Artelys Knitro 11.0

SOCP, API, Preconditioner and much more!
Artelys

- We specialize in **optimization, decision-support, modeling** and deliver efficient solutions to complex business problems.

**Domains of expertise**

- Energy
- Transport & Logistic
- Defense
- Numerical and Combinatorial Optimization

**Services**

- Auditing & Consulting
- On demand software
- Distribution and support of numerical optimization tools
- Training
KNITRO
- Industry leading solver for very large, difficult nonlinear optimization problems (NLP, MINLP)

FICO Xpress Optimization Suite
- High performance linear, quadratic and mixed integer programming solver (LP, MIP, QP)

Artelys Kalis
- Object-oriented environment to model and solve problems with constraints programming techniques

AMPL
- Comprehensive modeling language for Mathematical Programming
### Background
- Created in 2001 by Ziena Optimization
  - Spin-off of Northwestern University
- Now developed and supported by Artelys

### Key features
- **Efficient** and **robust** solution on **large scale** problems (~$10^5$ variables)
- **Four** active-set and interior-point algorithms for continuous optimization
- **MINLP algorithms** and **complementarity constraints** for discrete optimization
- **Many extra features** based on **customer feedbacks** or project requirements
- **Parallel multi-start method** for **global optimization**.
- **Easy to use and well documented**: [Online documentation](#)
Widely used in academia…
- **US Top Universities**: Berkeley, Columbia, Harvard, MIT, Stanford…
- **Worldwide Top Universities**: ETH Zürich, LSE, NUS (Singapore), Melbourne…

… and industry
- Economic consulting firms
- Financial institutions
- Mechanical engineering companies
- Oil & Gas companies
- Regulator & Policy maker
- Software developers
  - Used as a third-party optimization engine

More than 400 institutions in over 40 countries rely on Artelys Knitro
ARTELYS KNITRO: OVERVIEW

**Interfaces**

<table>
<thead>
<tr>
<th>Programming languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>C/C++</td>
</tr>
<tr>
<td>Python</td>
</tr>
<tr>
<td>Fortran</td>
</tr>
<tr>
<td>C#</td>
</tr>
<tr>
<td>Java</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Modeling languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMPL</td>
</tr>
<tr>
<td>GAMS</td>
</tr>
<tr>
<td>AIMMS</td>
</tr>
<tr>
<td>MATLAB</td>
</tr>
<tr>
<td>Microsoft Excel</td>
</tr>
</tbody>
</table>

**Supported platforms**

- **Windows**
  - Windows 32-bit, 64-bit
- **Linux**
  - Linux 64-bit
- **macOS**
  - macOS 64-bit

June 29, 2018
www.artelys.com
ARTELYS KNITRO: OVERVIEW

## Interfaces

### Programming languages
- C/C++
- Python
- Fortran
- Java
- C#
- Julia

### Modeling languages
- AMPL
- GAMS
- AIMMS
- MATLAB
- Microsoft Excel
- MPL

## Supported platforms

- **Windows**
  - Windows 32-bit, 64-bit
- **Linux**
  - Linux 64-bit
- **macOS**
  - macOS 64-bit
• **Variables** (x and y):
  – Continuous or discrete
  – Bounded or unbounded

• **Objective** (f) and **constraints** (g and h):
  – Linear or nonlinear
  – Smoothness: required, but may still work without it
  – Convexity: better but not required, local optimization or global optimization with multistart

• **Complementarity** constraints:
  – \( 0 \leq x_k \perp x_l \geq 0 \) is equivalent to: \( x_k \geq 0 \) and \( x_l \geq 0 \) and \( \{x_k = 0 \text{ or } x_l = 0\} \)
  – Applications: strategic bidding, economic models, equilibrium constraints, disjunctions

\[
\begin{align*}
\min_{x \in \mathbb{R}^n, y \in \mathbb{Z}^m} & \quad f(x, y) \\
\text{s.t.} & \quad g_i(x, y) \geq 0 \quad i \in I \\
& \quad h_j(x, y) = 0 \quad j \in J \\
& \quad 0 \leq x_k \perp x_l \geq 0 \quad (k, l) \in C \subset [1, n]^2 \\
& \quad l_x \leq x \leq u_x \\
& \quad l_y \leq y \leq u_y
\end{align*}
\]
Artelys Knitro 11.0 new features:

- **New SOCP Algorithm**
  - Detect conic constraints from quadratic structures
  - Designed for general nonlinear problems with SOC constraints

- **New C API**
  - Easier to use
  - Allows passing problem structure (eg linear, quadratic, conic constraints) with dedicated API and without providing Hessian

- **Preconditioning for all classes of problems**
  - Preconditioning can now be used for problems with equality and inequality constraints

- **New parallel linear solvers**
  - HSL MA86 (non-deterministic) and MA97 (deterministic)
  - Speedups on large scale problem with shared memory parallelism

- **Performance improvements**
  - Very large speedup on QCQPs (including nonlinear QCQP)
  - Speedups on general convex problems
  - Speedups on MINLP algorithms
Artelys Knitro 11.0 new features:

1. New SOCP Algorithm
   - Detect conic constraints from quadratic structures
   - Designed for general nonlinear problems with SOC constraints

2. New C API
   - Easier to use
   - Allows passing problem structure (e.g., linear, quadratic, conic constraints) with dedicated API and without providing Hessian

3. Preconditioning for all classes of problems
   - Preconditioning can now be used for problems with equality and inequality constraints

4. New parallel linear solvers
   - HSL MA86 (non-deterministic) and MA97 (deterministic)
   - Speedups on large scale problem with shared memory parallelism

5. Performance improvements
   - Very large speedup on QCQPs (including nonlinear QCQP)
   - Speedups on general convex problems
   - Speedups on MINLP algorithms
Artelys Knitro 11.0 new features:

- New SOCP Algorithm
  - Detect conic constraints from quadratic structures
  - Designed for **general nonlinear problems with SOC constraints**

- New C API
  - Easier to use
  - Allows passing problem structure (e.g., linear, quadratic, **conic constraints**) with dedicated API and without providing Hessian

- Preconditioning for all classes of problems
  - Preconditioning can now be used for problems with **equality** and inequality constraints

- New parallel linear solvers
  - HSL MA86 (non-deterministic) and MA97 (deterministic)
  - Speedups on large scale problem with shared memory parallelism

- Performance improvements
  - Very large speedup on QCQPs (including nonlinear QCQP)
  - Speedups on general convex problems
  - Speedups on MINLP algorithms
Artelys Knitro 11.0 new features:

- New SOCP Algorithm
  - Detect conic constraints from quadratic structures
  - Designed for general nonlinear problems with SOC constraints
- New C API
  - Easier to use
  - Allows passing problem structure (eg linear, quadratic, conic constraints) with dedicated API and without providing Hessian
- Preconditioning for all classes of problems
  - Preconditioning can now be used for problems with equality and inequality constraints
- New parallel linear solvers
  - HSL MA86 (non-deterministic) and MA97 (deterministic)
  - Speedups on large scale problem with shared memory parallelism
- Performance improvements
  - Very large speedup on QCQPs (including nonlinear QCQP)
  - Speedups on general convex problems
  - Speedups on MINLP algorithms
SOCP
Standard Second Order Cone (SOC) of dimension $k$ is

\[
\left\{ \begin{bmatrix} t \\ u \end{bmatrix} \mid u \in \mathbb{R}^{k-1}, t \in \mathbb{R}^1, \|u\| \leq t \right\}
\]

For $k=1$

\[
\{ t \mid t \in \mathbb{R}^1, 0 \leq t \}
\]

For $k=2$

\[
\left\{ \begin{bmatrix} t \\ u \end{bmatrix} \mid u \in \mathbb{R}^1, t \in \mathbb{R}^1, |u| \leq t \right\}
\]
Set $u = Ax + b$ and $t = c^T x + d$ to create general second order cone constraints of form

$$\|Ax + b\| \leq c^T x + d$$

Second Order Cone Program (SOCP):

$$\begin{align*}
\min_{x} & \quad f^T x \\
\text{s.t.} & \quad \|A_i x + b_i\| \leq c_i^T x + d_i, \quad i=1..m \\
& \quad G^T x + h \leq 0
\end{align*}$$

Convex QP and QCQP (and more) can be converted to SOCP
Applications

- Finance: portfolio optimization with loss risk constraints
- Facility location (e.g. antenna placement in wireless network)
- Robust optimization (under ellipsoid uncertainty)
- Robust least squares
- Grasping force optimization
- FIR filter design
- Truss design

See *Applications of Second-Order Cone Programming*, Lobo, Vandenberghe, Boyd, Lebret
Quadratic constraint

\[ x^T Q x + 2 q^T x + r \leq 0 \]
\[ \| Q^{1/2} x + Q^{-1/2} q \|^2 + r - q^T Q^{-1} q \leq 0 \]
\[ \| Q^{1/2} x + Q^{-1/2} q \| \leq (q^T Q^{-1} q - r)^{1/2} \]

Rotated cone constraint

\[ x^T x \leq yz, \quad y \geq 0, \quad z \geq 0 \]
\[ 4x^T x + y^2 + z^2 \leq 4yz + y^2 + z^2 \]
\[ \sqrt{4x^T x + (y - z)^2} \leq y + z \]
\[ \| 2x \| \leq y + z \]
\[ \| y - z \| \leq y + z \]
Knitro identifies the constraints in form
\[ \sum_{i=1}^{n} a_i x_i^2 \leq a_0 x_0^2, \ x_0 \geq 0 \]
and
\[ \sum_{i=2}^{n} a_i x_i^2 \leq a_0 x_0 x_1, \ x_0, x_1 \geq 0. \]
as second order cone constraints, and internally puts them into the standard form.

It also allows to input constraints in form
\[ \|Ax + b\| \leq cx + d \]
directly via the struct ‘L2norm’.

Currently, it does second order conic constraint identification on the presolved problem.
Knitro conic solver moves beyond SOCP (more general)

\[
\begin{align*}
\min_{x} & \quad f^T x \\
\text{s.t.} & \quad \|A_i x + b_i\| \leq c_i^T x + d_i \\
& \quad G^T x + h \leq 0
\end{align*}
\]

\[
\begin{align*}
\min_{x} & \quad f(x) \\
\text{s.t.} & \quad \|A_i x + b_i\| \leq c_i^T x + d_i \\
& \quad h(x) = 0 \\
& \quad g(x) \leq 0
\end{align*}
\]

- Handle any nonlinear problem with second order cone constraints, including non-convex
- First specialized solver of this kind!
- Extension of existing Knitro Interior/Direct algorithm
It generalizes the operations on the slack variables in the existing Knitro/Direct algorithm using the algebra associated with second order cones.

\[
\begin{align*}
\min \ f(x) \\
g(x) &\leq 0 \\
H(x) &\in K.
\end{align*}
\]

\[
H(x) := Mx + v = \begin{pmatrix} c_1 \\ A_1 \\ \vdots \\ c_t \\ A_t \end{pmatrix} x + \begin{pmatrix} d_1 \\ b_1 \\ \vdots \\ d_t \\ b_t \end{pmatrix}
\]

\[
\nabla f(x) + A^T \lambda - M^T w = 0
\]

\[
\begin{array}{l}
g(x) \leq 0, \lambda \geq 0 \\
H(x) \in K, w \in K \\
g(x).\lambda = 0 \\
H(x) \circ w = 0
\end{array}
\]
Can always write the cone constraint as a general NLP constraint:

$$\|u\| \leq t \rightarrow \sqrt{u_1^2 + u_2^2 + \ldots + u_{k-1}^2} \leq t$$

This does not work well in general; constraint is non-differentiable as $$\|u\| \to 0$$

- It is not uncommon that $$\|u\| \to 0$$ at the optimal solution

Can square the constraint, but then it is non-convex and degenerate at the solution if $$\|u\| \to 0$$

Can try to smooth or relax/perturb these constraints to avoid these issues, e.g.

$$\sqrt{u_1^2 + u_2^2 + \ldots + u_{k-1}^2 + \epsilon^2} \leq t$$

- This works better sometimes but is still not robust or nearly as effective as dealing with them directly
Consider the simple example:

\[
\begin{align*}
\min_x & \quad 0.5x_1 + x_2 \\
\text{s.t.} & \quad |x_1| \leq x_2
\end{align*}
\]

- The optimal solution is at (0,0)
- NLP form without conic detection can’t get dual feasible
- QCQP form (non-convex):
  \[
  \begin{align*}
  \min_x & \quad 0.5x_1 + x_2 \\
  \text{s.t.} & \quad x_1^2 \leq x_2^2, \ x_2 \geq 0
  \end{align*}
  \]
  solves in 50 iterations
- Conic formulation solves in 4 iterations

In fact, development of the conic extension started after having troubles with such a problem
- Steiner_model / Steiner_model_100 on the next slide
Compare Knitro (with and without special treatment of cone constraints) and Xpress on small SOCP models (iteration comparison)
Can utilize Knitro branch-and-bound algorithm to solve mixed-integer SOCP

... or more general mixed-integer models with SOC constraints

\[
\begin{align*}
\min_{x,y} & \quad f(x, y) \\
\text{s.t.} & \quad ||A_i x + b_i|| \leq c_i^T x + d_i, \quad i=1..k \\
& \quad h(x, y) = 0 \\
& \quad g(x, y) \leq 0 \\
& \quad y \text{ integer}
\end{align*}
\]
NEW KNITRO 11.0 API
Build optimization model piece-by-piece

- More flexible
- Easier problem modification
- More extendable (to multi-objective, statistical learning models, etc.)

Identify special structures (e.g. linear, quadratic, conic, etc)

- Identify more problem types (QCQP, SOCP, etc)
- Potential for more extensive presolve operations
- Faster (potentially parallel) evaluations of stored structures

Can combine exact and approximate derivatives
\begin{align*}
\text{min } f(x) + (1 - x_0)^2 \\
\text{s.t. } & x_0 x_1 \geq 1 \\
& x_0 + x_1^2 \geq 0, \quad x_0 \leq 0.5
\end{align*}
// Create a new Knitro solver instance.
KN_new(&kc);

// Add variables and constraints and set their bounds
KN_add_vars(kc, 2, NULL);
KN_set_var_upbnd(kc, 0, 0.5);

KN_add_cons(kc, 2, NULL);
double cLoBnds[2] = {1.0, 0.0};
KN_set_con_lobnds_all(kc, cLoBnds);

\[
\begin{align*}
\text{min } f(x) + (1 - x_0)^2 \\
\text{s.t. } & x_0 x_1 \geq 1 \\
& x_0 + x_1^2 \geq 0, \\& x_0 \leq 0.5
\end{align*}
\]
\[ \min f(x) + (1 - x_0)^2 \]

\[ \text{s.t. } x_0 x_1 \geq 1 \]

\[ x_0 + x_1^2 \geq 0, \quad x_0 \leq 0.5 \]

// Load quadratic structure x0*x1 for the first constraint
indexVar1 = 0; indexVar2 = 1; coef = 1.0;
KN_add_con_quadratic_struct_one (kc, 1, 0,
    &indexVar1, &indexVar2, &coef);
\[
\min f(x) + (1 - x_0)^2
\]
\[
\text{s.t. } x_0 x_1 \geq 1
\]
\[
x_0 + x_1^2 \geq 0, \quad x_0 \leq 0.5
\]

// Add linear term x0 in the second constraint
indexVar = 0; coef = 1.0;
KN_add_con_linear_struct_one (kc, 1, 1,
    &indexVar, &coef);

// Add quadratic term x1^2 in the second constraint
indexVar1 = 1; indexVar2 = 1; coef = 1.0;
KN_add_con_quadratic_struct_one (kc, 1, 1,
    &indexVar1, &indexVar2, &coef);
\[
\min f(x) + (1 - x_0)^2 \\
\text{s.t.} \quad x_0x_1 \geq 1 \\
\quad x_0 + x_1^2 \geq 0, \quad x_0 \leq 0.5
\]

// Pointer to structure holding information for callback
CB_context  *cb;

// Add a callback function "callbackEvalF" to evaluate the nonlinear
// (non-quadratic) part of the objective
KN_add_eval_callback (kc, KNTRUE, 0, NULL, f(x), &cb);

// Add the constant, linear, and quadratic terms in the objective.
KN_add_obj_constant(kc, 1.0);
indexVar = 0; coef = -2.0;
KN_add_obj_linear_struct(kc, 1, &indexVar, &coef);
indexVar1 = 1; indexVar2 = 1; coef = 1.0;
KN_add_obj_quadratic_struct(kc, 1, &indevVar1, &indexVar2, &coef);
\[
\begin{align*}
\min & \quad f(x) + (1 - x_0)^2 \\
\text{s.t.} & \quad x_0x_1 \geq 1 \\
& \quad x_0 + x_1^2 \geq 0, \quad x_0 \leq 0.5
\end{align*}
\]

// Set the non-default SQP algorithm
KN_set_int_param(kc, KN_PARAM_ALGORITHM, KN_ALG_ACT_SQP);

// Solve the problem.
KN_solve(kc);

// An example of obtaining solution information.
KN_get_solution(kc, &nStatus, &objSol, x, lambda);

// Delete the Knitro solver instance.
KN_free(&kc);
Comparing old API and new API on some large QCQP models

<table>
<thead>
<tr>
<th>Problem</th>
<th>#nnzJ</th>
<th>#nnzH</th>
<th>Old API (solve time)</th>
<th>New API (solve time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>qcqp1000-1nc</td>
<td>5,591</td>
<td>83,872</td>
<td>33.27</td>
<td>27.41</td>
</tr>
<tr>
<td>qcqp1000-2c</td>
<td>63,139</td>
<td>142,386</td>
<td>20.19</td>
<td>9.20</td>
</tr>
<tr>
<td>qcqp1000-2nc</td>
<td>63,139</td>
<td>131,114</td>
<td>17.71</td>
<td>8.38</td>
</tr>
<tr>
<td>qcqp1500-1c</td>
<td>180,041</td>
<td>438,989</td>
<td>1322.30</td>
<td>393.09</td>
</tr>
<tr>
<td>qcqp1500-1nc</td>
<td>180,041</td>
<td>409,820</td>
<td>230.93</td>
<td>330.52</td>
</tr>
<tr>
<td>qcqp500-3c</td>
<td>5,685</td>
<td>125,086</td>
<td>16.30</td>
<td>0.71</td>
</tr>
<tr>
<td>qcqp500-3nc</td>
<td>5,686</td>
<td>125,086</td>
<td>17.79</td>
<td>0.72</td>
</tr>
<tr>
<td>qcqp750-2c</td>
<td>10,792</td>
<td>281,514</td>
<td>56.37</td>
<td>2.21</td>
</tr>
<tr>
<td>qcqp750-2nc</td>
<td>10,792</td>
<td>281,514</td>
<td>55.37</td>
<td>2.25</td>
</tr>
</tbody>
</table>
Fallback step in projected conjugate gradient (PCG) with Knitro’s interior point

Compute search direction by solving a linear system

Bad inertia (no descent)  Too small steps in line-search

Projected conjugate gradient

\[
\min_{z} \frac{1}{2} z^T (\nabla^2 L + A_I^T S^{-1} A_I)z + r^T z \\
\text{s.t.} \quad A_E z = 0 \\
\quad z_L \leq z \leq z_U
\]
Fallback step in projected conjugate gradient (PCG) with Knitro’s interior point

Compute search direction by solving a linear system

Bad inertia (no descent)  Too small steps in line-search

Projected conjugate gradient

\[
\begin{align*}
\text{minimize}_z & \quad \frac{1}{2} z^T (\nabla^2 L + A_I^T S^{-1} A_I) z + r^T z \\
\text{s.t.} & \quad A_E z = 0 \\
& \quad z_L \leq z \leq z_U
\end{align*}
\]

Bottleneck: very sensitive to ill-conditioning
Fallback step in projected conjugate gradient (PCG) with Knitro’s interior point

Compute search direction by solving a linear system

- Bad inertia (no descent)
- Too small steps in line-search

Projected conjugate gradient

\[
\text{minimize}_z \frac{1}{2} z^T \left( \nabla^2 L + A_I^T S^{-1} A_I \right) z + r^T z \\
\text{s.t.} \quad A_E z = 0 \\
\quad z_L \leq z \leq z_U
\]

So just use a preconditioner!
CONDITIONING

Good conditioning

Bad conditioning
Too small steps in line-
search

Bad conditioning

Good conditioning

Preconditioner

$y = Mx$
Fallback step in projected conjugate gradient (PCG) with Knitro’s interior point

Compute search direction by solving a linear system

- Bad inertia (no descent)
- Too small steps in line-search

Projected conjugate gradient

\[
\text{minimize}_z \frac{1}{2} z^T (\nabla^2 L + A_I^T S^{-1} A_I) z + r^T z \\
\text{s.t.} \quad A_E z = 0 \\
\quad z_L \leq z \leq z_U
\]

Challenging for preconditioning
Incomplete Choleski factorization \((icfs\ module)\)

\[
\nabla^2_{xx} L + A_I^T S^{-1} \Lambda A_I \approx LL^T
\]

Steps to transform PCG direction so that \(A_E z = 0\)

1) Compute \(\tilde{r} := L^{-1} r\)
2) Form the dense matrix \(B := L^{-1} A_E^T\)
3) Compute \(C := B^T B\)
4) Solve \(Cw = B^T \tilde{r}\)
5) Compute \(\tilde{z} = \tilde{r} - Bw\)
6) Backsolve \(z = L^{-T} \tilde{z}\)

New Knitro options
- \(cg\_precond\) (0 or 1)
- \(cg\_pmem\) (density of the incomplete Cholesky factorization)
Nonlinear programs with inequality constraints only (alg=2, Knitro PCG)

Cumulative number of PCG iterations (alg=2)

- **jnlbrng1**: 36022 iterations (47 with preconditioner)
- **jnlbrng2**: 151306 iterations (38 with preconditioner)
- **jnlbrnga**: 36640 iterations (39 with preconditioner)
- **jnlbrngb**: 71712 iterations (1 with preconditioner)
Nonlinear programs with equality constraints (alg=2, Knitro PCG)

Cumulative number of PCG iterations with alg=2

# eqs  # ineqs
440  880
349  699
265  419
402  201
800  800
208  1176
279  123
632  233
352  210
441 1
316 101
277  40

PERFORMANCE IMPROVEMENTS
Nonlinear programs with equality constraints (alg=2, Knitro PCG)

PERFORMANCE IMPROVEMENTS

CPU time (sec) with alg=2

# eqs  # ineqs

<p>| | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>grow2.nl</td>
<td>629.467</td>
<td>85.861</td>
<td>349</td>
<td>800</td>
<td>208</td>
<td>279</td>
<td>632</td>
<td>352</td>
<td>441</td>
<td>500</td>
<td>1000</td>
</tr>
<tr>
<td>gouldq2.nl</td>
<td>8.410.904</td>
<td>8.410.904</td>
<td>800</td>
<td>1176</td>
<td>123</td>
<td>233</td>
<td>210</td>
<td>101</td>
<td>40</td>
<td>500</td>
<td>2000</td>
</tr>
<tr>
<td>maros.nl</td>
<td>993.186</td>
<td>993.186</td>
<td>265</td>
<td>201</td>
<td>800</td>
<td>800</td>
<td>800</td>
<td>800</td>
<td>800</td>
<td>800</td>
<td>2000</td>
</tr>
<tr>
<td>markowitz2.nl</td>
<td>190.072</td>
<td>190.072</td>
<td>208</td>
<td>279</td>
<td>632</td>
<td>352</td>
<td>441</td>
<td>500</td>
<td>1000</td>
<td>500</td>
<td>2000</td>
</tr>
<tr>
<td>optcdg3.nl</td>
<td>1547.565</td>
<td>1547.565</td>
<td>880</td>
<td>699</td>
<td>419</td>
<td>201</td>
<td>800</td>
<td>800</td>
<td>800</td>
<td>800</td>
<td>2000</td>
</tr>
<tr>
<td>pilot.nl</td>
<td>1913.868</td>
<td>1913.868</td>
<td>880</td>
<td>699</td>
<td>419</td>
<td>201</td>
<td>800</td>
<td>800</td>
<td>800</td>
<td>800</td>
<td>2000</td>
</tr>
<tr>
<td>pilot4.nl</td>
<td>1823.09</td>
<td>1823.09</td>
<td>880</td>
<td>699</td>
<td>419</td>
<td>201</td>
<td>800</td>
<td>800</td>
<td>800</td>
<td>800</td>
<td>2000</td>
</tr>
<tr>
<td>pilotnov.nl</td>
<td>305.075</td>
<td>305.075</td>
<td>880</td>
<td>699</td>
<td>419</td>
<td>201</td>
<td>800</td>
<td>800</td>
<td>800</td>
<td>800</td>
<td>2000</td>
</tr>
<tr>
<td>qscfxm2.nl</td>
<td>269.893</td>
<td>269.893</td>
<td>880</td>
<td>699</td>
<td>419</td>
<td>201</td>
<td>800</td>
<td>800</td>
<td>800</td>
<td>800</td>
<td>2000</td>
</tr>
<tr>
<td>qscfxs.nl</td>
<td>76.201</td>
<td>76.201</td>
<td>1176</td>
<td>123</td>
<td>233</td>
<td>210</td>
<td>1</td>
<td>101</td>
<td>40</td>
<td>500</td>
<td>2000</td>
</tr>
<tr>
<td>qship12s.nl</td>
<td>322.7</td>
<td>322.7</td>
<td>316</td>
<td>441</td>
<td>101</td>
<td>101</td>
<td>1</td>
<td>1</td>
<td>40</td>
<td>500</td>
<td>2000</td>
</tr>
<tr>
<td>qship04l.nl</td>
<td>2.168</td>
<td>2.168</td>
<td>277</td>
<td>277</td>
<td>277</td>
<td>277</td>
<td>277</td>
<td>277</td>
<td>277</td>
<td>277</td>
<td>277</td>
</tr>
</tbody>
</table>
Nonlinear programs with equality constraints (alg=2, Knitro PCG)

CPU time (sec) with alg=2

<table>
<thead>
<tr>
<th>Problem</th>
<th># eqs</th>
<th># ineqs</th>
<th>No preconditioner</th>
<th>With preconditioner</th>
</tr>
</thead>
<tbody>
<tr>
<td>grow22.nl</td>
<td>440</td>
<td>880</td>
<td>629.467</td>
<td>178.031</td>
</tr>
<tr>
<td>gouldqp2.nl</td>
<td>349</td>
<td>699</td>
<td>8.410,904</td>
<td>500</td>
</tr>
<tr>
<td>maros.nl</td>
<td>265</td>
<td>419</td>
<td>190.711</td>
<td>199.072</td>
</tr>
<tr>
<td>markowitz2.nl</td>
<td>402</td>
<td>201</td>
<td>1547.565</td>
<td>1582.309</td>
</tr>
<tr>
<td>optdeq3.nl</td>
<td>800</td>
<td>800</td>
<td>1913.868</td>
<td>1937.304</td>
</tr>
<tr>
<td>pilot.nl</td>
<td>208</td>
<td>1176</td>
<td>1913.868</td>
<td>1937.304</td>
</tr>
<tr>
<td>pilot4.nl</td>
<td>279</td>
<td>123</td>
<td>2851.39</td>
<td>322.7</td>
</tr>
<tr>
<td>pilotnov.nl</td>
<td>632</td>
<td>233</td>
<td>2851.39</td>
<td>322.7</td>
</tr>
<tr>
<td>qscfxm2.nl</td>
<td>352</td>
<td>210</td>
<td>2851.39</td>
<td>322.7</td>
</tr>
<tr>
<td>qseba.nl</td>
<td>441</td>
<td>1</td>
<td>2851.39</td>
<td>322.7</td>
</tr>
<tr>
<td>qship12s.nl</td>
<td>316</td>
<td>101</td>
<td>2851.39</td>
<td>322.7</td>
</tr>
<tr>
<td>qship04l.nl</td>
<td>277</td>
<td>40</td>
<td>2851.39</td>
<td>322.7</td>
</tr>
</tbody>
</table>

Bad trade-off
LINEAR SOLVERS
Knitro algorithms need to solve sparse symmetric indefinite linear systems, $Ax=b$, where $A$ and $b$ depends on the iteration

- Although the factorization is unique, each solver uses a different algorithm to compute it (multifrontal/supernodal; pivoting; etc.)
- The choice of the linear solver can change the number of iterations taken to solve a problem, and sometimes even the return status
- There is no linear solver best for all problems

Artelys Knitro 11.0 allows the use of two more parallel linear solvers

- HSL MA27 sequential
- HSL MA57 sequential
- MKL PARDISO parallel
- HSL MA86 parallel
- HSL MA97 parallel; bit-compatible (always give the same answer)
MA57 performs well for small and medium size problems

...but it might have no chance in the large scale

Problem MSK_STEP3

Number of variables: 55,163
Number of constraints: 108,911
Number of nonzeros in Jacobian: 54,330,672
Number of nonzeros in Hessian: 3,361,333

MA57: out of memory

<table>
<thead>
<tr>
<th></th>
<th>4 threads</th>
<th>8 threads</th>
<th>16 threads</th>
<th>30 threads</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MKLPARDISO</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># of iterations</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td>Total program time (secs)</td>
<td>4704.27100 (14239.626 CPU time)</td>
<td>3646.25830 (18465.613 CPU time)</td>
<td>3158.46362 (27331.607 CPU time)</td>
<td>3156.07690 (49937.773 CPU time)</td>
</tr>
<tr>
<td>KKT Factorization time/count</td>
<td>3168.19702 / 96</td>
<td>2104.03540 / 96</td>
<td>1592.80664 / 96</td>
<td>1589.04285 / 96</td>
</tr>
<tr>
<td><strong>MA86</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># of iterations</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td>Total program time (secs)</td>
<td>3363.03809 (11535.405 CPU time)</td>
<td>2370.03833 (14391.964 CPU time)</td>
<td>2111.79712 (23614.680 CPU time)</td>
<td>2265.83179 (47518.672 CPU time)</td>
</tr>
<tr>
<td>KKT Factorization time/count</td>
<td>2689.55811 / 90</td>
<td>1726.85120 / 90</td>
<td>1461.42456 / 90</td>
<td>1609.26416 / 90</td>
</tr>
</tbody>
</table>

June 27, 2018
Your turn!
Try it and let us know what you think…
BENCHMARK
Old benchmark overview:
- 1363 instances
- Classified along 40 labels
- Through text lists format: not user-friendly
  - to add tests
  - to add labels, modify them, ...

Current status:
- 5000+ instances, 50+ labels
- Categorical labels (small/medium/large)
  ; problem features (MINLP/QCQP/...)
- Standard benchmarks included
  - Mittelmann
  - GlobalLib-GAMS, MINLPLib2
  - Pglib-opf (for OPF)
- Database format (Excel..)
New tests integration process:

- Daily continuous integration
  - Non-regression tests
  - Comparison to a reference run
  - In terms of status, obj, cpu, #iters

- Daily tests report
  - Number of regressions
  - Classified regression table
  - Trends in the improvement of cpu, obj
  - Performance profiles in terms of cpu-time / number of iterations

- Deployment and run on the cluster
  - Use all available resources
  - 1363 instances benchmark is ran in 1h