Nonlinear Programming: Concepts, Algorithms and Applications

L. T. Biegler
Chemical Engineering Department
Carnegie Mellon University
Pittsburgh, PA
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

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

Large-Scale Nonlinear Programming
• Data Reconciliation
• Real-time Process Optimization

Further Applications
• Sensitivity Analysis for NLP Solutions
• Multiperiod 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**


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

\[
\begin{align*}
\min & \quad f(x, y) \\
\text{st} & \quad h(x, y) = 0 \\
& \quad g(x, y) \leq 0 \\
& \quad x \in \mathbb{R}^n, \quad y \in \{0, 1\}^m
\end{align*}
\]
# Optimization in Design, Operations and Control

<table>
<thead>
<tr>
<th></th>
<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>
<td>X</td>
</tr>
<tr>
<td>MENS</td>
<td>X</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></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reactors</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Equipment Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Flowsheeting</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Scheduling</td>
<td>X</td>
<td></td>
<td></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>
<td></td>
</tr>
<tr>
<td>Real-time optimization</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Linear MPC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Nonlinear MPC</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Hybrid</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Unconstrained Multivariable Optimization

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

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

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

$$\text{Min } \left\{ C_T \frac{\pi D^2}{2} + C_S \pi DL = \text{cost} \right\}$$

s.t. $$V - \frac{\pi D^2 L}{4} = 0$$

Convert to Unconstrained (Eliminate L)

$$\text{Min } \left\{ C_T \frac{\pi D^2}{2} + C_S \frac{4V}{D} = \text{cost} \right\}$$

$$\frac{d(\text{cost})}{dD} = C_T \pi D - \frac{4VC_S}{D^2} = 0$$

$$D = \left( \frac{4V}{\pi} \frac{C_S}{C_T} \right)^{1/3}$$

$$L = \left( \frac{4V}{\pi} \frac{C_T}{C_S} \right)^{1/3}$$

$$\Rightarrow \frac{L}{D} = \frac{C_T}{C_S}$$

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

- Find a *local minimum* point $x^*$ for $f(x)$ for feasible region defined by constraint functions: $f(x^*) \leq f(x)$ *for all x satisfying the constraints in some neighborhood around x* *(not for all $x \in X$)*

- Finding and verifying *global solutions* will not be considered here.

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

- A local solution to the NLP is also a global solution under the following *sufficient* conditions based on convexity.

- $f(x)$ 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 $X$. 

Linear Algebra - Background

Some Definitions

- Scalars - Greek letters, \( \alpha, \beta, \gamma \)
- Vectors - Roman Letters, lower case
- Matrices - Roman Letters, upper case
- Matrix Multiplication:
  \[ C = A \cdot B \text{ if } A \in \mathbb{R}^{n \times m}, B \in \mathbb{R}^{m \times p} \text{ and } C \in \mathbb{R}^{n \times p}, C_{ij} = \sum_k A_{ik} B_{kj} \]
- 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
  \[ \det(A) = \sum_i (-1)^{i+j} A_{ij} A_{\text{ij}} \text{ for any } j, \text{ or} \]
  \[ \det(A) = \sum_j (-1)^{i+j} A_{ij} A_{\text{ij}} \text{ for any } i, \text{ where } A_{\text{ij}} \text{ is the determinant of an order } n-1 \text{ matrix with row } i \text{ and column } j \text{ removed.} \]
  \[ \det(I) = 1 \]
- Singular Matrix: \( \det(A) = 0 \)
Linear Algebra - Background

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

\[
\nabla f = \begin{bmatrix}
\frac{\partial f}{\partial x_1} \\
\frac{\partial f}{\partial x_2} \\
\vdots \\
\frac{\partial f}{\partial x_n}
\end{bmatrix}
\]

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

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

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
  \[ \det(AB) = \det(A) \det(B); \quad \det(A) = \det(A^T) \]
  \[ \det(\alpha A) = \alpha^n \det(A); \quad \det(A) = \prod_i \lambda_i(A) \]

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

• **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 = \(\max_i |x_i|\))

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

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

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)

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

$V$ - 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 (or Negative) Curvature
Positive (or Negative) Definite Hessian

Both eigenvalues are strictly positive or negative
- A is positive definite or negative definite
- Stationary points are minima or 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

\[ \text{Min } Q(x) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}^T x + \frac{1}{2} x^T \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} x \]

\[ AV = V\Lambda \quad \text{with} \quad A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \]

\[ V^T AV = \Lambda = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \quad \text{with} \quad V = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \]

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

\[ z = V^T x = \begin{bmatrix} (x_1 - x_2)/\sqrt{2} \\ (x_1 + x_2)/\sqrt{2} \end{bmatrix} \]

\[ x = Vz = \begin{bmatrix} (x_1 + x_2)/\sqrt{2} \\ (-x_1 + x_2)/\sqrt{2} \end{bmatrix} \]

\[ z^* = \begin{bmatrix} 0 \\ 2/(3\sqrt{2}) \end{bmatrix} \]

\[ x^* = \begin{bmatrix} 1/3 \\ 1/3 \end{bmatrix} \]
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 the solution?

2. Benchmarks on Large Class of Test Problems

Representative Problem (Hughes, 1981)

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

\( a = [0.3, 0.6, 0.2] \)
\( b = [5, 26, 3] \)
\( c = [40, 1, 10] \)
\( x^* = [0.7395, 0.3144] \quad f(x^*) = 5.0893 \)
Three Dimensional Surface and Curvature for Representative Test Problem

Regions where minimum eigenvalue is less than:

\([0, -10, -50, -100, -150, -200]\)
What conditions characterize an optimal solution?

Unconstrained Local Minimum

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

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

For smooth functions, why are contours around optimum elliptical?

Taylor Series in n dimensions about \( x^* \):

\[ f(x) = f(x^*) + \nabla f(x^*)^T (x - x^*) + \frac{1}{2} (x - x^*)^T \nabla^2 f(x^*) (x - x^*) \]

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 $\approx 0$

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

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

- $f(x)$ is convex (concave) if for all $x \in \mathbb{R}^n$, $\nabla^2 f(x)$ is positive (negative) semidefinite
  i.e. $\min_j \lambda_j \geq 0$ (max$_j \lambda_j \leq 0$
- 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:
  $$\left\| x^{k+1} - x^* \right\| = K \left\| x^k - x^* \right\|^2$$
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 $B^k$.

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

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

  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)

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

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

where:

$$s = x^{k+1} - x^k$$
$$y = \nabla f(x^{k+1}) - \nabla f(x^k)$$
Quasi-Newton Methods

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

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

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

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 α is minimized.
3) BFGS is more stable and performs better than DFP, in general.
4) For n ≤ 100, these are the best methods for general purpose problems if second derivatives are not available.
Quasi-Newton Method - BFGS
Convergence Path

Starting Point
[0.8, 0.2] 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
Problem: \[ \text{Min}_x f(x) \]
\[ \text{s.t.} \]
\[ g(x) \leq 0 \]
\[ h(x) = 0 \]

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 Unique Optimum
- \( f(x) \) must be \textit{convex}, and
- feasible region must be convex,
  - i.e. \( g(x) \) are all \textit{convex}
  - \( h(x) \) are all \textit{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.

\[
x_1, y_1 \geq R_1 \quad x_1 \leq B - R_1, \quad y_1 \leq A - R_1 \\
x_2, y_2 \geq R_2 \quad x_2 \leq B - R_2, \quad y_2 \leq A - R_2 \\
x_3, y_3 \geq R_3 \quad x_3 \leq B - R_3, \quad y_3 \leq A - R_3 
\]

in box

\[
\begin{align*}
(x_1 - x_2)^2 + (y_1 - y_2)^2 &\geq (R_1 + R_2)^2 \\
(x_1 - x_3)^2 + (y_1 - y_3)^2 &\geq (R_1 + R_3)^2 \\
(x_2 - x_3)^2 + (y_2 - y_3)^2 &\geq (R_2 + R_3)^2 
\end{align*}
\]

no overlaps

\[x_1, x_2, x_3, y_1, y_2, y_3, A, B \geq 0\]
Characterization of Constrained Optima

Linear Program

(Alternate Optima)

Convex Objective Functions
Linear Constraints

Nonconvex Region
Multiple Optima

Nonconvex Objective
Multiple Optima
What conditions characterize an optimal solution?

Unconstrained Local Minimum

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

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

\[
\begin{align*}
\text{Min} & \quad f(x) \\
\text{s.t.} & \quad g(x) \leq 0 \\
\text{Analogy:} & \quad \text{Ball rolling down valley pinned by fence} \\
\text{Note:} & \quad \text{Balance of forces (}\nabla f, \nabla g_1) \\
\end{align*}
\]
Problem: Min $f(x)$

s.t. $g(x) \leq 0$
     $h(x) = 0$

Analogy: Ball rolling on rail pinned by fences
Balance of forces: $\nabla f, \nabla g_1, \nabla h$
Optimality conditions for local optimum

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)
\[ u \geq 0 \, \text{(Inequalities act in only one direction)} \]
\[ g(x^*) \leq 0, \ h(x^*) = 0 \, \text{(Feasibility)} \]
\[ u_j g_j(x^*) = 0 \, \text{(Complementarity: either } g_j(x^*) = 0 \text{ or } u_j = 0) \]
\[ u, v \text{ are "weights" for "forces," known as KKT multipliers, shadow prices, dual variables} \]

“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, \( [\nabla g_A(x^*) \ \nabla h(x^*)] \), to be linearly independent. Also, under LICQ, KKT multipliers are uniquely determined.”

Necessary (Sufficient) Second Order Conditions

- Positive curvature in "constraint" directions.
- \[ p^T \nabla^2 L(x^*) p \geq 0 \, \text{ (} p^T \nabla^2 L(x^*) p > 0 \text{)} \]
  where \( p \) are the constrained directions: \( \nabla g_A(x^*)^T p = 0, \ \nabla h(x^*)^T p = 0 \)
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:

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

First Order KKT conditions:

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

\[ u_1 (x-a) = 0 \]

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

\(-a \leq x \leq a\) \hspace{1cm} u_1, u_2 \geq 0

Consider three cases:

- \( u_1 > 0, u_2 = 0 \) \hspace{1cm} Upper bound is active, \( x = a, u_1 = -2a, u_2 = 0 \)
- \( u_1 = 0, u_2 > 0 \) \hspace{1cm} Lower bound is active, \( x = -a, u_2 = -2a, u_1 = 0 \)
- \( u_1 = u_2 = 0 \) \hspace{1cm} Neither bound is active, \( u_1 = 0, u_2 = 0, x = 0 \)

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

\[ \nabla_{xx} L(x^*, u^*) = 2 \]

\[ p^T \nabla_{xx} L(x^*, u^*) p = 2 (\Delta x)^2 > 0 \]
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:
\(\nabla L(x, u) = -2x + u_1 - u_2 = 0\)
\(u_1(x-a) = 0\)
\(u_2(-a-x) = 0\)
\(-a \leq x \leq a, u_1, u_2 \geq 0\)

Consider three cases:
• \(u_1 > 0, u_2 = 0\) Upper bound is active, \(x = a, u_1 = 2a, u_2 = 0\)
• \(u_1 = 0, u_2 > 0\) 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 \((x^*, u_1, u_2 = 0)\)
\(\nabla_{xx} L(x^*, u^*) = -2\)
\(p^T \nabla_{xx} L(x^*, u^*) p = -2(\Delta x)^2 < 0\)
**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:

\[
p^T \nabla_{xx} L (x^*, u^*, v^*) p > 0
\]

for all allowable directions, is *vacuously* satisfied - because there are *no* allowable directions that satisfy \( \nabla g_A(x^*)^T p = 0 \). 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

\[
[f(x^*, a + \Delta a) - f(x^*, a)]/\Delta a = 2a + \Delta a
\]

\[
df(x^*)/da = 2a = u_1
\]
Special Cases of Nonlinear Programming

Linear Programming:
\[ \text{Min } c^T x \]
\[ \text{s.t. } Ax \leq b \]
\[ Cx = d, \quad x \geq 0 \]
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.
Simplex Method

Min \(-2x_1 - 3x_2\)

s.t. \(2x_1 + x_2 \leq 5\)

\(x_1, x_2 \geq 0\)

\(\Rightarrow\)

Min \(-2x_1 - 3x_2\)

s.t. \(2x_1 + x_2 + x_3 = 5\)

\(x_1, x_2, x_3 \geq 0\)

(Add slack variable)

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

<table>
<thead>
<tr>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(x_3)</th>
<th>(f)</th>
<th>(b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>(x_1)</th>
<th>(x_2)</th>
<th>(x_3)</th>
<th>(f)</th>
<th>(b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>-4</td>
<td>0</td>
<td>-3</td>
<td>1</td>
<td>-15</td>
</tr>
</tbody>
</table>

\(f\) can no longer be decreased! \(Optimal\)

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

Problem: \[ \text{Min} \quad a^T x + \frac{1}{2} x^T B x \]
\[ \text{A} x \leq b \]
\[ C x = d \]

1) Can be solved using LP-like techniques:
   (Wolfe, 1959)
   \[ \Rightarrow \quad \text{Min} \quad \sum_j (z_{j+} + z_{j-}) \]
   \[ \text{s.t.} \quad a + Bx + A^T u + C^T v = z_+ - z_- \]
   \[ Ax - b + s = 0 \]
   \[ Cx - d = 0 \]
   \[ s, z_+ , z_- \geq 0 \]
   \[ \{ u_j , s_j = 0 \} \]
   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 \( i \)
\( r_i(t) \) - \((1 + \text{rate of return})\) for investment \( i \) in year \( t \).
\( \mu_i \) - average \( r(t) \) over \( T \) years, i.e.

\[
\mu_i = \frac{1}{T} \sum_{t=1}^{T} r_i(t)
\]

\[
\text{Max} \sum_i \mu_i x_i
\]

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

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

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

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

Variance/Covariance Matrix, S

$$\{ S \} _{ij} = \sigma _{ij} ^2 = \frac{1}{T} \sum _{t=1} ^{T} (r_i(t) - \mu _i)(r_j(t) - \mu _j)$$

Max $$x ^T S x$$

s . t. $$\sum _i x _i = 1$$

$$\sum _i \mu _i x _i \geq R$$

$$x _i \geq 0$$, etc .

Example: 3 investments

<table>
<thead>
<tr>
<th>Investment</th>
<th>$\mu _i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. IBM</td>
<td>1.3</td>
</tr>
<tr>
<td>2. GM</td>
<td>1.2</td>
</tr>
<tr>
<td>3. Gold</td>
<td>1.08</td>
</tr>
</tbody>
</table>

$$S = \begin{bmatrix} 3 & 1 & -0.5 \\ 1 & 2 & 0.4 \\ -0.5 & 0.4 & 1 \end{bmatrix}$$
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;
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.

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

LOWER LEVEL UPPER MARGINAL
- - - - EQU LP . . . . . . 1.000
- - - - EQU E2 1.000 1.000 1.000 1.200

LOWER LEVEL UPPER MARGINAL
- - - - VAR IBM 0.750 0.750 0.100 - - - - VAR GM . . 0.250 0.750 .
- - - - VAR GOLD . . . . . 0.750 . -0.120
- - - - VAR OBJLP -INF 1.275 +INF .

**** 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 3
BLOCKS OF VARIABLES 4 SINGLE VARIABLES 4
NON ZERO ELEMENTS 10 NON LINEAR N-Z 3
DERIVATIVE POOL 8 CONSTANT POOL 3
CODE LENGTH 95
GENERATION TIME = 2.360 SECONDS
EXECUTION TIME = 3.510 SECONDS
Portfolio Planning Problem - GAMS

SOLVE SUMMARY
MODEL PORTLP OBJECTIVE OBJLP
TYPE LP DIRECTION MAXIMIZE
SOLVER MINOS5 FROM LINE 34
**** 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 0
MINOS 5.3 (Nov. 1990) Ver: 225-DOS-02
B.A. Murtagh, University of New South Wales
and
P.E. Gill, W. Murray, M.A. Saunders and M.H. Wright
Systems Optimization Laboratory, Stanford University.

EXIT - - OPTIMAL SOLUTION FOUND
MAJOR ITNS, LIMIT 1
FUNOBJ, FUNCON CALLS 8
SUPERBASICS 1
INTERPRETER USAGE .21
NORM RG / NORM PI 3.732E-17
LOWER LEVEL UPPER MARGINAL
- - - - EQU QP . . . 1.000
- - - - EQU E1 1.150 1.150 +INF 1.216
- - - - EQU E2 1.000 1.000 1.000 -0.556
- - - - VAR IBM . 0.183 0.750 EPS
- - - - VAR GM . 0.248 0.750
- - - - VAR GOLD . 0.569 0.750
- - - - VAR OBJLP -INF 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)

Min $f(x)$

s.t. $g(x) + s = 0$ (add slack variable)
$c(x) = 0$
$a \leq x \leq b, s \geq 0$

Min $f(z)$

‘$\Rightarrow$’ s.t. $h(z) = 0$
$a \leq z \leq b$

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

Analogy to linear programming. Superbasics required only if nonlinear problem

- Solve unconstrained problem in space of superbasic variables.
  - Let $z^T = [z_S^T \ z_B^T \ z_N^T]$ optimize wrt $z_S$ with $h(z_S, z_B, z_N) = 0$
  - Calculate constrained derivative or reduced gradient wrt $z_S$.

- Dependent variables are $z_B \in \mathbb{R}^m$
- Nonbasic variables $z_N$ (temporarily) fixed
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 \( h(z) = 0 \), we have:

\[
dh = \left[ \frac{\partial h}{\partial z_S} \right]^T dz_S + \left[ \frac{\partial h}{\partial z_B} \right]^T dz_B = 0
\]

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

This leads to:

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

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

• Solve problem in reduced space of \( z_S \) variables
Example of Reduced Gradient

Min \( x_1^2 - 2x_2 \)

s.t. \( 3x_1 + 4x_2 = 24 \)

\( \nabla h^T = [3 \ 4], \ \nabla f^T = [2x_1 - 2] \)

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

\[
\frac{df}{dz_s} = \frac{\partial f}{\partial z_s} - \nabla_{z_s} h \left[ \nabla_{z_B} h \right]^{-1} \frac{\partial f}{\partial z_B}
\]

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

If \( \nabla h^T \) is \( m \times n \); \( \nabla_{z_S} h^T \) is \( m \times (n-m) \); \( \nabla_{z_B} h^T \) is \( m \times m \)

\( \frac{df}{dz_S} \) is the change in \( f \) along constraint direction per unit change in \( z_S \)
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 $h(z_S^k + \alpha d, z_B, z_N) = 0$
   - 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.
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 $h(z_S, z_B, z_N) = 0$ can get very expensive because $\nabla h$ is required
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 (MINOS/Augmented)**

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

\[
\begin{align*}
\text{Min} & \quad f(x) \\
\text{s.t.} & \quad Ax \leq b \\
& \quad Cx = d 
\end{align*}
\]

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 \( z^k \)
   \[
   D^k z = c^k
   \]
3. Let \( \psi = f(z) + v^T h(z) + \beta (h^T h) \) (Augmented Lagrange),
4. Solve linearly constrained problem:
   \[
   \begin{align*}
   \text{Min} & \quad \psi(z) \\
   \text{s.t.} & \quad Dz = c \\
   & \quad a \leq z \leq b 
   \end{align*}
   \]
   using reduced gradients to get \( z^{k+1} \)
5. Set \( k = k + 1 \), go to 3.
6. Algorithm terminates when no movement between steps 3) 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 $\|\nabla f(x^*)\| \leq 10^{-6}$
**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**

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

\[
h(x^*) = 0
\]

\[
g_A(x^*) = 0,
\]

where \(g_A\) are the active constraints.

**Newton - Step**

\[
\begin{bmatrix}
\nabla_{xx} L & \nabla g_A & \nabla h \\
\nabla g_A^T & 0 & 0 \\
\nabla h^T & 0 & 0 \\
\end{bmatrix}
\begin{bmatrix}
\Delta x \\
\Delta u \\
\Delta v \\
\end{bmatrix} = - \begin{bmatrix}
\nabla_x L (x^k, u^k, v^k) \\
g_A (x^k) \\
h(x^k) \\
\end{bmatrix}
\]

**Requirements:**
- \(\nabla_{xx} L\) must be calculated and should be ‘regular’
- correct active set \(g_A\)
- good estimates of \(u^k, v^k\)
1. **Wilson (1963)**
   - active set can be determined by solving QP:
     \[
     \begin{align*}
     \text{Min} & \quad \nabla f(x_k)^T d + 1/2 d^T \nabla_{xx} L(x_k, u_k, v_k) d \\
     \text{s.t.} & \quad g(x_k) + \nabla g(x_k)^T d \leq 0 \\
                & \quad h(x_k) + \nabla h(x_k)^T d = 0
     \end{align*}
     \]

2. **Han (1976), (1977), Powell (1977), (1978)**
   - approximate \( \nabla_{xx} L \) 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 $\nabla_{xx}L$?

- need to get second derivatives for $f(x)$, $g(x)$, $h(x)$.
- need to estimate multipliers, $u^k$, $v^k$; $\nabla_{xx}L$ may not be positive semidefinite

$\Rightarrow$ Approximate $\nabla_{xx}L(x^k, u^k, v^k)$ by $B^k$, a symmetric positive definite matrix.

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

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
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 \textit{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**

$$\psi(x) = f(x) + u^T g(x) + v^T h(x) + \frac{\eta}{2} \left\{ \sum (h_j(x))^2 + \sum \max (0, g_j(x))^2 \right\}$$
Newton-Like Properties for SQP

Fast Local Convergence

\[ B = \nabla_{xx} L \quad \text{Quadratic} \]
\[ \nabla_{xx} L \text{ is p.d and } B \text{ is Q-N} \quad \text{1 step Superlinear} \]
\[ B \text{ is Q-N update, } \nabla_{xx} L \text{ not p.d} \quad \text{2 step Superlinear} \]

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 \( \nabla f(x^k) \), \( \nabla g(x^k) \), \( \nabla h(x^k) \).

2. If \( k > 0 \), update \( B^k \) using the BFGS Formula.

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

If KKT error less than tolerance: \( \| \nabla L(x^*) \| \leq \varepsilon, \| h(x^*) \| \leq \varepsilon, \| g(x^*)_+ \| \leq \varepsilon \). 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

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

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

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

$x^* = [0.5, 0.375]$. 
Start from the origin \((x_0 = [0, 0]^T)\) with \(B^0 = I\), form:

\[
\text{Min } d_2 + \frac{1}{2} (d_1^2 + d_2^2)
\]

\[
\text{s.t. } d_2 \geq 0
\]
\[
d_1 + d_2 \geq 1
\]
\[
d = [1, 0]^T \text{ with } \mu_1 = 0 \text{ and } \mu_2 = 1.
\]
From $x_1 = [0.5, 0]^T$ with $B^l = I$
(no update from BFGS possible), form:

$$
\text{Min } d_2 + 1/2 (d_1^2 + d_2^2)
$$

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

\[
\begin{align*}
\min_{x \in \mathbb{R}^n} f(x) \\
\text{s.t.} & \quad c(x) = 0 \\
& \quad x \geq 0
\end{align*}
\]

Can generalize for \(a \leq x \leq b\)

Barrier Approach

\[
\begin{align*}
\min_{x \in \mathbb{R}^n} \varphi_\mu(x) &= f(x) - \mu \sum_{i=1}^{n} \ln x_i \\
\text{s.t.} & \quad c(x) = 0
\end{align*}
\]

\[\Rightarrow \text{As } \mu \to 0, \quad x^*(\mu) \to x^*\]  

Fiacco and McCormick (1968)
Solution of the Barrier Problem

⇒ Newton Directions (KKT System)

\[ \nabla f(x) + A(x)\lambda - v = 0 \]
\[ XVe - \mu e = 0 \]
\[ c(x) = 0 \]

⇒ Reducing the System

\[ d_v = \mu X^{-1}e - v - X^{-1}Vd_x \]

\[ \begin{bmatrix} \nabla \phi \mu \end{bmatrix} = \sum \begin{bmatrix} W + \Sigma A \end{bmatrix} \begin{bmatrix} A^T \lambda^+ \\ 0 \end{bmatrix} \begin{bmatrix} d_x \\ c \end{bmatrix} = X^{-1}V \]

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

**Merit Function**

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

Aug’d 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 \( \tau \sim 0.99 \).

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

• 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)

\[
\theta(x) = ||c(x)||
\]
Implementation Details

Modify KKT (full space) matrix if nonsingular

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

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

KKT matrix factored by MA27

Feasibility restoration phase

\[
\text{Min } \| c(x) \|_1 + \| x - x_k \|_Q^2
\]

\(x_l \leq x_k \leq x_u\)

Apply Exact Penalty Formulation

Exploit same structure/algorithm to reduce infeasibility
# Details of IPOPT Algorithm

## Options

### Line Search Strategies
- $L^2$ exact penalty merit function
- Augmented Lagrangian function
- Filter method (adapted from Fletcher and Leyffer)

### Hessian Calculation
- BFGS (reduced space)
- SR1 (reduced space)
- Exact full Hessian (direct)
- Exact reduced Hessian (direct)
- Preconditioned CG

## Comparison

- **34 COPS Problems**
  - (600 - 160,000 variables)
- **486 CUTE Problems**
  - (2 – 50,000 variables)
- **56 MITT Problems**
  - (12097 – 99998 variables)

## Performance Measure

- $r_{p,l} = \frac{\text{iter}_{p,l}}{\text{iter}_{p,\text{min}}}$
- $P(\tau) = \text{fraction of problems with } \log_2(r_{p,l}) < \tau$
IPOPT Comparison with KNITRO and LOQO

• IPOPT has better performance, robustness on CUTE, MITT and COPS problem sets
• Similar results appear with iteration counts
• Can be downloaded from http://www.coin-or.org
• See links for additional information
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.
   \[\Rightarrow\] Difficulty with many nonlinearities

---

**Small, Nonlinear Problems** - SQP solves QP's, not LCNLP's, fewer function calls.

**Large, Mostly Linear Problems** - MINOS performs sparse constraint decomposition. Works efficiently in reduced space if function calls are cheap!

**Exploit Both Features** – IPOPT takes advantages of few function evaluations and large-scale linear algebra, but requires exact second derivatives
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
A student version of GAMS is now 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. \( v(xy - z^2)^{1/2} = 3 \) \( \Rightarrow \) \( vu = 3 \)
   \( u^2 - (xy - z^2) = 0, \ u \geq 0 \)

3) Exploit linear constraints as much as possible, e.g. mass balance
   \[
   x_i L + y_i V = F z_i \ \Rightarrow \ l_i + v_i = f_i \\
   L - \sum l_i = 0
   \]

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. \( xT + y - T^2 y = 5 \)
   \( 4x - 5Ty + Tx = 7 \)
   \( 0 \leq T \leq 1 \)
   (If \( T \) is fixed \( \Rightarrow \) solve LP) \( \Rightarrow \) put \( T \) in outer optimization loop.
Process Optimization
Problem Definition and Formulation

State of Nature and Problem Premises

- Restrictions: Physical, Legal, Economic, Political, etc.
- Desired Objective: Yield, Economic, Capacity, etc.

Decisions

Mathematical Modeling and Algorithmic Solution

Process Model Equations

- Constraints
- Objective Function
- Additional Variables
Large Scale NLP Algorithms

Motivation: Improvement of Successive Quadratic Programming as Cornerstone Algorithm

→ Process optimization for design, control and operations

Evolution of NLP Solvers:

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

\[ \text{rSQP}++ \]

1999-: Simultaneous dynamic optimization over 1 000 000 variables and constraints

Current: Tailor structure, architecture and problems
Flowsheet Optimization Problems - Introduction

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 - Features

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 looks
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

Min \( f(x), \ s.t. \ g(x) \leq 0 \)

For single degree of freedom:

- work in space defined by curve below.
- requires repeated (expensive) recycle convergence
Expanded Region with Feasible Path
"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.
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 NH₃ product. Vapor from the two stage flash forms the recycle and is recompressed.
Ammonia Process Optimization

Optimization Problem

Max {Total Profit @ 15% over five years}

s.t. • 105 tons NH3/yr.
  • Pressure Balance
  • No Liquid in Compressors
  • $1.8 \leq \frac{H2/N2}{G} \leq 3.5$
  • \(T_{react} \leq 1000\)° F
  • NH3 purged \leq 4.5 lb mol/hr
  • NH3 Product Purity \geq 99.9 \%
  • Tear Equations

Performance Characterstics

• 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></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 $\varepsilon$!

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

Module Equations
\[ c(v, x, s, p, y) = 0 \]

Sensitivity Equations
\[
\begin{align*}
\frac{dy}{dx} & \\
\frac{ds}{dx} & \\
\frac{dy}{dp} & \\
\frac{ds}{dp} & 
\end{align*}
\]

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)
**Flash Recycle Optimization**

(2 decisions + 7 tear variables)

\[
\text{Max } S3(A)^2 - S3(B) - S3(A)^2 + S3(C) - S3(D) - (S3(E))^{1/2}
\]

**Ammonia Process Optimization**

(9 decisions and 6 tear variables)

**Flash Recycle Optimization**

(Mixer, Flash, P, Ratio, S1, S2, S3, S4, S5, S6, S7)

**Ammonia Process Optimization**

(Reactor, Hi P Flash, Lo P Flash, Ammonia Process Optimization)

**CPU Seconds (VS 3200)**

- Numerical
- Exact

**GRG**, **SQP**, **rSQP**
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 (\(\geq 100\ 000\))
- Many bounds and inequalities (\(\geq 100\ 000\))

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

- Many degrees of freedom (\(\geq 1000\))
  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:

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

- 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:

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

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

- \((A^T Y) d_Y = -c(x^k)\) is square, \(d_Y\) determined from bottom row.
- Cancel \(Y^T W Y\) and \(Y^T W Z\); (unimportant as \(d_Z, d_Y \to 0\))
- \((Y^T A) \lambda = -Y^T \nabla f(x^k)\), \(\lambda\) 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: \(d = Y d_Y + Z d_Z\)
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 ($A^T Z = 0$, $Y^T Z = 0$).

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

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

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)
6. Solve small QP for step \( d_Z \) in Null space.
   \[
   Min \quad (Z^T \nabla f(x^k) + w^k)^T \ d_Z + \frac{1}{2} d_Z^T B^k d_Z
   \]
   s.t. \( x_L \leq x^k + Y d_Y + Z d_Z \leq x_U \)
7. If error is less than tolerance stop. Else
8. Solve for multipliers using \( (Y^T A) \ \lambda = -Y^T \nabla f(x^k) \)
9. Calculate total step \( d = Y d_Y + Z d_Z \).
10. Find step size \( \alpha \) and calculate new point, \( x_{k+1} = x_k + \alpha \)
11. Continue from step 2 with \( k = k+1 \).
rSQP Results: Computational Results for General Nonlinear Problems
Vasantha Rajan 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></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>M</td>
<td>ME</td>
</tr>
<tr>
<td>Ramsey</td>
<td>34</td>
<td>23</td>
<td>10</td>
</tr>
<tr>
<td>Chenery</td>
<td>44</td>
<td>39</td>
<td>20</td>
</tr>
<tr>
<td>Korcege</td>
<td>100</td>
<td>96</td>
<td>78</td>
</tr>
<tr>
<td>Commodge</td>
<td>280</td>
<td>243</td>
<td>243</td>
</tr>
<tr>
<td>Ganges</td>
<td>357</td>
<td>274</td>
<td>274</td>
</tr>
</tbody>
</table>

* CPU Seconds - VAX 6320
## rSQP Results: Computational Results for Process Problems

Vasantharajan 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>Distill'n</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nonideal</td>
<td>569</td>
<td>567</td>
<td>567</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></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nonideal</td>
<td>977</td>
<td>975</td>
<td>975</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>
</tbody>
</table>

* CPU Seconds - VAX 6320 + MINOS (5.1) (F) Failed
## 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. SQP*</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>

The diagram illustrates a distillation process with trays labeled 1, 10, and 18, and heat exchangers denoted by $Q_{VK}$.
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
Optimization Case Study Characteristics

Model consists of 2836 equality constraints and only ten independent variables. It is also reasonably sparse and contains 24123 nonzero Jacobian elements.

\[ P = \sum_{i \in G} z_i C_i^G + \sum_{i \in E} z_i C_i^E + \sum_{m=1}^{NP} z_i C_i^{P_m} - U \]

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></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><strong>Heat Exchange Coefficient (TJ/°C)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Debutanizer Feed/Bottoms</td>
<td>6.565x10^-4</td>
<td>6.565x10^-4</td>
<td>5.000x10^-4</td>
<td>2.000x10^-4</td>
<td>6.565x10^-4</td>
<td>6.565x10^-4</td>
</tr>
<tr>
<td>Splitter Feed/Bottoms</td>
<td>1.030x10^-3</td>
<td>1.030x10^-3</td>
<td>5.000x10^-4</td>
<td>2.000x10^-4</td>
<td>1.030x10^-3</td>
<td>1.030x10^-3</td>
</tr>
<tr>
<td><strong>Pricing</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Propane ($/m^3)</td>
<td>180</td>
<td>180</td>
<td>180</td>
<td>180</td>
<td>300</td>
<td>180</td>
</tr>
<tr>
<td>Gasoline Base Price ($/m^3)</td>
<td>300</td>
<td>300</td>
<td>300</td>
<td>300</td>
<td>300</td>
<td>350</td>
</tr>
<tr>
<td>Octane Credit ($/(RON m^3))</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td>10</td>
</tr>
<tr>
<td><strong>Profit</strong></td>
<td>230968.96</td>
<td>239277.37</td>
<td>239267.57</td>
<td>236706.82</td>
<td>258913.28</td>
<td>370053.98</td>
</tr>
<tr>
<td>Change from base case</td>
<td>-</td>
<td>8308.41</td>
<td>(3.6%)</td>
<td>(2.5%)</td>
<td>27944.32</td>
<td>139085.02</td>
</tr>
<tr>
<td>($/d, %)</td>
<td></td>
<td></td>
<td>(3.6%)</td>
<td>(2.5%)</td>
<td>(12.1%)</td>
<td>(60.2%)</td>
</tr>
<tr>
<td><strong>Infeasible Initialization</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MINOS</td>
<td>Iterations</td>
<td>5 / 275</td>
<td>9 / 788</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>(Major/Minor)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU Time (s)</td>
<td>182</td>
<td>5768</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>rSQP</td>
<td>Iterations</td>
<td>5</td>
<td>20</td>
<td>12</td>
<td>24</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>(Major/Minor)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<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>
</tr>
<tr>
<td><strong>Parameter Initialization</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MINOS</td>
<td>Iterations</td>
<td>n/a</td>
<td>12 / 132</td>
<td>14 / 120</td>
<td>16 / 156</td>
<td>11 / 166</td>
</tr>
<tr>
<td></td>
<td>(Major/Minor)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPU Time (s)</td>
<td>n/a</td>
<td>462</td>
<td>408</td>
<td>1022</td>
<td>916</td>
</tr>
<tr>
<td>rSQP</td>
<td>Iterations</td>
<td>n/a</td>
<td>13</td>
<td>8</td>
<td>18</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>(Major/Minor)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<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>
</tr>
<tr>
<td><strong>Time rSQP / Time MINOS (%)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.8%</td>
<td>12.7%</td>
<td>10.7%</td>
<td>7.3%</td>
<td>5.7%</td>
<td>16.1%</td>
</tr>
</tbody>
</table>
Many degrees of freedom => full space IPOPT

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

- 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
Gasoline Blending

Pipelines

OIL TANKS

GAS STATIONS

Supply tanks

Intermediate tanks

Final Product tanks

FINAL PRODUCT TRUCKS

Gasoline Blending Here
Blending Problem & Model Formulation

\[ \text{max } \sum_t \left( \sum_k c_{t,k} f_{t,k} - \sum_i c_i f_{t,i} \right) \]

s.t.

\[ \sum_k f_{t,k} - \sum_i f_{t,ij} + v_{t+1,j} = v_t, j \]

\[ f_{t,k} - \sum_j f_{t,jk} = 0 \]

\[ \sum_k q_{t,j} f_{t,k} - \sum_i q_{t,ij} f_{t,ij} + q_{t+1,j} = q_t, j \]

\[ q_{t,k} - \sum_j q_{t,jk} = 0 \]

\[ q_{\text{min}} \leq q_{t,k} \leq q_{\text{max}} \]

\[ v_{t,j} \leq v_{t,j} \leq v_{j,\text{max}} \]

Model Formulation in AMPL

\( f \) & \( v \) ------ flowrates and tank volumes

\( q \) ------ tank qualities
Small Multi-day Blending Models
Single Qualities

Haverly, C. 1978 (HM)

<table>
<thead>
<tr>
<th></th>
<th>HM Day 1 (N= 13, M = 8, S = 8)</th>
<th>HM Day 25 (N= 325, M = 200, S = 200)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>no. of iterations</td>
<td>objective</td>
</tr>
<tr>
<td>LANCELOT</td>
<td>62</td>
<td>100</td>
</tr>
<tr>
<td>MINOS</td>
<td>15</td>
<td>400</td>
</tr>
<tr>
<td>SNOPT</td>
<td>36</td>
<td>400</td>
</tr>
<tr>
<td>KNITRO</td>
<td>38</td>
<td>100</td>
</tr>
<tr>
<td>LOOO</td>
<td>30</td>
<td>400</td>
</tr>
<tr>
<td>IPOPT, exact</td>
<td>31</td>
<td>400</td>
</tr>
<tr>
<td>IPOPT, L-BFGS</td>
<td>199</td>
<td>400</td>
</tr>
</tbody>
</table>

AHM Day 1 (N = 21, M = 14, S = 14)

<table>
<thead>
<tr>
<th></th>
<th>AHM Day 25 (N= 525, M = 300, S = 350)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>no. of iterations</td>
</tr>
<tr>
<td>LANCELOT</td>
<td>112</td>
</tr>
<tr>
<td>MINOS</td>
<td>29</td>
</tr>
<tr>
<td>SNOPT</td>
<td>60</td>
</tr>
<tr>
<td>KNITRO</td>
<td>44</td>
</tr>
<tr>
<td>LOOO</td>
<td>28</td>
</tr>
<tr>
<td>IPOPT, exact</td>
<td>28</td>
</tr>
<tr>
<td>IPOPT, L-BFGS</td>
<td>44</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>IHM Day 1 (N= 2003, M= 1595, S= 1449)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LANCELOT</td>
<td>388</td>
<td>$6.14 \times 10^4$</td>
<td>1.17 $\times 10^5$</td>
</tr>
<tr>
<td>MINOS</td>
<td>2238</td>
<td>$6.14 \times 10^4$</td>
<td>5.24 $\times 10^5$</td>
</tr>
<tr>
<td>SNOPT</td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>KNITRO</td>
<td>37</td>
<td>$1.00 \times 10^5$</td>
<td>1.58 $\times 10^5$</td>
</tr>
<tr>
<td>IPOPT, exact</td>
<td>21</td>
<td>$6.14 \times 10^4$</td>
<td>2.60</td>
</tr>
<tr>
<td>IPOPT, L-BFGS</td>
<td>52</td>
<td>$6.14 \times 10^4$</td>
<td>8.89</td>
</tr>
<tr>
<td>IHM Day 5 (N= 10 134, M= 8073, S= 7339)</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>8075</td>
<td>$1.39 \times 10^5$</td>
<td>3.08 $\times 10^5$</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>IPOPT, exact</td>
<td>39</td>
<td>$1.39 \times 10^5$</td>
<td>1.06 $\times 10^5$</td>
</tr>
<tr>
<td>IPOPT, L-BFGS</td>
<td>1000</td>
<td>$1.39 \times 10^5$</td>
<td>2.91 $\times 10^5$</td>
</tr>
<tr>
<td>IHM Day 10 (N= 20 826, M= 16 074, S= 15 206)</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>IPOPT, exact</td>
<td>65</td>
<td>$2.64 \times 10^4$</td>
<td>1.12 $\times 10^4$</td>
</tr>
<tr>
<td>IHM Day 15 (N= 31 743, M= 25 560, S= 23 073)</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>IPOPT, exact</td>
<td>110</td>
<td>$4.15 \times 10^4$</td>
<td>7.25 $\times 10^4$</td>
</tr>
</tbody>
</table>
Summary of Results – Dolan-Moré plot

Performance profile (iteration count)
Comparison of NLP Solvers: Data Reconciliation

Comparison of NLP Solvers: Data Reconciliation

Comparison of NLP Solvers: Data Reconciliation
Sensitivity Analysis for Nonlinear Programming

At nominal conditions, $p_0$

\[ \text{Min } f(x, p_0) \]
\[ \text{s.t. } c(x, p_0) = 0 \]
\[ a(p_0) \leq x \leq b(p_0) \]

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 parameteric uncertainties?
- Can this be analyzed easily?
Calculation of NLP Sensitivity

Take KKT Conditions

\[ \nabla L(x^*, p, \lambda, v) = 0 \]
\[ c(x^*, p_0) = 0 \]
\[ E^T x^* - bnd(p_0) = 0 \]

and differentiate and expand about \( p_0 \).

\[ \nabla_{px} L(x^*, p, \lambda, v)^T + \nabla_{xx} L(x^*, p, \lambda, v)^T \nabla_{x^T} + \nabla h(x^*, p, \lambda, v)^T \nabla_p \lambda^T + E \nabla_v v^T = 0 \]
\[ \nabla_{p} c(x^*, p_0)^T + \nabla_x c(x^*, p_0)^T \nabla_{p} x^T = 0 \]
\[ E^T (\nabla_p x^T - \nabla_p bnd^T) = 0 \]

Notes:
- A key assumption is that under strict complementarity, the active set will not change for small perturbations of \( p \).
- If an element of \( x^* \) is at a bound then \( \nabla_{p} x^i T = \nabla_{p} bnd^T \)
- Second derivatives are required to calculate sensitivities, \( \nabla_{p} x^T \)
- Is there a cheaper way to calculate \( \nabla_{p} x^T \)?
Decomposition for NLP Sensitivity

Let \( L(x^*, p, \lambda, \nu) = f(x^*) + \lambda^T c(x^*) + (E^T (x^* - \text{bnd}(p)))^T \nu \)

\[
\begin{bmatrix}
W & A & E \\
A^T & 0 & 0 \\
E^T & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\nabla_p x^{*T} \\
\nabla_p \lambda^T \\
\nabla_p \nu^T
\end{bmatrix} = -
\begin{bmatrix}
\nabla_{xp} L(x^*, p, \lambda, \nu)^T \\
\nabla_p c(x^*, p)^T \\
- E^T \nabla_p \text{bnd}^T
\end{bmatrix}
\]

• Partition variables into basic, nonbasic and superbasic

\( \nabla_p x^T = Z \nabla_p x_S^T + Y \nabla_p x_B^T + T \nabla_p x_N^T \)

• Set \( \nabla_p x_N^T = \nabla_p \text{bnd}_N^T \), nonbasic variables to rhs,

• Substitute for remaining variables

• Perform range and null space decomposition

• Solve only for \( \nabla_p x_S^T \) and \( \nabla_p x_B^T \)
Decomposition for NLP Sensitivity

\[
\begin{bmatrix}
Y^T W Y & Y^T W Y & Y^T A \\
Z^T W Y & Z^T W Z & 0 \\
A^T Y & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\nabla_p x_B^T \\
\nabla_p x_S^T \\
\nabla_p \lambda^T
\end{bmatrix} =
\begin{bmatrix}
Y^T (\nabla_x p L(x^*, p, \lambda, \nu)^T + W^T \nabla_p x_N^T) \\
Z^T (\nabla_x p L(x^*, p, \lambda, \nu)^T + W^T \nabla_p x_N^T) \\
\nabla_p c(x^*, p)^T + A^T \nabla_p x_N^T
\end{bmatrix}
\]

- Solve only for $\nabla_p x_B^T$ from bottom row and $\nabla_p x_S^T$ from middle row
- If second derivatives are not available, $Z^T W Z$, $Z^T W Y$ $\nabla_p x_B^T$ and $Z^T W T$ $\nabla_p x_N^T$ can be constructed by directional finite differencing
- If assumption of strict complementarity is violated, sensitivity can be calculated using a QP subproblem.
Second Order Tests

Reduced Hessian needs to be positive definite

At solution $x^*$: Evaluate eigenvalues of $Z^T \nabla_{xx} L^* Z$

- Strict local minimum if all positive.
- Nonstrict local minimum: If nonnegative, find eigenvectors for zero eigenvalues, regions of nonunique solutions.
- Saddle point: If any are negative, move along directions of corresponding eigenvectors and restart optimization.
Sensitivity for Flash Recycle Optimization
(2 decisions, 7 tear variables)

\[
\text{Max } S3(A)^2S3(B) - S3(A)^2 - S3(C)^3 + S3(D) - (S3(E))^2 \leq 2 \sqrt{2}
\]

- Second order sufficiency test:
- Dimension of reduced Hessian = 1
- Positive eigenvalue
- Sensitivity to simultaneous change in feed rate and upper bound on purge ratio
- Only 2-3 flowsheet perturbations required for second order information
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
- Only 5-6 extra perturbations for second derivatives
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
Multiperiod Flowsheet Example

Parameters | Period 1 | Period 2 | Period 3 | Period 4
---|---|---|---|---
E (kJ/mol) | 555.6 | 583.3 | 611.1 | 527.8
k₀ (1/h) | 10 | 11 | 12 | 9
F (kmol/h) | 45.4 | 40.8 | 24.1 | 32.6
Time (h) | 1000 | 4000 | 2000 | 1000
Multiperiod Design Model

\[ \min f_0(d) + \sum_i f_i(d, x_i) \]
\[ \text{s.t. } h_i(x, d) = 0, \ i = 1, \ldots, N \]
\[ g_i(x, d) \leq 0, \ i = 1, \ldots, N \]
\[ r(d) \leq 0 \]

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 \( \delta_i = d \)
**Multiperiod Decomposition Strategy**

**SQP Subproblem**

minimize $\phi = \nabla_d f_0^T \Delta d + \frac{1}{2} \Delta d^T \nabla_d^2 L_0 \Delta d +$

$$\sum_{i=1}^N (\nabla_p f_i^T p_i + \frac{1}{2} p_i^T \nabla_p^2 L_i p_i)$$

subject to

$$\bar{h}_i + \nabla_p \bar{h}_i^T p_i = 0 \quad i = 1, \ldots, N$$

$$r + \nabla_d r^T \Delta d \leq 0$$

$$p_i = \begin{bmatrix} x_i^{k+1} - x_i^k \\ s_i^{k+1} - s_i^k \\ \delta_i^{k+1} - \delta_i^k \end{bmatrix}$$

$$\bar{h}_i = \begin{bmatrix} h_i^k \\ g_i^k + s_i^k \\ -\Delta d \end{bmatrix}$$

$$\nabla_p \bar{h}_i = \begin{bmatrix} \nabla_p h_i^k \\ \nabla_p (g_i^k + s_i^k) \end{bmatrix}$$

- Block diagonal bordered KKT matrix (arrowhead structure)
- Solve each block sequentially (range/null dec.) to form small QP in space of $d$ variables
- Reassemble all other steps from QP solution
Multiperiod Decomposition Strategy

From decomposition of KKT block in each period, obtain the following directions that are parametric in \( \Delta d \):

\[
p_{Z_i} = A_{Z_i} + B_{Z_i} \Delta d \quad \text{and} \quad Z_i p_{Z_i} = Z_{A_i} + Z_{B_i} \Delta d
\]

\[
p_{Y_i} = A_{Y_i} + B_{Y_i} \Delta d \quad \text{and} \quad Y_i p_{Y_i} = Y_{A_i} + Y_{B_i} \Delta d
\]

Substituting back into the original QP subproblem leads to a QP only in terms of \( \Delta d \).

\[
\text{minimize} \quad \phi = \left[ \nabla_d f_0^T + \sum_{i=1}^{N} \{ \nabla_p f_i^T (Z_{B_i} + Y_{B_i}) + (Z_{A_i} + Y_{A_i})^T \nabla_p^2 L_i (Z_{B_i} + Y_{B_i}) \} \right] \Delta d
\]

\[
+ \frac{1}{2} \Delta d^T \left[ \nabla_d^2 L_0 + \sum_{i=1}^{N} \{ (Z_{B_i} + Y_{B_i})^T \nabla_p^2 L_i (Z_{B_i} + Y_{B_i}) \} \right] \Delta d
\]

subject to \( r + \nabla_d r \Delta d \leq 0 \)

Once \( \Delta d \) is obtained, directions are obtained from the above equations.
Multiperiod Decomposition Strategy

- $p_i$ steps are parametric in $\Delta d$ and their components are created independently.
- Decomposition linear in number of periods and trivially parallelizable.
- Choosing the active inequality constraints can be done through:
  - Active set strategy (e.g., bound addition)
  - Interior point strategy using barrier terms in objective
- Easy to implement in decomposition.
Multiperiod Flowsheet 1
(13+2) variables and (31+4) constraints (1 period)
262 variables and 624 constraints (20 periods)
Multiperiod Example 2 – Heat Exchanger Network
(12+3) variables and (31+6) constraints (1 period)
243 variables and 626 constraints (20 periods)
Summary and Conclusions
-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
-Multiperiod Optimization Problems
Optimization of Differential-Algebraic Equation Systems

L. T. Biegler
Chemical Engineering Department
Carnegie Mellon University
Pittsburgh, PA
DAE Optimization Outline

I Introduction
   Process Examples

II Parametric Optimization
   - Gradient Methods
     • Perturbation
     • Direct - Sensitivity Equations
     • Adjoint Equations

III Optimal Control Problems
   - Optimality Conditions
   - Model Algorithms
     • Sequential Methods
     • Multiple Shooting
     • Indirect Methods

IV Simultaneous Solution Strategies
   - Formulation and Properties
   - Process Case Studies
   - Software Demonstration
Dynamic Optimization Problem

\[
\begin{align*}
\min & \quad \Phi(z(t), y(t), u(t), p, t_f) \\
\text{s.t.} & \quad \frac{dz(t)}{dt} = f(z(t), y(t), u(t), t, p) \\
& \quad g(z(t), y(t), u(t), t, p) = 0 \\
& \quad z^0 = z(0) \\
& \quad z'^1 \leq z(t) \leq z'^2 \\
& \quad y'^1 \leq y(t) \leq y'^2 \\
& \quad u'^1 \leq u(t) \leq u'^2 \\
& \quad p'^1 \leq p \leq p'^2
\end{align*}
\]

\(t\), time  \quad \text{t}_f\), final time  \\
z, differential variables  \quad u, \text{control variables}  \\
y, algebraic variables  \quad p, \text{time independent parameters}
DAE Models in Process Engineering

Differential Equations
• Conservation Laws (Mass, Energy, Momentum)

Algebraic Equations
• Constitutive Equations, Equilibrium (physical properties, hydraulics, rate laws)
  • Semi-explicit form
  • Assume to be index one (i.e., algebraic variables can be solved uniquely by algebraic equations)
  • If not, DAE can be reformulated to index one (see Ascher and Petzold)

Characteristics
• Large-scale models – not easily scaled
• Sparse but no regular structure
• Direct linear solvers widely used
• Coarse-grained decomposition of linear algebra
Parameter Estimation

Catalytic Cracking of Gasoil (Tjoa, 1991)

\[ A \xrightarrow{p_1} Q, \quad Q \xrightarrow{p_2} S, \quad A \xrightarrow{p_3} S \]

\[ \dot{a} = -(p_1 + p_3) a^2 \]
\[ \dot{q} = -p_1 a^2 - p_2 q \]
\[ a(0) = 1, \quad q(0) = 0 \]

number of states and ODEs: 2
number of parameters: 3
no control profiles
constraints: \( p_L \leq p \leq p_U \)

Objective Function: Ordinary Least Squares

\( (p_1, p_2, p_3)^0 = (6, 4, 1) \)
\( (p_1, p_2, p_3)^* = (11.95, 7.99, 2.02) \)
\( (p_1, p_2, p_3)_{true} = (12, 8, 2) \)
Batch Distillation Multi-product Operating Policies

- Run between distillation batches
- Treat as boundary value optimization problem
  - When to switch from A to offcut to B?
  - How much offcut to recycle?
  - Reflux?
  - Boilup Rate?
  - Operating Time?
Nonlinear Model Predictive Control (NMPC)

\[ \begin{align*}
  & \min_u \quad \sum \| y(t) - y^{sp} \|_{Q^y} + \sum \| u(t^k) - u(t^{k-1}) \|_{Q^u} \\
  & s.t. \quad z'(t) = F(z(t), y(t), u(t), t) \\
  & \quad 0 = G(z(t), y(t), u(t), t) \\
  & \quad z(t) = z^{init} \\
  & \quad \text{Bound Constraint s} \\
  & \quad \text{Other Constraint s}
\end{align*} \]
Batch Process Optimization

Optimization of dynamic batch process operation resulting from reactor and distillation column

DAE models:
\[ z' = f(z, y, u, p) \]
\[ g(z, y, u, p) = 0 \]

number of states and DAEs: \( n_z + n_y \)

parameters for equipment design (reactor, column)

\( n_u \) control profiles for optimal operation

Constraints:
\[ u_L \leq u(t) \leq u_U \]
\[ y_L \leq y(t) \leq y_U \]
\[ z_L \leq z(t) \leq z_U \]
\[ p_L \leq p \leq p_U \]

Objective Function: amortized economic function at end of cycle time \( t_f \)

optimal reactor temperature policy

optimal column reflux ratio
**Reactor Design Example**

**Plug Flow Reactor Optimization**

The cracking furnace is an important example in the olefin production industry, where various hydrocarbon feedstocks react. Consider a simplified model for ethane cracking (Chen et al., 1996). The objective is to find an optimal profile for the heat flux along the reactor in order to maximize the production of ethylene.

\[
\text{Max } F_{\text{exit}}^{C_2H_4} \\
\text{s.t. DAE} \\
T_{\text{exit}} \leq 1180K
\]

The reaction system includes six molecules, three free radicals, and seven reactions. The model also includes the heat balance and the pressure drop equation. This gives a total of eleven differential equations.

\[
\begin{align*}
\text{CH}_2 & \rightarrow 2\text{CH}_3 \\
\text{H} + \text{CH}_2 & \rightarrow \text{H}_2 + \text{CH}_3 \\
\text{CH}_3 & \rightarrow \text{CH}_2 + \text{CH}_4 \\
\text{CH}_2 & \rightarrow \text{CH} + \text{H}_2 \\
\text{CH}_2 & \rightarrow \text{CH}_3 + \text{CH}_4 \\
\text{H} + \text{CH}_3 & \rightarrow \text{CH}_2 + \text{CH}_4
\end{align*}
\]

**Concentration and Heat Addition Profile**

![Graph showing concentration and heat addition profile](image_url)
Dynamic Optimization Approaches

1. DAE Optimization Problem
   - Discretize controls
   - Apply a NLP solver
   - Efficient for constrained problems

2. Sequential Approach
   - Efficient for constrained problems

3. Variational Approach
   - Inefficient for constrained problems

Pontryagin (1962)
Vassiliadis (1994)
Sequential Approaches - Parameter Optimization

Consider a simpler problem without control profiles:
e.g., equipment design with DAE models - reactors, absorbers, heat exchangers

\[ \text{Min } \Phi (z(t_f)) \]
\[ z' = f(z, p), \ z(0) = z_0 \]
\[ g(z(t_f)) \leq 0, \ h(z(t_f)) = 0 \]

By treating the ODE model as a "black-box" a sequential algorithm can be constructed that can be treated as a nonlinear program.

Task: How are gradients calculated for optimizer?
Gradient Calculation

Perturbation
Sensitivity Equations
Adjoint Equations

**Perturbation**
Calculate approximate gradient by solving ODE model \((np + 1)\) times

Let \(\psi = \Phi, g\) and \(h\) (at \(t = t_i\))

\[
\frac{d\psi}{dp_i} = \frac{\psi(p_i + \Delta p_i) - \psi(p_i)}{\Delta p_i}
\]

Very simple to set up

Leads to poor performance of optimizer and poor detection of optimum unless roundoff error \((O(1/\Delta p_i))\) and truncation error \((O(\Delta p_i))\) are small.

Work is proportional to \(np\) (expensive)
Direct Sensitivity

From ODE model:

\[ \frac{\partial}{\partial p} \{ \dot{z} = f(z,p,t), z(0) = z_0(p) \} \]

Define \( s_i(t) = \frac{\partial z(t)}{\partial p_i} i = 1, \ldots, np \)

\[ s_i' = \frac{d}{dt} (s_i) = \frac{\partial f}{\partial p_i} + \frac{\partial f^T}{\partial z} s_i, \quad s_i(0) = \frac{\partial z(0)}{\partial p_i} \]

(nz x np sensitivity equations)

- \( z \) and \( s_i, i = 1, \ldots, np \), can be integrated forward simultaneously.
- For implicit ODE solvers, \( s_i(t) \) can be carried forward in time after converging on \( z \)
- Linear sensitivity equations exploited in ODESSA, DASSAC, DASPK, DSL48s and a number of other DAE solvers

Sensitivity equations are efficient for problems with many more constraints than parameters \((1 + ng + nh > np)\)
Example: Sensitivity Equations

\[ z'_1 = z_1^2 + z_2^2 \]
\[ z'_2 = z_1 z_2 + z_1 p_b \]
\[ z_1 = 5, \quad z_2(0) = p_a \]
\[ s(t)_{a,j} = \frac{\partial z(t)}{\partial p_a}, \quad s(t)_{b,j} = \frac{\partial z(t)}{\partial p_b}, \quad j = 1, 2 \]
\[ s'_{a,1} = 2z_1 s_{a,1} + 2z_2 s_{a,2} \]
\[ s'_{a,2} = z_1 s_{a,2} + z_2 s_{a,1} + s_{a,1} p_b \]
\[ s_{a,1} = 0, \quad s_{a,2}(0) = 1 \]

\[ s'_{b,1} = 2z_1 s_{b,1} + 2z_2 s_{b,2} \]
\[ s'_{b,2} = z_1 + z_1 s_{b,2} + z_2 s_{b,1} + s_{b,1} p_b \]
\[ s_{b,1} = 0, \quad s_{b,2}(0) = 0 \]
Adjoint Sensitivity

Adjoint or Dual approach to sensitivity

Adjoin model to objective function or constraint

\( (\psi = \Phi, g \text{ or } h) \)

\[ \psi = \psi(t_f) - \int_0^{t_f} \lambda^T (z' - f(z, p, t)) dt \]

\[ \psi = \psi(t_f) + \lambda(0)^T z_0(p) - \lambda(t_f)^T z(t_f) + \int_0^{t_f} z^T \lambda' + \lambda^T F(z, p, t) dt \]

(\( \lambda(t) \) serve as multipliers on ODE’s)

Now, integrate by parts

\[ d\psi = \left[ \frac{\partial \psi(z(t_f))}{\partial z(t_f)} + \frac{\partial z_0(p)}{\partial p} \lambda(0) \right]^T dp + \int_0^{t_f} \left[ \lambda' + \frac{\partial f}{\partial z} \lambda \right]^T \delta z(t) + \left[ \frac{\partial f}{\partial p} \lambda \right]^T dp \ dt \]

and find \( d\psi / dp \) subject to feasibility of ODE’s

Now, set all terms not in \( dp \) to zero.
Adjoint System

\[ \lambda' = -\frac{\partial f}{\partial z} \lambda(t), \quad \lambda(t_f) = \frac{\partial \psi(z(t_f))}{\partial z(t_f)} \]

\[ \frac{d\psi}{dp} = \frac{\partial z_0(p)}{\partial p} \lambda(0) + \int_0^{t_f} \left[ \frac{\partial f}{\partial p} \lambda(t) \right] dt \]

Integrate model equations forward

Integrate adjoint equations backward and evaluate integral and sensitivities.

Notes:

nz (ng + nh + 1) adjoint equations must be solved backward (one for each objective and constraint function)

for implicit ODE solvers, profiles (and even matrices) can be stored and carried backward after solving forward for z as in DASPK/Adjoint (Li and Petzold)

more efficient on problems where: np > 1 + ng + nh
Example: Adjoint Equations

\[ z_1' = z_1^2 + z_2^2 \]
\[ z_1' = z_1, z_2 + z_1 p_b \]
\[ z_1 = 5, z_2(0) = p_a \]

Form \( \lambda' f(z, p, t) = \lambda_1(z_1^2 + z_2^2) + \lambda_2(z_1, z_2 + z_1 p_b) \)

\[ \lambda' = -\frac{\partial f}{\partial z} \lambda(t), \quad \lambda(t_f) = \frac{\partial \psi(z(t_f))}{\partial z(t_f)} \]

\[ \frac{d\psi}{dp} = \frac{\partial z_0(p)}{\partial p} \lambda(0) + \left[ \frac{\partial f}{\partial p} \lambda(t) \right] dt \]

then becomes:

\[ \lambda_1' = -2\lambda_1 z_1 - \lambda_2(z_2 + p_b), \quad \lambda_1(t_f) = \frac{\partial \psi(t_f)}{\partial z_1(t_f)} \]

\[ \lambda_2' = -2\lambda_1 z_2 - \lambda_2 z_1, \quad \lambda_2(t_f) = \frac{\partial \psi(t_f)}{\partial z_2(t_f)} \]

\[ \frac{d\psi(t_f)}{dp_a} = \lambda_1(0) \]

\[ \frac{d\psi(t_f)}{dp_b} = \int_0^{t_f} \lambda_2(t)z_1(t) dt \]
Example: Hot Spot Reactor

\[
\begin{align*}
\text{Min} & \quad \Phi = L - \int_{0}^{L} (T(t) - T_{S} / T_{R})\,dt \\
\text{s.t.} & \quad \frac{dq}{dt} = 0.3(1 - q(t))\exp[20 - 20/T(t)], \quad q(0) = 0 \\
& \quad \frac{dT}{dt} = -1.5(T(t) - T_{S} / T_{R}) + 2 / 3 \frac{dq}{dt}, \quad T(0) = 1 \\
& \quad \Delta H_{\text{feed}}(T_{R}, 110^\circ C) - \Delta H_{\text{product}}(T_{P}, T(L)) = 0 \\
& \quad T_{P} = 120^\circ C, \quad T(L) = 1 + 10^\circ C/T_{R}
\end{align*}
\]

Cases considered:
- Hot Spot - no state variable constraints
- Hot Spot with \( T(t) \leq 1.45 \)
Hot Spot Reactor: Unconstrained Case

Method: SQP (perturbation derivatives)

<table>
<thead>
<tr>
<th></th>
<th>L(norm)</th>
<th>T_R(K)</th>
<th>T_s(K)</th>
<th>T_P(K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>1.0</td>
<td>462.23</td>
<td>425.26</td>
<td>250</td>
</tr>
<tr>
<td>Optimal</td>
<td>1.25</td>
<td>500</td>
<td>470.1</td>
<td>188.4</td>
</tr>
</tbody>
</table>

13 SQP iterations / 2.67 CPU min. (µVax II)

Constrained Temperature Case: could not be solved with sequential method
Tricks to generalize classes of problems

Variable Final Time  (Miele, 1980)

Define  \( t = p_{n+1} \tau, \ 0 \leq \tau \leq 1, \ p_{n+1} = t_f \)

Let  \( dz/dt = (1/ p_{n+1}) \ dz/d \tau = f(z, p) \Rightarrow dz/d \tau = (p_{n+1}) f(z, p) \)

Converting Path Constraints to Final Time

Define measure of infeasibility as a new variable, \( z_{nz+1}(t) \) (Sargent & Sullivan, 1977):

\[
z_{nz+1}(t_f) = \sum_{j} \int_{0}^{t_f} \max(0, g_j(z(t), u(t))^2 \ dt
\]

or  \( \dot{z}_{nz+1}(t) = \sum_{j} \max(0, g_j(z(t), u(t))^2 \), \( z_{nz+1}(0) = 0 \)

Enforce  \( z_{nz+1}(t_f) \leq \varepsilon \) (however, constraint is degenerate)
Profile Optimization - (Optimal Control)

Optimal Feed Strategy (Schedule) in Batch Reactor
Optimal Startup and Shutdown Policy
Optimal Control of Transients and Upsets

**Sequential Approach:** Approximate control profile as through parameters (piecewise constant, linear, polynomial, etc.)

Apply NLP to discretization as with parametric optimization

Obtain gradients through adjoints (Hasdorff; Sargent and Sullivan; Goh and Teo) or sensitivity equations (Vassiliadis, Pantelides and Sargent; Gill, Petzold et al.)

**Variational (Indirect) approach:** Apply optimality conditions and solve as boundary value problem
Derivation of Variational Conditions
Indirect Approach

Optimality Conditions (Bound constraints on u(t))

\[ \text{Min } \phi(z(tf)) \]
\[ \text{s.t. } \frac{dz}{dt} = f(z, u), \quad z(0) = z_0 \]
\[ g(z(t_f)) \leq 0 \]
\[ h(z(t_f)) = 0 \]
\[ a \leq u(t) \leq b \]

Form Lagrange function - adjoin objective function and constraints:

\[ \phi = \phi(t_f) + g(z(t_f))^T \mu + h(z(t_f))^T v \]
\[ - \int_0^{t_f} \lambda^T (\dot{z} - f(z, u)) + \alpha_a^T (a - u(t)) + \alpha_b^T (u(t) - b) \, dt \]

Integrate by parts:

\[ \phi = \phi(t_f) + g(z(t_f))^T \mu + h(z(t_f))^T v + \lambda^T (0) z(0) - \lambda^T (t_f) z(t_f) \]
\[ + \int_0^{t_f} \lambda^T z + \lambda^T f(z, u) + \alpha_a^T (a - u(t)) + \alpha_b^T (u(t) - b) \, dt \]
Derivation of Variational Conditions

\[ \delta \phi = \left[ \frac{\partial \phi}{\partial z} + \frac{\partial g}{\partial z} \mu + \frac{\partial h}{\partial z} \nu - \lambda \right]^T \delta z(t_f) + \lambda^T(0) \delta z(0) \]

\[ + \int_0^{t_f} \left[ \dot{\lambda} + \frac{\partial f(z, u)}{\partial z} \lambda \right]^T \delta z(t) + \left[ \frac{\partial f(z, u)}{\partial u} \lambda + \alpha_b - \alpha_a \right]^T \delta u(t) \, dt \geq 0 \]

At optimum, \( \delta \phi \geq 0 \). Since \( u \) is the control variable, let all other terms vanish.

\[ \Rightarrow \delta z(t_f) : \quad \lambda(t_f) = \left\{ \frac{\partial \phi}{\partial z} + \frac{\partial g}{\partial z} \mu + \frac{\partial h}{\partial z} \nu \right\}_{t=t_f} \]

\[ \delta z(0) : \quad \lambda(0) = 0 \quad (\text{if } z(0) \text{ is not specified}) \]

\[ \delta z(t) : \quad \dot{\lambda} = -\frac{\partial H}{\partial z} = -\frac{\partial f}{\partial z} \lambda \]

Define Hamiltonian, \( H = \lambda^T f(z, u) \)

For \( u \) not at bound:

\[ \frac{\partial f}{\partial u} \lambda = \frac{\partial H}{\partial u} = 0 \]

\[ \alpha_a^T (a - u(t)) \]

\[ \alpha_b^T (u(t) - b) \]

\[ u_a \leq u(t) \leq u_b \]

\[ \alpha_a \geq 0, \alpha_b \geq 0 \]

For \( u \) at bounds:

\[ \frac{\partial H}{\partial u} = \alpha_a - \alpha_b \]

Upper bound, \( u(t) = b, \quad \frac{\partial H}{\partial u} = -\alpha_b \leq 0 \quad \text{Lower bound, } u(t) = a, \quad \frac{\partial H}{\partial u} = \alpha_a \geq 0 \]
Car Problem

Travel a fixed distance (rest-to-rest) in minimum time.

\[
\begin{align*}
\text{Min} & \quad t_f \\
\text{s.t.} & \quad x'' = u \\
& \quad a \leq u(t) \leq b \\
& \quad x(0) = 0, x(t_f) = L \\
& \quad x'(0) = 0, x'(t_f) = 0 \\
\end{align*}
\]

\[
\begin{align*}
\text{Min} & \quad x_3(t_f) \\
\text{s.t.} & \quad x_1' = x_2 \\
& \quad x_2' = u \\
& \quad x_3' = 1 \\
& \quad a \leq u(t) \leq b \\
& \quad x_1(0) = 0, x_1(t_f) = L \\
& \quad x_2(0) = 0, x_2(t_f) = 0 \\
\end{align*}
\]

Hamiltonian: \( H = \lambda_1 x_2 + \lambda_2 u + \lambda_3 \)

Adjoint: \( \dot{\lambda}_1 = 0 \Rightarrow \lambda_1(t) = c_1 \)
\[
\begin{align*}
\dot{\lambda}_2 & = -\lambda_1 \Rightarrow \lambda_2(t) = c_2 + c_1(t_f - t) \\
\dot{\lambda}_3 & = 0 \Rightarrow \lambda_3(t_f) = 1, \quad \lambda_3(t) = 1 \\
\frac{\partial H}{\partial u} & = \lambda_2 = c_2 + c_1(t_f - t) \begin{cases} t = 0, c_1 t_f + c_2 > 0, u = b \\
& t = t_f, c_2 > 0, u = a \\
\end{cases}
\end{align*}
\]

Crossover (\( \lambda_2 = 0 \)) occurs at \( t = t_s \)
Car Problem

Analytic Variational Solution

Optimal Profile

From state equations:
\[ x_1(t) = \begin{cases} 
1/2 bt^2, & t < t_s \\
1/2 \left(bt_s^2 - a(t_s - tf)^2\right), & t \geq t_s 
\end{cases} \]

\[ x_2(t) = \begin{cases} 
bt, & t < t_s \\
bt_s + a(t - t_s), & t \geq t_s 
\end{cases} \]

- Problem is linear in \( u(t) \). Frequently these problems have "bang-bang" character.
- For nonlinear and larger problems, the variational conditions can be solved numerically as boundary value problems.

Apply boundary conditions at \( t = tf \):
\[ x_1(tf) = 1/2 \left(b \ t_s^2 - a \ (t_s - tf)^2\right) = L \]
\[ x_2(tf) = bt_s + a \ (tf - t_s) = 0 \]
\[ \Rightarrow t_s = \left[ \frac{2L}{b(1-b/a)} \right]^{1/2} \]
\[ tf = (1-b/a) \left[ \frac{2L}{b(1-b/a)} \right]^{1/2} \]
Example: Batch reactor - temperature profile

Maximize yield of B after one hour’s operation by manipulating a transformed temperature, \( u(t) \).

\[
\Rightarrow \quad \text{Minimize } -z_B(1.0)
\]

s.t.
\[
\begin{align*}
z_A' &= -(u+u^2/2) z_A \\
z_B' &= u z_A \\
z_A(0) &= 1 \\
z_B(0) &= 0 \\
0 &\leq u(t) \leq 5
\end{align*}
\]

Adjoint Equations:
\[
\begin{align*}
H &= -\lambda_A(u+u^2/2) z_A + \lambda_B u z_A \\
\partial H/\partial u &= \lambda_A (1+u) z_A + \lambda_B z_A \\
\lambda'_A &= \lambda_A(u+u^2/2) - \lambda_B u, \quad \lambda_A(1.0) = 0 \\
\lambda'_B &= 0, \quad \lambda_B(1.0) = -1
\end{align*}
\]

Cases Considered
1. **NLP Approach** - piecewise constant and linear profiles.
2. **Control Vector Iteration**
Batch Reactor Optimal Temperature Program

Piecewise Constant

Results

Piecewise Constant Approximation with Variable Time Elements

Optimum B/A: 0.57105
Batch Reactor Optimal Temperature Program
Piecewise Linear

Results:
Piecewise Linear Approximation with Variable Time Elements
Optimum B/A: 0.5726
Equivalent # of ODE solutions: 32
Batch Reactor Optimal Temperature Program
Indirect Approach

Results:
Control Vector Iteration with Conjugate Gradients
Optimum (B/A): 0.5732
Equivalent # of ODE solutions: 58
Dynamic Optimization - Sequential Strategies

Small NLP problem, $O(np+nu)$ (large-scale NLP solver not required)
- Use NPSOL, NLPQL, etc.
- Second derivatives difficult to get

Repeated solution of DAE model and sensitivity/adjoint equations, scales with $nz$ and $np$
- Dominant computational cost
- May fail at intermediate points

Sequential optimization is not recommended for unstable systems. State variables blow up at intermediate iterations for control variables and parameters.

Discretize control profiles to parameters (at what level?)

Path constraints are difficult to handle exactly for NLP approach
Instabilities in DAE Models

This example cannot be solved with sequential methods (Bock, 1983):

\[
\begin{align*}
\frac{dy_1}{dt} &= y_2 \\
\frac{dy_2}{dt} &= \tau^2 y_1 + (\pi^2 - \tau^2) \sin(\pi t)
\end{align*}
\]

The characteristic solution to these equations is given by:

\[
\begin{align*}
y_1(t) &= \sin(\pi t) + c_1 \exp(-\tau t) + c_2 \exp(\tau t) \\
y_2(t) &= \pi \cos(\pi t) - c_1 \tau \exp(-\tau t) + c_2 \tau \exp(\tau t)
\end{align*}
\]

Both $c_1$ and $c_2$ can be set to zero by either of the following equivalent conditions:

**IVP** \[y_1(0) = 0, \quad y_2(0) = \pi\]

**BVP** \[y_1(0) = 0, \quad y_1(1) = 0\]
IVP Solution

If we now add roundoff errors \( e_1 \) and \( e_2 \) to the IVP and BVP conditions, we see significant differences in the sensitivities of the solutions.

For the IVP case, the sensitivity to the *analytic* solution profile is seen by large changes in the profiles \( y_1(t) \) and \( y_2(t) \) given by:

\[
y_1(t) = \sin (\pi \, t) + (e_1 - e_2/\tau) \exp(-\tau \, t)/2 \\
+ (e_1 + e_2/\tau) \exp(\tau \, t)/2
\]

\[
y_2(t) = \pi \cos (\pi \, t) - (\tau \, e_1 - e_2) \exp(-\tau \, t)/2 \\
+ (\tau \, e_1 + e_2) \exp(\tau \, t)/2
\]

Therefore, even if \( e_1 \) and \( e_2 \) are at the level of machine precision \(< 10^{-13}\), a large value of \( \tau \) and \( t \) will lead to unbounded solution profiles.
BVP Solution

On the other hand, for the boundary value problem, the errors affect the *analytic* solution profiles in the following way:

\[ y_1(t) = \sin (\pi t) + [e_1 \exp(\tau) - e_2] \exp(-\tau t)/[\exp(\tau) - \exp(-\tau)] \]

\[ + [e_1 \exp(-\tau) - e_2] \exp(\tau t)/[\exp(\tau) - \exp(-\tau)] \]

\[ y_2(t) = \pi \cos (\pi t) - \tau [e_1 \exp(\tau) - e_2] \exp(-\tau t)/[\exp(\tau) - \exp(-\tau)] \]

\[ + \tau [e_1 \exp(-\tau) - e_2] \exp(\tau t)/[\exp(\tau) - \exp(-\tau)] \]

Errors in these profiles never exceed \( t (e_1 + e_2) \), and as a result a solution to the BVP is readily obtained.
BVP and IVP Profiles

e_1, e_2 = 10^{-9}

Linear BVP solves easily

IVP blows up before midpoint
Dynamic Optimization Approaches

DAE Optimization Problem

Apply a NLP solver

Variational Approach

Pontryagin (1962)

Inefficient for constrained problems

Vassiliadis (1994)

Sequential Approach

Can not handle instabilities properly
Small NLP

Multiple Shooting

Discretize some state variables

Handles instabilities Larger NLP

Discretize controls

Efficient for constrained problems
Multiple Shooting for Dynamic Optimization
Divide time domain into separate regions

Integrate DAEs state equations over each region
Evaluate sensitivities in each region as in sequential approach wrt $u_{ij}$, $p$ and $z_j$
Impose matching constraints in NLP for state variables over each region
Variables in NLP are due to control profiles as well as initial conditions in each region
Multiple Shooting
Nonlinear Programming Problem

$$\min_{u_{i,j},p} \psi \left( z(t_f), y(t_f) \right)$$

s.t.

$$z\left(z_j, u_{i,j}, p, t_{j+1}\right) - z_{j+1} = 0$$

$$z^l_k \leq z(z_j, u_{i,j}, p, t_k) \leq z^u_k$$

$$y^l_k \leq y(z_j, u_{i,j}, p, t_k) \leq y^u_k$$

$$u^l_i \leq u_{i,j} \leq u^u_i$$

$$p^l \leq p \leq p^u$$

$$\left( \frac{dz}{dt} \right) = f(z, y, u_{i,j}, p), \quad z(t_j) = z_j$$

$$g(z, y, u_{i,j}, p) = 0$$

$$z^0_0 = z(0)$$

$$\min_{x \in \mathbb{R}^n} f(x)$$

s.t. $$c(x) = 0$$

$$x^L \leq x \leq x^u$$

Solved Implicitly
Consider: Jacobian of Constraint Matrix for NLP

- bound unstable modes with boundary conditions (dichotomy)
- can be done implicitly by determining stable pivot sequences in multiple shooting constraints approach
- well-conditioned problem implies dichotomy in BVP problem (deHoog and Mattheij)

Bock Problem (with $t = 50$)

- Sequential approach blows up (starting within $10^{-9}$ of optimum)
- Multiple Shooting optimization requires 4 SQP iterations
Dynamic Optimization – Multiple Shooting Strategies

Larger NLP problem $O(np+nu+NE \cdot nz)$
- Use SNOPT, MINOS, etc.
- Second derivatives difficult to get

Repeated solution of DAE model and sensitivity/adjoint equations, scales with $nz$ and $np$
- Dominant computational cost
- May fail at intermediate points

Multiple shooting can deal with unstable systems with sufficient time elements.

Discretize control profiles to parameters (at what level?)

Path constraints are difficult to handle exactly for NLP approach

Block elements for each element are dense!

Extensive developments and applications by Bock and coworkers using MUSCOD code
Dynamic Optimization Approaches

DAE Optimization Problem

Apply a NLP solver

Simultaneous Approach

Variational Approach

Sequential Approach

Pontryagin (1962)

Vassiliadis (1994)

Discretize controls

Inefficient for constrained problems

Can not handle instabilities properly Small NLP

Efficient for constrained problems

Discretize all state variables

Handles instabilities Large NLP

Small NLP Handles instabilities Large NLP
Nonlinear Dynamic Optimization Problem

Collocation on finite Elements

Continuous variables

Discretized variables

Nonlinear Programming Problem (NLP)
Discretization of Differential Equations
Orthogonal Collocation

**Given:** \( dz/dt = f(z, u, p), z(0)=\text{given} \)

Approximate \( z \) and \( u \) by Lagrange interpolation polynomials (order \( K+1 \) and \( K \), respectively) with interpolation points, \( t_k \)

\[
z_{K+1}(t) = \sum_{k=0}^{K} z_k \ell_k(t), \ell_k(t) = \prod_{\substack{j=0 \atop j \neq k}}^{K} \frac{(t-t_j)}{(t_k-t_j)} \implies z_{N+1}(t_k) = z_k
\]

\[
u_K(t) = \sum_{k=1}^{K} u_k \ell_k(t), \ell_k(t) = \prod_{\substack{j=1 \atop j \neq k}}^{K} \frac{(t-t_j)}{(t_k-t_j)} \implies u_N(t_k) = u_k
\]

Substitute \( z_{N+1} \) and \( u_N \) into ODE and apply equations at \( t_k \):

\[
r(t_k) = \sum_{j=0}^{K} z_j \dot{\ell}_j(t_k) - f(z_k, u_k) = 0, \quad k = 1, \ldots, K
\]
Collocation Example

\[ z_{K+1}(t) = \sum_{k=0}^{K} z_k \ell_k(t), \ell_k(t) = \prod_{j=0}^{K} \frac{(t-t_j)}{(t_k-t_j)} \implies z_{N+1}(t_k) = z_k \]

\[ t_0 = 0, t_1 = 0.21132, t_2 = 0.78868 \]

\[ \ell_0(t) = (t^2 - t + 1)/6, \quad \ell_0'(t) = t/3 - 1/6 \]
\[ \ell_1(t) = -8.195 t^2 + 6.4483 t, \quad \ell_1'(t) = 6.4483 - 16.39 t \]
\[ \ell_2(t) = 2.19625 t^2 - 0.4641 t, \quad \ell_2'(t) = 4.392 - 0.46412 \]

Solve \( z' = z^2 - 3 z + 2, z(0) = 0 \)

\[ \implies z_0 = 0 \]
\[ z_0 \ell_0(t_1) + z_1 \ell_1(t_1) + z_2 \ell_2(t_1) = z_1^2 - 3 z_1 + 2 \]
\[ (2.9857 z_1 + 0.46412) z_2 = z_1^2 - 3 z_1 + 2 \]
\[ z_0 \ell_0(t_2) + z_1 \ell_1(t_2) + z_2 \ell_2(t_2) = z_2^2 - 3 z_2 + 2 \]
\[ (-6.478 z_1 + 3 z_2) = z_2^2 - 3 z_2 + 2 \]

\[ z_0 = 0, z_1 = 0.291 (0.319), z_2 = 0.7384 (0.706) \]
\[ z(t) = 1.5337 t - 0.76303 t^2 \]
Converted Optimal Control Problem
Using Collocation

Min $\phi(z(t_f))$

s.t. $z' = f(z, u, p), z(0) = z_0$

$g(z(t), u(t), p) \leq 0$

$h(z(t), u(t), p) = 0$

to Nonlinear Program

$Min \quad \phi(z_f)$

$\sum_{j=0}^{K} z_j \ell_j (t_k) - f(z_k, u_k) = 0, \quad z_0 = z(0)$

$g(z_k, u_k) \leq 0$

$h(z_k, u_k) = 0$

$\sum_{j=0}^{K} z_j \ell_j (1) - z_f = 0$

How accurate is approximation
Results of Optimal Temperature Program
Batch Reactor (Revisited)

Results - NLP with Orthogonal Collocation
Optimum B/A - 0.5728
# of ODE Solutions - 0.7 (Equivalent)
Collocation on Finite Elements

\[ t_{ij} = \sum_{i=1}^{i=1} h_i + h_i \tau_j, \tau \in [0,1] \]

\[ \frac{dz}{d\tau} = h_i f(z, u) \]

Continuous Differential variables

\[ z(t) = \sum_{q=0}^{K} \ell_q(t) z_{iq} \]

Discontinuous Algebraic and Control variables

\[ y(t) = \sum_{q=1}^{k} \ell_q(t) y_{iq} \]

\[ u(t) = \sum_{q=1}^{k} \ell_q(t) u_{iq} \]

\[ r(t_{ik}) = \sum_{j=0}^{K} (z_{ij} \ell_j(\tau_k)) - h_i f(z_{ik}, u_{ik}, p) = 0, \quad k = 1,\ldots,K, \quad i = 1,\ldots,NE \]
Nonlinear Programming Problem

\[
\begin{align*}
\min & \quad \psi(z_f) \\
\text{s.t.} & \quad \sum_{j=0}^{K} (z_{ij} \ell_j(\tau_k)) - h_i f(z_{ik}, u_{ik}, p) = 0 \\
& \quad g(z_{i,k}, y_{i,k}, u_{i,k}, p) = 0 \\
& \quad \sum_{j=0}^{K} (z_{i-1,j} \ell_j(1)) - z_{i0} = 0, \quad i = 2, \ldots, NE \\
& \quad \sum_{j=0}^{K} (z_{NE,j} \ell_j(1)) - z_f = 0, \quad z_{10} = z(0)
\end{align*}
\]

Finite elements, \( h_i \), can also be variable to determine break points for \( u(t) \).

Add \( h_u \geq h_i \geq 0, \sum h_i = t_f \)

Can add constraints \( g(h, z, u) \leq \varepsilon \) for approximation error
Hot Spot Reactor Revisited

\[
\begin{align*}
\text{Min} \quad \Phi &= L - \int_0^L (T(t) - T_s / T_R) \, dt \\
\text{s.t.} \quad \frac{dq}{dt} &= 0.3(1 - q(t)) \exp[20 - 20/T(t)], \quad q(0) = 0 \\
\frac{dT}{dt} &= -1.5(T(t) - T_s / T_R) + 2/3 \frac{dq}{dt}, \quad T(0) = 1 \\
\Delta H_{\text{feed}}(T_R, 110^\circ C) - \Delta H_{\text{product}}(T_P, T(L)) &= 0 \\
T_P &= 120^\circ C, \quad T(L) = 1 + 10^\circ C/T_R
\end{align*}
\]

Cases considered:
- Hot Spot - no state variable constraints
- Hot Spot with \( T(t) \leq 1.45 \)
Base Case Simulation

Method: OCFE at initial point with 6 equally spaced elements

<table>
<thead>
<tr>
<th>L(norm)</th>
<th>T_R(K)</th>
<th>T_S(K)</th>
<th>T_P(K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>462.23</td>
<td>425.26</td>
<td>250</td>
</tr>
</tbody>
</table>

Base Case: 1.0 462.23 425.26 250
**Unconstrained Case**

Method: OCFE combined formulation with rSQP identical to integrated profiles at optimum

<table>
<thead>
<tr>
<th></th>
<th>L(norm)</th>
<th>T_R(K)</th>
<th>T_S(K)</th>
<th>T_P(K)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>1.0</td>
<td>462.23</td>
<td>425.26</td>
<td>250</td>
</tr>
<tr>
<td>Optimal</td>
<td>1.25</td>
<td>500</td>
<td>470.1</td>
<td>188.4</td>
</tr>
</tbody>
</table>

123 CPU s. (µVax II)

φ* = -171.5
Temperature Constrained Case

\[ T(t) \leq 1.45 \]

Method: OCFE combined formulation with rSQP, identical to integrated profiles at optimum

<table>
<thead>
<tr>
<th>L(norm)</th>
<th>( T_R(K) )</th>
<th>( T_S(K) )</th>
<th>( T_P(K) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial:</td>
<td>1.0</td>
<td>462.23</td>
<td>425.26</td>
</tr>
<tr>
<td>Optimal:</td>
<td>1.25</td>
<td>500</td>
<td>450.5</td>
</tr>
</tbody>
</table>

57 CPU s. (μVax II), \( \phi^* = -148.5 \)
Theoretical Properties of Simultaneous Method

A. Stability and Accuracy of Orthogonal Collocation

• Equivalent to performing a fully implicit Runge-Kutta integration of the DAE models at Gaussian (Radau) points
• 2K order (2K-1) method which uses K collocation points
• Algebraically stable (i.e., possesses A, B, AN and BN stability)

B. Analysis of the Optimality Conditions

• An equivalence has been established between the Kuhn-Tucker conditions of NLP and the variational necessary conditions
• Rates of convergence have been established for the NLP method
Simultaneous DAE Optimization

Case Studies
• Reactor - Based Flowsheets
• Fed-Batch Penicillin Fermenter
• Temperature Profiles for Batch Reactors
• Parameter Estimation of Batch Data
• Synthesis of Reactor Networks
• Batch Crystallization Temperature Profiles
• Grade Transition for LDPE Process
• Ramping for Continuous Columns
• Reflux Profiles for Batch Distillation and Column Design
• Source Detection for Municipal Water Networks
• Air Traffic Conflict Resolution
• Satellite Trajectories in Astronautics
• Batch Process Integration
• Optimization of Simulated Moving Beds
Production of High Impact Polystyrene (HIPS)
Startup and Transition Policies (Flores et al., 2005a)

Initiation reactions
- Thermal
  \[ 3M_S \xrightarrow{k_0} 2R_S^1 \]
- Chemical
  \[ I \xrightarrow{k_{I1}} 2R \]
  \[ R + M_S \xrightarrow{k_{R1}} R_S^1 \]
  \[ R + B_0 \xrightarrow{k_{RB}} B_R \]
  \[ B_R + M_S \xrightarrow{k_{RS}} B_{RS}^1 \]

Propagation reactions
- \[ R_S^i + M_S \xrightarrow{k_p} R_S^{i+1} \]
- \[ B_{RS}^i + M_S \xrightarrow{k_p} B_{RS}^{i+1} \]

Definite termination reactions
- Homopolymer
  \[ R_S^i + R_S^m \xrightarrow{k_t} P^{i+m} \]
- Grafting
  \[ R_S^i + B_R \xrightarrow{k_t} B_P^i \]
  \[ R_S^i + B_{RS}^m \xrightarrow{k_t} B_{RS}^{i+m} \]
- Crosslinking
  \[ B_R + B_R \xrightarrow{k_t} B_{EB} \]
  \[ B_{RS}^i + B_R \xrightarrow{k_t} B_{PB}^i \]
  \[ B_{RS}^i + B_{RS}^m \xrightarrow{k_t} B_{RS}^{i+m} \]

Transfer reactions
- Monomer
  \[ R_S^i + M_S \xrightarrow{k_{TR}} P^i + R_S^1 \]
  \[ B_{RS}^i + M_S \xrightarrow{k_{TB}} B_P^i + R_S^1 \]
- Grafting sites
  \[ R_S^i + B_0 \xrightarrow{k_{TR}} P^i + B_R \]
  \[ B_{RS}^i + B_0 \xrightarrow{k_{TB}} B_P^i + B_R \]
Phase Diagram of Steady States

Transitions considered among all steady state pairs

1: $Q_i = 0.0015$
2: $Q_i = 0.0025$
3: $Q_i = 0.0040$

1: $Q_{cw} = 10$
2: $Q_{cw} = 1.0$
3: $Q_{cw} = 0.1$
Startup to Unstable Steady State

- 926 variables
- 476 constraints
- 36 iters. / 0.95 CPU s (P4)
HIPS Process Plant (Flores et al., 2005b)

- Many grade transitions considered with stable/unstable pairs
- 1-6 CPU min (P4) with IPOPT
- Study shows benefit for sequence of grade changes to achieve wide range of grade transitions.
Batch Distillation – Optimization Case Study - 1

\[
\frac{dx_i, d}{dt} = \frac{V}{H_{\text{cond}}} \left[ y_{i,N} - x_{i,d} \right]
\]

\[
\frac{dx_i, 0}{dt} = \frac{V}{S} \left[ x_{i,d} - x_{i,0} \right]
\]

\[
\frac{dS}{dt} = -\frac{V}{R + 1}
\]

- Gauge effect of column holdups
- Overall profit maximization
- Make Tray Count Continuous
Optimization Case Study - 1

Modeling Assumptions

• Ideal Thermodynamics
• No Liquid Tray Holdup
• No Vapor Holdup
• Four component mixture ($\alpha = 2, 1.5, 1.25, 1$)
• Shortcut steady state tray model
  (Fenske-Underwood-Gilliland)
• Can be substituted by more detailed steady state models
  (Fredenslund and Galindez, 1988; Alkaya, 1997)

Optimization Problems Considered

• Effect of Column Holdup ($H_{\text{cond}}$)
• Total Profit Maximization
Maximum Distillate Problem

Comparison of optimal reflux profiles with and without holdup ($H_{\text{cond}}$)

Comparison of distillate profiles with and without holdup ($H_{\text{cond}}$) at 95.5% overall purity
Batch Distillation Profit Maximization

Max \{\text{Net Sales}(D, S_0)/(t_f + T_{\text{set}}) - TAC(N, V)\}

N = 20 trays, \(T_{\text{setup}} = 1\) hour
\(x_d = 0.98, \ x_{\text{feed}} = 0.50, \ \alpha = 2\)
\(C_{\text{prod}}/C_{\text{feed}} = 4.1\)
\(V = 120\) moles/hr, \(S_0 = 100\) moles.
Batch Distillation – Optimization Case Study - 2

\[
\frac{dx_{1,N+1}}{dt} = \frac{V}{H_{N+1}} \left[ y_{1,N} - x_{1,N+1} \right]
\]

\[
\frac{dx_{1,p}}{dt} = \frac{V}{H_p} \left[ y_{1,p-1} - y_{1,p} + \frac{R}{R+1} \left( x_{1,p+1} - x_{1,p} \right) \right]
\]

\[
\frac{dx_{1,0}}{dt} = \frac{V}{S} \left[ x_{1,0} - y_{1,0} + \frac{R}{R+1} \left( x_{1,1} - x_{1,0} \right) \right]
\]

\[
\frac{dD}{dt} = \frac{V}{R+1}
\]

\[
x_{i,0} = \left( S^0 - \sum_{p=1}^{N+1} H_p \right) x_{i,0} + \sum_{p=1}^{N+1} H_p x_{i,p}
\]

\[
\sum_{i} x_{i,p} = 1.0 \quad \sum_{i} y_{i,p} = 1.0
\]

**Ideal Equilibrium Equations**

\[ y_{i,p} = K_{i,p} \ x_{i,p} \]

**Binary Column (55/45, Cyclohexane, Toluene)**

\( S_0 = 200, \ V = 120, \ H_p = 1, \ N = 10, \sim 8000 \text{ variables,} \)

< 2 CPU hrs. (Vaxstation 3200)
Optimization Case Study - 2

Modeling Assumptions

• Ideal Thermodynamics
• Constant Tray Holdup
• No Vapor Holdup
• Binary Mixture (55 toluene/45 cyclohexane)
• 1 hour operation
• Total Reflux Initial Condition

Cases Considered

• Constant Composition over Time
• Specified Composition at Final Time
• Best Constant Reflux Policy
• Piecewise Constant Reflux Policy
Reflux Optimization Cases

Constant Purity over Time

\[ x_i(t) \geq 0.995 \]
\[ D^*(t_f) = 38.61 \]

Overall Distillate Purity

\[
\int x_d(t) \frac{V}{(R+1)} \, dt / D(t_f) \geq 0.998
\]
\[ D^*(t_f) = 42.34 \]

Shortcut Comparison

\[ D^*(t_f) = 37.03 \]
Reflux Optimization Cases

Constant Reflux over Time

\[ x_d(t) \frac{V}{(R+1)} \, dt / D(t_f) \geq 0.998 \]

\[ D^*(t_f) = 38.9 \]

Piecewise Constant Reflux over Time

\[ \int x_d(t) \frac{V}{(R+1)} \, dt / D(t_f) \geq 0.998 \]

\[ D^*(t_f) = 42.26 \]
Batch Reactive Distillation – Case Study 3

Reversible reaction between acetic acid and ethanol

\[
\text{CH}_3\text{COOH} + \text{CH}_3\text{CH}_2\text{OH} \leftrightarrow \text{CH}_3\text{COOCH}_2\text{CH}_3 + \text{H}_2\text{O}
\]

\[t = 0, \ x = 0.25\]
for all components

\[
\max \int_0^{t_f=1} D(t) \ dt
\]

s.t. DAE

\[
x_{\text{Ester}}^{D} \geq 0.4600
\]

Wajde & Reklaitis (1995)
Optimization Case Study - 3

Modeling Assumptions

• Ideal Thermodynamics
• Constant Tray Holdup
• No Vapor Holdup
• Tertiary Mixture (EtOH, HOAc, ETAc, H₂O)
• Cold Start Initial Condition

Cases Considered

• Specified Composition at Final Time
• Optimum Reflux Policy
• Various Trays Considered (8, 15, 25)
• 1 hour operation
Batch Reactive Distillation

Optimal Reflux Profiles

- 8 trays — 15 trays — 25 trays

Condenser Composition (8 trays)

Distillate Composition

- < 5000 variables
- < 260 DAEs
- 10 degrees of freedom
- 10 finite elements
- < 50 IPOPT iterations
- < 11 CPU minutes
## Batch Reactive Distillation

<table>
<thead>
<tr>
<th>Trays</th>
<th>DAEs</th>
<th>Discretized Variables</th>
<th>Iterations</th>
<th>CPU (s) Global</th>
<th>CPU (s) Elemental</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>98</td>
<td>1788</td>
<td>14</td>
<td>56.4</td>
<td>37.2</td>
</tr>
<tr>
<td>15</td>
<td>168</td>
<td>3048</td>
<td>32</td>
<td>245.7</td>
<td>207.5</td>
</tr>
<tr>
<td>25</td>
<td>258</td>
<td>4678</td>
<td>45</td>
<td>1083.2</td>
<td>659.3</td>
</tr>
</tbody>
</table>

### CPU Decomposition Time

![CPU Decomposition Time Diagram](image)
Nonlinear Model Predictive Control (NMPC)

\[
\begin{align*}
\text{min}_u & \quad \sum \|y(t) - y_{sp}\|_Q + \sum \|u(t^k) - u(t^{k-1})\|_{Q^u} \\
\text{s.t.} & \quad z'(t) = F(z(t), y(t), u(t), t) \\
& \quad 0 = G(z(t), y(t), u(t), t) \\
& \quad z(t) = z^{\text{init}} \\
& \quad \text{Bound Constraint} s \\
& \quad \text{Other Constraint} s
\end{align*}
\]
Dynamic optimization in a MATLAB Framework

**Dynamic Process**

**Dynamic Optimization Problem**

- **Process Model**
  \[ f(x', x, y, u, p, t) = 0 \]
  \[ g(x, y, u, p, t) = 0 \]

- **Inequality Constraints**
  \[ h(x', x, y, u, p, t) \leq 0 \]

- **Initial Conditions**
  \[ x(t_0) = x_0 \]

- **Constraints at Final Time**
  \[ \varphi(x(t_f), x(t_f), y(t_f), u(t_f), x_0, p, t_f) = 0 \]

- **Objective Function**
  \[ \min_{u(t), p, x_0, t_f} P(x(t_f), y(t_f), u(t_f), p, x_0, t_f) \]

**NLP Optimization Problem**

- **Process Model**
  \[ \hat{f}(\hat{x}, \hat{y}, \hat{u}, p, t_f, x_0) = 0 \]
  \[ \hat{g}(\hat{x}, \hat{y}, \hat{u}, p, t_f) = 0 \]

- **Inequality Constraints**
  \[ \hat{h}(\hat{x}, \hat{y}, \hat{u}, p, t_f) \leq 0 \]

- **Constraints at Final Time**
  \[ \varphi(x_N, y_N, u_N, p, t_f) = 0 \]

- **Objective Function**
  \[ \min_{x_N, y_N, u_N, p, t_f} \]
Tennessee Eastman Process

Unstable Reactor
11 Controls; Product, Purge streams
Model extended with energy balances
### Tennessee Eastman Challenge Process

#### DAE Model

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of differential equations</td>
<td>30</td>
</tr>
<tr>
<td>Number of algebraic variables</td>
<td>152</td>
</tr>
<tr>
<td>Number of algebraic equations</td>
<td>141</td>
</tr>
<tr>
<td>Difference (control variables)</td>
<td>11</td>
</tr>
</tbody>
</table>

#### NLP Optimization problem

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of variables of which are fixed</td>
<td>10920</td>
</tr>
<tr>
<td>Number of constraints</td>
<td>10260</td>
</tr>
<tr>
<td>Number of lower bounds</td>
<td>780</td>
</tr>
<tr>
<td>Number of upper bounds</td>
<td>540</td>
</tr>
<tr>
<td>Number of nonzeros in Jacobian</td>
<td>49230</td>
</tr>
<tr>
<td>Number of nonzeros in Hessian</td>
<td>14700</td>
</tr>
</tbody>
</table>

Method of Full Discretization of State and Control Variables

Large-scale Sparse block-diagonal NLP
## Setpoint change studies

<table>
<thead>
<tr>
<th>Process variable</th>
<th>Type</th>
<th>Magnitude</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Production rate change</td>
<td>Step</td>
<td>-15%</td>
<td>Make a step change to the variable(s) used to set the process production rate so that the product flow leaving the stripper column base changes from 14,228 to 12,094 kg h(^{-1})</td>
</tr>
<tr>
<td>Reactor operating pressure change</td>
<td>Step</td>
<td>-60 kPa</td>
<td>Make a step change so that the reactor operating pressure changes from 2805 to 2745 kPa</td>
</tr>
<tr>
<td>Purge gas composition of component B change</td>
<td>Step</td>
<td>+2%</td>
<td>Make a step change so that the composition of component B in the gas purge changes from 13.82 to 15.82%</td>
</tr>
</tbody>
</table>

Setpoint changes for the base case [Downs & Vogel]
Case Study:
Change Reactor pressure by 60 kPa

Control profiles
All profiles return to their base case values
Same production rate
Same product quality
Same control profile
Lower pressure – leads to larger gas phase (reactor) volume
Less compressor load
TE Case Study – Results I

Shift in TE process

Same production rate

More volume for reaction

Same reactor temperature

Initially less cooling water flow (more evaporation)
Case Study - Results II

Shift in TE process

Shift in reactor effluent to more condensables

Increase cooling water flow

Increase stripper steam to ensure same purity

Less compressor work
Case Study: Change Reactor Pressure by 60 kPa

- **Optimization with IPOPT**
  - 1000 Optimization Cycles
  - 5-7 CPU seconds
  - 11-14 Iterations

- **Optimization with SNOPT**
  - Often failed due to poor conditioning
  - Could not be solved within sampling times
  - > 100 Iterations
Optimization as a Framework for Integration

+ Directly handles interactions, multiple conditions
+ Trade-offs unambiguous, quantitative

- Larger problems to solve
- Need to consider a diverse process models

Research Questions

  How should diverse models be integrated?
  Is further algorithmic development needed?
Batch Integration Case Study

- What are the Interactions between Design and Dynamics and Planning?
- What are the differences between Sequential and Simultaneous Strategies?
- Especially Important in Batch Systems
Simultaneous Dynamic Optimization

- discretize (DAEs), state and control profiles
- large-scale optimization problem
- handles profile constraints directly
- incorporates equipment variables directly
- DAE model solved only once
- converges for unstable systems
Scheduling Formulation

- sequencing of tasks, products, equipment
- expensive discrete combinatorial optimization
- consider ideal transfer policies (UIS and ZW)
- closed form relations (Birewar and Grossmann, 1989)

Unlimited Int. Storage (UIS)
Short production cycle
Cycle time independent of sequence

Zero Wait (ZW)
Immediate transfer required
Slack times dependent on pair
Longer production cycle required
Case Study Example

4 stages, 3 products of different purity
Dynamic reactor - temperature profile
Dynamic column - reflux profile

\[
\begin{align*}
A + B & \rightarrow C \\
C + B & \rightarrow P + E \\
P + C & \rightarrow G
\end{align*}
\]

Process Optimization Cases

SQ - Sequential Design - Scheduling - Dynamics
SM - Simultaneous Design and Scheduling
   Dynamics with endpoints fixed.
SM* - Simultaneous Design, Scheduling and Dynamics
Scenarios in Case Study

Comparison of Dynamic vs. Best Constant Profiles

R0 - best constant temperature profile
R1 - optimal temperature policy

C0 - best constant reflux ratio
C1 - optimal reflux ratio
Results for Simultaneous Cases

- **R0/R1**
  - best constant / optimal temperature

- **C0/C1**
  - best constant / optimal reflux ratio

- ZW schedule becomes tighter
- less dependent on product sequences
Summary

**Sequential Approaches**
- Parameter Optimization
  - Gradients by: Direct and Adjoint Sensitivity Equations
- Optimal Control (Profile Optimization)
  - Variational Methods
  - NLP-Based Methods
- Require Repeated Solution of Model
- State Constraints are Difficult to Handle

**Simultaneous Approach**
- Discretize ODE's using orthogonal collocation on finite elements (solve larger optimization problem)
- Accurate approximation of states, location of control discontinuities through element placement.
- Straightforward addition of state constraints.
- Deals with unstable systems

**Simultaneous Strategies are Effective**
- Directly enforce constraints
- Solve model only once
- Avoid difficulties at intermediate points

**Large-Scale Extensions**
- Exploit structure of DAE discretization through decomposition
- Large problems solved efficiently with IPOPT
DAE Optimization Resources

References

Software
• Dynamic Optimization Codes
ACM – Aspen Custom Modeler
DynoPC - simultaneous optimization code (CMU)
COOPT - sequential optimization code (Petzold)
gOPT - sequential code integrated into gProms (PSE)
MUSCOD - multiple shooting optimization (Bock)
NOVA - SQP and collocation code (DOT Products)
• Sensitivity Codes for DAEs
DASOLV - staggered direct method (PSE)
DASPK 3.0 - various methods (Petzold)
SDASAC - staggered direct method (sparse)
DDASAC - staggered direct method (dense)
DynoPC – Windows Implementation

Model in FORTRAN
Data/Config.

F2c/C++
ADOL_C
Preprocessor

DDASSL
COLDAE
Simulator/Discretizer

F/DF

Outputs/Graphic & Texts

Interface of DynoPC

Reduction SQP

Optimal Solution
Interior Point
Calc. of independent variable move
For QP
Decomposition

LINEARIZATION
FILTER

Starting Point

Calc. of independent variable move
Decomposition
Line Search
FILTER

Calc. of independent variable move
Decomposition
Line Search
FILTER

Calculating the independent variable move
Decomposition
Line Search
FILTER

DynoPC – Windows Implementation
Example: Batch Reactor Temperature

\[ \text{Max } b(t_f) \]

\[ s.t. \]

\[ \frac{da}{at} = -k_1 \exp\left(-\frac{E_1}{RT}\right) \cdot a \]

\[ \frac{db}{at} = k_1 \exp\left(-\frac{E_1}{RT}\right) \cdot a - k_2 \exp\left(-\frac{E_2}{RT}\right) \cdot b \]

\[ a + b + c = 1 \]
Example: Car Problem

Min $t_f$

s.t. $z_1 = z_2$

$z_2' = u$

$z_2 \leq z_{\max}$

$-2 \leq u \leq 1$

subroutine model(nz,ny,nu,np,t,z,dmz,y,u,p,f)
double precision t, z(nz), dmz(nz), y(ny), u(nu), p(np)

double precision f(nz+ny)
f(1) = p(1)*z(2) - dmz(1)
f(2) = p(1)*u(1) - dmz(2)

return
end
Example: Crystallizer Temperature

SUBROUTINE model(nz,ny,nu,np,x,z,dmz,y,u,p,f)
implicit double precision (a-h,o-z)
double precision f(nz+ny),z(nz),dmz(nz),Y(ny),yp(4),u(1)
double precision kgr, ln0, ls0, kc, ke, kex, lau, deltT, alpha
dimension a(0:3), b(0:3)
data alpha/1.d-4/,a/-66.4309d0, 2.8604d0, -0.22579d0, 6.7117d-5/,
+ b/16.08852d0, -2.708263d0 ,0.670694d0, -3.5685d-4/, kgr/4.18d-3/,
+ en /1.1d0/, ln0/5.5d-5/, Bn /3.85d2/, em /5.72d-6/, ws0/2.d0/,
+ Ls0/5.4d-4/, Kc /35.d0/, Kex/65.d0/, are/5.8d0 /
+ amt/60.d0/, V0/1500.d0/, cw0/80.d0/,cw1/45.d0,,v1/200.d0/,
+ tm1/55.d0/,x6r/0d0/, tem/0.15d0/,claau/1580.d0/,lau/1.35d0/,
+ cp/0.4d0/,cbata/1.2d0/, calfa/.2d0/, cwt/10.d0/
ke = kex*area
x7i = cw0*lau/(100.d0-cw0)
v = (1.d0 - cw0/100.d0)*v0
w = lau*v0
yp(1) = (deltT + dsqrt(deltT**2 + alpha**2))*0.5d0
yp(2) = (a(0) + a(1)*yp(4) + a(2)*yp(4)**2 + a(3)*yp(4)**3)
yp(3) = (b(0) + b(1)*yp(4) + b(2)*yp(4)**2 + b(3)*yp(4)**3)
deltT = yp(2) - z(8)
yp(4) = 100.d0*z(7)/(lau+z(7))
f(1) = Kgr*z(1)**0.5*yp(1)**en - dmz(1)
f(2) = Bn*yp(1)**en*1.d-6 - dmz(2)
f(3) = (z(2)**dmz(1)+ dmz(2)* Ln0)**1.d+6*1.d-4) - dmz(3)
f(4) = (2.d0*cbata*z(3)*1.d+4*dmz(1)+dmz(2)*Ln0**2*1.d+6)-dmz(4)
f(5) = (3.d0*calfa*z(4)**dmz(1)+ dmz(2)**2*Ln0**3*1.d+6) - dmz(5)
f(6) = (3.d0*ws0/(ls0**3)*z(1)**2*dmz(1)+claau*V*dmz(5))-dmz(6)
f(7) = -dmz(6)/V - dmz(7)
f(8) = (Ke*dmz(6) - Ke*(z(8) - u(1)))/(w*cp) - dmz(8)
f(9) = y(1)+YP(3)- u(1)
return
de