Recent progress in PENNON

Michal Kočvara and Michael Stingl

UTIA AV ČR Prague and University of Erlangen-Nürnberg
Outline

1. PENNON — what is it?
2. basic algorithm
3. what’s new
   - PCG
   - modified Newton + TR
4. what’s missing (initialization!)
5. results
   - NLP (COPS3)
   - SDP (TOH collection)
   - BMI
PENNON (PENalty methods for NONlinear optimization) 
a collection of codes for NLP, (linear) SDP and BMI

– one algorithm to rule them all –
PENN NON (PENalty methods for NONlinear optimization)
a collection of codes for NLP, (linear) SDP and BMI

– one algorithm to rule them all –

READY

• PENNL P  AMPL, MATLAB, C/Fortran
• PENSDP  MATLAB/YALMIP, SDPA, C/Fortran
• PENBMI  MATLAB/YALMIP, C/Fortran
PENNNon (Penalty methods for Nonlinear optimization)
a collection of codes for NLP, (linear) SDP and BMI

– one algorithm to rule them all –

Ready

- PENNLp  AMPL, MATLAB, C/Fortran
- PENSDP  MATLAB/YALMIP, SDPA, C/Fortran
- PENBMI  MATLAB/YALMIP, C/Fortran

To come

- PENPMI  polynomial matrix inequalities
- PENNON  NLP + polynomial matrix inequalities
The NLP-SDP problem

\[
\min_{x \in \mathbb{R}^n} f(x) \quad \text{(NLP-SDP)}
\]
\[
\text{s.t.} \quad g_i(x) \leq 0, \quad i = 1, \ldots, m_g
\]
\[
\mathcal{A}(x) \preceq 0
\]

\(f, g_i : \mathbb{R}^n \rightarrow \mathbb{R}\) smooth
\(\mathcal{A} : \mathbb{R}^n \rightarrow S^{m_A}\) generally nonconvex
Based on the PBM method:

R. Polyak ’87
Ben-Tal, Zibulevsky ’92, ’97
Breitfeld, Shanno ’94

“Penalty/barrier function:”

\[ \varphi(t) \]

\[ \varphi'(t) \]
With $p_i > 0$ for $i \in \{1, \ldots, m\}$, we have

$$g_i(x) \leq 0 \iff p_i \varphi\left(\frac{g_i(x)}{p_i}\right) \leq 0, \quad i = 1, \ldots, m$$

and

$$A(x) \preceq 0 \iff \Phi_P(A(x)) \preceq 0.$$
With $p_i > 0$ for $i \in \{1, \ldots, m\}$, we have

$$g_i(x) \leq 0 \iff p_i \varphi(g_i(x)/p_i) \leq 0, \quad i = 1, \ldots, m$$

and

$$A(x) \preceq 0 \iff \Phi_P(A(x)) \preceq 0.$$  

The corresponding *augmented Lagrangian*:

$$F(x, u, U, p, P) = f(x) + \sum_{i=1}^{m_g} u_i p_i \varphi(g_i(x)/p_i) + \langle U, \Phi_P(A(x)) \rangle_{S_{mA}}$$
Augmented Lagrangian:

\[ F(x, u, U, p, P) = f(x) + \sum_{i=1}^{m_g} u_i p_i \varphi_g (g_i(x)/p_i) + \langle U, \Phi_P (A(x))\rangle_{s_mA} \]

PENNON algorithm:

(i) Find \( x^{k+1} \) satisfying \( \| \nabla_x F(x, u^k, U^k, p^k, P^k) \| \leq \varepsilon^k \)

(ii) \( u_i^{k+1} = u_i^k \varphi'_g (g_i(x^{k+1})/p_i^k), \quad i = 1, \ldots, m_g \)

\( \quad U^{k+1} = D_A \Phi_P (A(x); U^k) \)

(iii) \( p_i^{k+1} < p_i^k, \quad i = 1, \ldots, m_g \)

\( \quad P^{k+1} < P^k \)
PENNON algorithm

Augmented Lagrangian:

\[ F(x, u, U, p, P) = f(x) + \sum_{i=1}^{m_g} u_i p_i \varphi_g(g_i(x)/p_i) + \langle U, \Phi_P(A(x)) \rangle_{\Sigma A} \]

PENNON algorithm:

(i) Find \( x^{k+1} \) satisfying \( \|\nabla_x F(x, u^k, U^k, p^k, P^k)\| \leq \varepsilon^k \)

(ii) \( u_i^{k+1} = u_i^k \varphi'_g(g_i(x^{k+1})/p_i^k), \quad i = 1, \ldots, m_g \)

\( U^{k+1} = D_A \Phi_p(A(x); U^k) \)

(iii) \( p_i^{k+1} < p_i^k, \quad i = 1, \ldots, m_g \)

\( P^{k+1} < P^k \)

Step (i): (modified) Newton’s method (or TR)
Experience so far... 

(... about a year ago)

NLP
- very good for convex problems
- mixed experience with nonconvex problems (many fails)

SDP (linear)
- one of the fastest codes, in average
- troubles with large-scale problems (typical for IP codes)

What’s new:
- implementation of PCG for the Newton system
- modified Newton + TR
- many small modifications to increase robustness
When PCG helps (NLP) ?

NLP: sparse problems with dense columns in $\nabla g$

Minimize $F(x, u, p) := f(x) + \sum_{i=1}^{mg} u_i p_i \varphi(g_i(x)/p_i)$ by Newton.

Hessian:

$$H(x) = \ldots + \nabla g_i(x) \varphi'' \nabla g_i(x)^T + \varphi(x) ' \nabla^2 g_i(x) + \ldots$$

Solve $H(x) d = -\nabla g(x)$ by preconditioned CG method.
At each iteration need the product $Hz$. 
When PCG helps (NLP)?

NLP: sparse problems with dense columns in $\nabla g$

Minimize $F(x, u, p) := f(x) + \sum_{i=1}^{m_g} u_i p_i \varphi(g_i(x)/p_i)$ by Newton.

Hessian:

$$H(x) = \ldots + \nabla g_i(x) \varphi'' \nabla g_i(x)^T + \varphi(x)' \nabla^2 g_i(x) + \ldots$$

Solve $H(x)d = -\nabla g(x)$ by preconditioned CG method. At each iteration need the product $Hz$.

PENNON: assemble only “sparse part” of $H$, the rest by direct computation:

$$Hz = H_{sp}z + \sum_{\text{dense}} \gamma_i \gamma_i^T z$$
NLP: sparse problems with dense columns in $\nabla g$

Minimize $F(x,u,p) := f(x) + \sum_{i=1}^{m_g} u_i p_i \varphi(g_i(x)/p_i)$ by Newton.

Hessian:

$$H(x) = \ldots + \nabla g_i(x) \varphi'' \nabla g_i(x)^T + \varphi(x)\nabla^2 g_i(x) + \ldots$$

Solve $H(x)d = -\nabla g(x)$ by preconditioned CG method.
At each iteration need the product $Hz$.

PENNON: assemble only “sparse part” of $H$, the rest by direct computation:

$$Hz = H_{sp}z + \sum_{\text{dense}} \gamma_i \gamma_i^T z$$

Consequence: preconditioner can only use Hessian-vector products
When PCG helps (NLP) ?

NLP: sparse problems with dense columns in $\nabla g$

Minimize $F(x, u, p) := f(x) + \sum_{i=1}^{mg} u_i p_i \varphi(g_i(x)/p_i)$ by Newton.

Hessian:

$$H(x) = \ldots + \nabla g_i(x) \varphi'' \nabla g_i(x)^T + \varphi(x)' \nabla^2 g_i(x) + \ldots$$

Solve $H(x)d = -\nabla g(x)$ by preconditioned CG method.
At each iteration need the product $Hz$.

PENNON: assemble only “sparse part” of $H$, the rest by direct computation:

$$Hz = H_{sp}z + \sum_{\text{dense}} \gamma_i \gamma_i^T z$$

Example: lane-emden40 from COPS3 (19241 vars, 81 nl $\leq$)

Cholesky: 1600 MB, 26 min  PCG: 300 MB, 1 min 40 sec
Preconditioners

Should be:
– efficient (obvious but often difficult to reach)
– simple (low complexity)
– only use Hessian-vector product (NOT Hessian elements)
Preconditioners

Should be:

- efficient (obvious but often difficult to reach)
- simple (low complexity)
- only use Hessian-vector product (NOT Hessian elements)

- Diagonal
- Symmetric Gauss-Seidel
- L-BFGS (Morales-Nocedal, SIOPT 2000)
- A-inv (approximate inverse) (Benzi-Collum-Tuma, SISC 2000)

“Improves the CG performance on extremely ill-conditioned systems.”

preconditioner:

\[ M = C_k C_k^T, \quad C_{k+1} \leftarrow \alpha C_k + \beta C_k p_k p_k^T, \quad C_1 = \gamma I \]

\( \alpha, \beta, p_k \) ... by matrix-vector products

VERY preliminary results (MATLAB implementation)
Example: problem Theta2 from SDPLIB ($n = 498$)
Example: problem Theta2 from SDPLIB ($n = 498$)
1. Given a current iterate \((x, U, p)\), compute the gradient \(g\) and
Hessian \(H\) of \(F\) at \(x\).
2. Try to factorize \(H\) by Cholesky decomposition. If \(H\) is
factorizable, set \(\hat{H} = H\) and go to Step 4.
3. Compute \(\beta \in [−\lambda_{\text{min}}, −2\lambda_{\text{min}}]\), where \(\lambda_{\text{min}}\) is the minimal
eigenvalue of \(H\) and set
\[
\hat{H} = H + \beta I.
\]
4. Compute the search direction
\[
d = −\hat{H}^{-1}g.
\]
5. Perform line-search in direction \(d\). Denote the step-length by \(s\).
6. Set
\[
x_{\text{new}} = x + sd.
\]
Choose initial $\hat{\beta} > 0$. Perform Cholesky factorization of $H + \hat{\beta}I$. If the factorization fails, go to Step (i); otherwise go to Step (iii).

(i) Set $\hat{\beta} \leftarrow 2\hat{\beta}$.

(ii) Perform Cholesky factorization of $H + \hat{\beta}I$. If the factorization fails, go to Step (i); otherwise stop and return $\beta = \hat{\beta}$.

(iii) Set $\hat{\beta} \leftarrow \hat{\beta}/2$.

(iv) Perform Cholesky factorization of $H + \hat{\beta}I$. If the factorization fails stop and return $\beta = 2\hat{\beta}$; otherwise go to Step (iii).
Numerical experience:
linesearch often fails when close to the solution → low accuracy

Remedy:
When stopping crit. low ($10^{-4} - 10^{-6}$) switch to Trust-Region

In this way we obtain solutions with very high accuracy.
Modified Newton + Trust-Region

Example \texttt{minsurf100} from COPS3 (5000 vars., box constr.)

<table>
<thead>
<tr>
<th>it</th>
<th>obj</th>
<th>\texttt{&lt;U,G(x)&gt;}</th>
<th>\texttt{dF}</th>
<th>feas</th>
<th>pmin</th>
<th>Nwt</th>
<th>Fact</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.32e+00</td>
<td>1.2e+00</td>
<td>4.5e-02</td>
<td>2.8e-01</td>
<td>1.0e+00</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2.63e+00</td>
<td>9.6e+02</td>
<td>3.8e-03</td>
<td>8.5e-02</td>
<td>1.0e+00</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2.42e+00</td>
<td>1.5e+02</td>
<td>3.2e-03</td>
<td>1.7e-01</td>
<td>1.0e-01</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>2.50e+00</td>
<td>1.2e-03</td>
<td>2.8e-05</td>
<td>1.7e-09</td>
<td>1.0e-06</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>2.50e+00</td>
<td>1.2e-03</td>
<td>2.5e-07</td>
<td>9.1e-11</td>
<td>1.0e-06</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>2.50e+00</td>
<td>1.2e-04</td>
<td>1.6e-05</td>
<td>7.8e-11</td>
<td>1.0e-07</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>2.50e+00</td>
<td>1.2e-05</td>
<td>7.2e-07</td>
<td>4.7e-13</td>
<td>1.0e-08</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>2.50e+00</td>
<td>1.2e-05</td>
<td>2.9e-08</td>
<td>5.3e-14</td>
<td>1.0e-08</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>2.50e+00</td>
<td>1.2e-05</td>
<td>2.8e-09</td>
<td>8.7e-14</td>
<td>1.0e-08</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>2.50e+00</td>
<td>1.2e-05</td>
<td>1.5e-09</td>
<td>1.8e-13</td>
<td>1.0e-08</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>2.50e+00</td>
<td>1.2e-05</td>
<td>4.2e-10</td>
<td>2.2e-14</td>
<td>1.0e-08</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>
COPS 3.0 (E. Dolan, J. Moré, T. Munson, Argonne Nat. Lab.)

selection of difficult nonlinearly constrained optimization problems from applications in

- optimal design
- fluid dynamics
- parameter estimation
- optimal control
- etc.

Report compares FILTER, KNITRO, LOQO, MINOS, SNOPT.
Test results: COPS 3.0

COPS 3.0 (E. Dolan, J. Moré, T. Munson, Argonne Nat. Lab.)

selection of difficult nonlinearly constrained optimization problems from applications in
  – optimal design
  – fluid dynamics
  – parameter estimation
  – optimal control
  – etc.

Report compares FILTER, KNITRO, LOQO, MINOS, SNOPT.

PENNNon 1.0
can solve about 50% of the problems for default parameter setting
parameter tuning → 80–90% of the problems (cf. Polyak-Griva)

PENNNon 2.0
default parameters → all problems but one (solved with different pars.)
Test results: COPS 3.0

Performance profile on COPS3

- KNITRO
- LOQO
- PENNON

Recent progress in PENNON – p.19/24
# Test results: COPS 3.0

<table>
<thead>
<tr>
<th>problem</th>
<th>FILTER</th>
<th>KNITRO</th>
<th>LOQO</th>
<th>SNOPT</th>
<th>PENNON</th>
</tr>
</thead>
<tbody>
<tr>
<td>polygon200</td>
<td>‡</td>
<td>59.5</td>
<td>‡</td>
<td>‡</td>
<td>44</td>
</tr>
<tr>
<td>elec200</td>
<td>‡</td>
<td>21</td>
<td>‡</td>
<td>103</td>
<td>36</td>
</tr>
<tr>
<td>chain800</td>
<td>125</td>
<td>0.6</td>
<td>1.7</td>
<td>126</td>
<td>12</td>
</tr>
<tr>
<td>pinene400</td>
<td>1296</td>
<td>2.4</td>
<td>4.8</td>
<td>96</td>
<td>26</td>
</tr>
<tr>
<td>channel800</td>
<td>343</td>
<td>2.7</td>
<td>8.8</td>
<td>58</td>
<td>12</td>
</tr>
<tr>
<td>robot800</td>
<td>182</td>
<td>6.3</td>
<td>‡</td>
<td>102</td>
<td>184</td>
</tr>
<tr>
<td>rocket1600</td>
<td>264</td>
<td>310</td>
<td>7.6</td>
<td>143</td>
<td>70</td>
</tr>
<tr>
<td>gasoil400</td>
<td>69</td>
<td>18</td>
<td>3</td>
<td>25</td>
<td>22</td>
</tr>
<tr>
<td>torsion100</td>
<td>545</td>
<td>‡</td>
<td>2</td>
<td>91</td>
<td>2</td>
</tr>
<tr>
<td>bearing100</td>
<td>1058</td>
<td>478</td>
<td>‡</td>
<td>68</td>
<td>2</td>
</tr>
<tr>
<td>minsurf100</td>
<td>424</td>
<td>‡</td>
<td>‡</td>
<td>‡</td>
<td>4</td>
</tr>
<tr>
<td>tri_turtle</td>
<td>137</td>
<td>4</td>
<td>‡</td>
<td>‡</td>
<td>4</td>
</tr>
<tr>
<td>duct15</td>
<td>199</td>
<td>16</td>
<td>28</td>
<td>1768</td>
<td>16</td>
</tr>
<tr>
<td>lane_emd40</td>
<td>‡</td>
<td>82</td>
<td>1372</td>
<td>‡</td>
<td>196</td>
</tr>
<tr>
<td>dirichlet40</td>
<td>‡</td>
<td>329</td>
<td>1276</td>
<td>‡</td>
<td>396</td>
</tr>
<tr>
<td>glider400</td>
<td>‡</td>
<td>578</td>
<td>‡</td>
<td>‡</td>
<td>‡(46)</td>
</tr>
</tbody>
</table>
When PCG helps (SDP)?

Linear SDP, dense Hessian

\[ A = \sum_{i=1}^{n} A_i, \quad A_i \in \mathbb{R}^{m \times m} \]

Complexity of Hessian evaluation

- \( O(m_A^3n + m_A^2n^2) \) for dense matrices
- \( O(m_A^2n + K^2n^2) \) for sparse matrices
  \( (K \ldots \text{max. number of nonzeros in } A_i, \ i = 1, \ldots, n) \)

Complexity of Cholesky algorithm - linear SDP

- \( O(n^3) \) \( (\ldots \text{from PCG we expect } O(n^2)) \)

Problems with large \( n \) and small \( m \):
CG better than Cholesky (expected)
Hessian free methods

Use finite difference formula for Hessian-vector products:

\[ \nabla^2 F(x_k)v \approx \frac{\nabla F(x_k + hv) - \nabla F(x_k)}{h} \]

with \( h = (1 + \|x_k\|_2 \sqrt{\varepsilon}) \)

Complexity: Hessian-vector product = gradient evaluation
need for Hessian-vector-product type preconditioner

Limited accuracy (4–5 digits)
Stopping criterium for PENNON

Exact Hessian: $10^{-7}$ (7–8 digits in objective function)
Approximate Hessian: $10^{-4}$ (4–5 digits in objective function)

Stopping criterium for CG/QMR ???

\[ H_d = -g, \text{ stop when } \|H_d + g\| / \|g\| \leq \epsilon \]
Test results: linear SDP, dense Hessian

Stopping criterium for PENNON

Exact Hessian: \(10^{-7}\) (7–8 digits in objective function)
Approximate Hessian: \(10^{-4}\) (4–5 digits in objective function)

Stopping criterium for CG/QMR ???

\[ \mathbf{H} \mathbf{d} = -\mathbf{g}, \text{ stop when } \| \mathbf{H} \mathbf{d} + \mathbf{g} \| / \| \mathbf{g} \| \leq \epsilon \]

Experiments: \(\epsilon = 10^{-2}\) sufficient.
\(\rightarrow\) often very low (average) number of CG iterations

Complexity: \(n^3 \rightarrow kn^2, k \approx 4 - 8\)

Practice: effect not that strong, due to other complexity issues
Problems with large $n$ and small $m$

Library of examples with large $n$ and small $m$
(courtesy of Kim Toh)

CG-exact much better than Cholesky
CG-approx much better than CG-exact
<table>
<thead>
<tr>
<th>problem</th>
<th>n</th>
<th>m</th>
<th>PENSDP</th>
<th>PENSDP (APCG)</th>
<th>SDPLR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU</td>
<td>CG</td>
<td>CPU</td>
<td>iter</td>
<td></td>
</tr>
<tr>
<td>ham_7_5_6</td>
<td>1793</td>
<td>126</td>
<td>4</td>
<td>52</td>
<td>1</td>
</tr>
<tr>
<td>ham_9_8</td>
<td>2305</td>
<td>423</td>
<td>210</td>
<td>66</td>
<td>46</td>
</tr>
<tr>
<td>ham_8_3_4</td>
<td>16129</td>
<td>81274</td>
<td>104</td>
<td>52</td>
<td>21</td>
</tr>
<tr>
<td>ham_9_5_6</td>
<td>53761</td>
<td>1984</td>
<td>71</td>
<td>71</td>
<td>102</td>
</tr>
<tr>
<td>theta42</td>
<td>200</td>
<td>4722</td>
<td>51</td>
<td>269</td>
<td>393</td>
</tr>
<tr>
<td>theta6</td>
<td>4375</td>
<td>2327</td>
<td>108</td>
<td>308</td>
<td>1221</td>
</tr>
<tr>
<td>theta62</td>
<td>13390</td>
<td>68374</td>
<td>196</td>
<td>240</td>
<td>1749</td>
</tr>
<tr>
<td>theta8</td>
<td>7905</td>
<td>11947</td>
<td>263</td>
<td>311</td>
<td>1854</td>
</tr>
<tr>
<td>theta82</td>
<td>23872</td>
<td>m</td>
<td>650</td>
<td>267</td>
<td>4650</td>
</tr>
<tr>
<td>theta83</td>
<td>39862</td>
<td>m</td>
<td>1715</td>
<td>277</td>
<td>7301</td>
</tr>
<tr>
<td>theta10</td>
<td>12470</td>
<td>57516</td>
<td>492</td>
<td>278</td>
<td>4636</td>
</tr>
<tr>
<td>theta102</td>
<td>37467</td>
<td>m</td>
<td>1948</td>
<td>340</td>
<td>12275</td>
</tr>
<tr>
<td>theta103</td>
<td>62516</td>
<td>m</td>
<td>6149</td>
<td>421</td>
<td>17687</td>
</tr>
<tr>
<td>theta104</td>
<td>87845</td>
<td>m</td>
<td>8400</td>
<td>269</td>
<td></td>
</tr>
<tr>
<td>theta12</td>
<td>17979</td>
<td>t</td>
<td>843</td>
<td>240</td>
<td>8081</td>
</tr>
<tr>
<td>keller4</td>
<td>5101</td>
<td>3264</td>
<td>52</td>
<td>432</td>
<td>244</td>
</tr>
<tr>
<td>sanr200-0.7</td>
<td>6033</td>
<td>6664</td>
<td>52</td>
<td>278</td>
<td>405</td>
</tr>
<tr>
<td>problem</td>
<td>n</td>
<td>m</td>
<td>PENSDP CPU</td>
<td>PENSDP (APCG) CPU</td>
<td>SDPLR CPU</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----</td>
<td>----</td>
<td>------------</td>
<td>-------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>ham_7_5_6</td>
<td>1793</td>
<td>128</td>
<td>126</td>
<td>4</td>
<td>52</td>
</tr>
<tr>
<td>ham_9_8</td>
<td>2305</td>
<td>512</td>
<td>423</td>
<td>210</td>
<td>66</td>
</tr>
<tr>
<td>ham_8_3_4</td>
<td>16129</td>
<td>256</td>
<td>81274</td>
<td>104</td>
<td>52</td>
</tr>
<tr>
<td>ham_9_5_6</td>
<td>53761</td>
<td>512</td>
<td>1984</td>
<td>71</td>
<td>71</td>
</tr>
<tr>
<td>theta42</td>
<td>200</td>
<td>5986</td>
<td>4722</td>
<td>51</td>
<td>269</td>
</tr>
<tr>
<td>theta6</td>
<td>4375</td>
<td>300</td>
<td>2327</td>
<td>108</td>
<td>308</td>
</tr>
<tr>
<td>theta62</td>
<td>13390</td>
<td>300</td>
<td>68374</td>
<td>196</td>
<td>240</td>
</tr>
<tr>
<td>theta8</td>
<td>7905</td>
<td>400</td>
<td>11947</td>
<td>263</td>
<td>311</td>
</tr>
<tr>
<td>theta82</td>
<td>23872</td>
<td>400</td>
<td>m</td>
<td>650</td>
<td>267</td>
</tr>
<tr>
<td>theta83</td>
<td>39862</td>
<td>400</td>
<td>m</td>
<td>1715</td>
<td>277</td>
</tr>
<tr>
<td>theta10</td>
<td>12470</td>
<td>500</td>
<td>57516</td>
<td>492</td>
<td>278</td>
</tr>
<tr>
<td>theta102</td>
<td>37467</td>
<td>500</td>
<td>m</td>
<td>1948</td>
<td>340</td>
</tr>
<tr>
<td>theta103</td>
<td>62516</td>
<td>500</td>
<td>m</td>
<td>6149</td>
<td>421</td>
</tr>
<tr>
<td>theta104</td>
<td>87845</td>
<td>500</td>
<td>m</td>
<td>8400</td>
<td>269</td>
</tr>
<tr>
<td>theta12</td>
<td>17979</td>
<td>600</td>
<td>t</td>
<td>843</td>
<td>240</td>
</tr>
<tr>
<td>keller4</td>
<td>5101</td>
<td>171</td>
<td>3264</td>
<td>52</td>
<td>432</td>
</tr>
<tr>
<td>sanr200-0.7</td>
<td>6033</td>
<td>200</td>
<td>6664</td>
<td>52</td>
<td>278</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>problem</th>
<th>n</th>
<th>m</th>
<th>PENSDP (APCG)</th>
<th>RENDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>theta83</td>
<td>39862</td>
<td>400</td>
<td>460</td>
<td>440</td>
</tr>
<tr>
<td>theta103</td>
<td>62516</td>
<td>500</td>
<td>1440</td>
<td>850</td>
</tr>
<tr>
<td>theta123</td>
<td>90020</td>
<td>600</td>
<td>5286</td>
<td>1530</td>
</tr>
</tbody>
</table>