1 Origins

Large-scale solvers such as CONOPT [9, 2], LANCELOT [6, 7, 19], MINOS [23, 24], and SNOPT [10, 12] are designed to solve constrained optimization problems in the following fairly general form:

\[
\begin{align*}
\text{NCO} \quad & \text{minimize} & \phi(x) \\
& \text{subject to} & \ell \leq \begin{pmatrix} x \\ Ax \\ c(x) \end{pmatrix} \leq u,
\end{align*}
\]

where \( A \) is a sparse matrix, \( c(x) \) is a vector of smooth nonlinear functions, and the bounds \( \ell \) and \( u \) are likely to exist as shown. (Some could be infinite and some could imply equality constraints: \( \ell_j = u_j \).)

In the early days of optimization, it was a challenge to optimize a nonlinear function \( \phi(x) \) with no constraints. Bounds on the variables \( \ell_j \leq x_j \leq u_j \) could be accommodated with reasonable ease by active-set strategies, but more general constraints tended to be imposed via penalty terms and barrier terms in the objective.

Often it is general enough to deal with nonlinear equality constraints and simple bounds:

\[
\begin{align*}
\text{NCB} \quad & \text{minimize} & \phi(x) \\
& \text{subject to} & \ c(x) = 0, \ \ell \leq x \leq u,
\end{align*}
\]

where \( c(x) \in \mathbb{R}^m \) may include linear functions, and \( x \) includes slack variables to deal with inequality constraints.

Some applications require all iterates to be feasible. CFSQP [4] is one solver that achieves this. The Generalized Reduced Gradient (GRG) approach exemplified by CONOPT is slightly different. Assuming the current point is feasible, it generates the next search direction by assuming \( c(x) \) is linear. It may evaluate \( c(x + \alpha p) \) at infeasible points, but it restores feasibility at each stage before “moving on”.

Otherwise, most methods are content to satisfy \( c(x) = 0 \) only in the limit. The main approaches of interest are penalty methods, augmented Lagrangian methods, and nonlinear interior methods, all of which solve (approximately) sequences of optimization subproblems that are simpler than the original NCO or NCB.

1.1 The Lagrangian

A fundamental function associated with problem NCB (with or without the bound constraints) is the Lagrangian:

\[
L(x, y) = \phi(x) - y^T c(x).
\]

For a given vector \( y \), its derivatives \( \nabla_x L \) and \( \nabla^2_{xx} L \) are

\[
\begin{align*}
g_L(x, y) &= g_0(x) - J(x)^T y, \\
H_L(x, y) &= H_0(x) - \sum_i y_i H_i(x),
\end{align*}
\]

where \( g_0(x) = \nabla \phi(x) \) is the objective gradient, \( H_0(x) = \nabla^2 \phi(x) \) is the objective Hessian, \( J(x) \) is the Jacobian of \( c(x) \) (the \( m \times n \) matrix whose \( i \)th row is the transpose of \( \nabla c_i(x) \)), and \( H_i(x) = \nabla^2 c_i(x) \) is the Hessian of the \( i \)th constraint function. In general, \( y \) will be an approximation to the dual variables associated with the constraints \( c(x) = 0 \).
2 Penalty methods

For this section we assume there are only equality constraints \( c_i(x) = 0, \ i = 1:m \):

\[
\begin{array}{ll}
\text{NEC} & \text{minimize} \quad \phi(x) \\
& \text{subject to} \quad c(x) = 0.
\end{array}
\]

A locally optimal solution \((x^*, y^*)\) satisfies the first-order KKT conditions for NEC:

\[
\begin{align*}
c(x) &= 0, \quad \text{(4)} \\
J(x)^T y &= g_0(x). \quad \text{(5)}
\end{align*}
\]

In real life we are always finding a balance between what is desirable (our objective function) and what is legally achievable (the constraints that prevent infinite profit!). This multiple objective point of view suggests solving the unconstrained problem

\[
\text{PP}(\rho) \quad \text{minimize}_x \quad P(x, \rho) = \phi(x) + \frac{1}{2} \rho \|c(x)\|^2
\]

with some penalty parameter \( \rho > 0 \). We call \( P(x, \rho) \) the quadratic penalty function.

We envisage a trajectory of points \( x_\rho \) that solve \( \text{PP}(\rho) \) for an increasing sequence of \( \rho \) values. We must let \( \rho \) become large to achieve near feasibility, but at least the penalty function is smooth. We may therefore apply Newton or quasi-Newton methods for unconstrained optimization. The derivatives of the penalty function are

\[
\begin{align*}
g(x, \rho) &\equiv \nabla P(x, \rho) = g_L(x, y), \\
H(x, \rho) &\equiv \nabla^2 P(x, \rho) = H_L(x, y) + \rho J(x)^T J(x),
\end{align*}
\]

where \( g_L \) and \( H_L \) are the Lagrangian derivatives (2)–(3) with \( y = -\rho c(x) \) in this case. Note that \( g(x, \rho) = 0 \) at \( x_\rho \) (an unconstrained minimizer of the penalty function). Defining \( y_\rho = -\rho c(x_\rho) \), we see that \((x_\rho, y_\rho)\) is the exact solution of a perturbed form of problem NEC:

\[
\begin{array}{ll}
\text{NEC}_\rho & \text{minimize}_x \quad \phi(x) \\
& \text{subject to} \quad c(x) = c(x_\rho).
\end{array}
\]

Also, if the Jacobian \( J(x^*) \) has full row rank and \( x^* \) is a unique local minimizer for NEC (i.e., the reduced Hessian for NEC is positive definite), we can show that the full Hessian \( H(x, \rho) \) is positive definite at \((x_\rho, y_\rho)\) for sufficiently large \( \rho \). Thus, the penalty function is convex near \( x = x_\rho \) for large \( \rho \), and the minimizer \( x_\rho \) exists.

At first glance, Newton’s method for minimizing \( \text{PP}(\rho) \) would obtain a search direction \( p \) by solving \( H(x, \rho)p = -g(x, \rho) \), i.e., the system

\[
(H_L + \rho J^T J)p = -(g_0 + \rho J^T c), \quad \text{(6)}
\]

where \( H_L \) is defined with \( y = -\rho c \). System (6) is ill-conditioned for large \( \rho \) (assuming \( m < n \)). This is one reason why the early unconstrained optimizers proved unsuccessful with quadratic penalty functions. It was some time before a cure for the ill-conditioning was recognized, but indeed there is one. Following Gould [14] we define \( q = -\rho(c + Jp) \) at the current \( x \). The Newton system is then equivalent to

\[
\begin{pmatrix} H_L & J^T \\ J & -1/\rho I \end{pmatrix} \begin{pmatrix} p \\ -q \end{pmatrix} = - \begin{pmatrix} g_0 \\ c \end{pmatrix}, \quad \text{(7)}
\]

which contains no large numbers and may be preferable for sparsity reasons anyway. If \((x, y)\) is close to a local optimum \((x^*, y^*)\) and if \( \rho \) is large, any ill-conditioning in (7) reflects the sensitivity of \((x^*, y^*)\) to perturbations in the data.

Unfortunately, although \( \rho \) can be computed reliably when \( \rho \) is large, this doesn’t save the quadratic penalty method. When \( c \) is not very small, \( p \) leads away from the linearization of \( c(x) = 0 \) at the current \( x \), and Newton’s method is likely to be too slow.
3 Equality constraints

We continue to study problem NEC, bearing in mind the difficulties encountered with the quadratic penalty method when \( \rho \) becomes very large. Again let \((x^*, y^*)\) be a local minimizer, and assume that the Jacobian \( J(x) = \nabla c(x) \) has full row rank at \( x = x^* \). The first-order optimality conditions that \((x^*, y^*)\) must satisfy are

\[
\begin{align*}
  c(x) &= 0, \quad \text{(8)} \\
  g_0(x) - J(x)^T y &= 0. \quad \text{(9)}
\end{align*}
\]

The Lagrangian associated with problem NEC is \( L(x, y) = \phi(x) - y^T c(x) \) (1). We see that the Lagrangian gradient \( \nabla_x L \) must be zero at \((x^*, y^*)\). The required solution is a stationary point of the Lagrangian. However, in general we cannot find \( x^* \) by minimizing \( L(x, y) \) as a function of \( x \), even if we set \( y = y^* \). The problem \( \min x_1^2 + x_2^2 \) st \( x_1^2 + x_2^2 = 1 \) illustrates what goes wrong no matter what the value of \( y \).

The second-order optimality condition for \((x^*, y^*)\) to be an isolated local minimizer is that the Lagrangian Hessian \( H_L(x^*, y^*) \equiv \nabla^2_{x^*} L(x^*, y^*) \) should be positive definite within the null space of \( J(x^*) \). That is, the Hessian should satisfy \( z^T H_L(x^*, y^*) z > 0 \) for all nonzero vectors \( z \) satisfying \( J(x^*) z = 0 \).

The following result on quadratic forms is relevant.

**Theorem 1 (Debreu [8])** Let \( H \) be an \( n \times n \) symmetric matrix and \( J \) an \( m \times n \) matrix with \( m \leq n \). If \( z^T H z > 0 \) for every nonzero \( z \) satisfying \( J z = 0 \), then for all \( \rho \) sufficiently large, \( H + \rho J^T J \) is positive definite.

This suggests that we should add to the Lagrangian a term whose Hessian is \( \rho J(x)^T J(x) \). We already know such a function: it appears in the quadratic penalty method.

3.1 The augmented Lagrangian

The augmented Lagrangian associated with problem NEC is

\[
L(x, y, \rho) = \phi(x) - y^T c(x) + \frac{1}{2\rho} \| c(x) \|^2, \quad \text{(10)}
\]

It may be thought of as a modification to the Lagrangian closely related to \( L(x, \tilde{y}) \), where \( \tilde{y} \equiv y - \rho c(x) \), or as a shifted quadratic penalty function. For a given \( y \) and \( \rho \), its derivatives \( \nabla_x L \) and \( \nabla^2_{xx} L \) are

\[
\begin{align*}
  g_L(x, y, \rho) &= g_0(x) - J(x)^T \tilde{y}, \quad \text{(11)} \\
  H_L(x, y, \rho) &= H_0(x) - \sum \tilde{y}_i H_i(x) + \rho J(x)^T J(x). \quad \text{(12)}
\end{align*}
\]

The augmented Lagrangian method for solving problem NEC proceeds by choosing \( y \) and \( \rho \) judiciously and then minimizing \( L(x, y, \rho) \) as a function of \( x \). The resulting \( x \) is used to choose a new \( y \) and \( \rho \), and the process repeats. The auxiliary vector \( \tilde{y} \) simplifies the above notation and proves to be useful in its own right.

Note that if \( \rho \) is reasonably large, minimizing \( L \) will tend to make \( \| c(x) \| \) small (as for the penalty method), even if \( y \) is somewhat arbitrary. Also, \( H_L \) will tend to have positive curvature in the right space and a minimizer is likely to exist.

On the other hand, if \( y \) is close to \( y^* \), since minimizing \( L \) makes \( \| g_L \| \) small, we see that \((x, y) \approx (x^*, y^*)\) almost satisfies (9). If it also happens that \( \| c(x) \| \) is small (because \( \rho \) is large enough), \((x, y) \) will almost satisfy (8) as well.

The strategy is to check that \( \| c(x) \| \) is suitably small after each (approximate) minimization of \( L \). If so, \( y \) is updated to \( \tilde{y} = y - \rho c(x) \). If not, \( \rho \) is increased and \( y \) remains the same. Under favorable conditions, \((x, y) \rightarrow (x^*, y^*) \) before \( \rho \) becomes too large.
4 LANCELOT

We now return to the general optimization problem

\[
\begin{align*}
\text{minimize} & \quad \phi(x) \\
\text{subject to} & \quad c(x) = 0, \quad \ell \leq x \leq u,
\end{align*}
\]

where \(c(x) = 0\) includes linear and nonlinear constraints, and \(x\) includes slack variables where necessary to deal with inequalities. The large-scale solver LANCELOT [6, 7, 19] treats NCB by applying an augmented Lagrangian method to a sequence of bound-constrained subproblems of the form

\[
(BC_k) \quad \begin{align*}
\text{minimize} & \quad L(x, y_k, \rho_k) = \phi(x) - y_k^T c(x) + \frac{1}{2} \rho_k \|c(x)\|^2 \\
\text{subject to} & \quad \ell \leq x \leq u.
\end{align*}
\]

A large-scale trust-region method is applied to each BCL subproblem. It takes care of the bound constraints directly. (They are not part of the augmented Lagrangian.)

We call the LANCELOT approach a bound-constrained Lagrangian method, in anticipation of other methods that minimize the augmented Lagrangian subject to additional constraints.

4.1 The BCL algorithm

The LANCELOT BCL method is summarized in Algorithm 1. Each subproblem \((BC_k)\) is solved with a specified optimality tolerance \(\omega_k\), generating an iterate \(x_k^*\) and the associated Lagrangian gradient \(z_k^* \equiv \nabla L(x_k^*, y_k, \rho_k)\). If \(\|c(x_k^*)\|\) is sufficiently small, the iteration is regarded as “successful” and an update to \(y_k\) is computed from \(x_k^*\). Otherwise, \(y_k\) is not altered but \(\rho_k\) is increased.

Key properties are that the subproblems are solved inexactly, the penalty parameter is increased only finitely often, and the multiplier estimates \(y_k\) need not be assumed bounded.

Under certain conditions, all iterations are eventually successful, the \(\rho_k\)'s remain constant, the iterates converge linearly, and the algorithm terminates in a finite number of iterations.

4.2 Solving the BCL subproblems

LANCELOT contains a solver SMIN for minimizing a nonlinear objective function subject to bounds. SBMIN is used to obtain an approximate solution for each BCL subproblem \((BC_k)\). It employs an elaborate active-set strategy (involving generalized Cauchy points and a trust-region constraint) to determine a set of active bounds. It then applies a modified Newton or quasi-Newton method to optimize the augmented Lagrangian objective \(L(x, y, \rho)\) with respect to the moving variables.

In MATLAB notation, most of the sparse-matrix computation is performed in solving linear systems of the form

\[
H_L(m, m) p_M = -g_L(m),
\]

where \(m\) is an index set denoting rows and columns corresponding to moving variables. (Reduced Hessians \(Z^T H Z\) and reduced gradients \(Z^T g\) are easy to form when the constraints are simple bounds!)

The sparse-matrix package MA27 is used to factorize \(H(m, m)\) or \(H(m, m) + E\), where \(E\) is a diagonal modification to make the system positive definite. As \(m\) changes, Schur-complement updates are made in order to re-use the factors.

For very large problems, the conjugate-gradient method (CG) is used to compute \(p_M\), with assistance from a range of of preconditioning options. If \(H_L(m, m)\) is not positive
Algorithm 1: BCL (Bound-Constrained Lagrangian Method).

Input: \( x_0, y_0, z_0 \)

Output: \( x^*, y^*, z^* \)

Set penalty parameter \( \rho_1 > 0 \), scale factor \( \tau > 1 \), and constants \( \alpha, \beta > 0 \) with \( \alpha < 1 \).
Set positive convergence tolerances \( \eta_\ast, \omega_\ast \ll 1 \) and infeasibility tolerance \( \eta_1 > \eta_\ast \).

\( k \leftarrow 0, \text{ converged } \leftarrow \text{false} \)

repeat

\( k \leftarrow k + 1 \)

Choose optimality tolerance \( \omega_k > 0 \) such that \( \lim_{k \to \infty} \omega_k \leq \omega_\ast \).

Find \((x_k^*, z_k^*)\) that solves (BC\(_k\)) to within tolerance \( \omega_k \).

if \( \|c(x_k^*)\| \leq \max(\eta_\ast, \eta_k) \) then

\( y_k \leftarrow y_k - \rho_k c(x_k^*)\)

\( x_k \leftarrow x_k^*, \ y_k \leftarrow y_k^*, \ z_k \leftarrow z_k^* \)

[update solution estimates]

\( \text{then converged } \leftarrow \text{true} \)

[keep \( \rho_k \)]

[decrease \( \eta_k \)]

else

\( \rho_{k+1} \leftarrow \tau \rho_k \)

\( \eta_{k+1} \leftarrow \eta_k / (1 + \beta \rho_{k+1}) \)

[increase \( \rho_k \)]

[may increase or decrease \( \eta_k \)]

end

t until converged

\( x^* \leftarrow x_k, \ y^* \leftarrow y_k, \ z^* \leftarrow z_k \)

definite, PCG may terminate with a direction of infinite descent or a direction of negative curvature.

A feature of LANCELOT and SBMIN is their provision for nonlinear functions that are group partially separable. This facilitates the formation of \( g_L \) and \( H_L \). It is also useful for computing the matrix-vector products \( H_L(M, M)v \) required by PCG.

4.3 Partial augmented Lagrangians

Subproblem (BC\(_k\)) above treats the bound constraints directly. Similar theory applies if other constraints from \( c(x) = 0 \) are imposed directly in the subproblem constraints rather than via the augmented Lagrangian objective.

For example, Conn et al. [5] show how to treat general linear constraints directly as well as bounds. Probably this was inspired by the strangely poor performance of LANCELOT on linear programs. (However, the approach has not been implemented.)

An important application of this approach is to nonlinear network problems with side constraints. Network constraints are well suited to the reduced-gradient method because the null-space operator \( Z \) is extremely sparse and can be formed explicitly. General linear and nonlinear constraints are best included within the augmented Lagrangian objective.

4.4 Further reading

The augmented Lagrangian method for handling equality constraints was originally called the method of multipliers (Hestenes [15], Powell [26]). Powell viewed the augmented Lagrangian as a shifted quadratic penalty function. An important reference for augmented Lagrangian methods is Bertsekas [3]. Nocedal and Wright [25, Ch 7] give an overview.
Algorithm 2: NCL (Nonlinearly Constrained Lagrangian Method).

Input: $x_0, r_0, y_0, z_0$
Output: $x^*, r^*, y^*, z^*$

Set penalty parameter $\rho_1 > 0$, scale factor $\tau > 1$, and constants $\alpha, \beta > 0$ with $\alpha < 1$.
Set positive convergence tolerances $\eta_\omega, \omega_\omega \ll 1$ and infeasibility tolerance $\eta_1 > \eta_\omega$.

$k \leftarrow 0$, converged $\leftarrow$ false

repeat
  $k \leftarrow k + 1$
  Choose optimality tolerance $\omega_k > 0$ such that $\lim_{k \to \infty} \omega_k \leq \omega_\omega$.
  Find $(x_k^*, r_k^*, y_k^*, z_k^*)$ that solves (NC$_k$) to within tolerance $\omega_k$.
  if $\|r_k^*\| \leq \max(\eta_\omega, \eta_k)$ then
    $y_k^* \leftarrow y_k + \rho_k r_k^*,$ $x_k \leftarrow x_k^*$, $y_k \leftarrow y_k^*$, $z_k \leftarrow z_k^*$ [update solution estimates]
    if $(x_k, y_k, z_k)$ solves NCB to within $\omega$, then converged $\leftarrow$ true
    $\rho_{k+1} \leftarrow \rho_k$ [keep $\rho_k$]
    $\eta_{k+1} \leftarrow \eta_k/(1 + \rho_k^{\alpha})$ [decrease $\eta_k$]
  else
    $\rho_{k+1} \leftarrow \tau \rho_k$
    $\eta_{k+1} \leftarrow \eta_k/(1 + \rho_k^{\alpha})$ [increase $\rho_k$]
  [may increase or decrease $\eta_k$]
end
until converged

$x^* \leftarrow x_k$, $r^* \leftarrow r_k$, $y^* \leftarrow y_k$, $z^* \leftarrow z_k$

5 Algorithm NCL

Returning to problem NCB and LANCELOT’s subproblems (BC$_k$), if we introduce $m$ variables $r = -c(x)$, we obtain nonlinearly constrained subproblems of the form

\[
\begin{align*}
\text{(NC$_k$)} & \quad \begin{array}{c}
\text{minimize} \\
L(x, r, y_k, \rho_k) = \phi(x) + y_k^T r + \frac{1}{2} \rho_k \|r\|^2
\end{array} \\
\text{subject to} & \quad c(x) + r = 0, \quad \ell \leq x \leq u.
\end{align*}
\]

Algorithm 1 becomes Algorithm 2, in which the final value of the new variables $r$ should become close to zero: $\|r^*\| \leq \eta_\omega$. Algorithm 2 is an alternative implementation of LANCELOT. Important properties of the subproblems (NC$_k$) follow.

1. If the constraints $c(x) = 0$ in NCB were $c(x) \geq 0$, the constraints in (NC$_k$) would become $c(x) + r \geq 0$. Similarly for a mixture of equalities and inequalities.
2. The subproblems are feasible, like (BC$_k$).
3. The subproblem constraints are linearly independent.
4. Only the objective function of (NC$_k$) changes with $k$.
5. If an active-set solver like MINOS or SNOPT were applied to (NC$_k$), it would be natural for the basic variables to be $r$ throughout. All of $x$ would be superbasic or nonbasic. Warm starts would be straightforward, but there could be many superbasics. For large problems, this would be inefficient with the reduced-gradient implementations of MINOS and SNOPT, but will be acceptable with SNOPT 9, which switches from SQOPT [11] to SQIC [13] to solve QP subproblems with many superbasics.
6. When an interior solver like IPOPT [16] or KNITRO [18] is used for each (NC$_k$) subproblem, the presence of $r$ does not affect the sparsity of the factorizations normally used to compute search directions. Also, warm starts (for $k \geq 2$) have unexpectedly proved to be practical (see below).
5.1 An implementation of algorithm NCL

Algorithm NCL was developed during experimentation with a class of challenging problems arising in the modeling of taxation policy [17]. These have potentially millions of nonlinear inequality constraints $c(x) \geq 0$ in thousands of variables $x$. With $x = (c, y)$, they take the form

$$\text{TAX maximize } \sum_i \lambda_i U^i(c_i, y_i)$$

subject to

$$U^i(c_i, y_i) - U^i(c_j, y_j) \geq 0 \quad \text{for all } i, j$$

$$\lambda^T(y - c) \geq 0$$

$$c, y \geq 0,$$

where $c_i$ and $y_i$ are the consumption and income of taxpayer $i$, and $\lambda$ is a vector of positive weights. The utility functions $U^i$ are each of the form

$$U(c, y) = \frac{(c - \alpha)^{1-1/\gamma}}{1 - 1/\gamma} - \frac{\psi(y/w)^{1/\eta+1}}{1/\eta+1},$$

where $w$ is the wage rate (of dimension $na$) and $\eta, \alpha, \psi, \gamma$ are taxpayer heterogeneities (dimensions $nb, nc, nd, ne$). Problem TAX and algorithm NCL have been implemented in AMPL [1] and results are reported in [20, 21, 22]. Experience so far includes the following.

1. The utility function $U(c, y)$ must be extended to exist when $c \geq \alpha$ and $c < \alpha$.

2. With $na = 5, 9, 11, 17, 21$ and $nb = nc = 3$, $nd = ne = 2$, the largest model with $na = 21$ has $m > 570,000$ inequality constraints $c(x) \geq 0$ and $n > 1500$ variables $(c, y)$. More than $6.6m$ inequality constraints are within $10^{-6}$ of being active at a solution. Thus, the constraints $c(x) \geq 0$ of problem TAX do not satisfy the linear independence constraint qualification (LICQ), and this is why normal solvers have difficulty. Subproblem (NC$_k$)'s regularized constraints $c(x) + r \geq 0$ do satisfy LICQ.

3. LANCELOT was nearly able to solve the smallest model ($na = 5$), but the subproblems (BC$_k$) were increasingly expensive. After 8 hours, LANCELOT was interrupted with $k = 11$, $\rho_k = 10^6$, and objective value correct to about 5 digits.

4. NCL/IPOPT was able to use warm-start options

\begin{verbatim}
warm_start_init_point=yes
mu_init=1e-4
\end{verbatim}

for $k \geq 2$ to solve subproblem (NC$_k$) more efficiently than with default options. Further efficiency was obtained by reducing $\mu_{\text{init}}$ to $1e-4, 1e-5, 1e-6, 1e-7, 1e-8$ for $k = 2, 4, 6, 8, 10$ respectively.

5. NCL/KNITRO was able to use warm-start options

\begin{verbatim}
algorithm=1
bar_directinterval=0
bar_initpt=2
bar_murule=1
bar_initmu=1e-4
bar_slackboundpush=1e-4
\end{verbatim}

for $k \geq 2$ to solve subproblem (NC$_k$) more efficiently than with default options. Further efficiency was obtained by reducing $\text{bar}_{\text{initmu}}$ and $\text{bar}_{\text{slackboundpush}}$ to $1e-4, 1e-5, 1e-6, 1e-7, 1e-8$ for $k = 2, 4, 6, 8, 10$ respectively.

6. Compared with LANCELOT’s 8 hours on the smallest model ($na = 5$), NCL/IPOPT and NCL/KNITRO needed about 2.5 and 1.0 minutes respectively. On the largest model ($na = 21$) they needed about 4.8 and 0.8 hours respectively.

We conclude that the folklore regarding the difficulty of warm-starting interior methods may need to be upgraded, at least in the context of the sequence of subproblems generated by algorithm NCL.
5.2 Numerical results

Several examples of problem TAX in section 5.1 were modeled in AMPL and run with realistic data on a 2.5GHz Mac laptop. Initial values for \((c, y)\) were computed from the problem

\[
\begin{align*}
\text{maximize} & \quad \sum_i \lambda_i U^i(c_i, y_i) \\
\text{subject to} & \quad \lambda^T(y - c) = 0 \\
& \quad c, y \geq 0.1,
\end{align*}
\]

which SNOPT solved easily in 1 or 2 seconds. SNOPT and IPOPT were not able to solve the original problem TAX for any of the problem sizes tried (3D, 4D, 5D with \(na = 12\)). KNITRO was able to solve all cases, but with sharply increasing iterations and time. Algorithm NCL was successful with either IPOPT or KNITRO as subproblem solver and warm starts for both solvers for \(k \geq 2\). NCL/KNITRO was significantly more efficient than NCL/IPOPT. About 10 NCL iterations reduced \(\|r\|_\infty\) below \(10^{-6}\) in all cases. We consider these successful results on difficult practical problems that don’t satisfy LICQ.

\[
\begin{array}{cccccccc}
3D, na = 12, nb = nc = 3 & \quad \text{Obj} = -6.794e+03 \\
\hline
m & n & \text{Itns} & \text{Time} & \text{Itns} & \text{Time} & \text{Itns} & \text{Time} \\
11556 & 216 & 169 & 7 & 9 & 777 & 29 & 9 & 471 & 25 \\
\hline
4D, na = 12, nb = nc = 3, nd = 2 & \quad \text{Obj} = -1.293e+05 \\
\hline
m & n & \text{Itns} & \text{Time} & \text{Itns} & \text{Time} & \text{Itns} & \text{Time} \\
46440 & 432 & 1687 & 371 & 11 & 744 & 393 & 11 & 563 & 135 \\
\hline
5D, na = 12, nb = nc = 3, nd = ne = 2 & \quad \text{Obj} = -1.738e+05 \\
\hline
m & n & \text{Itns} & \text{Time} & \text{Itns} & \text{Time} & \text{Itns} & \text{Time} \\
186193 & 854 & 11966 & 29562 & 10 & 11607 & 28417 & 10 & 2169 & 3303 \\
\end{array}
\]

Exercises

1. Derive Newton’s method for solving the nonlinear equations (8)–(9) for \((x, y)\). Show that the search direction \((p, q)\) satisfies the KKT-type system

\[
\begin{pmatrix}
H_L(x, y) & J^T \\
J & \quad 0
\end{pmatrix}
\begin{pmatrix}
p \\
-\frac{1}{\rho} q
\end{pmatrix}
= - \begin{pmatrix}
g_0 - J^T y \\
- \frac{1}{\rho} c
\end{pmatrix},
\]

where \(H_L(x, y)\) defines the Hessian of the Lagrangian as in (3).

2. Derive Newton’s method for minimizing the augmented Lagrangian (10) wrto \(x\). Show that the search direction \(p\) satisfies the KKT-type system

\[
\begin{pmatrix}
H_L(x, y, 0) & J^T \\
J & - \frac{1}{\rho} I
\end{pmatrix}
\begin{pmatrix}
p \\
-\frac{1}{\rho} \tilde{y}
\end{pmatrix}
= - \begin{pmatrix}
g_0 - J^T y \\
- \frac{1}{\rho} c
\end{pmatrix},
\]

where \(H_L(x, y, \rho)\) and \(\tilde{y} = y - \rho c\) define the Hessian of the augmented Lagrangian as in (12) and \(q = -\rho(c + J p)\).
References