Recent developments in DNOPT and SNOPT

Philip E. Gill & Elizabeth Wong

Department of Mathematics
University of California, San Diego

Joint work with Michael Saunders

SIAM Optimization Meeting 2014, San Diego
1. Introduction
2. Recent developments
3. Using second derivatives
4. Conclusion
Outline

1. Introduction
2. Recent developments
3. Using second derivatives
4. Conclusion
**Introduction**

**minimize** \[ f(x) \]
\[ \text{subject to } c(x) = 0, \ x \geq 0 \]

- Large-scale nonlinear problems
- \( f(x) \) is smooth with gradient \( g(x) \)
- \( c(x) \) is a vector of nonlinear constraints
- \( J(x) \) is the sparse \( m \times n \) Jacobian matrix of \( c(x) \)
Overview

We are interested in sequential quadratic programming (SQP) methods to solve the NLP:

At each major iteration:
- Form a subproblem that minimize a quadratic model of the objective function subject to linearized constraints at the current point
- Solve the QP subproblem (minor iterations)
- Update the point
- Check whether the new point is “good” enough
- Repeat until converged
Overview

We are interested in sequential quadratic programming (SQP) methods to solve the NLP:

At each major iteration:

- Form a subproblem that minimize a quadratic model of the objective function subject to linearized constraints at the current point
- Solve the QP subproblem (minor iterations)
- Update the point
- Check whether the new point is “good” enough
- Repeat until converged
SNOPT7 Features

**SNOPT7** is a Fortran 77 implementation of a particular SQP method

- Limited/full-memory quasi-Newton approximation of Lagrangian Hessian
  - ⇒ Convex QP subproblems
- Uses the convex QP solver **SQOPT** for subproblems
  - Reduced-Hessian, reduced-gradient active-set method
  - Solves dense systems of the form $Z^THZp = -Z^Tg$, where the size of $Z^THZ$ is the number of superbasic variables $n_s$
SNOPT7 Features

**SNOPT7** is a Fortran 77 implementation of a particular SQP method

- Limited/full-memory quasi-Newton approximation of Lagrangian Hessian
  - Convex QP subproblems
- Uses the convex QP solver **SQOPT** for subproblems
  - Reduced-Hessian, reduced-gradient active-set method
  - Solves dense systems of the form $Z^THZp = -Z^Tg$, where the size of $Z^THZ$ is the number of superbasic variables $n_s$
SNOPT7 Features

SNOPT7 is a Fortran 77 implementation of a particular SQP method

- Limited/full-memory quasi-Newton approximation of Lagrangian Hessian
  ⇒ Convex QP subproblems
- Uses the convex QP solver SQOPT for subproblems
  - Reduced-Hessian, reduced-gradient active-set method
  - Solves dense systems of the form $Z^T H Z p = -Z^T g$, where the size of $Z^T H Z$ is the number of superbasic variables $n_s$
SNOPT7 Features

- Exploit sparsity in the problem
- Differentiate between linear and nonlinear variables and constraints
- Use all or some first derivatives

Deficiencies of SNOPT7

1. Fortran 77 \( \Rightarrow \) no dynamic allocation; user needs to estimate space
2. Problems with large number of superbasics \((Z^T H Z p = -Z^T g)\), SNOPT7 enters CG mode
3. No second derivative information
SNOPT7 Features

- Exploit sparsity in the problem
- Differentiate between linear and nonlinear variables and constraints
- Use all or some first derivatives

Deficiencies of SNOPT7

1. Fortran 77 ⇒ no dynamic allocation; user needs to estimate space
2. Problems with large number of superbasics ($Z^THZp = -Z^Tg$), SNOPT7 enters CG mode
3. No second derivative information
SNOPT7 Features

- Exploit sparsity in the problem
- Differentiate between linear and nonlinear variables and constraints
- Use all or some first derivatives

Deficiencies of SNOPT7

1. Fortran 77 ⇒ no dynamic allocation; user needs to estimate space
2. Problems with large number of superbasics
   \[(Z^THZp = -Z^Tg), \text{ SNOPT7 enters CG mode}\]
3. No second derivative information
SNOPT7 Features

- Exploit sparsity in the problem
- Differentiate between linear and nonlinear variables and constraints
- Use all or some first derivatives

Deficiencies of SNOPT7

1. Fortran 77 $\Rightarrow$ no dynamic allocation; user needs to estimate space
2. Problems with large number of superbasics $(Z^THZp = -Z^Tg)$, SNOPT7 enters CG mode
3. No second derivative information
SNOPT7 Features

- Exploit sparsity in the problem
- Differentiate between linear and nonlinear variables and constraints
- Use all or some first derivatives

Deficiencies of SNOPT7

1. Fortran 77 \Rightarrow no dynamic allocation; user needs to estimate space
2. Problems with large number of superbasics ($Z^T H Z p = - Z^T g$), SNOPT7 enters CG mode
3. No second derivative information
SNOPT7 Features

- Exploit sparsity in the problem
- Differentiate between linear and nonlinear variables and constraints
- Use all or some first derivatives

Deficiencies of SNOPT7

1. Fortran 77 ⇒ no dynamic allocation; user needs to estimate space
2. Problems with large number of superbasics \((Z^T H Z p = - Z^T g)\), SNOPT7 enters CG mode
3. No second derivative information
SNOPT7 Features

- Exploit sparsity in the problem
- Differentiate between linear and nonlinear variables and constraints
- Use all or some first derivatives

Deficiencies of SNOPT7

1. **Fortran 77**  ⇒  no dynamic allocation; user needs to estimate space

2. Problems with large number of superbasics  
   \( Z^T H Z p = - Z^T g \), SNOPT7 enters CG mode

3. No second derivative information
SNOPT7 Features

- Exploit sparsity in the problem
- Differentiate between linear and nonlinear variables and constraints
- Use all or some first derivatives

Deficiencies of SNOPT7

1. Fortran 77 ⇒ no dynamic allocation; user needs to estimate space
2. Problems with large number of superbasics \((Z^THZp = -Z^Tg)\), SNOPT7 enters CG mode
3. No second derivative information
Results on CUTEst test set

- 1092 problems from the CUTEst test set
- Biggest problem has \((m, n) \approx (250000, 250000)\)
- Time limit of two hours per problem

We compared the number of function evaluations of
- **SNOPT7** with no superbasic limit (CG mode for \(> 2000\))
- **SNOPT7** with superbasic limit of 2000
- **IPOPT** with ma57
Performance profile of function evaluations

With time, IPOPT does best, solving all problems in about 1.5 days; SNOPT solves in just over 2 days.

UCSD Center for Computational Mathematics
Performance profile of function evaluations

With time, IPOPT does best, solving all problems in about 1.5 days; SNOPT solves in just over 2 days.
Recent Developments

- **SNOPT9**, Fortran 2003 version **SNOPT7**
- Automatic allocation of workspace
- "Simpler" user interface
- New QP solver **SQIC**
  - Combination of variable-reduction and block-matrix method
  - Can use third-party linear solvers (**LUSOL**, **HSL_MA57**, **UMFPACK**, **SuperLU**)
  - No CG!
- Utilizing second-derivative information
  - concurrent QP convexification
  - post-convexification
- Added option for circular buffer, limited-memory quasi-Newton [Bradley 2010]
Recent Developments

- **SNOPT**9, Fortran 2003 version **SNOPT**7
  - Automatic allocation of workspace
  - “Simpler” user interface
  - New QP solver **SQIC**
    - Combination of variable-reduction and block-matrix method
    - Can use third-party linear solvers (**LUSOL**, **HSL_MA57**, **UMFPACK**, **SuperLU**)
    - No CG!
  - Utilizing second-derivative information
    - concurrent QP convexification
    - post-convexification
  - Added option for circular buffer, limited-memory quasi-Newton [Bradley 2010]
Recent Developments

- **SNOPT9**, Fortran 2003 version **SNOPT7**
- Automatic allocation of workspace
  - "Simpler" user interface
- New QP solver **SQIC**
  - Combination of variable-reduction and block-matrix method
  - Can use third-party linear solvers (**LUSOL**, **HSL_MA57**, **UMFPACK**, **SuperLU**)
  - No CG!
- Utilizing second-derivative information
  - concurrent QP convexification
  - post-convexification
- Added option for circular buffer, limited-memory quasi-Newton [Bradley 2010]
**Recent Developments**

- **SNOPT9**, Fortran 2003 version **SNOPT7**
- Automatic allocation of workspace
- “Simpler” user interface
  - New QP solver **SQIC**
    - Combination of variable-reduction and block-matrix method
    - Can use third-party linear solvers (*LUSOL*, *HSL_MA57*, *UMFPACK*, *SuperLU*)
    - No CG!
- Utilizing second-derivative information
  - concurrent QP convexification
  - post-convexification
- Added option for circular buffer, limited-memory quasi-Newton [Bradley 2010]
Recent Developments

- **SNOPT9**, Fortran 2003 version **SNOPT7**
- Automatic allocation of workspace
- “Simpler” user interface
- New QP solver **SQIC**
  - Combination of variable-reduction and block-matrix method
  - Can use third-party linear solvers (**LUSOL**, **HSL_MA57**, **UMFPACK**, **SuperLU**)
  - No CG!
- Utilizing second-derivative information
  - concurrent QP convexification
  - post-convexification
- Added option for circular buffer, limited-memory quasi-Newton [Bradley 2010]
**Recent Developments**

- **SNOPT9**, Fortran 2003 version **SNOPT7**
- Automatic allocation of workspace
- "Simpler" user interface
- New QP solver **SQIC**
  - Combination of variable-reduction and block-matrix method
  - Can use third-party linear solvers (**LUSOL**, **HSL_MA57**, **UMFPACK**, **SuperLU**)
  - No CG!
- Utilizing second-derivative information
  - concurrent QP convexification
  - post-convexification
- Added option for circular buffer, limited-memory quasi-Newton [Bradley 2010]
Recent Developments

- **SNOPT9**, Fortran 2003 version **SNOPT7**
- Automatic allocation of workspace
- “Simpler” user interface
- New QP solver **SQIC**
  - Combination of variable-reduction and block-matrix method
  - Can use third-party linear solvers (**LUSOL**, **HSL_MA57**, **UMFPACK**, **SuperLU**)
  - No CG!
- Utilizing second-derivative information
  - concurrent QP convexification
  - post-convexification
- Added option for circular buffer, limited-memory quasi-Newton [Bradley 2010]
Conjugate gradient vs block-matrix mode

- 85 problems where SNOPT7 hit the superbasic limit
- Compared SNOPT7 with CG mode and SNOPT9 with block-matrix mode
- Time limit of one hour per problem
QP subproblem

We minimize a quadratic model of the objective subject to linearized constraints

\[
\begin{align*}
\text{minimize} & \quad g^T(x - x_0) + \frac{1}{2}(x - x_0)^T H(x - x_0) \\
\text{subject to} & \quad c + J(x - x_0) = 0, \quad x \geq 0
\end{align*}
\]

- In SNOPT7, \( H_k \) is a positive-semidefinite approximation of the Hessian of the Lagrangian function \( \nabla^2 \mathcal{L}(x_k, \pi_k) \)
- In SNOPT9, \( H_k \) is the exact Hessian of the Lagrangian function (for some QP subproblems)
- \( \pi \) are the multipliers for the equality constraints
- \( z = g + H(x - x_0) - J^T \pi \) are the multipliers for the bounds
Using exact second derivatives

When using the exact Hessian, $H$ may not be positive semidefinite and the QP subproblem may be indefinite. To avoid an indefinite subproblem, we convexify $H$, but only when the QP direction has negative curvature.

If a QP search direction $p$ has negative curvature $p^T Hp < 0$, then we define $\sigma > 0$ to diagonally modify $H$ such that

- $p^T Hp + \sigma > 0$
- $H \leftarrow H + \sigma e_s e_s^T$
Using exact second derivatives

When using the exact Hessian, $H$ may not be positive semidefinite and the QP subproblem may be indefinite. To avoid an indefinite subproblem, we convexify $H$, but only when the QP direction has negative curvature.

If a QP search direction $p$ has negative curvature $p^T Hp < 0$, then we define $\sigma > 0$ to diagonally modify $H$ such that

- $p^T Hp + \sigma > 0$
- $H \leftarrow H + \sigma e_s e_s^T$
Concurrent QP convexification

Suppose we have a nonoptimal multiplier $z_s < 0$ at $x$

In SQIC, a QP search direction $p$ is computed such that

$$p = P \begin{pmatrix} p_B \\ e_s \end{pmatrix}$$

where $B$ is the set of basic (free/inactive) variables ($x_i > 0$). Remaining variables are nonbasic (fixed/active) variables.

Thus, the curvature along $p$ is

$$p^T H p = \begin{pmatrix} p_B^T & e_s^T \end{pmatrix} P^T H P \begin{pmatrix} p_B \\ e_s \end{pmatrix}$$

$$= \ldots \text{some terms} \ldots + e_s^T H e_s$$

$$= \ldots \text{some terms} \ldots + h_{ss}$$
Concurrent QP convexification

Suppose we have a nonoptimal multiplier \( z_s < 0 \) at \( x \).

In SQIC, a QP search direction \( p \) is computed such that

\[
p = P \left( \begin{pmatrix} p_B \\ e_s \end{pmatrix} \right)
\]

where \( B \) is the set of basic (free/inactive) variables (\( x_i > 0 \)). Remaining variables are nonbasic (fixed/active) variables.

Thus, the curvature along \( p \) is

\[
p^T H p = \left( p_B^T \ e_s^T \right) P^T H P \begin{pmatrix} p_B \\ e_s \end{pmatrix}
\]

\[
= \ldots \text{some terms} \ldots + e_s^T H e_s
\]

\[
= \ldots \text{some terms} \ldots + h_{ss}
\]
Concurrent QP convexification

If $p^T Hp < 0$, then we perturb the curvature by adding $\sigma$ to $(s, s)$-th element of $H$

$$H \Rightarrow H + \sigma e_s e_s^T = \bar{H}$$
$$p^T Hp \Rightarrow \text{some terms...} + (h_{ss} + \sigma)$$
How do we define $\sigma$?

Obviously, $\sigma > \sigma_{\text{min}} = -p^T Hp$ for positive curvature.

What happens to the multipliers $z = g + H(x - x_0) - J^T \pi$ when we perturb $H$ to $\tilde{H} = H + \sigma e_s e_s^T$?

$$z \leftarrow g + \tilde{H}(x - x_0) - J^T \pi = z + \sigma e_s e_s^T (x - x_0)$$

$\Rightarrow$ Only $z_s$ is perturbed by $\sigma (x - x_0)_s$
How do we define $\sigma$?

Obviously, $\sigma > \sigma_{\text{min}} = -p^T Hp$ for positive curvature

What happens to the multipliers $z = g + H(x - x_0) - J^T \pi$ when we perturb $H$ to $\bar{H} = H + \sigma e_s e_s^T$?

$$z \leftarrow g + \bar{H}(x - x_0) - J^T \pi = z + \sigma e_s e_s^T (x - x_0)$$

$\Rightarrow$ Only $z_s$ is perturbed by $\sigma (x - x_0)_s$
Concurrent QP convexification

How do we define $\sigma$?
Obviously, $\sigma > \sigma_{\text{min}} = -p^T H p$ for positive curvature

What happens to the multipliers $z = g + H(x - x_0) - J^T \pi$ when we perturb $H$ to $\tilde{H} = H + \sigma e_s e_s^T$?

$$z \leftarrow g + \tilde{H}(x - x_0) - J^T \pi = z + \sigma e_s e_s^T(x - x_0)$$

$\Rightarrow$ Only $z_s$ is perturbed by $\sigma(x - x_0)_s$
Concurrent QP convexification

How do we define $\sigma$?

Obviously, $\sigma > \sigma_{\text{min}} = -p^T H p$ for positive curvature.

What happens to the multipliers $z = g + H(x - x_0) - J^T \pi$ when we perturb $H$ to $\bar{H} = H + \sigma e_s e_s^T$?

\[
z \leftarrow g + \bar{H}(x - x_0) - J^T \pi = z + \sigma e_s e_s^T(x - x_0)
\]

$\Rightarrow$ Only $z_s$ is perturbed by $\sigma(x - x_0)_s$
Recall $z_s < 0$. Let $z_s(\sigma) = z_s + \sigma(x - x_0)_s$

Case 1: Assume $(x - x_0)_s > 0$ so $z_s(\sigma)$ is an increasing function
Recall $z_s < 0$. Let $z_s(\sigma) = z_s + \sigma (x - x_0)_s$

Case 1: Assume $(x - x_0)_s > 0$ so $z_s(\sigma)$ is an increasing function
Recall $z_s < 0$. Let $z_s(\sigma) = z_s + \sigma(x - x_0)_s$

Case 1: Assume $(x - x_0)_s > 0$ so $z_s(\sigma)$ is an increasing function
Recall $z_s < 0$. Let $z_s(\sigma) = z_s + \sigma(x - x_0)_s$

Case 1: Assume $(x - x_0)_s > 0$ so $z_s(\sigma)$ is an increasing function
Recall $z_s < 0$. Let $z_s(\sigma) = z_s + \sigma(x - x_0)_s$

Case 1: Assume $(x - x_0)_s > 0$ so $z_s(\sigma)$ is an increasing function
Recall $z_s < 0$. Let $z_s(\sigma) = z_s + \sigma(x - x_0)_s$

Case 1: Assume $(x - x_0)_s > 0$ so $z_s(\sigma)$ is an increasing function
Recall $z_s < 0$. Let $z_s(\sigma) = z_s + \sigma(x - x_0)_s$

Case 1: Assume $(x - x_0)_s > 0$ so $z_s(\sigma)$ is an increasing function
Recall $z_s < 0$. Let $z_s(\sigma) = z_s + \sigma(x - x_0)_s$

Case 1: Assume $(x - x_0)_s > 0$ so $z_s(\sigma)$ is an increasing function
Case 1: \((x - x_0)_s > 0\)

\[ \sigma = \max\{\sigma_z, 2\sigma_{\text{min}}\} \]

- \(z_s(\sigma)\) no longer nonoptimal. No step taken. Check for other nonoptimal multipliers and continue with the QP algorithm
- Minimal changes to algorithm
- No extra factorizations or solves necessary
Case 2: Assume \((x - x_0)_s \leq 0\). Then \(z_s(\sigma) < 0\) for all \(\sigma > 0\)

Choose \(\sigma\) to limit the optimal step length

\[
\alpha = -\frac{z_s}{p^THp} \leftarrow -\frac{z_s + \sigma (x - x_0)_s}{p^THp + \sigma}
\]

Define a target step length \(\alpha_T\) and compute \(\sigma_T\) such that \(\alpha = \alpha_T\).

\[
\Rightarrow \quad \sigma = \max(2\sigma_{\text{min}}, \sigma_T).
\]

- \(z_s(\sigma)\) is still nonoptimal; continue as usual with perturbed multiplier value
- No extra factorizations or solves necessary (Directions and factors are in terms of basic variables; \(s\) is nonbasic)
Case 2: Assume \((x - x_0)_s \leq 0\). Then \(z_s(\sigma) < 0\) for all \(\sigma > 0\).

Choose \(\sigma\) to limit the optimal step length

\[
\alpha = -\frac{z_s}{p^T H p} \leftarrow -\frac{z_s + \sigma(x - x_0)_s}{p^T H p + \sigma}
\]

Define a target step length \(\alpha_T\) and compute \(\sigma_T\) such that \(\alpha = \alpha_T\).

\[\Rightarrow \sigma = \max(2\sigma_{\text{min}}, \sigma_T).\]

- \(z_s(\sigma)\) is still nonoptimal; continue as usual with perturbed multiplier value
- No extra factorizations or solves necessary (Directions and factors are in terms of basic variables; \(s\) is nonbasic)
Case 2: Assume $(x - x_0)_s \leq 0$. Then $z_s(\sigma) < 0$ for all $\sigma > 0$

Choose $\sigma$ to limit the optimal step length

$$\alpha = -\frac{z_s}{p^THp} \leftarrow -\frac{z_s + \sigma(x - x_0)_s}{p^THp + \sigma}$$

Define a target step length $\alpha_T$ and compute $\sigma_T$ such that $\alpha = \alpha_T$.

$\Rightarrow \quad \sigma = \max(2\sigma_{\min}, \sigma_T)$.

- $z_s(\sigma)$ is still nonoptimal; continue as usual with perturbed multiplier value
- No extra factorizations or solves necessary (Directions and factors are in terms of basic variables; $s$ is nonbasic)
If the QP algorithm terminates optimally, we have a solution $(x_{QP}, \pi_{QP}, z_{QP})$ for the perturbed subproblem

$$\begin{align*}
\text{minimize} & \quad g^T(x - x_0) + \frac{1}{2}(x - x_0)^T(H + D)(x - x_0) \\
\text{subject to} & \quad c + J(x - x_0) = 0, \quad x \geq 0
\end{align*}$$

where $D$ is a diagonal, positive-semidefinite matrix.
Post-QP convexification

Given the QP solution \((x_{QP}, \pi_{QP}, z_{QP})\), the SQP direction is

\[ p = x_{QP} - x_0 \]

Compute the next iterate using a line-search on the augmented Lagrangian merit function

\[ M(x, \pi) = f(x) - \pi^T(c(x)) + \frac{1}{2} \rho \|c(x)\|^2 \]

- To satisfy conditions of descent in SNOPT, we may perturb \(H\)
- Perturbation requires minimal change

\[ p^T Hp \leftarrow p^T Hp + \gamma \geq \frac{|g_L^T p|}{\|g_L\| \|p\|} \text{ and } \pi \leftarrow \pi - \gamma c \]

- Prevent unnecessary increase of penalty parameter
Problem HS61: 2 constraint, 3 variables

---

**SNOPT9 with quasi-Newton**

<table>
<thead>
<tr>
<th>Major Minors</th>
<th>Step</th>
<th>nCon</th>
<th>Feasible</th>
<th>Optimal</th>
<th>MeritFunction</th>
<th>nS</th>
<th>Penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>59</td>
<td>1</td>
<td>1.3E-01</td>
<td>160</td>
<td>3.0E+00</td>
<td>2.3E+00</td>
<td>-1.5383319E+02</td>
<td>1</td>
</tr>
<tr>
<td>60</td>
<td>1</td>
<td>1.8E-01</td>
<td>162</td>
<td>2.5E+00</td>
<td>2.3E+00</td>
<td>-1.5090078E+02</td>
<td>1</td>
</tr>
<tr>
<td>61</td>
<td>1</td>
<td>2.6E-01</td>
<td>164</td>
<td>1.9E+00</td>
<td>2.1E+00</td>
<td>-1.4796506E+02</td>
<td>1</td>
</tr>
<tr>
<td>62</td>
<td>1</td>
<td>3.8E-01</td>
<td>166</td>
<td>1.2E+00</td>
<td>1.6E+00</td>
<td>-1.4551659E+02</td>
<td>1</td>
</tr>
<tr>
<td>63</td>
<td>1</td>
<td>1.0E+00</td>
<td>167</td>
<td>9.4E-02</td>
<td>2.9E-02</td>
<td>-1.4377037E+02</td>
<td>1</td>
</tr>
<tr>
<td>64</td>
<td>1</td>
<td>8.4E-02</td>
<td>169</td>
<td>8.6E-02</td>
<td>8.6E-02</td>
<td>-1.4364945E+02</td>
<td>1</td>
</tr>
<tr>
<td>65</td>
<td>1</td>
<td>4.2E-01</td>
<td>171</td>
<td>5.0E-02</td>
<td>9.3E-02</td>
<td>-1.4365368E+02</td>
<td>1</td>
</tr>
<tr>
<td>66</td>
<td>1</td>
<td>1.0E+00</td>
<td>172</td>
<td>1.8E-04</td>
<td>6.2E-04</td>
<td>-1.4364614E+02</td>
<td>1</td>
</tr>
<tr>
<td>67</td>
<td>1</td>
<td>1.0E+00</td>
<td>173</td>
<td>(1.3E-09)</td>
<td>2.8E-05</td>
<td>-1.4364614E+02</td>
<td>1</td>
</tr>
<tr>
<td>68</td>
<td>1</td>
<td>1.0E+00</td>
<td>174</td>
<td>(5.5E-12)</td>
<td>(1.1E-06)</td>
<td>-1.4364614E+02</td>
<td>1</td>
</tr>
</tbody>
</table>

Problem name: HS61

- No. of iterations: 72
- Objective value: -1.4364614220E+02
- No. of major iterations: 68
- Linear objective: 0.0000000000E+00
- Penalty parameter: 2.595E+00
- Nonlinear objective: -1.4364614220E+02
- No. of calls to funobj: 174
- No. of calls to funcon: 174

---

**SNOPT9 with exact Hessian**

<table>
<thead>
<tr>
<th>Major Minors</th>
<th>Step</th>
<th>nCon</th>
<th>Feasible</th>
<th>Optimal</th>
<th>MeritFunction</th>
<th>nS</th>
<th>Penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>1</td>
<td>1.0E+00</td>
<td>31</td>
<td>(9.0E-08)</td>
<td>1.6E-03</td>
<td>-1.4364614E+02</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>1.0E+00</td>
<td>32</td>
<td>(1.7E-08)</td>
<td>7.1E-04</td>
<td>-1.4364614E+02</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>1.0E+00</td>
<td>33</td>
<td>(3.2E-09)</td>
<td>3.1E-04</td>
<td>-1.4364614E+02</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>1.0E+00</td>
<td>34</td>
<td>(6.1E-10)</td>
<td>1.3E-04</td>
<td>-1.4364614E+02</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>1.0E+00</td>
<td>35</td>
<td>(1.1E-10)</td>
<td>5.8E-05</td>
<td>-1.4364614E+02</td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>1.0E+00</td>
<td>36</td>
<td>(2.2E-11)</td>
<td>2.5E-05</td>
<td>-1.4364614E+02</td>
<td>1</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>1.0E+00</td>
<td>37</td>
<td>(4.1E-12)</td>
<td>1.1E-05</td>
<td>-1.4364614E+02</td>
<td>1</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>1.0E+00</td>
<td>38</td>
<td>(7.7E-13)</td>
<td>4.8E-06</td>
<td>-1.4364614E+02</td>
<td>1</td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>1.0E+00</td>
<td>39</td>
<td>(1.4E-13)</td>
<td>2.1E-06</td>
<td>-1.4364614E+02</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>1.0E+00</td>
<td>40</td>
<td>(2.7E-14)</td>
<td>(9.0E-07)</td>
<td>-1.4364614E+02</td>
<td>1</td>
</tr>
</tbody>
</table>

Problem name: HS61

- No. of iterations: 37
- Objective value: -1.4364614220E+02
- No. of major iterations: 30
- Linear objective: 0.0000000000E+00
- Penalty parameter: 3.866E+00
- Nonlinear objective: -1.4364614220E+02
- No. of calls to funobj: 40
- No. of calls to funcon: 40
- Calls for the Hessian: 26
- Hessian products: 0
### Problem HS38: 1 constraint, 4 variables

---

**SNOPT9 with quasi-Newton**

<table>
<thead>
<tr>
<th>Major Minors</th>
<th>Step</th>
<th>nObj</th>
<th>Feasible</th>
<th>Optimal</th>
<th>Objective</th>
<th>nS</th>
</tr>
</thead>
<tbody>
<tr>
<td>87</td>
<td>1.0E+00</td>
<td>107</td>
<td>2.8E-01</td>
<td>6.2459917E-03</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>88</td>
<td>1.0E+00</td>
<td>108</td>
<td>2.0E-01</td>
<td>1.8582680E-03</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>89</td>
<td>1.0E+00</td>
<td>109</td>
<td>1.8E-01</td>
<td>1.2670335E-04</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>1.0E+00</td>
<td>110</td>
<td>3.9E-02</td>
<td>3.494491E-05</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>91</td>
<td>1.0E+00</td>
<td>111</td>
<td>1.8E-02</td>
<td>3.3437832E-06</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>1.0E+00</td>
<td>112</td>
<td>6.0E-03</td>
<td>4.7524755E-07</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>93</td>
<td>1.0E+00</td>
<td>113</td>
<td>1.3E-03</td>
<td>1.4403776E-08</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>1.0E+00</td>
<td>114</td>
<td>1.7E-04</td>
<td>1.7608449E-10</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>1.0E+00</td>
<td>115</td>
<td>5.8E-06</td>
<td>3.8753016E-13</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>96</td>
<td>1.0E+00</td>
<td>116</td>
<td>(1.6E-07)</td>
<td>2.9152976E-16</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Problem name HS38

- No. of iterations: 105
- Objective value: 2.9152976482E-16
- No. of major iterations: 96
- Linear objective: 0.0000000000E+00
- Penalty parameter: 0.000E+00
- Nonlinear objective: 2.9152976482E-16
- No. of calls to funobj: 116
- No. of calls to funcon: 0

---

**SNOPT9 with exact Lagrangian Hessian**

<table>
<thead>
<tr>
<th>Major Minors</th>
<th>Step</th>
<th>nObj</th>
<th>Feasible</th>
<th>Optimal</th>
<th>Objective</th>
<th>nS</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>1.0E+00</td>
<td>61</td>
<td>2.0E+00</td>
<td>8.3116690E-01</td>
<td>4 L</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>1.0E+00</td>
<td>62</td>
<td>5.2E+00</td>
<td>6.3686465E-01</td>
<td>4 L</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>1.0E+00</td>
<td>63</td>
<td>6.6E-01</td>
<td>2.7536799E-01</td>
<td>4 L</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>4.7E-01</td>
<td>65</td>
<td>2.8E+00</td>
<td>1.5105949E-01</td>
<td>4 L</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>1.0E+00</td>
<td>66</td>
<td>8.0E-01</td>
<td>4.5350619E-02</td>
<td>4 L</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>1.0E+00</td>
<td>67</td>
<td>1.2E+00</td>
<td>1.0046352E-02</td>
<td>4 s L</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>1.0E+00</td>
<td>68</td>
<td>1.2E-01</td>
<td>5.7842738E-04</td>
<td>4 L</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>1.0E+00</td>
<td>69</td>
<td>2.7E-02</td>
<td>4.6893472E-06</td>
<td>4 L</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>1.0E+00</td>
<td>70</td>
<td>8.1E-05</td>
<td>2.3522524E-10</td>
<td>4 L</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>1.0E+00</td>
<td>71</td>
<td>(1.2E-08)</td>
<td>9.2391628E-19</td>
<td>4 L</td>
<td></td>
</tr>
</tbody>
</table>

Problem name HS38

- No. of iterations: 62
- Objective value: 9.2391628256E-19
- No. of major iterations: 45
- Linear objective: 0.0000000000E+00
- Penalty parameter: 0.000E+00
- Nonlinear objective: 9.2391628256E-19
- No. of calls to funobj: 71
- No. of calls to funcon: 0
- Calls for the Hessian: 35
- Hessian products: 0
Conclusions

- New QP solver **SQIC**
- Preliminary implementation of second derivatives in **SNOPT9**
- A better choice for minimum value of $\sigma$?
- Deciding when to use the Hessian of the Lagrangian
- Line search based on descent direction and negative curvature?
- Convergence to second-order point?
Software information at http://ccom.ucsd.edu/~optimizers

