMPL
Small Multi-day Blending Models

Single Qualities

Haverly, C. 1978 (HM)

Audet & Hansen 1998 (AHM)

<table>
<thead>
<tr>
<th>no. of iterations</th>
<th>objective</th>
<th>CPU (s)</th>
<th>normalized CPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HM Day 1</strong> (N = 13, M = 8, S = 8)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LANCELOT</td>
<td>62</td>
<td>190</td>
<td>0.10</td>
</tr>
<tr>
<td>MINOS</td>
<td>15</td>
<td>400</td>
<td>0.04</td>
</tr>
<tr>
<td>SNOPT</td>
<td>26</td>
<td>400</td>
<td>0.02</td>
</tr>
<tr>
<td>KNITRO</td>
<td>38</td>
<td>190</td>
<td>0.14</td>
</tr>
<tr>
<td>LOGO</td>
<td>30</td>
<td>200</td>
<td>0.10</td>
</tr>
<tr>
<td>IPOPT, exact</td>
<td>51</td>
<td>400</td>
<td>0.01</td>
</tr>
<tr>
<td>IPOPT, L-BFGS</td>
<td>150</td>
<td>400</td>
<td>0.08</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>no. of iterations</th>
<th>objective</th>
<th>CPU (s)</th>
<th>normalized CPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AHM Day 1</strong> (N = 21, M = 14, S = 14)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LANCELOT</td>
<td>112</td>
<td>49.2</td>
<td>0.52</td>
</tr>
<tr>
<td>MINOS</td>
<td>29</td>
<td>9.00</td>
<td>0.01</td>
</tr>
<tr>
<td>SNOPT</td>
<td>69</td>
<td>49.2</td>
<td>0.01</td>
</tr>
<tr>
<td>KNITRO</td>
<td>44</td>
<td>31.6</td>
<td>0.15</td>
</tr>
<tr>
<td>LOGO</td>
<td>28</td>
<td>49.2</td>
<td>0.10</td>
</tr>
<tr>
<td>IPOPT, exact</td>
<td>28</td>
<td>49.2</td>
<td>0.01</td>
</tr>
<tr>
<td>IPOPT, L-BFGS</td>
<td>44</td>
<td>49.2</td>
<td>0.02</td>
</tr>
</tbody>
</table>

Honeywell Blending Model – Multiple Days

48 Qualities

<table>
<thead>
<tr>
<th>no. of iterations</th>
<th>objective</th>
<th>CPU (s)</th>
<th>normalized CPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HBM Day 6</strong> (N = 1014, M = 8073, S = 7339)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LANCELOT</td>
<td>388</td>
<td>6.14 x 10^5</td>
<td>1.17 x 10^5</td>
</tr>
<tr>
<td>MINOS</td>
<td>2238</td>
<td>6.14 x 10^5</td>
<td>5.24 x 10^4</td>
</tr>
<tr>
<td>SNOPT</td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>KNITRO</td>
<td>37</td>
<td>1.00 x 10^7</td>
<td>1.58 x 10^7</td>
</tr>
<tr>
<td>LOGO</td>
<td>1.00</td>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>IPOPT, exact</td>
<td>21</td>
<td>6.14 x 10^5</td>
<td>2.60</td>
</tr>
<tr>
<td>IPOPT, L-BFGS</td>
<td>52</td>
<td>6.14 x 10^5</td>
<td>8.89</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>no. of iterations</th>
<th>objective</th>
<th>CPU (s)</th>
<th>normalized CPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HBM Day 10</strong> (N = 20925, M = 16074, S = 15205)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LANCELOT</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>MINOS</td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>SNOPT</td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>KNITRO</td>
<td>q</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>LOGO</td>
<td>a</td>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>IPOPT, exact</td>
<td>65</td>
<td>2.64 x 10^6</td>
<td>1.12 x 10^6</td>
</tr>
<tr>
<td>IPOPT, L-BFGS</td>
<td>100</td>
<td>2.64 x 10^6</td>
<td>8.89</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>no. of iterations</th>
<th>objective</th>
<th>CPU (s)</th>
<th>normalized CPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HBM Day 15</strong> (N = 31743, M = 25560, S = 23673)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LANCELOT</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>MINOS</td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>SNOPT</td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>KNITRO</td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>LOGO</td>
<td>b</td>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>IPOPT, exact</td>
<td>110</td>
<td>4.15 x 10^6</td>
<td>7.25 x 10^5</td>
</tr>
</tbody>
</table>
Summary of Results – Dolan-Moré plot

Performance profile (iteration count)

Comparison of NLP Solvers: Data Reconciliation

Degrees of Freedom

Degrees of Freedom

CPU Time (s, norm.)
Comparison of NLP solvers
(latest Mittelmann study)

Mittelmann NLP benchmark (10-26-2008)

<table>
<thead>
<tr>
<th>Solver</th>
<th>Limits</th>
<th>Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPOPT</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>KNITRO</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>LOQO</td>
<td>23</td>
<td>4</td>
</tr>
<tr>
<td>SNOPT</td>
<td>56</td>
<td>11</td>
</tr>
<tr>
<td>CONOPT</td>
<td>55</td>
<td>11</td>
</tr>
</tbody>
</table>

117 Large-scale Test Problems
500 - 250 000 variables, 0 – 250 000 constraints

Typical NLP algorithms and software

- **SQP** - NPSOL, VF02AD, NLPQL, fmincon
- **reduced SQP** - SNOPT, rSQP, MUSCOD, DMO, LSSOL...

- Reduced Grad. rest. - GRG2, GINO, SOLVER, CONOPT
- Reduced Grad no rest. - MINOS
- Second derivatives and barrier - IPOPT, KNITRO, LOQO

Interesting hybrids -

- FSQP/cFSQP - SQP and constraint elimination
- LANCELOT (Augmented Lagrangian w/ Gradient Projection)
At nominal conditions, $p_0$

$$\begin{align*}
\text{Min } f(x, p_0) \\
\text{s.t. } c(x, p_0) = 0 \\
a(p_0) \leq x \leq b(p_0)
\end{align*}$$

How is the optimum affected at other conditions, $p \neq p_0$?

- Model parameters, prices, costs
- Variability in external conditions
- Model structure

- How sensitive is the optimum to parametric uncertainties?
- Can this be analyzed easily?

**Second Order Optimality Conditions:**

**Reduced Hessian needs to be positive semi-definite**

- **Nonstrict local minimum:** If nonnegative, find eigenvectors for zero eigenvalues, $\Rightarrow$ regions of nonunique solutions

- **Saddle point:** If any are eigenvalues are negative, move along directions of corresponding eigenvectors and restart optimization.
IPOPT Factorization Byproducts:
Tools for Postoptimality and Uniqueness

Modify KKT (full space) matrix if nonsingular

\[
\begin{bmatrix}
W_k + \Sigma_k + \delta_1 I & A_k \\
A_k^T & -\delta_2 I
\end{bmatrix}
\]

- \(\delta_1\) - Correct inertia to guarantee descent direction
- \(\delta_2\) - Deal with rank deficient \(A_k\)

KKT matrix factored by indefinite symmetric factorization

- Solution with \(\delta_1, \delta_2 = 0\) \(\Rightarrow\) sufficient second order conditions
- Eigenvalues of reduced Hessian all positive – unique minimizer and multipliers
- Else:
  - Reduced Hessian available through sensitivity calculations
  - Find eigenvalues to determine nature of stationary point

NLP Sensitivity

Parametric Programming

\[
\begin{align*}
\min_{x, p} \quad & f(x, p) \\
\text{s.t.} \quad & c(x, p) = 0 \\
& x \geq 0
\end{align*}
\]

Solution Triplet

\[
s^*(p)^T = [x^T \lambda^T \nu^T]
\]

Optimality Conditions

\[
\begin{align*}
\nabla_x f(x, p) + \nabla_c c(x, p) \lambda - \nu &= 0 \\
c(x, p) &= 0 \\
XVe &= 0
\end{align*}
\]

NLP Sensitivity \(\Rightarrow\) Rely upon Existence and Differentiability of \(s^*(p)\)

\(\Rightarrow\) Main Idea: Obtain \(\frac{\partial s^*}{\partial p}_{p_0}\) and find \(\tilde{s}^*(p_{1})\) by Taylor Series Expansion

\[
\tilde{s}^*(p_{1}) \approx s^*(p_{0}) + \frac{\partial s}{\partial p}_{p_0} \bigg|_{p_0} (p_{1} - p_{0})
\]
NLP Sensitivity Properties (Fiacco, 1983)

Assume sufficient differentiability, LICQ, SSOC, SC:

Intermediate IP solution \( (s(\mu) - s^*) = O(\mu) \)

Finite neighborhood around \( p_0 \) and \( \mu = 0 \) with same active set exists and is unique

\[
\left. \frac{\partial s}{\partial p} \right|_{p_0} = O(\mu)
\]

\[
s(p) - [s(p_0) + \left. \frac{\partial s}{\partial p} \right|_{p_0}^T (p - p_0)] = O((p - p_0)^2)
\]

\[
s(p) - [s(p_0, \mu) + \left. \frac{\partial s}{\partial p} \right|_{p_0, \mu}^T (p - p_0)] = O((p - p_0)^2) + O(\mu)
\]

NLP Sensitivity

Obtaining \( \left. \frac{\partial s}{\partial p} \right|_{p_0} \)

Optimality Conditions of \( P(p) \)

\[
\begin{align*}
\nabla_x \mathcal{L} &= \nabla_x f(x, p) + \nabla_x c(x, p) \lambda - \nu = \zeta \\
\lambda c(x, p) &= \zeta \\
X V c &= \zeta
\end{align*}
\]

\( Q(s, p) = 0 \)

Apply Implicit Function Theorem to \( Q(s, p) = 0 \) around \( (p_0, s^*(p_0)) \)

\[
\left. \frac{\partial Q(s^*(p_0), p_0)}{\partial s} \right|_{p_0} \frac{\partial s}{\partial p} + \left. \frac{\partial Q(s^*(p_0), p_0)}{\partial p} \right|_{p_0} = 0
\]

KKT Matrix IPOPT

\[
\begin{bmatrix}
W(s^*(p_0)) & A(x^*(p_0)) \\
A(x^*(p_0))^T & 0 \\
V^*(p_0) & X^*(p_0)
\end{bmatrix}
\begin{bmatrix}
\frac{\partial x}{\partial p} \\
\frac{\partial \lambda}{\partial p} \\
\frac{\partial c}{\partial p}
\end{bmatrix}
\]

\[
\begin{bmatrix}
\nabla_{x, p} \mathcal{L}(s^*(p_0)) \\
\nabla_x c(x^*(p_0))
\end{bmatrix} = 0
\]

⇒ Already Factored at Solution
⇒ Sensitivity Calculation from Single Backsolve
⇒ Approximate Solution Retains Active Set
Sensitivity for Flash Recycle Optimization
(2 decisions, 7 tear variables)

- Second order sufficiency test:
- Dimension of reduced Hessian = 1
- Positive eigenvalue
- Sensitivity to simultaneous change in feed rate and upper bound on purge ratio

Ammonia Process Optimization
(9 decisions, 8 tear variables)

- Second order sufficiency test:
- Dimension of reduced Hessian = 4
- Eigenvalues = [2.8E-4, 8.3E-10, 1.8E-4, 7.7E-5]
- Sensitivity to simultaneous change in feed rate and upper bound on reactor conversion
Multi-Scenario Optimization

1. Design plant to deal with different operating scenarios (over time or with uncertainty)

2. Can solve overall problem simultaneously
   • large and expensive
   • polynomial increase with number of cases
   • must be made efficient through specialized decomposition

3. Solve also each case independently as an optimization problem (inner problem with fixed design)
   • overall coordination step (outer optimization problem for design)
   • require sensitivity from each inner optimization case with design variables as external parameters

Design Under Uncertain Model Parameters and Variable Inputs

\[
\min_{\theta \in \Theta} \mathbb{E}[P(d, z, y, \theta)],
\]
\[
s.t. \quad h(d, z, y, \theta) = 0,
\]
\[
g(d, z, y, \theta) \leq 0
\]

\(\mathbb{E}[P, \ldots]\) : expected value of an objective function
\(h\) : process model equations
\(g\) : process model inequalities
\(y\) : state variables (x, T, p, etc)
\(d\) : design variables (equipment sizes, etc)
\(\theta_p\) : uncertain model parameters
\(\theta_v\) : variable inputs \(\theta = [\theta_p^T \theta_v^T]\)
\(z\) : control/operating variables (actuators, flows, etc)
(may be fixed or a function of (some) \(\theta\))
(*single or two stage formulations*)
Multi-scenario Models for Uncertainty

\[
\begin{align*}
\min_{d,z} \mathbb{E}_{\theta \in \Theta}[P(d, z, y, \theta)], \\
\text{s.t. } & h(d, z, y, \theta) = 0, \\
& g(d, z, y, \theta) \leq 0
\end{align*}
\]

Multi-scenario Models for Variability

\[ \min_{d, z(\theta)} E_{\theta \in \Theta} [P(d, z(\theta), y, \theta)], \]
\[ s.t. \ h(d, z(\theta), y, \theta) = 0, \]
\[ g(d, z(\theta), y, \theta) \leq 0 \]


Multi-scenario Models for Variability

\[ \min f_0(d) + \sum_j \omega_j f_j (d, z_j, y_j, \theta_j) \]
\[ s.t. h_j (d, z_j, y_j, \theta_j) = 0 \]
\[ g_j (d, z_j, y_j, \theta_j) \leq 0 \]

Multi-scenario Models for Both

\[
\begin{align*}
\text{Min}_{d, z(\theta_v)} & \quad E_{\theta \in \Theta} [P(d, z(\theta_v), y, \theta), \\
\text{s.t.} & \quad h(d, z(\theta_v), y, \theta) = 0, \\
& \quad g(d, z(\theta_v), y, \theta) \leq 0]
\end{align*}
\]


---

Multi-scenario Models for Both

\[
\begin{align*}
\text{Min} & \quad f_0(d) + \sum_{i,k} \omega_{ik} f_{ik}(d, z_k, y_{ik}, \theta_{v,k}, \theta_{p,i}) \\
\text{s.t.} & \quad h_{ik}(d, z_k, y_{ik}, \theta_{v,k}, \theta_{p,i}) = 0 \\
& \quad g_{ik}(d, z_k, y_{ik}, \theta_{v,k}, \theta_{p,i}) = 0
\end{align*}
\]

Example: Williams-Otto Process
(Rooney, B., 2003)

\[ A + B \xrightarrow{\alpha_1} C \]
\[ C + B \xrightarrow{\alpha_2} P + E \]
\[ P + C \xrightarrow{\alpha_3} G \]

- Uncertain model parameters: \( a_1, a_2 \) and \( a_3 \)
- Varying process parameters: \( F_A = 10000(1 \pm \delta) \) and \( F_B = 40000(1 \pm \delta) \)

Uncertainty and Variability: Williams-Otto Process
(Rooney, B., 2003)

- Uncertain model parameters, \( a_1, a_2 \) and \( a_3 \)
- Varying process parameters: \( F_A = 10000(1 \pm \delta) \) and \( F_B = 40000(1 \pm \delta) \)
Solving Multi-scenario Problems: Interior Point Method

Min \( f_0(d) + \sum_j \omega_j f_j(d, z_j, y_j, \theta_j) \)
\( s.t. h_j(d, z_j, y_j, \theta_j) = 0 \)
\( g_j(d, z_j, y_j, \theta_j) + s_j = 0, s_j \geq 0 \)

Min \( f_0(p) + \sum_j \omega_j f_j(p, x_j) \)
\( s.t. c_j(p, x_j) = 0, \quad p, x_j \geq 0 \)

\[
\begin{align*}
\min \ f_0(p) + \sum_j \omega_j f_j(p, x_j) - \mu \left( \sum_{j \in J_B} \ln x_j + \sum_{j \in J_K} \ln p_j \right) \\
\text{s.t. } c_j(p, x_j) = 0 \\
\mu^i \rightarrow 0 \Rightarrow [x(\mu^i), p(\mu^i)] \rightarrow [x^*, p^*]
\end{align*}
\]

Newton Step for IPOPT

\[
\begin{bmatrix}
K_1 & \cdots & K_N \\
\vdots & & \vdots \\
K_1 & \cdots & K_N \\
\end{bmatrix}
\begin{bmatrix}
w_1^f \\
w_2^f \\
w_3^f \\
\vdots \\
w_N^f \\
\end{bmatrix}
\begin{bmatrix}
u_1 \\
u_2 \\
u_3 \\
\vdots \\
u_N \\
\end{bmatrix}
= -
\begin{bmatrix}
r_1 \\
r_2 \\
r_3 \\
\vdots \\
r_N \\
\end{bmatrix}
\]

\[
K_i = 
\begin{bmatrix}
(\nabla_{x_i} x_i L_k^k + (X_i^k)^{-1} y_i^k) & \nabla_{x_i} c_i (x_i^k, p_k) \\
\nabla_{x_i} c_i (x_i^k, p_k)^T & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
u_i \\
\Delta x_i \\
\Delta \lambda_i \\
u_p \\
\Delta p
\end{bmatrix}
= 
\begin{bmatrix}
u_i \\
\Delta x_i \\
\Delta \lambda_i \\
u_p \\
\Delta p
\end{bmatrix}
\]

\[
K_p = 
\begin{bmatrix}
\nabla_{p} L^k + (p_k)^{-1} y_p^k \\
\nabla_{p} c_i (p_k)^T \\
\nabla_{p} c_i (p_k)^T \\
\nabla_{p} c_i (p_k)^T
\end{bmatrix}
\]

\[
\begin{bmatrix}
w_i \\
\Delta \lambda_i \\
\Delta \lambda_i \\
w_i
\end{bmatrix}
= 
\begin{bmatrix}
w_i \\
\Delta \lambda_i \\
\Delta \lambda_i \\
w_i
\end{bmatrix}
\]
**Schur Complement Decomposition Algorithm**

Key Steps:
1. \( (K_{pp} - \sum_{i} w_i^T K_i^{-1} w_i) \Delta u_p = r_p - \sum_{i} w_i^T K_i^{-1} r_i \)
2. \( K_i \Delta u_i = r_i - w_i \Delta u_p \)

**Nonlinear Optimization Engines**

Evolution of NLP Solvers:

\[ \textit{process optimization for design, control and operations} \]

\[ \text{SQP} \rightarrow \text{rSQP} \rightarrow \text{IPOPT} \]

IPOPT 3.x

'00s: Simultaneous dynamic optimization over 1 000 000 variables and constraints

Object Oriented Codes to tailor structure, architecture to problems
Multi-scenario Design Model

\[
\begin{align*}
\text{Min } & f_0(d) + \sum_i f_i(d, x_i) \\
\text{s.t. } & h_i(x_i, d) = 0, \ i = 1, \ldots, N \\
& g_i(x_i, d) \leq 0, \ i = 1, \ldots, N \\
& r(d) \leq 0
\end{align*}
\]

Variables:
- \( x \): state (\( z \)) and control (\( u \)) variables in each operating period
- \( d \): design variables (e.g. equipment parameters) used
- \( \delta_i \): substitute for \( d \) in each period and add \( \delta_i = d \)

Composite NLP

\[
\begin{align*}
\text{Min } & \sum_i (f_i(\delta_i, x_i) + f_0(\delta_i)/N) \\
\text{s.t. } & h_i(x_i, \delta_i) = 0, \ i = 1, \ldots, N \\
& g_i(x_i, \delta_i) + s_i = 0, \ i = 1, \ldots, N \\
& 0 \leq s_i, \ d - \delta_i \geq 0, \ i = 1, \ldots, N \\
& r(d) \leq 0
\end{align*}
\]

Internal Decomposition Implementation

- Water Network Base Problem
  - 36,000 variables
  - 600 common variables
- Testing
  - Vary # of scenarios
  - Vary # of common variables
Parallel Schur-Complement Scalability

Multi-scenario Optimization
- Single Optimization over many scenarios, performed on parallel cluster

Water Network Case Study
- 1 basic model
  - Nominal design optimization
- 32 possible uncertainty scenarios
  - Form individual blocks

Determine Injection time profiles as common variables

Characteristics
- 36,000 variables per scenario
- 600 common variables

Summary and Conclusions

**Optimization Algorithms**
- Unconstrained Newton and Quasi Newton Methods
- KKT Conditions and Specialized Methods
- Reduced Gradient Methods (GRG2, MINOS)
- Successive Quadratic Programming (SQP)
- Reduced Hessian SQP
- Interior Point NLP (IPOPT)

**Process Optimization Applications**
- Modular Flowsheet Optimization
- Equation Oriented Models and Optimization
- Realtime Process Optimization
- Blending with many degrees of freedom

**Further Applications**
- Sensitivity Analysis for NLP Solutions
- Multi-Scenario Optimization Problems