GAMS/CONOPT3

ARKI Consulting and Development
Bagsvaerdvej 246A, DK-2880 Bagsvaerd, Denmark
Tel: +45 44 49 03 23, Fax: +45 44 49 03 33, Email: info@arki.dk

1. INTRODUCTION

ARKI Consulting & Development A/S is happy to release a test version of CONOPT3, our new version of the old CONOPT2 solver. This note will describe the main new features and possibilities that CONOPT3 offers.

During the initial test period for CONOPT3 we encourage users to report errors and problems with CONOPT3. The best will be to email us a zip-file with the project that created the error or problem with a short note explaining how to run the model and what went wrong to adrud@arki.dk. Please also report the version number. The very first CONOPT3 version will have the following banner

```
    C O N O P T 3   Windows NT/95/98  version 3.00A-010-044
```

that identifies the version as 3.00A. Later versions will be 3.00B, 3.00C, etc. We will try to correct errors and forward you a new version within a few days.

The most important new feature in CONOPT3 is a sequential quadratic programming (SQP) component that uses exact second deritatives to compute better search directions. The SQP component is used during Phase 2 and Phase 4 as an alternative to the existing quasi-Newton method, and CONOPT3 will select this component when statistics for the current behavior of the model indicates that it is useful. The SQP component has two sub-components. CONOPT3 will estimate and update a reduced Hessian in much the same way as CONOPT2 for models with a small number of superbasic variables. CONOPT3 will use a conjugate gradient or scaled conjugate gradient methods to solve the SQPs for models with many superbasic variables or for models with a fairly dense matrix of second derivatives. CONOPT3 can use second derivatives in two forms: as a sparse matrix of second derivatives or as the product of the second derivative matrix with a search direction. The selected method depends on memory availability and performance statistics.

There are two consequences of the new SQP components. Models that used to spend many iterations in Phase 2 or 4 will often solve more quickly. And many models with a large number of superbasic variables, that only could be solved slowly if they could be solved at all, can now be solved faster and more reliably.

The second important feature in CONOPT3 is a new scaling algorithm that seems to work better than the algorithm in CONOPT2. Results have been good enough to make scaling the default option.

Finally, the linear algebra used inside CONOPT3 has been revised and it should now be possible to solve larger models than before.

Although we expect and hop that CONOPT3 will be more efficient than our older solvers for most models we realize that there is no such thing as the best solver. CONOPT2 will still be better for many models and we will continue to discribute both for some time, but new developments will concentrate on CONOPT3. For some time it will therefore be up to the user to decide which solver to use.

2. INSTALLATION

CONOPT3 is distributed as a file named gmsco3a.zip that contains an "incremental installation" to GAMS. Copy the file to your GAMS systems directory and run the installation program, gamsinst. Gamsinst will unpack gmsco3a.zip, ask for selections of default solvers, and update the necessary systems files. Gamsinst will also remove gmsco3a.zip.

Once CONOPT3 has been installed, you can use it as any other NLP solver by specifying "Option NLP=CONOPT3;" before the SOLVE statement in your GAMS file, or by adding "NLP=CONOPT3" to the GAMS command line.

3. NEW ITERATION AND LISTING OUTPUT

On most machines you will by default get a logline on your screen or terminal at regular intervals. The iteration log may look something like this:

```
  Iter Phase Ninf   Infeasibility   RGmax    NSB   Step InItr MX OK
     0   0        8.5269660493E+06 (Input point)
                               Pre-triangular equations:         0
                               Post-triangular equations:       77
     1   0        4.8543741567E+02 (After pre-processing)
     2   0        6.0216212296E+01 (After scaling)

** Feasible solution. Value of objective =   -28477.2352091

  Iter Phase Ninf     Objective     RGmax    NSB   Step InItr MX OK
    11   4        1.3693898216E+03 3.9E+05    48 3.3E-02      F  T
    16   4        1.9946873966E+03 2.8E+04    48 1.0E+00    4 F  T
    21   4        2.0947557037E+03 2.0E+04    51 1.0E+00    3 F  T
    26   4        2.2000738205E+03 1.7E+04    53 1.0E+00    6 F  T
    31   4        2.3593746000E+03 1.3E+02    52 1.0E+00   15 F  T
    36   4        2.4483342343E+03 4.3E+01    53 1.0E+00    7 F  T
    41   4        2.5095845119E+03 1.4E+00    66 1.0E+00   16 F  T
    46   4        2.5119405663E+03 1.1E-03    76 1.0E+00   13 F  T
    50   4        2.5119405663E+03 3.9E-09    76

** Optimal solution. Reduced gradient less than tolerance.
```

Compared to CONOPT2 the main difference is indicated by the change in the header line where SlpIt (SLP Iteration) is replace by InItr (Inner Iteration). If the column InItr has a number it is the number of inner iterations in a sub-algorithm that finds a good search direction that subsequently is used as the basis for a line search. When CONOPT3 is in Phase 1 or 3 the model behaves fairly linearly and the sub-algorithm is a sequential linear programming (SLP) algorithm. When CONOPT is in Phase 2 or 4 the model has significant nonlinear components and the sub-algorithm is the new sequential quadratic programming (SQP) algorithm.

In the example shown above, almost all iterations use the SQP algorithm. The numbers can be interpreted in the same way as for SLP:

- Iter it the iteration number.
- Phase describes the phase of the optimization:
  - 0: an initial cheap Newton process used to find a feasible or almost feasible solution.
  - 1: Infeasible and behaving almost as a linear model
  - 2: Infeasible with significant nonlinear components
  - 3: Feasible and behaving almost as a linear model
  - 4: Feasible with significant nonlinear components
- Objective: The value of the objective function at the end of the iteration. The objective function is computed as the true objective defined by the modeler plus an adjustment for small infeasibilities. The objective function should usually change monotonically, but the adjustment can in some cases be so large and inaccurate that the objective will move in the wrong direction. The documentation for CONOPT2 has some comments about stalling caused by this phenomenon.
- Rgmax is the largest reduced cost at the start of the iteration.
- NSB is the number of superbasic variables (or the dimension of the current search space or the degress of freedom).
- Step is the steplength. When the search direction is determined with SLP or SQP the steplength is the fraction of the step suggested by the SLP or SQP algorithm, and 1.0 means that the full step was taken.
- InItr is the number of iterations in the SLP or SQP algorithm.
- MX is T (true) if the step was limited by bounds on variables and F (false) if the step was determined by nonlinearities.
- OK is T (true) if the linesearch was well behaved and F (false) if CONOPT3 could not move to the local optimum along the search direction because it could not maintain feasibility.

The listing file (*.lst) will have a few extra lines with information about first and second derivatives. The first looks as follows:

```
The model has 537 variables and 457 constraints
with 1597 Jacobian elements, 380 of which are nonlinear.
The Hessian of the Lagrangian has 152 elements on the diagonal,
228 elements below the diagonal, and 304 nonlinear variables.
```

The first two lines repeat information given in the GAMS model statistics and the last two lines describe second order information. CONOPT3 uses the matrix of second derivatives (the Hessian) of a linear combination of the objective and the constraints (the Lagrangian). The Hessian is symmetric and the statistics show that it has 152 elements on the diagonal and 228 below for a total of 380 elements in this case. This compares favorably to the number of elements in the matrix of first derivatives (the Jacobian).

For some models you may see the following message instead:

```
Second order sparsety pattern was not generated.
The Hessian of the Lagrangian became too dense because of
equation obj.
You may try to increase Rvhess from its default value of 10.
```

CONOPT3 has interrupted the creation of the matrix of second derivatives because it became too dense. A dense matrix of second derivatives will need more memory than CONOPT3 initially has allocated for it, and it may prevent CONOPT3 from performing the optimization with default memory allocations. In addition, it is likely that a dense Hessian will make the SQP iterations so slow that the potential saving in number of iterations is used up computing and manipulating the Hessian.

GAMS/CONOPT3 can use second derivatives even if the Hessian is not available. A special version of the function evaluation routine can compute the Hessian multiplied by a vector (the so-called directional second derivative) without computing the Hessian itself. This routine is used when the Hessian is not available. The directional second derivative approach will require one function evaluation call per inner SQP iteration instead of one Hessian evaluation per SQP sub-model.

In this particular case, the offending GAMS equation is "obj". You may consider rewriting this equation. Look for nonlinear functions applied to long expressions such as `log(sum(i,x(i));` An expression like this will create a dense Hessian with `card(i)` rows and columns. You should consider introducing an intermediate variable that is equal to the long expression and then apply the nonlinear function to this single variable.

The time spend on the new types of function and derivative evaluations are reported in the listing file in a section like this:

```
CONOPT time Total                            0.734 seconds
  of which: Function evaluations             0.031 =  4.3%
            1st Derivative evaluations       0.020 =  2.7%
            2nd Derivative evaluations       0.113 = 15.4%
            Directional 2nd Derivative       0.016 =  2.1%
```

The function evaluations and 1st derivatives are similar to those reported by CONOPT2. 2nd derivative evaluations are computations of the Hessian of the Lagrangian, and directional 2nd derivative evaluations are computations of the Hessian multiplied by a vector, computed without computing the Hessian itself. The lines for 2nd derivatives will only be present if CONOPT3 has used this type of 2nd derivative.

4. THE NEW CONOPT3 OPTIONS

CONOPT3 follows the tradition of CONOPT2 and has been designed to be self-tuning. Most tolerances and options are dynamic and you should in most cases not need any options. If you do need to change tolerances or options, it can be done in the CONOPT Options file. The name of the Options file is on most systems `"conopt3.opt"` for the new CONOPT3. You must as usual tell the solver that you want to use an options file with the statement "<model>.OPTFILE = 1;" in your GAMS source file before the SOLVE statement or by adding "optfile=1" on the GAMS command line.

The CONOPT Options file consists in its standard form of a number of lines like these:

```
Dohess := true;
Lsscal := true;
Lmscal := 2;
```

The important new or changed options are described in the following table:

`Dohess`	A logical variable that controls the creation of the Hessian (matrix of second derivatives). The default value depends on the model. If the number of equalities is very close to the number of non-fixed variables then the solution is assumed to be in a corner point or in a very low dimensional space where second derivatives are not needed, and `dohess` is initialized to false. Otherwise `dohess` is initialized to true. If `dohess` is false you will not get statistics about the Hessian in the listing file. It takes some time to generate second order information and it uses some space. If CONOPT3 generates this information for your model but it does not use it, i.e. if you see that no time is spend on 2nd derivative evaluations, then you may experiment with `dohess` turned off. If the number of Hessian elements is very large you may also try turning `dohess` off. Note that CONOPT3 still can use directional second derivatives and therefore use its SQP algorithm in the cases where the Hessian is not available.
`Rvhess`	A real number that controls the space available for creation of the Hessian. The maximum number of nonzero elements in the Hessian and in some intermediate terms used to compute it is limited by Rvhess times the number of Jacobian elements (first derivatives). The default value of Rvhess is 10, which means that the Hessian should not be denser than 10 second derivatives per first derivative.
`Lfilos`	Iteration Log frequency for SLP and SQP iterations. A log line is printed to the screen every `lfilos` iterations while using the SLP or SQP mode. The default value depends on the size of the model: it is 1 for large models with more than 2000 constraints or 3000 variables, 5 for medium sized models

	with more than 500 constraints or 1000 variables, and 10 for smaller models.
`Lfnsup`	Maximum Hessian dimension. If the number of superbasics exceeds `lfnsup` CONOPT3 will no longer keep a reduced Hessian. However, it can still use second derivatives in combination with a conjugate gradient algorithm. The default value of `lfnsup` is 500. It is usually not a good idea to increase `lfnsup` much beyond its default value. The time used to manipulate a very large reduced Hessian matrix is often large compared to the potential reduction in the number of iterations. (Note: CONOPT2 and CONOPT3 react very differently to changes in `lfnsup`.)
`Lsscal`	Logical switch for scaling. A logical switch that turns scaling on (with the value `t` or `true`) or off (with the value `f` or `false`). The default value is `t`, i.e. scaling is by default used.
`Lmscal`	Scaling method. The default value is 2 corresponding to a new method where scaling is based on a moving average of the sizes of variables and Jacobian elements. The method used in CONOPT2 can be selected by setting `lmscal` to 0.
`Rtmaxs`	Scale factors larger than `rtmaxs` are rounded down to `rtmaxs`. The default value is 1048576 (=2**20).
`Rtmins`	Scale factors smaller than `rtmins` are rounded up to `rtmins`. The default value is 1/128. All scale factors are powers of 2 to avoid round-off errors from the scaling procedure. `Rtmins` and `Rtmaxs` are on purpose not symmetric around 1. A large `Rtmaxs` allows us to reduce very large variables to a reasonable size. A very small `Rtmins` will result in very small terms being blown up and it could result in feasibility problems if the small terms were really 'noise'.

5. UPDATE LOG

The changes from version 3.00A to version 3.00F are:

Enhancements or changes in functionality:

- CONOPT3 will by default not use the code for first derivatives created by GAMS. Instead it uses an alternative method for computing derivatives directly from the GAMS instructions that define the function values, the so-called backward method. The objective is to save memory by storing a smaller number of GAMS instructions. You can still turn the old method on with the option "Test1f := false;" in the CONOPT3 options file.

- CONOPT will be default use the nonlinear instructions as created by GAMS. For some models the second derivative computations can be very slow with these instructions. If you have one of these models you can now try the option "TestBL := true;" and the instructions will be changed into a format more suitable for second derivative computations.

Bugfixes:

- A Systems Error 153 has been removed.
- Variables with very small reduced costs were in some models set to their upper or lower bounds, far from the optimal value.
- A CNS model could be declared infeasible with variable number 0 at infinity and GAMS could not translate the messages. These models will now stop with a "too small pivot" message.
- DNLP models could give an error saying that the Hessian of the Lagrangian was inconsistent with the nonlinearity pattern of the Jacobian.
- The new factorization routine did not always stop when it ran out of memory and it could result in a crash. Fixed in 3.00F.