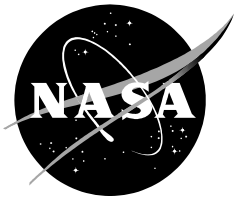
NASA/TM—2016–219104



Use of the NLP10x10 Sequential Quadratic Programming Algorithm to Solve Rotorcraft Hub Loads Minimisation Problems

Jane Anne Leyland
Ames Research Center
Moffett Field, California

May 2016

NASA STI Program ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

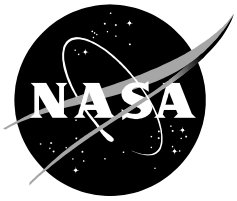
- Access the NASA STI program home page at http://www.sti.nasa.gov

- E-mail your question to help@sti.nasa.gov

- Phone the NASA STI Information Desk at 757-864-9658

- Write to:
  NASA STI Information Desk
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199

NASA/TM—2016–219104



Use of the NLP10x10 Sequential Quadratic Programming Algorithm to Solve Rotorcraft Hub Loads Minimisation Problems

Jane Anne Leyland
Ames Research Center
Moffett Field, California

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, CA 94035-1000

May 2016

Available from:

NASA STI Support Services
Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199
757-864-9658

National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312
webmail@ntis.gov
703-605-6000

This report is also available in electronic form at

http://ntrs.nasa.gov

Table of Contents

List of Tables

List of Appendices

Separate Volumes:

Nomenclature

A_0	Previous duty cycle "actual" Control Amplitude Vector.
A_{0_r}	The *r-th* component of the previous duty cycle "actual" Control Amplitude A_0 – vector.
A_{00}	The optional bias constant vector A_{00} in the equation that defines the previous duty cycle "actual" Control Amplitude A_0 – vector.
A_{00_r}	The *r-th* component of an optional bias vector constant A_{00} in the equation that defines A_{0_r}.
C_1	Input coefficient in the equation that defines the T-Matrix.
C_2	Input coefficient in the equation that defines the A_0 – vector.
C_3	Input coefficient in the equation that defines the ϕ_0 – vector.
C_4	Input coefficient in the equation that defines the ΔZ_A – vector.
CTHHC	Continuous-Time Higher Harmonic Control.
$[CV]$	Control $(N_\theta \times 1)$ θ – vector. $[CV] \equiv \theta$.
$[EC]$	Predicted measurement/end conditions $(N_Z \times 1)$ Z – vector. $[EC] \equiv Z(\theta)$.
FEV	Total number of function evaluations required for convergence to the solution $(N_\theta \times 1)$ θ_{Sol} – vector.
$g[Z(\theta)]$	Scalar performance index function that defines the performance index J. $g[Z(\theta)]$ is a function of the plant output measurement Z – vector, which is a function of the control θ – vector. In general, these functions can be non-linear.
I_Z	Set of all $q \ni Z_q \in Z$.
I_θ	Set of all $p \ni \theta_p \in \theta$.

Nomenclature (continued)

IMSL	International Mathematics and Statistics Library, Inc.—a commercial collection of software libraries of numerical analysis functionality that are implemented in C, Java, C#.NET, and Fortran computer programming languages.
$ISEED1_k$	Input (4 x 1) seed argument for the $k-th$ call to the random number generator function RANQ(•) in the equation that defines $T_{q,p}$. Updated automatically on completion of the generation of the random number.
$ISEED2_k$	Input (4 x 1) seed argument for the $k-th$ call to the random number generator function RANQ(•) in the equation that defines A_{0_r}. Updated automatically on completion of the generation of the random number.
$ISEED3_k$	Input (4 x 1) seed argument for the $k-th$ call to the random number generator function RANQ(•) in the equation that defines ϕ_{0_r}. Updated automatically on completion of the generation of the random number.
$ISEED4_k$	Input (4 x 1) seed argument for the $k-th$ call to the random number generator function RANQ(•) in the equation that defines ΔZ_{A_q}. Updated automatically on completion of the generation of the random number.
ITR	Total number of optimisation iterations required for convergence to the solution $\left(N_\theta \text{ x } 1\right)$ θ_{Sol} – vector.
J	Scalar performance index defined by $g\left[Z(\theta)\right]$. In general, this function can be non-linear. For the problems considered in this research, J is a scalar performance index that is a quadratic function of the plant output measurement vector (i.e., the Z – vector). In this case, Z is a linear function of the control θ – vector, and J is a quadratic function of the control θ – vector.
k	Index number for the input seed argument for calls to the random number generator function RANQ(•) in the equations that synthetically define $T_{q,p}$, A_{0_r}, $\theta_{p_{Initial}}$, ϕ_{0_r}, and ΔZ_{A_q}, where: $k \in \left[1, 2, 3, \bullet, \bullet, \bullet, \bullet, \rightarrow +\infty\right)$.
N_{EQ}	Number of elements (dimension) in the Equality Constraint $\phi(\theta)$ –vector.

Nomenclature (continued)

N_{IEQ}	Number of elements (dimension) in the Inequality Constraint $\psi(\theta)$–vector. $N_{IEQ} = N_{IEQ_1} + N_{IEQ_2}$.
N_{IEQ_1}	Number of elements in the First Inequality Constraint ${}^1\psi(\theta)$–sub-vector.
N_{IEQ_2}	Number of elements in the Second Inequality Constraint ${}^2\psi(\theta)$–sub-vector.
N_Z	Number of elements (dimension) in the predicted measurement Z – vector.
N_θ	Number of elements (dimension) in the control θ – vector.
NLP	Non-Linear Programming algorithm.
NLPQLP	Non-Linear Programming (NLP) algorithm that employs the Sequential Quadratic Programming (SQP) algorithm as its core algorithm.
p	Index number for the control θ – vector elements.
q	Index number for the predicted measurement Z – vector elements.
QPP	Quadratic Programming Problem.
QPS	Quadratic Programming Sub-problem.
r	Index number for the Equality Constraint $\phi(\theta)$–vector elements; also an index number for the control θ – vector elements such that $$r = \left(\frac{p+1}{2}\right) \qquad \forall \quad p \in I_\theta.$$
$\text{RANQ}(\bullet)$	Uniform Random Number Distribution Function which yields a uniformly random real number $\in [0, 1)$.
s	Index number for the Inequality Constraint $\psi(\theta)$–vector elements.
SMART	Smart Material Advanced Rotor Technology.
SQP	Sequential Quadratic Programming algorithm.
T	System or transfer $\left(N_Z \times N_\theta\right)$ matrix either defined by direct input or synthetically determined.

Nomenclature (continued)

$\mathsf{T}_{q,p}$	The $(q, p) - th$ element of the system or transfer $\left(\mathsf{N}_Z \text{ x } \mathsf{N}_\theta\right)$ matrix.
W_Z	Diagonal $\left(\mathsf{N}_Z \text{ x } \mathsf{N}_Z\right)$ weighting matrix in the performance index. The default setting that is the identity matrix; can be redefined by input.
W_θ	Diagonal $\left(\mathsf{N}_\theta \text{ x } \mathsf{N}_\theta\right)$ weighting matrix in the W_θ term of the regulator performance index J. The default setting that is the null matrix; can be redefined by input.
$W_{\dot{\theta}}$	Diagonal $\left(\mathsf{N}_\theta \text{ x } \mathsf{N}_\theta\right)$ weighting matrix in the $W_{\dot{\theta}}$ term of the regulator performance index J. The default setting that is the null matrix; can be redefined by input.
Z	$\equiv Z(\theta)$ and is used when the omission of the explicit dependence on the control θ – vector does not present any confusion, ambiguity, or vagueness.
$Z(\theta)$	Predicted measurement $\left(\mathsf{N}_Z \text{ x } 1\right)$ Z – vector evaluated during the optimisation or regulator process and is a function of the control θ – vector. In general, this function can be non-linear. For the problems analysed in this research, $Z(\theta)$ is a linear function of the control θ – vector.
Z_A	Actual measurement $\left(\mathsf{N}_Z \text{ x } 1\right)$ Z – vector that would normally be evaluated during the previous duty cycle or at a reference epoch time. Z_A is either directly input or synthetically determined.
Z_q	The $q - th$ element of the predicted measurement Z – vector.
ΔZ_A	Random component of the synthetically determined actual measurement Z_A – vector.
ΔZ_{A_q}	Random component of the $q - th$ element of the synthetically determined actual measurement Z_A – vector.
α	Adjustment coefficient $\alpha \in [0, 1]$ used in some feedback control applications to adjust the explicit solution to the regulator problem to enhance control convergence in real time $(1.0\mathsf{D}+00)$.
ε	Input small value constant selected to prevent $\theta_{0_{Initial}}$ from being identically on a bound (i.e., the least upper bound [l.u.b.] or the greatest lower bound [g.l.b.]).

Nomenclature (continued)

θ	Control $\left(\mathrm{N}_{\theta}\ \mathrm{x}\ 1\right)$ θ – vector.
$\dot{\theta}$	Time rate of change of the control $\left(\mathrm{N}_{\theta}\ \mathrm{x}\ 1\right)$ θ – vector.
θ_0	Solution $\left(\mathrm{N}_{\theta}\ \mathrm{x}\ 1\right)$ θ_{Sol} – vector that would normally be evaluated during the previous duty cycle or at a reference epoch time. θ_0 is either directly input or synthetically determined.
θ_{MAX_p}	Least upper bound (l.u.b.) for the $p-th$ element of the control $\left(\mathrm{N}_{\theta}\ \mathrm{x}\ 1\right)$ θ – vector.
θ_{MIN_p}	Greatest lower bound (g.l.b.) for the $p-th$ element of the control $\left(\mathrm{N}_{\theta}\ \mathrm{x}\ 1\right)$ θ – vector.
θ_p	The $p-th$ element of the control $\left(\mathrm{N}_{\theta}\ \mathrm{x}\ 1\right)$ θ – vector.
$\dot{\theta}_p$	The $p-th$ element of the time rate of change of the control $\left(\mathrm{N}_{\theta}\ \mathrm{x}\ 1\right)$ θ – vector.
$\theta_{p_{Initial}}$	The $p-th$ element of the solution $\left(\mathrm{N}_{\theta}\ \mathrm{x}\ 1\right)$ θ_{Sol}– vector that would normally be evaluated during the previous duty cycle or at a reference epoch time. $\theta_{p_{Initial}}$ is synthetically determined.
θ_{Sol}	Solution $\left(\mathrm{N}_{\theta}\ \mathrm{x}\ 1\right)$ θ_{Sol} – vector evaluated during the optimisation or regulator process.
θ_{Sol_p}	The $p-th$ element of the solution control $\left(\mathrm{N}_{\theta}\ \mathrm{x}\ 1\right)$ θ_{Sol} – vector evaluated during the optimisation or regulator process.
$\phi(\theta)$	Equality Constraint $\left(\mathrm{N}_{\mathrm{EQ}}\ \mathrm{x}\ 1\right)$ $\phi(\theta)$ –vector function; in general, can be dependent on the θ – vector and be non-linear.
$\phi_r(\theta)$	The $r-th$ element of the Equality Constraint $\left(\mathrm{N}_{\mathrm{EQ}}\ \mathrm{x}\ 1\right)$ $\phi(\theta)$ –vector function.

Nomenclature (concluded)

ϕ_0 Previous duty cycle "actual" Phase Angle Vector.

ϕ_{0_r} The *r-th* component of the previous duty cycle "actual" Control Phase Angle ϕ_0 – Vector. ϕ_{0_r} is adjusted to be within the principal cycle ([0 deg, 360 deg) or [–180 deg, +180 deg)).

ϕ_{00} Optional bias constant vector ϕ_{00} in the equation that defines the previous duty cycle "actual" Control Phase Angle ϕ_0 – Vector.

ϕ_{00_r} The *r-th* component of an optional bias vector constant ϕ_{00} in the equation that defines ϕ_{0_r}.

$\psi(\theta)$ Complete Inequality Constraint $\left(\left[\mathsf{N}_{\mathsf{IEQ}} = \mathsf{N}_{\mathsf{IEQ}_1} + \mathsf{N}_{\mathsf{IEQ}_2}\right] \times 1\right)$ $\psi(\theta)$–vector function; in general, can be dependent on the θ – vector and be non-linear. $\psi(\theta)$ is comprised of the two sub-vector functions ${}^1\psi(\theta)$ and ${}^2\psi(\theta)$. Specifically:

$$\psi(\theta) = \begin{bmatrix} {}^1\psi(\theta) \\ {}^2\psi(\theta) \end{bmatrix}.$$

${}^1\psi(\theta)$ First Inequality Constraint $\left(\mathsf{N}_{\mathsf{IEQ}_1} \times 1\right)$ ${}^1\psi(\theta)$–sub-vector function with elements of the First Inequality Constraint Form.

${}^2\psi(\theta)$ Second Inequality Constraint $\left(\mathsf{N}_{\mathsf{IEQ}_2} \times 1\right)$ ${}^2\psi(\theta)$–sub-vector function with elements of the Second Inequality Constraint Form.

$\psi_s(\theta)$ The $s-th$ element of the Complete Inequality Constraint $\left(\mathsf{N}_{\mathsf{IEQ}} \times 1\right)$ vector $\psi(\theta)$–function.

Summary

Previous research and experimentation on the use of a non-linear programming constrained optimisation technique to define an optimal control vector for rotorcraft applications indicated that the use of this methodology was feasible and desirable in many cases. In particular, non-linear programming methods that solve a sequence of related quadratic-programming sub-problems were used successfully to solve these problems. Accordingly, a licence for one of the latest versions of Professor Klaus Schittkowski's very successful Sequential Quadratic Programming NLPQLP software was obtained and used to experiment with and analyse typical optimisation problems encountered in various rotorcraft wind tunnel and flight tests. This research resulted in the development of the general NLPQLP Optimisation System that could be used to solve problems encountered in various rotorcraft applications where there is a linear dependence of the measurement vector on the control vector, and where equality and/or inequality constraints might be imposed. This development was accomplished on a mainframe computer not part of actual wind tunnel and/or flight-test experiment, but in a format that was transferable to laptop computers for use in a wind tunnel. Emphasis was directed toward obtaining efficiency, robustness, and speed in computation.

The NLP10x10 System was developed in support of the five-bladed SMART Active Flap Rotor Hub Loads analytical minimisation research. The design and development of the NLP10x10 Optimisation System was tailored to address the problem of how to minimise a performance metric function of measured hub load harmonic angular couple components by optimising the control vector harmonic flap angular couple components subject to constraints on the amplitudes of those components. In addition, to facilitate real-time wind tunnel experimentation, the NLP10x10 System provided the ability to rapidly select/change the particular hub load harmonic angular couple components and/or the particular control vector harmonic angular couple components considered in the optimisation procedure. This capability allows the singling out of particular hub load frequencies and/or particular flap angle frequencies for analysis during testing operations.

The NLP10x10 System was used very successfully for the SMART Active Flap Rotor Hub Loads minimisation problems considered in the study, the results of which were presented at the American Helicopter Society Fifth Decennial Aeromechanics Specialists' Conference in January 2014. Excellent agreement between cases initiated with best-guess starting estimates for the control vector elements and cases initiated with zero-control-vector starting element estimates resulted, indicating the robustness of the NLP10x10 algorithm.

1.0 Introduction

Previous research and experimentation on the use of a non-linear programming (NLP) constrained optimisation technique to define an optimal control vector for rotorcraft applications indicated that the use of this methodology was feasible and desirable in many cases (refs. 1–8). Use of this methodology for rotorcraft applications resulted in better controller performance in many cases compared to results obtained with the widely used solution to the regulator problem. This research, which resulted in the development of the NLP10x10 Optimisation System described herein, was an evolution process and is described in references 1 through 8. It was accomplished on a mainframe computer, not part of actual wind tunnel and/or flight-test experiments, but in a format that was transferable to laptop computers for use in a wind tunnel. The initial research (ref. 1) on the development, design, and use of a non-linear programming method to optimise rotorcraft aeromechanical behaviour indicated that the general non-linear programming method coded as the NCONF/DNCONF subroutine system, which is available in the IMSL MATH/LIBRARY (ref. 9), worked very well solving the required constrained optimisation problems and was significantly superior to methods previously used for solving problems of this type. This IMSL NCONF/DNCONF subroutine system, which dates back to 1989, was used successfully on the research described in references 1 through 5.

This IMSL NCONF/DNCONF subroutine system is based on the work by Schittkowski, Gill et al., Powell, and Stoer (refs. 10–16). This method solves the general non-linear programming problem by solving a sequence of related quadratic programming sub-problems (QPSs). One advantage of this method is that quadratic programming problems (QPPs) can be solved efficiently. A very important property of QPPs is that if the quadratic coefficient matrix in the performance index is positive definite, the problem has a unique solution, which is, of course, the global solution. This means that the sequence of solutions to the QPSs will converge to the global solution of the general problem in the limit, providing that the quadratic coefficient matrix in the performance index remains positive definite in the process.

To conduct a wind tunnel experiment, the optimisation code is installed on a computer within the wind tunnel or on a portable computer such as a laptop that could be brought into the wind tunnel. Research on available suitable optimisation codes revealed that Professor Klaus Schittkowski had revised and updated his Sequential Quadratic Programming (SQP) method, which is part of the IMSL MATH/LIBRARY, via several versions of the code that improved performance and eliminated errors, and that it was available by licence as a stand-alone code (i.e., not part of a library). NASA obtained a licence, and Version 3.1, dated February 2010 (ref. 17 and Appendix A) was obtained and installed on both the Mac Pro desktop computer and the Hewlett-Packard Alpha mainframe computer. The code that was installed on the Mac Pro desktop computer should be transportable to a Mac laptop computer for use in a wind tunnel. Version 3.1 of the NLPQLP System (ref. 17 and Appendix A) was used in the development of the NLPQLP Sequential Programming Algorithm System (refs. 6–8), the development of the NLP10x10 System described herein, and the "*Application of Sequential Quadratic Programming*

to Minimise SMART Active Flap Rotor Hub Loads" research analysis (ref. 18) presented at the Fifth Decennial AHS Aeromechanics Specialists' Conference in San Francisco, California in January 2014.

In 2006, DARPA, Boeing, the U.S. Army, and NASA completed a test of the Boeing Smart Material Advanced Rotor Technology (SMART) bearingless rotor in the USAF National Full-Scale Aerodynamics Complex 40- by 80-Foot Wind Tunnel at NASA Ames Research Center (refs. 19 and 20). During the test, hub loads were successfully minimised using the Continuous-Time Higher Harmonic Control (CTHHC) algorithm (ref. 20). The specific requirements for the minimisation of the SMART Active Flap Rotor Hub Loads provided the motivation for the development of the NLP10x10 System. The design and development of the NLP10x10 Optimisation System described herein was tailored to address the problem of how to minimise a performance metric function of measured hub load harmonic angular couple components by optimising the control vector harmonic flap angular couple components subject to constraints on the amplitudes of those components. In addition, to facilitate real-time wind tunnel experimentation, the NLP10x10 System provided the ability to rapidly select/change the particular hub load harmonic angular couple components and/or the particular control vector harmonic angular couple components included in the optimisation procedure.

2.0 Technical

The details, experimentation, and evaluation of the newly developed NLP10x10 Optimisation System are described in this document. This system has the capability to rapidly select/change the dimensions of the problem in the real-time wind tunnel experimentation environment.

It is assumed that the plant model can be adequately modelled by a linear global plant matrix (referred to as the T-Matrix) that linearly relates a measurement/end conditions vector comprised of the sine and cosine components of the measurement/end conditions angular couples to a control vector comprised of the sine and cosine components of the control angular couples. It is presumed that the sine and cosine components of the measurement/end conditions vector angular couples are observable and controllable by the sine and cosine components of the control vector angular couples. This model is widely used in rotorcraft aeromechanical behaviour studies and is assumed for the problems addressed in this research.

The background formulation, methodology, and solution that lead to the development of the NLP10x10 Optimisation System described herein are presented in section 2.1. The typical, most inclusive SMART Active Flap Rotor Hub Loads Minimisation problem (ref. 18) that was the basis for the NLP10x10 Optimisation System had a maximum of 10 elements for both the control vector and end conditions vector. This was the most inclusive NLP10x10 problem considered in the development of the NLP10x10 Optimisation System. The methodology of problem dimension reduction (i.e., reduction of the dimension of the control vector and/or the end conditions vector) is described in section 2.2.

In an actual wind tunnel or flight test experiment optimisation application, the required input would be the actual test data that perhaps might be reformatted for computer compatibly. To expedite the testing, experimentation, and evaluation of the NLPQLP System, the input data can be synthetically determined if desirable. This synthesis process is described in section 2.3.

Listings of the FORTRAN Code for the NLP10x10 Optimisation System are provided in section 2.4, and a listing of typical Input to the NLP10x10 Optimisation System is presented in section 2.5. Output of selected SMART rotor cases of special interest that were simulated as part of the "*Application of Sequential Quadratic Programming to Minimise SMART Active Flap Rotor Hub Loads*" research analysis (ref. 18) are listed in section 2.6.

2.1 Background

The previous research and experimentation described in references 1 through 8 utilised very general non-linear programming (NLP) methodology (i.e., the NLPQLP Sequential Programming Algorithm System) to solve various typical rotorcraft problems. Use of this very general methodology provided considerable flexibility in previous research. This flexibility was particularly useful in the selection of neural-network parameters in the non-linear neural-network plant model described in references 2 through 5. The most inclusive general non-linear programming problem that the NLPQLP System can address is defined in section 2.1.1. The optimisation problems addressed in the research described herein are special cases of this most inclusive general non-linear programming problem that can be solved with the NLPQLP System (refs. 9 and 17).

The solution process used in the NLPQLP System to solve this most inclusive general non-linear programming problem addresses a sequence of related QPPs, each of which are readily solved. This solution process is described in section 2.1.2.

2.1.1 The General Non-Linear Programming Problem

The optimisation problems that are addressed in the research described herein are special cases of the general non-linear programming (NLP) problem that can be solved with the NLPQLP System (refs. 9 and 17, and Appendix A). The solution process of the NLP problem seeks the optimal control vector (θ_{Sol}) that minimises a performance index (J) subject to constraints on the control vector (θ). The performance index and constraints on the control vector can, in general, be non-linear. The general optimisation problem that can be solved with the NLPQLP System is:

Determine the θ_{Sol} – vector that solves the problem:

$$\underset{\theta_p \in \theta}{Minimise} \quad J = g\left[Z(\theta)\right] \qquad for \quad p \in I_\theta$$

$$where \qquad Z = Z(\theta) = \left[\bullet \ \bullet \ \bullet \ \left\{ Z_q \,\middle|\, q \in I_Z \right\} \ \bullet \ \bullet \ \bullet \right]^T$$

$$and \qquad \theta = \left[\bullet \ \bullet \ \bullet \ \left\{ \theta_p \,\middle|\, p \in I_\theta \right\} \ \bullet \ \bullet \ \bullet \right]^T$$

$$I_Z = \left\{ \bullet \ \bullet \ \bullet \ \left\{ \forall \ q \ \ni \ Z_q \in Z \right\} \ \bullet \ \bullet \ \bullet \right\}$$

$$I_\theta = \left\{ \bullet \ \bullet \ \bullet \ \left\{ \forall \ p \ \ni \ \theta_p \in \theta \right\} \ \bullet \ \bullet \ \bullet \right\}$$

Subject to:

$$\left.\begin{array}{l} \theta_{\mathrm{MIN}_p} \le \theta_p \le \theta_{\mathrm{MAX}_p} \\ \theta_{\mathrm{MIN}_p} \in (-\infty, +\infty) \\ \theta_{\mathrm{MAX}_p} \in (-\infty, +\infty) \end{array}\right\} \left\{ \begin{array}{l} \text{Direct Constraints on the Control } \theta\text{ – vector} \\ \text{Elements } \theta_p \text{ for: } \quad p \in I_\theta \end{array} \right.$$

$$\phi(\theta) = 0 \quad \left\} \begin{array}{l} \text{General Equality Constraint Vector Function} \\ \text{with Dimension } N_{EQ} \end{array} \right.$$

$$\psi(\theta) \ge 0 \quad \left\} \begin{array}{l} \text{General Inequality Constraint Vector Function} \\ \text{with Dimension } N_{IEQ} \end{array} \right.$$

2.1.2 A Solution to the General Non-Linear Programming Problem

Investigation of various methods to solve the general non-linear programming (NLP) problem led to the selection (ref. 1) of the highly successful modern methods of Schittkowski, Powell, Stoer, and Gill et al. (refs. 10–16). These methods solve the general NLP problem by solving a sequence of related quadratic programming sub-problems (QPSs) until either convergence to the desired solution is obtained, or the specified maximum number of iterations (i.e., the maximum number of QPSs to be solved) is reached. This process is described in detail in reference 6.

One important advantage of this technique is that quadratic programming problems (QPPs) can be solved efficiently. A very important property of QPPs is that if the quadratic coefficient matrix in the performance index is positive definite, then the problem has a unique solution, which is, of course, the global solution. This means that the sequence of solutions to the QPSs will converge to the global solution of the general problem in the limit providing the quadratic coefficient matrix in the performance index remains positive definite in the process.

This NLP solution process was coded in FORTRAN and is available as IMSL library routines (specifically, IMSL main driver routines DNCONF and DNCONG described in reference 9). This IMSL NCONF/DNCONF subroutine system, which dates back to 1989, worked quite well in the research described in references 1, 2, 4, and 6 and has proven to be quite robust and efficient in those applications. Professor Klaus Schittkowski has subsequently revised and updated his Sequential Quadratic Programming (SQP) method, which is part of the IMSL MATH/LIBRARY, via several versions of the code that improved performance and eliminated errors. This revised SQP code is available by licence as a stand-alone code (NLPQLP). NASA obtained a licence from Schittkowski and Version 3.1, dated February 2010, was installed on both a Mac Pro desktop computer and a Hewlett-Packard Alpha mainframe computer. To conduct a wind tunnel experiment, the optimisation code would be installed on a computer within the wind tunnel or on a portable computer such as a laptop that could be brought into the wind tunnel. The code that was installed on the Mac Pro should be transportable to a Mac laptop computer for use in the wind tunnel. Version 3.1 of the NLPQLP System (ref. 17 and Appendix A), was used for the study described herein.

The successive QPSs solved with the NLPQLP System are formulated by using a quadratic approximation of the general NLP performance index $J = g[Z(\theta)]$ and linear approximations of the general NLP equality and inequality constraint functions $\phi(\theta)$ and $\psi(\theta)$. These approximations are obtained by simple replacement of the $g[Z(\theta)]$, $\phi(\theta)$, and $\psi(\theta)$ functions with their appropriately truncated matrix Taylor Series expansions. Specifically, the matrix Taylor Series expansion for $J = g[Z(\theta)]$ is truncated after the quadratic term. The quadratic coefficient of the truncated $g[Z(\theta)]$ is the Hessian of $g[Z(\theta)]$ $\left(i.e., \frac{\partial^2 g(\theta)}{\partial \theta^2} \right)$. If the Hessian does not remain positive definite during the iteration process, the algorithm adjusts it so that it is positive definite in order to ensure global optimality of this newly defined QPS. The matrix Taylor Series expansions for constraint functions $\phi(\theta)$ and $\psi(\theta)$ are truncated after their linear terms. The linear coefficients of these constraint functions are the gradients of these functions $\left(i.e., \frac{\partial \phi(\theta)}{\partial \theta} \ and \ \frac{\partial \psi(\theta)}{\partial \theta} \right)$.

If optimality, as measured by satisfying the Kuhn–Tucker (KT) optimality criterion at the completion of an iteration step, and the specified maximum number of iterations have not been reached, the Hessian and constraint gradients are updated (refs. 9, 17, and 18, and Appendix A) and a new QPS is defined. A new iteration is then attempted starting with the last value of θ and the newly updated Hessian and constraint gradients. This process is continued until either convergence is obtained or an error termination occurs (e.g., exceeds the specified maximum allowed number of iterations).

2.2 NLP10x10 Problems

The most general SMART Active Flap Rotor Hub Loads Minimisation problem considered in this research, which was the motivation for the development of the NLP10x10 Optimisation System, is defined in section 2.2.1. In an actual wind tunnel or flight test experiment, it is desirable to be able to select the control elements to be optimised and the measurement elements to be included in the performance index that is minimised. This selection ability can result in fewer than the maximum number of control elements and measurement elements that are considered for a specific problem. In this case, a method to reduce the dimension of the problem is required. An example of SMART rotor model dimension reduction is also presented in section 2.2.1.

The most inclusive NLP10x10 problem that is the basis for the NLP10x10 Optimisation System is presented in section 2.2.2, and the general methodology to reduce its dimensions is described in section 2.2.3.

2.2.1 The SMART Active Flap Rotor Hub Loads Minimisation Problem

The motivation for the development of the NLP10x10 Optimisation System is the SMART Active Flap Rotor Hub Loads Minimisation problem described in reference 18. The SMART rotor is a five-bladed rotor with flaps on the trailing edge of each blade that can be used to reduce the rotor hub loads. The goal of this problem is to minimise a performance measure/index defined by the sum of the squares of rotor hub loads at a selected frequency. The five-per-revolution (5P) frequency hub loads were selected for the analysis described in reference 18. This process assumes that these loads (i.e., the sine and cosine components of the hub loads) are observable and controllable by flap deflections (i.e., the sine and cosine components of the flap deflections) at frequencies of 2P, 3P, 4P, 5P, and 6P. Constraints on the individual and sum of flap deflections can be imposed as required. A linear global plant model (i.e., a T-Matrix) that linearly relates the measurement vector $[Z]$ (i.e., the sine and cosine components of the hub loads) to the control vector $[\theta]$ (i.e., the sine and cosine components of the flap deflections) was assumed. This model is widely used in rotorcraft aeromechanical behaviour studies and was assumed for the problems that were solved as part of this research.

The most general SMART Active Flap Rotor Hub Loads Minimisation problem addressed in this research and treated by the NLP10x10 System assumes that the General End Conditions vector $[Z]$ is a (10 x 1) vector composed of the sine and cosine components of the axial, side, and normal 5P hub shear forces (FXS, FXC, FYS, FYC, FZS, and FZC), and the roll and pitch 5P hub moments (MXS, MXC, MYS, and MYC); and that the General Control vector $[\theta]$ is a (10 x 1) vector composed of the sine and cosine components of the 2P, 3P, 4P, 5P, and 6P flap deflection angles (D2S, D2C, D3S, D3C, D4S, D4C, D5S, D5C, D6S, and D6C). The corresponding T-Matrix is a (10 x 10) matrix. This general model relates the General End Conditions vector $[Z]$ to the General Control vector $[\theta]$ according to:

$$Z = Z(\theta) = Z_A + \mathsf{T}(\theta - \theta_0)$$

where $[Z_A]$ is the actual $[Z]$ vector measured during a previous duty cycle and $[\theta_0]$ is the solution $[\theta]$ vector in that same previous duty cycle.

The General End Conditions $[Z]$ and General Control $[\theta]$ vectors have the forms:

$$[Z] = \begin{bmatrix} Z_1 \\ Z_2 \\ Z_3 \\ Z_4 \\ Z_5 \\ Z_6 \\ Z_7 \\ Z_8 \\ Z_9 \\ Z_{10} \end{bmatrix} = \begin{bmatrix} \text{FXS} \\ \text{FXC} \\ \text{FYS} \\ \text{FYC} \\ \text{FZS} \\ \text{FZC} \\ \text{MXS} \\ \text{MXC} \\ \text{MYS} \\ \text{MYC} \end{bmatrix} \quad \textit{and} \quad [\theta] = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \\ \theta_7 \\ \theta_8 \\ \theta_9 \\ \theta_{10} \end{bmatrix} = \begin{bmatrix} \text{D2S} \\ \text{D2C} \\ \text{D3S} \\ \text{D3C} \\ \text{D4S} \\ \text{D4C} \\ \text{D5S} \\ \text{D5C} \\ \text{D6S} \\ \text{D6C} \end{bmatrix}$$

where Z_{A_i} and Z_i are the *i-th* elements of $[Z_A]$ and $[Z]$ vectors, respectively; and θ_j and θ_{0_j} are the *j-th* elements of the $[\theta]$ and $[\theta_0]$ vectors, respectively.

This most general SMART Active Flap Rotor Hub Loads Minimisation problem addressed in this research minimises a performance index J subject to 20 direct constraints on the values of the elements of the control $[\theta]$ vector, five computed inequality constraints on the angular control couple element amplitudes, and an inequality constraint on the summation of angular control couple element amplitudes.

Specifically, this most general SMART Active Flap Rotor Hub Loads Minimisation problem is:

$$\underset{\theta_p \in \theta}{\textit{Minimise}} \qquad J = Z^{\mathsf{T}} W_Z Z \qquad \textit{for} \quad p \in I_\theta$$

$$\textit{where} \qquad \theta = \left[\bullet \quad \bullet \quad \bullet \quad \left\{ \theta_p \,\middle|\, p \in I_\theta \right\} \quad \bullet \quad \bullet \quad \bullet \right]^{\mathsf{T}}$$

$$I_\theta = \left\{ 1, \quad 2, \quad 3, \quad \bullet \quad \bullet \quad \bullet \quad 10 \right\}$$

$$\textit{and} \qquad W_Z = \textit{is the diagonal } (10 \times 10) \textit{ weighting matrix}$$

and subject to the following Direct and Computed Inequality Constraints.

The Direct Constraints on the control $[\theta]$ vector elements are:

$$\left.\begin{array}{l} \theta_{\mathrm{MIN}_p} \le \theta_p \le \theta_{\mathrm{MAX}_p} \\ \theta_{\mathrm{MIN}_p} \in (-\infty, +\infty) \\ \theta_{\mathrm{MAX}_p} \in (-\infty, +\infty) \end{array}\right\} \left\{\begin{array}{l} \text{Direct Constraints on the Sinusoidal} \\ \text{Control } \theta - \text{vector Elements } \theta_p \text{ for} \\ p \in \{1,\ 2,\ 3,\ \bullet\ \bullet\ \bullet\ 10\} \end{array}\right.$$

The General Computed Inequality Constraint vector $\psi(\theta)$ is comprised of the two sub-vector functions ${}^1\psi(\theta)$ and ${}^2\psi(\theta)$, where:

$$\psi(\theta) = \begin{bmatrix} {}^1\psi(\theta) \\ ---- \\ {}^2\psi(\theta) \end{bmatrix} = \begin{bmatrix} \psi_1(\theta) \\ \psi_2(\theta) \\ \psi_3(\theta) \\ \psi_4(\theta) \\ \psi_5(\theta) \\ ---- \\ \psi_6(\theta) \end{bmatrix}$$

The First Computed Inequality Constraint sub-vector function ${}^1\psi(\theta)$ is:

$$\left.\begin{bmatrix} \psi_1(\theta) = \psi_{1_{\mathrm{MAX}}} - \sqrt{\theta_1^2 + \theta_2^2} \ge 0 \\ \psi_2(\theta) = \psi_{2_{\mathrm{MAX}}} - \sqrt{\theta_3^2 + \theta_4^2} \ge 0 \\ \psi_3(\theta) = \psi_{3_{\mathrm{MAX}}} - \sqrt{\theta_5^2 + \theta_6^2} \ge 0 \\ \psi_4(\theta) = \psi_{4_{\mathrm{MAX}}} - \sqrt{\theta_7^2 + \theta_8^2} \ge 0 \\ \psi_5(\theta) = \psi_{5_{\mathrm{MAX}}} - \sqrt{\theta_9^2 + \theta_{10}^2} \ge 0 \end{bmatrix}\right\} \left\{\begin{array}{l} \text{Inequality Constraints} \\ \text{on the Angular Control} \\ \text{Element Amplitudes} \end{array}\right.$$

and the Second Computed Inequality Constraint sub-vector function ${}^2\psi(\theta)$ is:

$$\left.\left[\psi_6(\theta) = \psi_{6_{\mathrm{MAX}}} - \sum_{r \in I_\psi} \sqrt{\theta_{2r-1}^2 + \theta_{2r}^2} \ge 0\right]\right\} \left\{\begin{array}{l} \text{Inequality Constraint on the} \\ \text{Summation of Angular Control} \\ \text{Couple Element Amplitudes} \end{array}\right.$$

$$for \qquad I_\psi = \{1,\ 2,\ 3,\ 4,\ 5\}$$

Some of the specific problems analysed during the SMART Active Flap Rotor Hub Loads Minimisation study required reduction of the elements of the General End Conditions vector $[Z]$ and/or the General Control Vector $[\theta]$ and/or the General Constraint Vector $\psi(\theta)$ to define the required model for the case to be minimised. For these cases, both the sine and cosine elements of the appropriate angular couple(s) in the General End Conditions vector $[Z]$, and/or the General Control vector $[\theta]$ are eliminated defining a reduced End Conditions vector $[Z_R]$ and/or a reduced Control vector $[\theta_R]$. Those computed constraints that are not required are likewise eliminated defining a reduced Constraint Vector $\psi_R(\theta)$ and reduced sub-vector functions ${}^1\psi_R(\theta)$ and/or ${}^2\psi_R(\theta)$. It is noted that it is not necessary to define any constraints at all. Unconstrained cases are simply defined by not specifying any constraints; hence the constraint vector is identical to the null vector in this case. These eliminations define a reduced model with corresponding reduced T_R-Matrix according to:

$$Z_R = Z_R(\theta) = Z_{A_R} + \mathsf{T}_R\left(\theta_R - \theta_{0_R}\right)$$

$$\psi_R(\theta) = \begin{bmatrix} {}^1\psi_R(\theta) \\ ---- \\ {}^2\psi_R(\theta) \end{bmatrix}$$

where $[Z_{A_R}]$ is the reduced actual $[Z_A]$ vector measured during a previous duty cycle and $[\theta_{0_R}]$ is the reduced solution $[\theta_0]$ vector in that same previous duty cycle. The $[Z_R]$, $[\theta_R]$, $\psi_R(\theta)$, ${}^1\psi_R(\theta)$, and ${}^2\psi_R(\theta)$ vectors are the reduced $[Z]$, $[\theta]$, $\psi(\theta)$, ${}^1\psi(\theta)$, and ${}^2\psi(\theta)$ vectors, respectively. The T_R-Matrix is the corresponding reduced T-Matrix.

An example of model dimension reduction is shown next. In this case, only the 4P flap deflection angle is assumed for the control vectors $[\theta_R]$ and $[\theta_{0_R}]$, and only the X and Y components of the 5P hub shear loads are assumed for the measurement vectors $[Z_R]$ and $[Z_{A_R}]$. Consequently, the reduced control vectors are (2 x 1) and the reduced measurement vectors are (4 x 1). The corresponding reduced T-Matrix (T_R-Matrix) is (4 x 2) and the reduced diagonal weighting W_Z – Matrix $(W_{Z_R}$ – Matrix$)$ is (4 x 4). An amplitude constraint is imposed on the amplitude of the 4P control angular couple. The corresponding reduced constraint vector $[\psi_R(\theta)]$ is simply a scalar (1 x 1).

The corresponding reduced matrix and vector forms are:

$$\left[Z_R\right] = \begin{bmatrix} Z_{R_1} \\ Z_{R_2} \\ Z_{R_3} \\ Z_{R_4} \end{bmatrix} = \begin{bmatrix} \mathrm{FXS} \\ \mathrm{FXC} \\ \mathrm{FYS} \\ \mathrm{FYC} \end{bmatrix}, \qquad \left[\theta_R\right] = \begin{bmatrix} \theta_{R_1} \\ \theta_{R_2} \end{bmatrix} = \begin{bmatrix} \mathrm{D4S} \\ \mathrm{D4C} \end{bmatrix},$$

$$\left[\mathsf{T}_R\right] = \begin{bmatrix} \mathsf{T}_{R_{1,1}} & \mathsf{T}_{R_{1,2}} \\ \mathsf{T}_{R_{2,1}} & \mathsf{T}_{R_{1,2}} \\ \mathsf{T}_{R_{3,1}} & \mathsf{T}_{R_{1,2}} \\ \mathsf{T}_{R_{4,1}} & \mathsf{T}_{R_{1,2}} \end{bmatrix}, \qquad \left[\mathsf{W}_{Z_R}\right] = \begin{bmatrix} \mathsf{W}_{R_1} & 0 & 0 & 0 \\ 0 & \mathsf{W}_{R_2} & 0 & 0 \\ 0 & 0 & \mathsf{W}_{R_3} & 0 \\ 0 & 0 & 0 & \mathsf{W}_{R_4} \end{bmatrix}$$

and

$$\left[\psi_R(\theta)\right] = \begin{bmatrix} {}^1\psi_R(\theta) \\ \hline\hline {}^2\psi_R(\theta) \end{bmatrix} = \left[\begin{array}{c} \psi_{R_1}(\theta) = \psi_{R_{1_{MAX}}} - \sqrt{\theta_{R_1}^2 + \theta_{R_2}^2} \geq 0 \\ \hline\hline \psi_{R_2}(\theta) \equiv 0 \end{array}\right]$$

then

$$\left[\psi_R(\theta)\right] = \left[\begin{array}{c} \psi_{R_1}(\theta) = \psi_{R_{1_{MAX}}} - \sqrt{\mathrm{D4S}^2 + \mathrm{D4C}^2} \geq 0 \\ \hline\hline \psi_{R_2}(\theta) \equiv 0 \end{array}\right]$$

$$\left[\psi_R(\theta)\right] = \left[\sqrt{\mathrm{D4S}^2 + \mathrm{D4C}^2} \leq \psi_{R_{1_{MAX}}}\right]$$

The corresponding reduced performance index J_R is:

$$J_R = W_{Z_1}\,\mathrm{FXS}^2 + W_{Z_2}\,\mathrm{FXC}^2 + W_{Z_3}\,\mathrm{FYS}^2 + W_{Z_4}\,\mathrm{FYC}^2$$

This reduced optimisation problem is then:

$$\underset{\theta_R}{Minimise} \qquad J_R = Z_R^{\mathsf{T}}\, W_{Z_R}\, Z_R$$

$$subject\ to: \qquad \theta_{R_{\mathrm{MIN}_{p=1}}} \le \theta_{R_1} \le \theta_{R_{\mathrm{MAX}_{p=1}}}$$

$$\theta_{R_{\mathrm{MIN}_{p=2}}} \le \theta_{R_2} \le \theta_{R_{\mathrm{MAX}_{p=2}}}$$

$$and:$$

$$\sqrt{\mathrm{D4S}^2 + \mathrm{D4C}^2} \le \psi_{R_{1_{MAX}}}$$

2.2.2 The Most Inclusive NLP10x10 Problem

As mentioned previously, the motivation for the definition of the most inclusive NLP10x10 problem, which is the basis for the NLP10x10 System, is the SMART Active Flap Rotor Hub Loads Minimisation problem described in section 2.2.1. The General Control vector $[\theta]$ is (10 x 1) and is comprised of five angular control couples. It is noted that if a greater number of control elements are required, it is a relatively simple matter to provide the required capacity by changing the dimensions of the appropriate arrays and the limits of the appropriate "DO Loops." Each angular control couple element is comprised of the sine and cosine components of the radius vector of the angular couple rather than its amplitude/magnitude R and its phase angle ϕ. This selection is done to avoid the pathological computation situation of the numerical phase angle discontinuity that exists at the beginning and end of a principal cycle because computational angles are usually limited to a principal cycle (e.g., [0 deg, 360 deg) or [–180 deg, +180 deg)). Specifically:

$$\left[\theta\right] = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \\ \theta_7 \\ \theta_8 \\ \theta_9 \\ \theta_{10} \end{bmatrix} = \left[\theta_C\right] = \begin{bmatrix} \theta_{C_1} \\ \theta_{C_2} \\ \theta_{C_3} \\ \theta_{C_4} \\ \theta_{C_5} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \\ \begin{bmatrix} \theta_3 \\ \theta_4 \end{bmatrix} \\ \begin{bmatrix} \theta_5 \\ \theta_6 \end{bmatrix} \\ \begin{bmatrix} \theta_7 \\ \theta_8 \end{bmatrix} \\ \begin{bmatrix} \theta_9 \\ \theta_{10} \end{bmatrix} \end{bmatrix}$$

where

$$\left[\theta_{C_j}\right] = \begin{bmatrix} \theta_{2j-1} = R_j \bullet Sine(\phi_j) \\ \theta_{2j} = R_j \bullet Cosine(\phi_j) \end{bmatrix} \quad for \quad j \in \{1, 2, 3, 4, 5\}$$

and where

R_j is the magnitude of the radius vector of the *j-th* angular control couple,

ϕ_j is the phase angle of the radius vector of the *j-th* angular control couple, and

j is the index number *j* of the *j-th* angular control couple in the angular couple control vector $\left[\theta_C\right]$.

The General End Conditions vector [Z] is (10 x 1) and is comprised of five angle couples. As in the case of the control vector $\left[\theta\right]$, if a greater number of end condition elements are required, it is a relatively simple matter to provide the required capacity by changing the dimensions of the appropriate arrays and the limits of the appropriate "DO Loops." Likewise, as in the case of the control vector $\left[\theta\right]$, each angular end conditions couple element is comprised of the sine and cosine components of the radius vector of the angular couple, rather than its amplitude/magnitude R and its phase angle ϕ, to avoid the pathological computation situation of the numerical phase angle discontinuity that exists at the beginning and end of a principal cycle. Specifically:

$$\left[Z\right] = \begin{bmatrix} Z_1 \\ Z_2 \\ Z_3 \\ Z_4 \\ Z_5 \\ Z_6 \\ Z_7 \\ Z_8 \\ Z_9 \\ Z_{10} \end{bmatrix} = \left[Z_C\right] = \begin{bmatrix} Z_{C_1} \\ Z_{C_2} \\ Z_{C_3} \\ Z_{C_4} \\ Z_{C_5} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} \\ \begin{bmatrix} Z_3 \\ Z_4 \end{bmatrix} \\ \begin{bmatrix} Z_5 \\ Z_6 \end{bmatrix} \\ \begin{bmatrix} Z_7 \\ Z_8 \end{bmatrix} \\ \begin{bmatrix} Z_9 \\ Z_{10} \end{bmatrix} \end{bmatrix}$$

where

$$\left[Z_{C_i}\right] = \begin{bmatrix} Z_{2i-1} = R_i \bullet Sine\left(\phi_i\right) \\ Z_{2i} = R_i \bullet Cosine\left(\phi_i\right) \end{bmatrix} \quad for \quad i \in \left\{1, 2, 3, 4, 5\right\}$$

and where

R_i is the magnitude of the radius vector of the *i-th* angular end conditions couple,

ϕ_i is the phase angle of the radius vector of the *i-th* angular end conditions couple, and

i is the index number *i* of the *i-th* angular end conditions couple in the angular end conditions couple vector $\left[Z_C\right]$.

The most inclusive (10 x 10) T-Matrix NLP Control problem that is the basic core of the NLP10x10 System is a special control problem considered in references 6, 7, and 8. This problem, referred to herein as the most inclusive NLP10x10 problem, is a slight generalisation of the SMART Active Flap Rotor Hub Loads Minimisation problem described in section 2.2.1. The system or transfer T-Matrix is likewise (10 x 10), and the number of elements (dimension) in both the control θ – vector and predicted measurement *Z* – vector is also ten. These vector elements are grouped as pairs of the sine and cosine components of angular couples as described previously. No Equality Constraints are imposed; hence there is no Equality Constraint $\phi(\theta)$ –vector. The number of elements in the Inequality Constraint $\psi(\theta)$ vector is six, comprised of the five-element First Inequality Constraint ${}^1\psi(\theta)$ –sub-vector and the single-element Second Inequality Constraint ${}^2\psi(\theta)$ –sub-vector. The basic (10 x 10) T- Matrix NLP Control problem with all constraints is defined as:

Determine the θ–vector, θ_{Sol}, which solves the problem:

$$\underset{\theta_p \in \theta}{Minimise} \quad J = Z^{\mathsf{T}} W_Z Z \qquad for \quad p \in I_\theta$$

$$where \qquad Z = Z(\theta) = Z_A + \mathsf{T}(\theta - \theta_0)$$

$$Z = Z(\theta) = \left[\bullet \ \bullet \ \bullet \ \left\{ Z_q \middle| q \in I_Z \right\} \ \bullet \ \bullet \ \bullet \right]^{\mathsf{T}}$$

$$W_Z = \left\{ \begin{bmatrix} \bullet & 0 & 0 & 0 & 0 \\ 0 & W_q & 0 & 0 & 0 \\ 0 & 0 & \bullet & 0 & 0 \\ 0 & 0 & 0 & \bullet & 0 \\ 0 & 0 & 0 & 0 & \bullet \end{bmatrix} \middle| q \in I_Z \right\}$$

$$\theta = \left[\bullet \ \bullet \ \bullet \ \left\{ \theta_p \middle| p \in I_\theta \right\} \ \bullet \ \bullet \ \bullet \right]^{\mathsf{T}}$$

$$I_Z = \{1, \quad 2, \quad 3, \quad \bullet \quad \bullet \quad \bullet \quad 10\}$$

$$I_\theta = \{1, \quad 2, \quad 3, \quad \bullet \quad \bullet \quad \bullet \quad 10\}$$

Subject to the following Direct and Computed Inequality Constraints:

The **Direct Inequality Constraints** on the magnitudes of the control $[\theta]$ vector elements are:

$$\left.\begin{array}{l} \theta_{MIN_p} \le \theta_p \le \theta_{MAX_p} \\ \theta_{MIN_p} \in (-\infty, +\infty) \\ \theta_{MAX_p} \in (-\infty, +\infty) \end{array}\right\} \left\{\begin{array}{l} \text{Direct Constraints on the Sinusoidal} \\ \text{Control } \theta - \text{vector Elements } \theta_p \text{ for} \\ p \in \{1, \quad 2, \quad 3, \quad \bullet \quad \bullet \quad \bullet \quad 10\} \end{array}\right.$$

The Computed Inequality Constraint vector $\psi(\theta)$ is comprised of the two sub-vector functions ${}^{1}\psi(\theta)$ and ${}^{2}\psi(\theta)$, where:

$$\psi(\theta) = \begin{bmatrix} {}^{1}\psi(\theta) \\ ---- \\ {}^{2}\psi(\theta) \end{bmatrix} = \begin{bmatrix} \psi_1(\theta) \\ \psi_2(\theta) \\ \psi_3(\theta) \\ \psi_4(\theta) \\ \psi_5(\theta) \\ ---- \\ \psi_6(\theta) \end{bmatrix}$$

The elements of the first sub-vector ${}^{1}\psi(\theta)$ are the constraints on amplitude of each harmonic couple. These are:

$$\left.\begin{bmatrix} \psi_1(\theta) = \psi_{1_{MAX}} - \sqrt{\theta_1^2 + \theta_2^2} \geq 0 \\ \psi_2(\theta) = \psi_{2_{MAX}} - \sqrt{\theta_3^2 + \theta_4^2} \geq 0 \\ \psi_3(\theta) = \psi_{3_{MAX}} - \sqrt{\theta_5^2 + \theta_6^2} \geq 0 \\ \psi_4(\theta) = \psi_{4_{MAX}} - \sqrt{\theta_7^2 + \theta_8^2} \geq 0 \\ \psi_5(\theta) = \psi_{5_{MAX}} - \sqrt{\theta_9^2 + \theta_{10}^2} \geq 0 \end{bmatrix}\right\} \begin{array}{l} \text{First Inequality Constraint} \\ {}^{1}\psi(\theta) - \text{sub-vector} \end{array}$$

The element of the second sub-vector ${}^{2}\psi(\theta)$ is the constraint on the summation of the amplitudes of each harmonic couple. This constraint is:

$$\left.\begin{bmatrix} \psi_6(\theta) = \psi_{6_{MAX}} - \sum_{r \in I_\psi} \sqrt{\theta_r^2 + \theta_r^2} \geq 0 \end{bmatrix}\right\} \begin{array}{l} \text{Second Inequality Constraint} \\ {}^{2}\psi(\theta) - \text{sub-vector} \end{array}$$

where

$$I_\psi = \{1,\ 2,\ 3,\ 4,\ 5\}$$

2.2.3 Reduction of the Most Inclusive NLP10x10 Problem

The first step in the definition and solution of a specific NLP minimisation problem is to define the most general problem that might be considered—that is, the greatest set of control vector and end conditions harmonic couple elements that could be required. For this particular software system (the NLP10x10 System), the maximum number of end condition harmonic elements $\mathcal{M}_Z$ is five, the maximum number of control vector harmonic elements $\mathcal{N}_\theta$ is likewise five, and the maximum number of constraint elements, if any, is six $\left(i.e.,\ \mathcal{N}_\theta + 1\right)$. This selection was made for the SMART Rotor Hub Loads Minimisation problem because the SMART rotor has five blades. It is emphasised that either or both $\mathcal{M}_Z$ and $\mathcal{N}_\theta$ can have values other than five. In this case, the dimensions of the various appropriate arrays and element limits in the NLP10x10 software would have to be changed accordingly. This would be a relatively simple task.

The corresponding "greatest" T-Matrix must then be determined or defined. Next, the specific problem to be considered must be defined—that is, the required set of control vector and end conditions harmonic couple elements and the constraints, if any, for this specific problem must be defined. Then if the set of control vector harmonic couple elements and/or the set of end conditions harmonic couple elements are smaller than the greatest element sets that have been defined, it is necessary to reduce these corresponding sets and the T-Matrix accordingly. Finally, the required constraints must be specified.

Let $\mathcal{M}_Z$ denote the total number of selected end conditions NP load couples (i.e., the sine and cosine pair of the selected NP load element) from $\left[Z_{\mathrm{C}}\right]$ to comprise the reduced $\left[Z_{\mathrm{C_R}}\right]$ and $\left[Z_{\mathrm{A_{C_R}}}\right]$ vectors, respectively; $\mathcal{N}_\theta$ denote the total number of selected control angle harmonic couples (i.e., the sine and cosine pair of the selected control harmonic angle) from $\left[\theta_{\mathrm{C}}\right]$ to comprise the reduced $\left[\theta_{\mathrm{C_R}}\right]$ vector; $\left\{\mathcal{H}_Z\right\}$ be the set of all index numbers of the selected end conditions NP load couples; $\left\{\mathcal{H}_\theta\right\}$ be the set of all index numbers of the selected control harmonic angle couples; $\left\{\mathcal{H}_{\mathrm{C1}}\right\}$ be the set of all index numbers of the selected individual control harmonic angle amplitude constraints; and $\left\{\mathcal{H}_{\mathrm{C2}}\right\}$ be the set of all index numbers of the selected individual control harmonic angle amplitudes to be included in the summation of amplitudes constraint. Then:

$$\left\{\mathcal{H}_Z\right\} = \left\{ h_{Z_i} \;\middle|\; \begin{array}{r} h_{Z_i} \text{ is a selected end conditions angular couple} \\ \text{index number,} \quad \text{where } i \in [1, 2, 3, 4, 5] \end{array} \right\}$$

$$\left\{\mathcal{H}_\theta\right\} = \left\{ h_{\theta_j} \;\middle|\; \begin{array}{r} h_{\theta_j} \text{ is a selected control anglar couple index number,} \\ \text{where } j \in [1, 2, 3, 4, 5] \end{array} \right\}$$

Then define the elements of $\left[Z_{C_R} \right]$, $\left[Z_{A_{C_R}} \right]$, and $\left[\theta_{C_R} \right]$ to be:

$$Z_{R_i} = Z_k \quad \text{for} \quad i \in \{1, 2, \bullet\bullet\bullet 2\mathcal{M}_Z\}, \qquad \text{where } \mathcal{M}_Z \le 5$$
$$\text{and} \quad k = \begin{cases} 2h_{Z_i} - 1 \text{ for } i \text{ odd} \\ 2h_{Z_i} \text{ for } i \text{ even} \end{cases}$$

$$Z_{A_{R_i}} = Z_{A_k} \quad \text{for} \quad i \in \{1, 2, \bullet\bullet\bullet 2\mathcal{M}_Z\}, \qquad \text{where } \mathcal{M}_Z \le 5$$
$$\text{and} \quad k = \begin{cases} 2h_{Z_i} - 1 \text{ for } i \text{ odd} \\ 2h_{Z_i} \text{ for } i \text{ even} \end{cases}$$

$$\theta_{R_j} = \theta_l \quad \text{for} \quad j \in \{1, 2, \bullet\bullet\bullet 2\mathcal{N}_\theta\}, \qquad \text{where } \mathcal{N}_\theta \le 5$$
$$\text{and} \quad l = \begin{cases} 2h_{\theta_j} - 1 \text{ for } j \text{ odd} \\ 2h_{\theta_j} \text{ for } j \text{ even} \end{cases}$$

The corresponding reduced T-Matrix $[T_R]$ is defined by eliminating the rows and columns from [T] that correspond to the elements eliminated from $[Z]$, $\left[Z_A \right]$, and $[\theta]$ when constructing $\left[Z_R \right]$, $\left[Z_{A_R} \right]$, and $\left[\theta_R \right]$, respectively.

The optimisation problem seeks to minimise the three types of constraints in the performance index J. They are: 1) constraints imposed directly on each element $\left[\theta_{R_j} \right]$ of the reduced $\left[\theta_R \right]$ vector, 2) constraints imposed on selected computed amplitudes of flap deflection angles of the reduced $\left[\theta_R \right]$ vector, and 3) constraints imposed on the summation of selected computed amplitudes of flap deflection angles. Specifically, the minimisation problem is:

Minimise: $J = \sum_{i=1}^{2\mathcal{M}_Z} W_{R_i} Z_{R_i}^2$

where W_{R_i} *is the Weighting Coefficient for the* $Z_{R_i}^2$ *term.*

Subject to:

Control Element Constraints:

$$Z_{MIN_j} \le Z_j \le Z_{MAX_j} \quad \ni \quad j \in \left\{1, 2, \bullet\bullet\bullet 2\mathcal{N}_\theta\right\}$$

Control Amplitudes Constraints:

$$A_q = \sqrt{Z_{2q-1}^2 + Z_{2q}^2} \le A_{MAX_q} \quad \ni \quad q \in \left\{\mathcal{H}_{C1}\right\}$$

where $\left\{\mathcal{H}_{C1}\right\} = \left\{ h_{C1_j} \middle| \begin{array}{l} h_{C1_j} \text{ is the index number of the selected individual control} \\ \text{harmonic angle amplitude constraint} \quad \ni \quad h_{C1_j} \in \left\{\mathcal{H}_\theta\right\} \end{array} \right\}$

Control Amplitude Summation Constraint:

$$\sum_{p \in \mathcal{H}_{C2}} A_p \le A_{MAX_\Sigma} \qquad \text{for} \quad p \in \left\{\mathcal{H}_{C2}\right\}$$

where $\left\{\mathcal{H}_{C2}\right\} = \left\{ h_{C2_j} \middle| \begin{array}{l} h_{C2_j} \text{ is the index number of the selected individual} \\ \text{control harminic angle amplitude to be included} \\ \text{in the sumation constraint} \quad \ni \quad h_{C2_j} \in \left\{\mathcal{H}_\theta\right\} \end{array} \right\}$

This problem is a special case of the most inclusive NLP10x10 problem described in previous sections 2.2.2 and 2.2.3.

2.3 Synthetic Data

The specific problems solved during this study were selected to be representative of actual problems to be solved during experimentation and/or testing of various rotorcraft configurations. It is assumed that the tasking of these T-Matrix NLP Control problems occurs within the framework of real-time controller duty cycles. Tasking of one of these T-Matrix NLP Control problems during the current duty cycle requires: 1) a previously identified T-Matrix, and 2) the actual control θ_0– vector and the actual measurement Z_A– vector pair determined during a previous duty cycle or at a reference epoch time. Preparations for such experimentation and/or testing requires the definition of the specific NLP problem (i.e., the performance index, the dimension and elements of the control vector and the measurement vector, and all constraints) to be solved. The main driver program, which defines the problem to be solved and tasks the NLPQLP System to solve it, must be coded, verified for proper functioning, and tuned-up so that it will reliably and efficiently solve the required problems during actual tests. This, of course, must precede the test and the generation of actual associated test data.

In an actual wind tunnel or flight test, the identification and/or acquisition of the required data would normally be tasked during a previous duty cycle or at a reference epoch time and would be transmitted to the NLP program host computer for the solution of the NLP problem. The solution control θ_{Sol}– vector would then be transmitted back to the controller during the current duty cycle. It is necessary to have input/output (I/O) interface/compatibility between the controller and the NLP host computer in order to successfully transmit usable data between the controller and the computer. This issue is not addressed in this document.

Although it might be possible to use similar test data from another test for verification and tuning purposes, in general this is cumbersome and can be unnecessarily time consuming. In addition, there could be I/O interface/compatibility issues associated with use of test data from another test. Synthetic determination of the T-Matrix, the actual control θ_0– vector, and the actual measurement Z_A– vector data avoids problems associated with use of test data from another test, and provides a simple and rapid method to obtain the required data during verification and tuning.

A synthesis procedure was designed to provide the T-Matrix and a previous duty cycle “actual” control θ_0– vector/”actual” measurement Z_A– vector pair required for the verification of the NLP main driver codes, and subsequent tuning of the operation of these codes to solve the type of problems expected to be encountered during the test. In order to develop a realistic test of the main driver codes and the NLPQLP System, a small degree of randomness was included in the synthetic modelling. Uniformly distributed pseudo-random numbers were selected for this synthesis process (see section 2.3.1).

First, the T-Matrix is determined by defining its elements from pseudo-random numbers between –1.0 inclusive and +1.0 exclusive (see section 2.3.2). Scaling coefficients are provided to ensure that the norm to the T-Matrix is within acceptable limits. Next, the previous duty cycle “actual” Control Amplitude A_0 and corresponding “actual” Control Phase Angle ϕ_0 vectors are likewise determined by defining their elements from pseudo-random numbers between –1.0 inclusive and +1.0 exclusive (see section 2.3.2) and using scaling coefficients. The elements of the ϕ_0–vector are adjusted to be within the principal cycle [0 deg, 360 deg). The “actual” control

θ_0– vector is then determined from the A_0 and adjusted ϕ_0 vectors. In this case, the elements of the θ_0– vector are constrained to be within specified limits à la "external limiting." Although the definition of the T-Matrix and the "actual" control θ_0– vector is accomplished using pseudo-random numbers, these elements do not have to be defined randomly and could be directly input or generated otherwise. These elements were generated randomly for convenience. The essential requirements for these elements are that the scaling and limits are reasonable and that the resulting values are realistic. The degree of randomness to the model is introduced in the definition of the "actual" measurement Z_{A}– vector. The "actual" measurement Z_{A}– vector is defined by adding uniformly distributed pseudo-random numbers to the elements of the $\mathsf{T}\theta_0$ product used in the definition of the Z_{A}-vector (see section 2.3.2).

2.3.1 Definition of the Random Number Generator Function RANQ(SEED)

The random number generator function RANQ(•) employed for data synthesis from both the Hewlett-Packard Alpha mainframe computer and the Mac Pro desktop computer produces a real uniformly pseudo-random number between 0.0 inclusive and 1.0 exclusive.

The RANQ(•) random number generator requires provision of an integer seed in the calling argument. This argument (i.e., the seed) has four elements and is consequently an integer vector with dimension (4 x 1). Each element of the initial entry seed for the first call to RANQ(•) should have a value between 0 and 4095. The fourth (i.e., the last) element must be an odd integer. The values of these seed elements are updated during each subsequent call to RANQ(•) to be used in the following call to RANQ(•).

2.3.2 Determination of the Synthetic T-Matrix, the "Actual" Control θ_0– Vector, and the "Actual" Measurement Z_{A}– Vector

The methodology employed to synthetically determine the T-Matrix, the previous duty cycle "actual" control θ_0– vector, and the previous duty cycle "actual" measurement Z_{A}– vector is:

First, Synthesise the T-Matrix *according to:*

$$\mathsf{T}_{q,p} = \mathsf{C}_1\left[2 \bullet \mathsf{RANQ}\left(\mathsf{ISEED1}_k\right) - 1\right]$$

where

k = *is the index number of the input seed for calls to* $\mathsf{RANQ}(\bullet)$.

and

$$\mathsf{T} = \begin{bmatrix} \bullet & \bullet & \bullet \\ \bullet & \mathsf{T}_{q,p} & \bullet \\ \bullet & \bullet & \bullet \end{bmatrix} \; is \; \left(\mathsf{N}_Z \times \mathsf{N}_\theta\right) \quad \forall \; q \in I_Z \;\; and \;\; \forall \; p \in I_\theta$$

Next, synthesise the previous duty cycle "actual" Control Amplitude A_0 *and Control Phase Angle* ϕ_0 *vectors according to*:

$$\left.\begin{aligned} \mathsf{A}_{0_r} &= \mathsf{A}_{00_r} + \mathsf{C}_2\left[2 \bullet \mathsf{RANQ}\left(\mathsf{ISEED2}_k\right) - 1\right] \\ &\textit{and} \\ \phi_{0_r} &= \phi_{00_r} + \mathsf{C}_3\left[2 \bullet \mathsf{RANQ}\left(\mathsf{ISEED3}_k\right) - 1\right] \end{aligned}\right\} \textit{ for } \left\{\begin{array}{c} \forall\ p \in I_\theta \\ \textit{and where} \\ r = \left(\dfrac{p+1}{2}\right) \end{array}\right.$$

where A_{00_r} and ϕ_{00_r} are optional bias constants, and the resulting ϕ_{0_r} is adjusted to be within the principal cycle ([0 deg, 360 deg) or [–180 deg, +180 deg)).

Then define the previous duty cycle "actual" $\theta_0 -$ Vector *according to*:

$$\theta_{p_{Initial}} = \left\{\begin{array}{ll} \mathsf{A}_{0_r} \bullet Sine\left(\phi_{0_r}\right) & \textit{for odd values of } p \\ \mathsf{A}_{0_r} \bullet Cosine\left(\phi_{0_r}\right) & \textit{for even values of } p \end{array}\right\} \textit{ for } \left\{\begin{array}{c} \forall\ p \in I_\theta \\ \textit{and where} \\ r = \left(\dfrac{p+1}{2}\right) \end{array}\right.$$

Subject to:

$$\left.\begin{array}{l} \theta_{\mathsf{MIN}_p} \le \theta_{p_{Initial}} \le \theta_{\mathsf{MAX}p} \\ \theta_{\mathsf{MIN}_p} \in \left(-\infty, +\infty\right) \\ \theta_{\mathsf{MAX}_p} \in \left(-\infty, +\infty\right) \end{array}\right\} \left\{\begin{array}{l} \text{Direct Constraints on the Control} \\ \theta - \text{vector Elements } \theta_p \text{ for: } p \in I_\theta \end{array}\right.$$

if $\theta_{p_{Initial}} \le \theta_{\mathsf{MIN}_p} + \varepsilon$ *then* $\theta_{p_{Initial}} = \theta_{\mathsf{MIN}_p} + \varepsilon$

if $\theta_{p_{Initial}} \ge \theta_{\mathsf{MAX}_p} - \varepsilon$ *then* $\theta_{p_{Initial}} = \theta_{\mathsf{MIN}_p} - \varepsilon$

otherwise there is no change to the value of $\theta_{p_{Initial}}$ *as determined by the random equation above.*

then

$$\theta_0 = \left[\bullet\ \bullet\ \bullet\ \left\{\theta_{p_{Initial}} \,\middle|\, p \in I_\theta\right\}\ \bullet\ \bullet\ \bullet\right]^{\mathsf{T}}$$

Finally, synthesise the previous duty cycle "actual" $Z_{\mathsf{A}} -$ Vector *according to*:

$$\Delta Z_{A_q} = C_4 \left[2 \bullet \mathrm{RANQ}\left(\mathrm{ISEED4}_k \right) - 1 \right] \qquad \forall \quad q \in I_Z$$

then

$$\Delta Z_A = \left[\bullet \quad \bullet \quad \bullet \quad \left\{ \Delta Z_{A_q} \,\middle|\, q \in I_Z \right\} \quad \bullet \quad \bullet \quad \bullet \right]^T$$

and

$$Z_A = T\theta_0 + \Delta Z_A$$

where

$\mathrm{RANQ}(\bullet)$ *is the Uniform Random Number Distribution Function which yields a uniformly random real number* $\in [0,1)$

$$I_Z = \left\{ \bullet \quad \bullet \quad \bullet \quad \left\{ \forall \quad q \quad \ni \quad Z_q \in Z \right\} \quad \bullet \quad \bullet \quad \bullet \right\}$$

$$I_\theta = \left\{ \bullet \quad \bullet \quad \bullet \quad \left\{ \forall \quad p \quad \ni \quad \theta_p \in \theta \right\} \quad \bullet \quad \bullet \quad \bullet \right\}$$

2.4 The NLP10x10 System Fortran Codes

The NLP10x10 System Main Driver code and supporting routines not part of the NLPQLP System itself (ref. 17 and Appendix A), and the NLPQLP System itself were written in Fortran 77 (refs. 21–23) and installed on both a Hewlett-Packard Alpha Server GS1280 7/1150 mainframe computer with the Open VMS Version 8.2 Operating System, and a Mac Pro desktop computer with the Mac OS X, Version 10.8.4 Operating System for the results presented in this report. The VMS FORTRAN Compiler (ref. 21) was used to compile the code on the mainframe computer. Initially (refs. 6–8) the G95 FORTRAN Compiler (ref. 22) was used to compile the code on the Mac Pro Desktop Computer. Subsequently, the gfortran FORTRAN Compiler (ref. 23) was installed on the Mac Pro desktop computer and was used to compile the code described herein. It is noted that different computers can, in general, produce slightly different numerical results because of differences in data formatting and arithmetic processing. The codes for both the Hewlett-Packard Alpha mainframe computer and the Mac Pro desktop computer are stand-alone codes and do not require any special software libraries. Accordingly, the associated software and codes installed on the Mac Pro desktop computer should be transportable to a Mac laptop computer for use in a wind tunnel. It is also noted that, for reference purposes, the explicit solution to the regulator problem (Appendix B) is included in the main driver code.

The DCL Command File Code for the Hewlett-Packard Alpha Mainframe Computer is listed in Appendix C, and listings of the codes for the NLP10x10 System Main Driver and supporting subroutines not part of the NLPQLP System (ref. 17 and Appendix A) are presented in Appendix D.

2.5 Input to the NLP10x10 System

Input to the NLPQLP System is accomplished via NAMELIST CDATA. This section contains a list of all the CDATA input parameters. The initial default values for these input parameters are shown in parentheses after each input parameter definition. The listings of the SMART Rotor Hub Loads minimisation cases presented in Appendices E and F include the input parameters required for the particular case being presented. It is noted that it is not necessary to input all available input parameters; only those parameters required for the particular problem being defined are required when using the NAMELIST input procedure. Also, if more than a particular value of an input parameter is input, the last value input is the value assumed to be the input value. A list of all the CDATA input parameters follows.

NAMELIST CDATA INPUT PARAMETERS

Input Parameter	Definition
A00(NCC)	Initial estimates of the Control Amplitude Vector element values (i.e., the initial estimates of the element values of the vector of radius magnitudes of the angular control couples $[R_0]$). Also, A00(NCC) are the optional bias constant vector A_{00} elements in the equation that randomly defines the "actual" Control Amplitude A_0 – vector element values in the reference previous duty cycle or at the reference epoch time for synthetic data definition only. $(\text{NCC}*1.0\text{D}+00)$
ACC	Desired final accuracy. The termination accuracy should not be much smaller than the accuracy by which the gradients are computed (Appendix A). $(1.0\text{D}-07)$
ACCQP	Selection pointer that defines the procedure by which the tolerance required by the QP solver to perform several tests is selected (Appendix A). $(0.0\text{D}+00)$ If ACCQP ≤ 0, the machine precision is computed internally in NLPQLP and subsequently multiplied by $1.0\text{D}-07$. > 0, the tolerance required by the QP solver to perform several tests (e.g., whether or not the optimality conditions are satisfied, or whether or not a number is considered to be zero).
AL0(NCC)	Greatest least bounds ([g.l.b.]s) of the elements of the Control Amplitude Vectors $([R \text{ and } R_0])$. $(\text{NCC}*-1.0\text{D}-07)$

NAMELIST CDATA INPUT PARAMETERS (CONTINUED)

Input Parameter	Definition
ALPHA	Adjustment coefficient $\alpha \in [0.0\mathrm{D}+00, 1.0\mathrm{D}+00]$ used in some feedback control applications to adjust the explicit solution to the regulator problem (Appendix B) to enhance control convergence in real time. $(1.0\mathrm{D}+00)$
APRV0(NCC)	Previous "Actual" Control Amplitude Vector element values (i.e., the vector of radius magnitudes of the angular control couples $[R_0]$ element values) in the reference previous duty cycle or at the reference epoch time before compression processing. (No initial data values set.)
AU0(NCC)	Least upper bounds ([l.u.b.]s) of the elements of the Control Amplitude Vectors $([R \text{ and } R_0])$. $(\mathrm{NCC}*3.0\mathrm{D}+00)$
CRAN1	Input coefficient C_1 in the equation that randomly defines the T-Matrix. $(2.0\mathrm{D}+00)$
CRAN2	Input coefficient C_2 in the equation that randomly defines the A_0 – vector. $(3.0\mathrm{D}+00)$
CRAN3	Input coefficient C_3 in the equation that randomly defines the ϕ_0 – Vector. $(1.0\mathrm{D}+00)$
CRAN4	Input coefficient C_4 in the equation that randomly defines the ΔZ_A – Vector. $(1.0\mathrm{D}+00)$
CV00(NMAX)	Initial "Actual" NLP Control Vector Estimates Vector $[\theta - \text{Vector}]$ before being adjusted to be within limits before compression processing. (No initial data values set.)
CVOUT	Special high accuracy control vector output option. (5)

If CVOUT

≤ 0, No special high accuracy control vector data is output.

> 0, Special high accuracy control vector data is output.

Then if IOUT

$= 1$, Only special high accuracy NLP control vector elements are output.

$= 2$, Special high accuracy NLP and Regulator control vector elements and the root sum square (RSS) of their differences are output.

$= 3$, Only special high accuracy Regulator control vector elements are output.

NAMELIST CDATA INPUT PARAMETERS (CONTINUED)

Input Parameter	Definition
CVPRV0(NMAX)	Actual Previous "Actual" NLP Control Vector $[\theta_0]$ element values in the reference previous duty cycle or at the reference epoch time before compression processing. (No initial data values set.)
ECPRV0(NZZ)	Actual Previous "Actual" NLP End Conditions Vector $[Z_A]$ element values in the reference previous duty cycle or at the reference epoch time before compression processing. (No initial data values set.)
EPS	Small value used to prevent the elements of the various control vector elements from violating their specified bounds (i.e., the values of AL0(NCC) and AU0(NCC)). $(1.0\mathrm{D}-07)$
ICASE	Case number. After the first case, ICASE is automatically incremented by 1 in each subsequent case unless specifically input by ICASE. (1)
ICYCL0	Maximum number of ±360 deg adjustments to the randomly defined values of the elements of the Phase Angle Control Vector $(\phi_0 - \text{Vector})$ in the reference previous duty cycle or at the reference epoch time. (2000)
IDATA	Input Data Type Specification Flag. (1)
	If IDATA
	$=1,$ The T-Matrix, the A_0 – vector, the $\phi_0 - \text{Vector}$, and the $\Delta Z_A - \text{Vector}$ are randomly defined.
	$=2,$ The Previous "Actual" NLP Control Vector $[\theta_0 - \text{Vector}]$ is directly input, and then the Previous "Actual" Control Amplitude Vector $(A_0 - \text{Vector})$ and the Previous "Actual" Control Phase Angle Vector $(\phi_0 - \text{Vector})$ are computed.
	$=3,$ The Previous "Actual" Control Amplitude Vector $(A_0 - \text{Vector})$ and the Previous "Actual" Control Phase Angle Vector $(\phi_0 - \text{Vector})$ are directly input, and then the Previous "Actual" NLP Control Vector $[\theta_0 - \text{Vector}]$ is computed.
	$>3,$ Error exit from the current case.
IN	Integer that specifies the number of the desired logical input unit [e.g., the READ statement for the NAMELIST CDATA in the program code starts with "READ(IN, CDATA)"]. (5)

NAMELIST CDATA INPUT PARAMETERS (CONTINUED)

Input Parameter	Definition
IOPT	Pointer flag that specifies the problem(s) to be solved. (1) If IOPT $= 1,$ Only solve the NLP Problem. $= 2,$ Solve both the NLP Problem and the Regulator Problem. $= 3,$ Only solve the Regulator Problem.
IOUT	Integer that specifies the number of the desired logical output unit [e.g., the WRITE statements in the program code start with "WRITE(IOUT,••••)"] (Appendix A). (6)
IPRINT	NLPQLP program output level selection pointer (Appendix A). (2) If IPRINT $= 0,$ No NLPQLP program output is provided. $= 1,$ Only the final convergence analysis is output. $= 2,$ One line of intermediate results is output for each iteration. $= 3,$ Additional detailed information is output for each iteration. $= 4,$ Additional line search date is output.
ISEED1(4)	Input (4 x 1) seed arguments for calls to the random number generator function RANQ(•) in the equation that randomly defines the system or transfer T-Matrix. It is updated automatically upon completion of the generation of the random number. (2395, 4013, 3813, 1837)
ISEED2(4)	Input (4 x 1) seed arguments for calls to the random number generator function RANQ(•) in the equation that randomly defines the previous duty cycle "actual" Control Amplitude A_0 Vector. It is updated automatically upon completion of the generation of the random number. (1843, 4011, 3364, 2835)
ISEED3(4)	Input (4 x 1) seed arguments for calls to the random number generator function RANQ(•) in the equation that defines the previous duty cycle "actual" Phase Angle ϕ_0 Vector. It is updated automatically upon completion of the generation of the random number. (3962, 1111, 3215, 2637)

NAMELIST CDATA INPUT PARAMETERS (CONTINUED)

Input Parameter	Definition
ISEED4(4)	Input (4 x 1) seed arguments for calls to the random number generator function RANQ(•) in the equation that defines the previous duty cycle "actual" measurement $\Delta Z_A -$ vector. It is updated automatically upon completion of the generation of the random number. (2397, 1504, 4031, 3173)
ITOUT	Compression data output level option flag. (1)

If Abs(ITOUT)

- $\geq 1,$ The Control Vector and End Conditions Vector data are output after the final (i.e., the Second) compression; and if $\text{ITOUT} < 0,$ the T-Matrix is also output.
- $\geq 2,$ The Control Vector and End Conditions Vector data are output for the initial data before compression and the data after the final (i.e., the Second) compression; and if $\text{ITOUT} < 0,$ the T-Matrix is also output.
- $\geq 3,$ The Control Vector and End Conditions Vector data are output for the initial data before compression and the data after the intermediate and after the final (i.e., the First and Second) compressions; and if $\text{ITOUT} < 0,$ the T-Matrix is also output.

L	Number of parallel systems, that is the number of function calls during a line search at predetermined iterates (Appendix A). (1)
LQL	Logical parameter that is used to select the method that is used to solve the Quadratic Programming (QP) sub-problem. (Appendix A). $(.\text{TRUE}.)$

If LQL

- = .TRUE., then the QP is solved proceeding from a full positive definite quasi-Newton matrix.
- = .FALSE., then the QP is solved by using a Cholesky decomposition (LDL) and updated internally so that the matrix C always consists of the lower triangular factor and D of the diagonal.

NAMELIST CDATA INPUT PARAMETERS (CONTINUED)

Input Parameter	Definition
LSAVE(NCC)	Angular End Conditions Couples Amplitude Constraint Elimination Pointer Vector (LSAVE(K), for K = 1, MI+1). $(6*1)$ If LSAVE(K), for K = 1, MI ≤ 0, Delete the k-th angular end conditions couple amplitude constraint from the end conditions amplitude constraint sub-vector $[{}^{1}\psi(\theta)]$. > 0, Retain the k-th angular end conditions couple amplitude constraint from the end conditions amplitude constraint sub-vector $[{}^{1}\psi(\theta)]$. If LSAVE(MI+1) ≤ 0, Delete the angular end conditions couple amplitude summation constraint from the end conditions amplitude constraint sub-vector $[{}^{2}\psi(\theta)]$. > 0, Retain the angular end conditions couple amplitude summation constraint from the end conditions amplitude constraint sub-vector $[{}^{2}\psi(\theta)]$.
MAXASUM	Least upper bound (l.u.b.) of the angular end conditions couple amplitude summation constraint. $(12.0D+00)$
MAXFUN	Integer variable that specifies the maximum number of function calls during a line search. MAXFUN is only required if L = 1. In this case, MAXFUN ≤ 50 (Appendix A). (30)
MAXIT	Integer variable that specifies the maximum number of outer iterations. One iteration corresponds to one formulation and solution to the quadratic programming sub-problem, or alternatively, one evaluation of the gradient (Appendix A). (100)
MAXNM	Selection pointer that specifies the stack size for storing merit function values or the line search procedure to be used (Appendix A). (0) If MAXNM ≤ 0, specifies that a monotone line search is performed. > 0, the integer variable that specifies the stack size for storing merit function values at previous iterations for a non-monotone line search. MAXNM should be ≤ 50.

NAMELIST CDATA INPUT PARAMETERS (CONTINUED)

Input Parameter	Definition
MI	Total number of inequality constraints; that is, the number of angular end conditions couple amplitude constraints that are specified plus the angular end conditions couple amplitude summation constraint if it is also specified. (0)
MINASUM	Greatest least bound (g.l.b.) of the angular end conditions couple amplitude summation constraint. $(0.0\mathrm{D}+00)$
MODE	Integer parameter used to select the desired version of NLPQLP to be used. (Appendix A). (0)

If MODE

$= 0,$ The normal execution (i.e., reverse communication) is specified.

$= 1,$ The initial guess for multipliers in U and the Hessian of the Lagrangian function in C and D are provided. If LQL = .TRUE., D is ignored. If LQL = .FALSE., the lower part of C has to contain the lower triangular factor of an LDL decomposition and the diagonal part D.

$= 2,$ Initial scaling (Oren-Luenberger) is done after the first step. BFGS updates are started from a multiple of the identity matrix.

$> 2,$ Initial and repeated scaling is done every MODE steps and the BFGS matrix is reset to a multiple of the identity matrix.

MSAVE0(NZZ) Angular End Conditions Couples Elimination Pointer Vector $(\mathrm{MSAVE0(J),\ for\ J = 1, NZ0/2})$. $(10*1)$

If MSAVE0(J), for J = 1, NZ0/2

$\leq 0,$ Delete the j-th angular end conditions couple from the angular end conditions couple vector $[Z_c]$.

$> 0,$ Retain the j-th angular end conditions couple from the angular end conditions couple vector $[Z_c]$.

MULT Run the Next Case indicator flag. (0)

If MULT

$\leq 0,$ Terminate the job. Do not run any additional cases.

$> 0,$ Increment the case number (ICASE) by one and run the next case. Reset MULT to zero upon completion of the case. This requires that MULT be input > 0 for each subsequent case to be run.

NAMELIST CDATA INPUT PARAMETERS (CONTINUED)

Input Parameter	Definition
NSAVE0(NXX)	Angular Control Couples Elimination Pointer Vector (NSAVE0(I), for I = 1, NX0/2). $(10*1)$ If NSAVE0(I), for I = 1, NX0/2 ≤ 0, Delete the i-th angular control couple from the angular control couples vector $[\theta_C]$. > 0, Retain the i-th angular control couple from the angular control couples vector $[\theta_C]$.
NX0	Initial dimension of the initial NLP Control Vector (θ) before compression processing. (10)
NZ0	Initial dimension of the initial NLP End Conditions Vector $[Z]$ before compression processing. (10)
OPTEND	Debug case stop location flag. (3) If OPTEND $= 1$, Terminate the case after compression processing of the angular conditions couple vector and the angular control couples vector. $= 2$, Terminate the case after the selection of the constraint functions before running the NLPQLP Optimisation procedure. $= 3$, Run the case to completion as specified (i.e., completion of the NLPQLP Optimisation procedure and/or the Regulator Problem procedure.
PHASE0(NCC)	Initial estimates of the Control Phase Angle Vector element values (i.e., the initial estimates of the element values of the vector of phase angles of the angular control couples $[\phi]$) before compression processing. Also, PHASE0(NCC) are the optional bias constant vector ϕ_{00} elements in the equation that randomly defines the "Actual" Control Phase Angle ϕ_0 – Vector element values in the reference previous duty cycle or at the reference epoch time for synthetic data definition only. (NCC * 90.0D+00)

NAMELIST CDATA INPUT PARAMETERS (CONCLUDED)

Input Parameter	Definition
PHSPRV0(NCC)	Previous "Actual" Control Phase Angle ϕ_0 – Vector element values (i.e., the vector of phase angle magnitudes of the angular control couples $[\phi_0]$ element values) in the reference previous duty cycle or at the reference epoch time. (No initial data values set.)
RHOB	Parameter used to specify restarts in the event that IFAIL = 2 (Appendix A). $(100.0\text{D}+00)$ If RHOB $\leq 1.0\text{D}+00$, If IFAIL = 2 (see Appendix A), NO restart is performed. $> 1.0\text{D}+00$, If IFAIL = 2 (see Appendix A), the BFGS-update matrix is reset to RHOB • I, where I is the identity matrix. The number of restarts is limited by the value of MAXFUN. It is recommended that RHOB > 1.0D+00.
STPMIN	Minimum step length selection parameter. (Appendix A). $(0.0\text{D}+00)$ If STPMIN $\leq 0.0\text{D}+00$, and/or if L = 1, STPMIN = ACC is assumed (Appendix A). $> 0.0\text{D}+00$, and L > 1, the step length reduction factor is then STPMIN**[1/(L–1)]. In this case, it is recommended that the value of L be on the order of the accuracy by which the functions are computed (Appendix A).
T0(NZZ,NXX)	Initial T-Matrix before compression processing. $(\text{NZNX}*0.0\text{D}+00)$
WDT0(NZZ)	Diagonal of the weighting matrix W_Z in the performance index J before compression processing. $(\text{NZZ}*1.0\text{D}+00)$
WDX(NXX)	Diagonal of the weighting matrix $W_{\dot{\theta}}$ in the $\dot{\theta}$ term of the performance index J of the regulator problem (Appendix B). $(\text{NXX}*1.0\text{D}+00)$
WX(NXX)	Diagonal of the weighting matrix W_θ in the θ term of the performance index J of the regulator problem (Appendix B). $(\text{NXX}*1.0\text{D}+00)$
WZ(NZZ)	Diagonal of the weighting matrix W_Z in the Z term of the performance index J of the regulator problem (Appendix B). $(\text{NZZ}*1.0\text{D}+00)$

VECTOR AND ARRAY DIMENSION ARGUMENTS

Dimension Argument	Definition
NCC	Dimension used for amplitude, phase angle, and constraint vectors. (6)
NMAX	Row dimension of the approximate Hessian matrix of the Lagrangian function (Appendix A). NMAX must be at least two greater than NXX. NMAX is also used for some of the NLP control vectors. (14)
NXX	Dimension used for some of the NLP control vectors. (10)
NZZ	Dimension used for some of the NLP end conditions vectors. (10)
NZNX	Equivalent single dimension of the two dimensional T-Matrices and the two dimensional weighting coefficient matrices. NZNX ≡ NZZ • NXX. (100)

2.6 SMART Rotor Hub Loads Minimisation Cases

Parametric values of the flap deflection frequency control elements D2, D3, D4, D5, and D6, and the Hub Load end conditions elements FX, FY, FZ, MX, and MY were analysed using the NLP10x10 System on both a mainframe and a desktop computer. The basic parametric cases that were considered are defined in Table 1. These cases assumed a best-guess starting estimate for the control vectors. Values of the initial performance index (J0), the solution performance index (J), the number of iterations required for solution (ITR), and the number of function evaluations required for solution (FEV) for each case are also presented in this table. Constraint sensitivity to the value of the upper constraint bound was determined for two cases of interest, cases 20 and 25. The results of this constraint sensitivity study for best-guess control vector starting estimates for cases 20 and 25 are presented in Tables 2 and 3, respectively. The cases defined in Tables 1, 2, and 3 were re-run using zero-control-vector starting estimates instead of the best-guess starting control vector estimates used originally. The results of these zero-control-vector starting estimate cases are presented in Tables 4, 5, and 6. These results, which are essentially the same as the best-guess control vector starting estimate cases, illustrate the robustness of the NLP10x10 System.

The cases defined in Tables 1 through 6 were run on both a Hewlett-Packard Alpha Server GS1280 7/1150 mainframe computer with the Open VMS Version 8.2 Operating System, and a Mac Pro desktop computer with the Mac OS X, Version 10.8.4 Operating System. Agreement between the cases run on these two computers is excellent, and correspondingly Tables 1 through 6 represent the case results from both computers. Listings of the cases run on the Hewlett-Packard Alpha mainframe computer are presented in Appendix E, and the cases run on the Mac Pro desktop computer are presented in Appendix F. Zero-control-vector starting estimates were assumed for these cases.

3.0 Results and Conclusions

The NLP10x10 System is a specially tailored version of the system described in reference 6 designed to facilitate the solution of rotor hub loads minimisation problems both analytically and in test environments, including real-time applications. The logic included in the NLP10x10 System to easily and quickly reduce the dimension of the control and end conditions greatly enhances the use of this system in both analytic studies and real-time test operations.

This system was used very successfully in the SMART Active Flap Rotor Hub Loads minimisation problems described in references 6 and 18. The results of this study were presented at the American Helicopter Society Fifth Decennial Aeromechanics Specialists' Conference in January 2014 (ref. 18). The dimensions of the control and end conditions vectors were varied in the problems analysed in this study. Specifically, control variable vector dimensions of 2, 4, 6, and 10, and end conditions vector dimensions of 2, 6, and 10 were considered. These problems shared the same T-Matrix plant model that was reduced appropriately by input flags for the particular problem being solved. The results of the cases analysed using the NLP10x10 System are summarised in Tables 1 through 4, and the listings of these cases are presented in Appendices E and F.

Excellent agreement between cases initiated with best-guess starting estimates for the control vector elements and cases initiated with zero-control-vector elements resulted, indicating the robustness of the NLP10x10 algorithm. Excellent agreement was also obtained between cases run on a Hewlett-Packard Alpha Server GS1280 7/1150 mainframe computer and a Mac Pro desktop computer. The NLP10x10 System code used on both computers was identical, and because these computers had different operating systems and Fortran compilers, the code for the NLP10x10 System was transportable between these computers. This fact suggests that the NLP10x10 System code could be easily transported to other computers.

4.0 References

1. Leyland, J. A.: A Higher Harmonic Optimal Controller to Optimise Rotorcraft Aeromechanical Behaviour. NASA Technical Memorandum 110390, Mar. 1996.

2. Leyland, J. A.: A Closed-Loop Optimal Neural-Network Controller to Optimise Rotorcraft Aeromechanical Behaviour, Volume 1, Theory and Methodology. NASA TM-2001-209622, Mar. 2001.

3. Leyland, J. A.: A Closed-Loop Optimal Neural-Network Controller to Optimise Rotorcraft Aeromechanical Behaviour, Volume 2, Output From Two Sample Cases. NASA TM-2001-209623, Mar. 2001.

4. Leyland, J. A.: A General Regularisation Functional to Enhance the Update Convergence of a Neural-Network Rotorcraft Behaviour Model, Volume 1, Theory and Methodology. NASA TM-2002-211843, Oct. 2002.

5. Leyland, J. A.: A General Regularisation Functional to Enhance the Update Convergence of a Neural-Network Rotorcraft Behaviour Model, Volume 2, Output From Two Sample Cases. NASA TM-2002-211844, Oct. 2002.

6. Leyland, J. A.: Use of the NLPQLP Sequential Programming Algorithm to Solve Rotorcraft Aeromechanical Constrained Optimisation Problems, Volume 1: Theory and Methodology. NASA TM-2014-216632, Jan. 2014.

7. Leyland, J. A.: Use of the NLPQLP Sequential Programming Algorithm to Solve Rotorcraft Aeromechanical Constrained Optimisation Problems, Appendix B: Cases Run on the Hewlett-Packard Alpha Mainframe Computer. NASA TM-2016-216632, Apr. 2014.

8. Leyland, J. A.: Use of the NLPQLP Sequential Programming Algorithm to Solve Rotorcraft Aeromechanical Constrained Optimisation Problems, Appendix C: Cases Run on the Mac Pro Desktop Computer. NASA TM-2016-216632, Apr. 2014.

9. Anon.: IMSL MATH/LIBRARY User's Manual, FORTRAN Subroutines for Mathematical Applications, Volume 3, Chapter 8: Nonlinearly Constrained Minimisation Using Finite Difference Gradients. Version 1.1, MALB-USM-UNBND-EN8901-1.1, pp. 895–902, Jan. 1989.

10. Schittkowski. K.: Non-Linear Programming Codes. *Lecture Notes in Economics and Mathematics*, **183**, Springer-Verlag, Berlin, Germany, 1980.

11. Schittkowski, K.: On the Convergence of a Sequential Quadratic Programming Method With an Augmented Lagrangian Line Search Function. *Mathematik Operationsforschung und Statistik, Serie Optimization*, **14**, pp. 197–216, 1983.

12 Schittkowski, K.: NLPQL: a FORTRAN Subroutine Solving Constrained Non-Linear Programming Problems. (C. L. Monma, Ed.), *Annals of Operations Research*, **5**, pp. 485–500, 1986.

13. Gill, P. E.; Murray, W.; Saunders, M. A.; and Wright,. M. H.: Model Building and Practical Aspects of Non-Linear Programming. *Computational Mathematical Programming*, (K. Schittkowski, Ed.), NATO ASI Series, **15**, Springer-Verlag, Berlin, Germany, 1985.

14. Powell, M. J. D.: A Fast Algorithm for Non-Linearly Constrained Optimisation Calculations. *Numerical Analysis Proceedings*, Dundee, 1977.

15. Powell, M. J. D.: A Fast Algorithm for Non-Linearly Constrained Optimisation Calculations. *Lecture Notes in Economics and Mathematics*, **630**, Springer-Verlag, Berlin, Germany, pp. 144–157, 1978.

16. Stoer, J.: Principles of Sequential Quadratic Programming Methods in Solving Non-Linear Problems. *Computational Mathematical Programming*, (K. Schittkowski, Ed.), NATO ASI Series, **15**, Springer-Verlag, Berlin, Germany, 1985.

17. Schittkowski, K.: NLPQLP: a FORTRAN Implementation of a Sequential Quadratic Programming Algorithm With Distributed and Non-Monotone Line Search – User's Guide, Version 3.1. Department of Computer Science, University of Bayreuth, Germany, Feb. 2010.

18. Kottapalli, S., and Leyland, J. A.: Application of Sequential Quadratic Programming to Minimise SMART Active Flap Rotor Hub Loads. AHS Fifth Decennial Aeromechanics Specialists' Conference, San Francisco, CA, Jan. 20–22, 2014.

19. Straub, F. K.; Anand, V. R.; Birchette, T. S.; and Lau, B. H.: Wind Tunnel Test of the SMART Active Flap Rotor. AHS 65th Annual Forum Proceedings, Grapevine, TX, May 27–29, 2009.

20. Hall, S. R.; Anand, V. R.; Straub, F. K.; and Lau, B. H.: Active Flap Control of the SMART Rotor for Vibration Reduction. AHS 65th Annual Forum Proceedings, Grapevine, TX, May 27–29, 2009.

21. Anon.: Programming in VAX FORTRAN, Software Version: V4.0. *VAX/VMS Manual No. AA-D034D-TE*, Digital Equipment Corporation (DEC), Maynard, Mass., Sept. 1984.

22. Vaught, A.: G95 FORTRAN Compiler, Mesa, Ariz., Oct. 2006.

23. The GFORTRAN Team: Using GNU Fortran for GCC, Version 4.6.4. Free Software Foundation, Boston, Mass., 2011.

TABLE 1. BEST-GUESS STARTING ESTIMATE FOR CV0

CASE	End Conditions Vector 5P Hub Loads										Control Vector Flap Deflection Frequency									
1	FXS	FXC	FYS	FYC	FZS	FZC	MXS	MXC	MYS	MYC	2S	2C	3S	3C	4S	4C	5S	5C	6S	6C
2	FXS	FXC	FYS	FYC	FZS	FZC	MXS	MXC	MYS	MYC			3S	3C	4S	4C	5S	5C		
3	FXS	FXC	FYS	FYC	FZS	FZC	MXS	MXC	MYS	MYC	2S	2C								
4	FXS	FXC	FYS	FYC	FZS	FZC	MXS	MXC	MYS	MYC			3S	3C						
5	FXS	FXC	FYS	FYC	FZS	FZC	MXS	MXC	MYS	MYC					4S	4C				
6	FXS	FXC	FYS	FYC	FZS	FZC	MXS	MXC	MYS	MYC							5S	5C		
7	FXS	FXC	FYS	FYC	FZS	FZC	MXS	MXC	MYS	MYC									6S	6C
8	FXS	FXC	FYS	FYC	FZS	FZC					2S	2C								
9	FXS	FXC	FYS	FYC	FZS	FZC							3S	3C						
10	FXS	FXC	FYS	FYC	FZS	FZC									4S	4C				
11	FXS	FXC	FYS	FYC	FZS	FZC											5S	5C		
12	FXS	FXC	FYS	FYC	FZS	FZC													6S	6C
13							MXS	MXC	MYS	MYC	2S	2C								
14							MXS	MXC	MYS	MYC			3S	3C						
15							MXS	MXC	MYS	MYC					4S	4C				
16							MXS	MXC	MYS	MYC							5S	5C		
17							MXS	MXC	MYS	MYC									6S	6C
18	FXS	FXC									2S	2C								
19	FXS	FXC											3S	3C						
20	FXS	FXC													4S	4C				
21	FXS	FXC															5S	5C		
22	FXS	FXC																	6S	6C
23			FYS	FYC							2S	2C								
24			FYS	FYC									3S	3C						
25			FYS	FYC											4S	4C				
26			FYS	FYC													5S	5C		
27			FYS	FYC															6S	6C
28					FZS	FZC					2S	2C								
29					FZS	FZC							3S	3C						
30					FZS	FZC									4S	4C				
31					FZS	FZC											5S	5C		
32					FZS	FZC													6S	6C
33	FXS	FXC	FYS	FYC							2S	2C								
34	FXS	FXC	FYS	FYC									3S	3C						
35	FXS	FXC	FYS	FYC											4S	4C				
36	FXS	FXC	FYS	FYC													5S	5C		
37	FXS	FXC	FYS	FYC															6S	6C

TABLE 1. BEST-GUESS STARTING ESTIMATE FOR CV0 (CONTINUED)

CASE	J0	J	ITR	FEV
1	4 E+00	5 E-07	14	188
2	2 E+04	2 E+02	13	108
3	9 E+03	9 E+02	5	22
4	7 E+04	2 E+03	7	26
5	3 E+04	7 E+03	9	31
6	1 E+04	6 E+03	7	27
7	3 E+04	1 E+04	8	29
8	8 E+03	6 E+02	5	21
9	6 E+04	7 E+02	9	32
10	3 E+04	4 E+03	9	32
11	1 E+04	3 E+03	6	23
12	3 E+04	6 E+03	8	29
13	1 E+03	5 E+01	9	30
14	7 E+03	7 E+02	6	22
15	3 E+03	5 E+02	7	26
16	3 E+03	2 E+03	6	22
17	2 E+03	2 E+03	7	27
18	2 E+03	6 E-11	6	24
19	2 E+04	2 E-08	7	25
20	1 E+04	7 E-08	6	21
21	5 E+03	9 E-11	6	24
22	1 E+04	3 E-10	9	32
23	2 E+03	5 E-11	6	24
24	2 E+04	4 E-10	7	25
25	1 E+04	6 E-09	6	20
26	5 E+03	1 E-10	6	23
27	1 E+04	8 E-09	7	24
28	3 E+03	4 E-08	8	28
29	2 E+04	2 E-08	6	23
30	1 E+04	7 E-11	8	28
31	1 E+03	3 E-10	7	27
32	9 E+03	8 E-11	10	33
33	4 E+03	1 E+01	5	22
34	4 E+04	6 E+01	7	26
35	2 E+04	2 E+03	8	29
36	1 E+04	1 E+02	6	23
37	2 E+04	9 E+02	7	25

TABLE 2. CASE 20 CONSTRAINT ANALYSIS—BEST-GUESS STARTING ESTIMATE FOR CV0

CASE	[EC]		[CV]		Comments	Upper Bound Constraint	Constraint Value
20	FXS	FXC	4S	4C	Constrained	Not Active	10.0000
920	FXS	FXC	4S	4C	Constrained	Active	9.0000
820	FXS	FXC	4S	4C	Constrained	Active	8.0000
720	FXS	FXC	4S	4C	Constrained	Active	7.0000
620	FXS	FXC	4S	4C	Constrained	Active	6.0000
520	FXS	FXC	4S	4C	Constrained	Active	5.0000
420	FXS	FXC	4S	4C	Constrained	Active	4.0000
320	FXS	FXC	4S	4C	Constrained	Active	3.0000
220	FXS	FXC	4S	4C	Constrained	Active	2.0000
120	FXS	FXC	4S	4C	Constrained	Active	1.0000
0620	FXS	FXC	4S	4C	Constrained	Active	0.6000
0320	FXS	FXC	4S	4C	Constrained	Active	0.3000
0120	FXS	FXC	4S	4C	Constrained	Active	0.1000

CASE	J0	J	ITR	FEV	Amp (Deg)	Phase (Deg)
20	1.1937xE+04	6.9909xE-08	6	21	9.8456	-62.8965
920	1.1937xE+04	7.9594xE+00	7	25	9.0000	-63.3113
820	1.1937xE+04	3.7949xE+01	7	22	8.0000	-63.9154
720	1.1937xE+04	9.0310xE+01	7	22	7.0000	-64.6929
620	1.1937xE+04	1.6517xE+02	6	20	6.0000	-65.7312
520	1.1937xE+04	2.6276xE+02	6	19	5.0000	-67.1885
420	1.1937xE+04	3.8354xE+02	6	19	4.0000	-69.3846
320	1.1937xE+04	5.2863xE+02	6	19	3.0000	-73.0803
220	1.1937xE+04	7.0166xE+02	6	19	2.0000	-80.6709
120	1.1937xE+04	9.2801xE+02	7	22	1.0000	-107.2343
0620	1.1937xE+04	1.6541xE+03	10	33	0.6000	-146.8446
0320	1.1937xE+04	6.0832xE+03	9	30	0.3000	-147.9234
0120	1.2987xE+04	1.1442xE+04	10	58	0.1000	-148.0954

TABLE 3. CASE 25 CONSTRAINT ANALYSIS—BEST-GUESS STARTING ESTIMATE FOR CV0

CASE	[EC]		[CV]		Comments	Upper Bound Constraint	Constraint Value
25	FYS	FYC	4S	4C	Constrained	Not Active	10.0000
525	FYS	FYC	4S	4C	Constrained	Active	5.0000
425	FYS	FYC	4S	4C	Constrained	Active	4.0000
325	FYS	FYC	4S	4C	Constrained	Active	3.0000
225	FYS	FYC	4S	4C	Constrained	Active	2.0000
125	FYS	FYC	4S	4C	Constrained	Active	1.0000
0625	FYS	FYC	4S	4C	Constrained	Active	0.6000
0325	FYS	FYC	4S	4C	Constrained	Active	0.3000
0125	FYS	FYC	4S	4C	Constrained	Active	0.1000

CASE	J0	J	ITR	FEV	Amp (Deg)	Phase (Deg)
25	1.0866xE+04	5.7041xE-09	6	20	5.3291	130.7954
525	1.0866xE+04	3.3992xE+00	7	22	5.0000	131.3135
425	1.0866xE+04	5.5726xE+01	6	19	4.0000	133.4172
325	1.0866xE+04	1.7260xE+02	6	19	3.0000	136.9533
225	1.0866xE+04	3.5892xE+02	7	22	2.0000	144.1937
125	1.0866xE+04	6.4775xE+02	7	22	1.0000	169.0030
0625	1.0866xE+04	1.2387xE+03	10	33	0.6000	-150.0216
0325	1.0866xE+04	5.2845xE+03	9	29	0.3000	-147.9257
0125	1.1894xE+04	1.0422xE+04	13	79	0.1000	-147.6405

Notes:

-150.0216 Degrees = +209.9784 Degrees

-147.9257 Degrees = +212.0743 Degrees

-147.6405 Degrees = +212.3595 Degrees

TABLE 4. ZERO VECTOR STARTING ESTIMATE FOR CV0

CASE	End Conditions Vector 5P Hub Loads										Control Vector Flap Deflection Frequency									
1	FXS	FXC	FYS	FYC	FZS	FZC	MXS	MXC	MYS	MYC	2S	2C	3S	3C	4S	4C	5S	5C	6S	6C
2	FXS	FXC	FYS	FYC	FZS	FZC	MXS	MXC	MYS	MYC			3S	3C	4S	4C	5S	5C		
3	FXS	FXC	FYS	FYC	FZS	FZC	MXS	MXC	MYS	MYC	2S	2C								
4	FXS	FXC	FYS	FYC	FZS	FZC	MXS	MXC	MYS	MYC			3S	3C						
5	FXS	FXC	FYS	FYC	FZS	FZC	MXS	MXC	MYS	MYC					4S	4C				
6	FXS	FXC	FYS	FYC	FZS	FZC	MXS	MXC	MYS	MYC							5S	5C		
7	FXS	FXC	FYS	FYC	FZS	FZC	MXS	MXC	MYS	MYC									6S	6C
8	FXS	FXC	FYS	FYC	FZS	FZC					2S	2C								
9	FXS	FXC	FYS	FYC	FZS	FZC							3S	3C						
10	FXS	FXC	FYS	FYC	FZS	FZC									4S	4C				
11	FXS	FXC	FYS	FYC	FZS	FZC											5S	5C		
12	FXS	FXC	FYS	FYC	FZS	FZC													6S	6C
13							MXS	MXC	MYS	MYC	2S	2C								
14							MXS	MXC	MYS	MYC			3S	3C						
15							MXS	MXC	MYS	MYC					4S	4C				
16							MXS	MXC	MYS	MYC							5S	5C		
17							MXS	MXC	MYS	MYC									6S	6C
18	FXS	FXC									2S	2C								
19	FXS	FXC											3S	3C						
20	FXS	FXC													4S	4C				
21	FXS	FXC															5S	5C		
22	FXS	FXC																	6S	6C
23			FYS	FYC							2S	2C								
24			FYS	FYC									3S	3C						
25			FYS	FYC											4S	4C				
26			FYS	FYC													5S	5C		
27			FYS	FYC															6S	6C
28					FZS	FZC					2S	2C								
29					FZS	FZC							3S	3C						
30					FZS	FZC									4S	4C				
31					FZS	FZC											5S	5C		
32					FZS	FZC													6S	6C
33	FXS	FXC	FYS	FYC							2S	2C								
34	FXS	FXC	FYS	FYC									3S	3C						
35	FXS	FXC	FYS	FYC											4S	4C				
36	FXS	FXC	FYS	FYC													5S	5C		
37	FXS	FXC	FYS	FYC															6S	6C

TABLE 4. ZERO VECTOR STARTING ESTIMATE FOR CV0 (CONTINUED)

CASE	J0	J	ITR	FEV
1	4 E+04	5 E-07	19	269
2	4 E+04	2 E+02	13	109
3	4 E+04	9 E+02	9	32
4	4 E+04	2 E+03	8	29
5	4 E+04	7 E+03	9	32
6	4 E+04	6 E+03	8	31
7	4 E+04	1 E+04	8	29
8	4 E+04	6 E+02	8	29
9	4 E+04	7 E+02	8	28
10	4 E+04	4 E+03	8	28
11	4 E+04	3 E+03	8	31
12	4 E+04	6 E+03	7	25
13	4 E+03	5 E+01	7	24
14	4 E+03	7 E+02	8	27
15	4 E+03	5 E+02	6	24
16	4 E+03	2 E+03	7	25
17	4 E+03	2 E+03	7	26
18	1 E+04	1 E-08	8	29
19	1 E+04	4 E-11	8	28
20	1 E+04	2 E-08	7	24
21	1 E+04	1 E-10	8	29
22	1 E+04	5 E-11	9	32
23	1 E+04	7 E-11	7	26
24	1 E+04	4 E-11	8	28
25	1 E+04	6 E-09	6	20
26	1 E+04	3 E-10	7	25
27	1 E+04	7 E-12	7	26
28	1 E+04	4 E-10	10	33
29	1 E+04	9 E-13	8	29
30	1 E+04	7 E-11	8	28
31	1 E+04	2 E-10	6	24
32	1 E+04	2 E-08	10	34
33	3 E+04	1 E+01	8	30
34	3 E+04	6 E+01	8	29
35	3 E+04	2 E+03	6	20
36	3 E+04	1 E+02	8	29
37	3 E+04	9 E+02	7	26

TABLE 5. CASE 20 CONSTRAINT ANALYSIS—ZERO VECTOR STARTING ESTIMATE FOR CV0

CASE	[EC]		[CV]		Comments	Upper Bound Constraint	Constraint Value
20	FXS	FXC	4S	4C	Constrained	Not Active	10.0000
920	FXS	FXC	4S	4C	Constrained	Active	9.0000
820	FXS	FXC	4S	4C	Constrained	Active	8.0000
720	FXS	FXC	4S	4C	Constrained	Active	7.0000
620	FXS	FXC	4S	4C	Constrained	Active	6.0000
520	FXS	FXC	4S	4C	Constrained	Active	5.0000
420	FXS	FXC	4S	4C	Constrained	Active	4.0000
320	FXS	FXC	4S	4C	Constrained	Active	3.0000
220	FXS	FXC	4S	4C	Constrained	Active	2.0000
120	FXS	FXC	4S	4C	Constrained	Active	1.0000
0620	FXS	FXC	4S	4C	Constrained	Active	0.6000
0320	FXS	FXC	4S	4C	Constrained	Active	0.3000
0120	FXS	FXC	4S	4C	Constrained	Active	0.1000

CASE	J0	J	ITR	FEV	Amp (Deg)	Phase (Deg)
20	1.4844xE+04	1.6010xE-08	7	24	9.8456	-62.8965
920	1.4844xE+04	7.9594xE+00	7	25	9.0000	-63.3113
820	1.4844xE+04	3.7949xE+01	7	22	8.0000	-63.9154
720	1.4844xE+04	9.0310xE+01	7	22	7.0000	-64.6929
620	1.4844xE+04	1.6517xE+02	6	20	6.0000	-65.7312
520	1.4844xE+04	2.6276xE+02	6	19	5.0000	-67.1885
420	1.4844xE+04	3.8354xE+02	6	19	4.0000	-69.3846
320	1.4844xE+04	5.2863xE+02	6	19	3.0000	-73.0803
220	1.4844xE+04	7.0166xE+02	6	19	2.0000	-80.6709
120	1.4844xE+04	9.2801xE+02	7	22	1.0000	-107.2343
0620	1.4844xE+04	1.6541xE+03	10	33	0.6000	-146.8446
0320	1.4844xE+04	6.0832xE+03	9	30	0.3000	-147.9234
0120	1.4844xE+04	1.1442xE+04	13	69	0.1000	-148.0957

TABLE 6. CASE 25 CONSTRAINT ANALYSIS—ZERO VECTOR STARTING ESTIMATE FOR CV0

CASE	[EC]		[CV]		Comments	Upper Bound Constraint	Constraint Value
25	FYS	FYC	4S	4C	Constrained	Not Active	10.0000
525	FYS	FYC	4S	4C	Constrained	Active	5.0000
425	FYS	FYC	4S	4C	Constrained	Active	4.0000
325	FYS	FYC	4S	4C	Constrained	Active	3.0000
225	FYS	FYC	4S	4C	Constrained	Active	2.0000
125	FYS	FYC	4S	4C	Constrained	Active	1.0000
0625	FYS	FYC	4S	4C	Constrained	Active	0.6000
0325	FYS	FYC	4S	4C	Constrained	Active	0.3000
0125	FYS	FYC	4S	4C	Constrained	Active	0.1000

CASE	J0	J	ITR	FEV	Amp (Deg)	Phase (Deg)
25	1.3724xE+04	5.6974xE-09	6	20	5.3291	130.7954
525	1.3724xE+04	3.3992xE+00	6	19	5.0000	131.3135
425	1.3724xE+04	5.5726xE+01	6	19	4.0000	133.4172
325	1.3724xE+04	1.7260xE+02	6	19	3.0000	136.9533
225	1.3724xE+04	3.5892xE+02	7	22	2.0000	144.1937
125	1.3724xE+04	6.4775xE+02	7	21	1.0000	169.0030
0625	1.3724xE+04	1.2387xE+03	10	33	0.6000	-150.0216
0325	1.3724xE+04	5.2845xE+03	9	29	0.3000	-147.9257
0125	1.3724xE+04	1.0422xE+04	12	64	0.1000	-147.6405

Notes:

-150.0216 Degrees = +209.9784 Degrees

-147.9257 Degrees = +212.0743 Degrees

-147.6405 Degrees = +212.3595 Degrees

Appendix A

NLPQLP: A Fortran Implementation of a Sequential Quadratic Programming Algorithm with Distributed and Non-Monotone Line Search - User's Guide, Version 3.1

By Professor Klaus Schittkowski
Department of Computer Science
University of Bayreuth, Germany

Table of Contents

NLPQLP: A Fortran Implementation of a Sequential Quadratic Programming Algorithm with Distributed and Non-Monotone Line Search

- User's Guide, Version 3.1 -

Address: Prof. K. Schittkowski
Department of Computer Science
University of Bayreuth
D - 95440 Bayreuth

Phone: (+49) 921 557750

E-mail: klaus.schittkowski@uni-bayreuth. de

Web: http://www.klaus-schittkowski.de

Date: May, 2010

Abstract

The Fortran subroutine NLPQLP solves smooth nonlinear programming problems by a sequential quadratic programming (SQP) algorithm. This version is specifically tuned to run under distributed systems controlled by an input parameter (l). In case of computational errors as for example caused by inaccurate function or gradient evaluations, a non-monotone line search is activated. Numerical results are included which show that in case of noisy function values, a significant improvement of the performance is achieved compared to the version with monotone line search. Further stabilization is obtained by performing internal restarts in case of errors when computing the search direction due to inaccurate derivatives. The new version of NLPQLP successfully solves more than 90% of our 306 test examples subject to a stopping tolerance of 10^{-7}, although at most two digits in function values are correct in the worst case and although numerical differentiation leads to additional truncation errors. In addition, automated initial and periodic scaling with restarts is implemented. The usage of the code is documented and illustrated by an example.

Keywords: SQP, sequential quadratic programming, nonlinear programming, non-monotone line search, numerical algorithm, distributed computing, Fortran code.

1 Introduction

We consider the general optimization problem to minimize an objective function f under nonlinear equality and inequality constraints,

$$\begin{aligned} & \min f(x) \\ x \in I\!R^n : \; & g_j(x) = 0, \qquad j = 1, \bullet\bullet\bullet, m_e \\ & g_j(x) \geq 0, \qquad j = m_e + 1, \bullet\bullet\bullet, m \\ & x_l \leq x \leq x_u \end{aligned} \tag{1}$$

where x is an *n-dimensional* parameter vector. It is assumed that all problem functions $f(x)$ and $g_j(x),\ j = 1, \bullet\bullet\bullet, m$ are continuously differentiable on the whole $I\!R^n$.

Sequential quadratic programming (SQP) is the standard general purpose method to solve smooth nonlinear optimization problems, at least under the following assumptions:

- The problem is not too large.
- Functions and gradients can be evaluated with sufficiently high precision.
- The problem is smooth and well scaled.

The original code NLPQL of Schittkowski [47] is a Fortran implementation of a sequential quadratic programming (SQP) algorithm. The numerical algorithm is based on extensive comparative numerical tests, see Schittkowski [40, 44, 42], Schittkowski et al. [56], Hock and Schittkowski [25], and on further theoretical investigations published in [41, 43, 45, 46]. The algorithm is extended to solve also nonlinear least squares problems efficiently, see [49] or [51], and to handle problems with very many constraints, see [54]. To conduct the numerical tests, a random test problem generator is developed for a major comparative study, see [40], Two collections with together 306 test problems are published in Hock and Schittkowski [25] and in Schittkowski [48]. Fortran source codes and a test frame can be downloaded from the home page of the author,

http://www.klaus-schittkowski.de

Many of them became part of the Cute test problem collection of Bongartz et al. [7]. About 80 test problems based on a Finite Element formulation are collected for a comparative evaluation in Schittkowski et al. [56]. A set of 1,300 least squares test problems solved by an extension of the code NLPQL to retain typical features of a Gauss-Newton algorithm, is described in [51].

Also these problems can be downloaded from the home page of the author together with an interactive software system called EASY-FIT, see [52].

Moreover, there exist hundreds of commercial and academic applications of NLPQL, for example

1. mechanical structural optimization, see Schittkowski, Zillober, Zotemantel [56] and Kneppe, Krammer, Winkler [28],
2. data fitting and optimal control of transdermal pharmaceutical systems, see Boderke, Schittkowski, Wolf [3] or Blatt, Schittkowski [6],
3. computation of optimal feed rates for tubular reactors, see Birk, Liepelt, Schittkowski, and Vogel [5],
4. food drying in a convection oven, see Frias, Oliveira, and Schittkowski [15],
5. optimal design of horn radiators for satellite communication, see Hartwanger, Schittkowski, and Wolf [23],
6. receptor-ligand binding studies, see Schittkowski [50],
7. optimal design of surface acoustic wave filters for signal processing, see Bfinner, Schittkowski, and van de Braak [8].

Previous and present versions of NLPQLP are part of commercial libraries, modeling systems, or optimization systems like

- IMSL Library (Visual Numerics Inc., Houston) for general nonlinear programming (Version 1.0, 1981),
- ANSYS/POPT (CAD-FEM, Grafing) for structural optimization,
- DesignXplorer (ANSYS, Canonsburg) for structural design optimization,
- STRUREL (RCP, Munich) for reliability analysis,
- TEMPO (OECD Reactor Project, Halden) for control of power plants,
- Microwave Office Suit (Applied Wave Research, El Segundo) for electronic design,
- MOOROPT (Marintek, Trondheim) for the design of mooring systems,
- iSIGHT (Enginious Software/Dassault) for multi-disciplinary CAE,
- POINTER (Synaps, Atlanta) for design automation,
- EXCITE (AVL, Graz) for non-linear dynamics of power units,
- ModeFRONTIER (ESTECO, Trieste) for integrated multi-objective and multidisciplinary design optimization,

- TOMLAB/MathLab (Tomlab Optimization, Vasteras, Sweden) for general nonlinear programming, least squares optimization, data fitting in dynamical systems,
- EASY-FIT (Schittkowski, Bayreuth) for data fitting in dynamical systems,
- OptiSLang (DYNARDO, Weimar), for structural design optimization,
- AMESim (IMAGINE, Roanne), for multidisciplinary system design,
- LMS OPTIMUS (NOESIS, Leuven, Belgium) for multi-disciplinary CAE,
- RADIOSS/M-OPT (MECALOG/Altair, Antony, France) for multi-disciplinary CAE,
- CHEMASIM (BASF, Ludwigshafen) for the design of chemical reactors.

Customers include, among many others, AMD, Astrium, BASF, Bayer, Bell Labs, BMW, Chevron Research, DLR, Dow Chemical, DuPont, EADS, EMCOSS, ENSIGC, EPCOS, ESA-ESOC, Eurocopter, Fantoft Prosess, General Electric, Hoechst, Hidro-electrica Espanola, IABG, IBM, Institute for Energy Technology Halden, KFZ Karlsruhe, Kongsberg Maritime, Lockheed Martin, Loral Space Systems, Markov Processes, Marintek, MTU, NASA Langley, NASA Ames, Nevesbu, National Airspace Laboratory, Norsk Hydro Research, Norwegian Computing Center, Norwegian Defense Agency, OECDHalden, Philips, Polysar, ProSim, Rolls-Royce, Shell, Siemens, Sintef, Solar Turbines, Statoil, TNO, Transpower, USAF Research Lab, Wright R & D Center and in addition dozens of academic research institutions all over the world.

The general availability of parallel computers and in particular of distributed computing in networks motivates a careful redesign of the original implementation NLPQL to allow simultaneous function evaluations. The resulting extensions are implemented and the code is called NLPQLP. An additional input parameter l is introduced for the number of parallel machines, that is the number of function calls to be executed simultaneously. In case of $l-1$, NLPQLP is more or less identical to NLPQL besides of additional changes of the code. Otherwise, the line search procedure is modified to allow parallel function calls, which can also be applied for approximating gradients by difference formulae. The mathematical background is outlined, in particular the modification of the line search algorithm to retain convergence under parallel systems. It must be emphasized that dis tributed computation of function values is only simulated throughout the paper. It is up to the user to adopt the code to a particular parallel environment.

However, SQP methods are quite sensitive subject to round-off or any other errors in function and especially gradient values. If objective or constraint functions cannot be computed within machine accuracy or if the accuracy by which gradients are approximated is above the termination tolerance, the code could break down typically with the error message IFAIL = 4. In this situation, the line search cannot be terminated within a given number of iterations and the algorithm is stopped.

All new versions since 2.0 makes use of non-monotone fine search in the error situation described above. The idea is to replace the reference value of the fine search termination check, $\psi_{r_k}(x_k, v_k)$, by

$$\max\left\{ \psi_{r_j}(x_j, v_j) \ : \ j = k-p, \bullet\bullet\bullet, k \right\}$$

where $\psi_r(x, v)$ is a merit function and p a given parameter. The general idea is not new and for example described in Dai [11], where a general convergence proof for the unconstrained case is presented. The general idea goes back to Grippo, Lampariello, and Lucidi [18], and was extended to constrained optimization and trust region methods in a series of subsequent papers, see Bonnans et al. [4], Deng et al. [13], Grippo et al. [19, 20], Ke and Han [26], Ke et al. [27], Lucidi et al. [30], Panier and Tits [34], Raydan [39], and Toint [59, 60]. However, there is a basic difference in the methodology: Our goal is to allow monotone line searches as long as they terminate successfully, and to apply a non-monotone one only in a special error situation.

Despite of strong analytical results, SQP methods do not always terminate successfully. Besides of the difficulties leading to the usage of non-monotone line search, it might happen that the search direction as computed from a certain quadratic programming sub-problem, is not a downhill direction of the merit function needed to perform a fine search. Possible reasons are again severe errors in function and especially gradient evaluations, or a violated regularity condition concerning linear independency of gradients of active constraints (LICQ). In the latter case, the optimization problem is not modelled in a suitable way to solve it directly by an SQP method. Our new version performs an automated restart as soon as a corresponding error message appears. The BFGS quasi-Newton matrix is reset to a multiple of the identity matrix and the matrix update procedure starts from there.

Scaling is an extremely important issue and an efficient procedure is difficult to derive in the general case without knowing too much about the numerical structure of the optimisation problem. If requested by the user, the first BFGS update is started from a multiple of the identity matrix, which takes into account information from the solution of the initial quadratic programming sub-problem. This restart can be repeated periodically with successively adapted scaling parameters.

In Section 2 we outline the general mathematical structure of an SQP algorithm, the non-monotone line search, and the modifications to run the code under distributed systems. Section 3 contains some numerical results obtained for a set of 306 standard test problems of the collections published in Hock and Schittkowski [25] and in Schittkowski [48]. They show the sensitivity of the new version with respect to the number of parallel machines and the influence of gradient approximations under uncertainty. Moreover, we test the non-monotone line search versus the monotone one, and generate noisy test problems by adding random errors to function values and by inaccurate gradient approximations. This situation appears frequently in practical environments, where complex simulation codes prevent accurate responses and where gradients can only be computed by a difference formula. The usage of the Fortran subroutine is documented in Section 4 and Section 5 contains an illustrative examples.

2 Sequential Quadratic Programming Methods

Sequential quadratic programming or SQP methods belong to the most powerful nonlinear programming algorithms we know today for solving differentiable nonlinear programming problems of the form (1). The theoretical background is described e.g. in Stoer [58] in form of a review, or in Spellucci [57] in form of an extensive text book. From the more practical point of view, SQP methods are also introduced in the books of Papalambros, Wilde [35] and Edgar, Himmelblau [14]. Their excellent numerical performance is tested and compared with other methods in Schittkowski [40], and since many years they belong to the most frequently used algorithms to solve practical optimization problems.

To facilitate the notation of this section, we assume that upper and lower bounds $x_u$and xi are not handled separately, i.e., we consider the somewhat simpler formulation

$$\begin{array}{rll} & \min f(x) & \\ x \in I\!R^n : & g_j(x) = 0, & j = 1, \bullet\bullet\bullet, m_e \\ & g_j(x) \geq 0, & j = m_e + 1, \bullet\bullet\bullet, m \end{array} \qquad (2)$$

It is assumed that all problem functions $f(x)$ and $g_j(x),\ j = 1, \bullet\bullet\bullet, m$ are continuously differentiable on $I\!R^n$.

The basic idea is to formulate and solve a quadratic programming sub-problem in each iteration which is obtained by linearising the constraints and approximating the Lagrangian function

$$L(x,u) := f(x) - \sum_{j=1}^{m} u_j g_j(x) \qquad (3)$$

quadratically, where $x \in I\!R^n$ is the primal variable and $u = (u_1, \bullet\bullet\bullet, u_m)^T \in I\!R^m$ the multiplier vector.

To formulate the quadratic programming sub-problem, we proceed from given iterates $x \in I\!R^n$ an approximation of the solution, $v \in I\!R^m$ an approximation of the multipliers, and $B_k \in I\!R^{n\times n}$, an approximation of the Hessian of the Lagrangian function. Then one has to solve the quadratic programming problem

$$\begin{aligned} & \min \tfrac{1}{2} d^T B_k d \; + \; \nabla f(x_k)^T d \\ d \in I\!R^n \; : \; & \nabla g_j \left(x_k\right)^T d \; + \; g_j \left(x_k\right) = 0 \, , \qquad j \; = 1, \bullet \bullet \bullet , \, m_e \\ & \nabla g_j \left(x_k\right)^T d \; + \; g_j \left(x_k\right) \geq 0 \, , \qquad j \; = m_e + 1, \bullet \bullet \bullet , \, m \end{aligned} \tag{4}$$

Let d_k be the optimal solution and u_k the corresponding multiplier of this sub-problem. A new iterate is obtained by

$$\begin{pmatrix} x_{k+1} \\ v_{k+1} \end{pmatrix} \; : \; = \begin{pmatrix} x_k \\ v_k \end{pmatrix} \; + \; \alpha_k \begin{pmatrix} d_k \\ u_k - v_k \end{pmatrix} \tag{5}$$

where $\alpha_k \in \left(0, 1\right]$ is a suitable step length parameter.

Although we are able to guarantee that the matrix B_k is positive definite, it is possible that (4) is not solvable due to inconsistent constraints. One possible remedy is to introduce an additional variable $\delta \in I\!R$ leading to a modified quadratic programming problem, see Schittkowski [47] for details.

The steplength parameter α_k is required in (5) to enforce global convergence of the SQP method, i.e., the approximation of a point satisfying the necessary Karush-Kuhn-Tucker optimality conditions when starting from arbitrary initial values, typically a user-provided $x_0 \in I\!R^n$ and $v_0 \in 0, \; B_0 \in I$. α_k should satisfy at least a sufficient decrease condition of a merit function $\phi_r(\alpha)$ given by

$$\phi_r(\alpha) \; : \; = \psi_r \left[\begin{pmatrix} x \\ v \end{pmatrix} \; + \; \alpha \begin{pmatrix} d \\ u - v \end{pmatrix} \right] \tag{6}$$

with a suitable penalty function by $\psi_r \left(x, u\right)$. Implemented is the augmented Lagrangian function

$$\psi_r \left(x, u\right) \; : = \; f(x) \; - \; \sum_{j \, \in \, J} \left[v_j g_j (x) - \tfrac{1}{2} r_j g_j (x)^2 \right] \; - \; \tfrac{1}{2} \sum_{j \, \in \, K} \frac{v_j^2}{r_j} \tag{7}$$

with $\qquad J \; : = \left\{ 1, \bullet \bullet \bullet , \, m_e \right\} \cup \left\{ j \; : \; m_e < \; j \; \leq \; m, \; g_j \left(x\right) \leq \; v_j / r_j \right\}$

and $K := \{1, \bullet\bullet\bullet, m\} \setminus J,$ cf. Schittkowski [45].

The objective function is *penalized* as soon as an iterate leaves the feasible domain. The corresponding penalty parameters r_j, $j = 1, \bullet\bullet\bullet, m$ that control the degree of constraint violation, must carefully be chosen to guarantee a descent direction of the merit function, see Schittkowski [45] or Wolfe [61] in a more general setting, i.e., to get

$$\phi'_{r_k}(0) = \nabla \psi_{r_k}(x_k, v_k)^T \begin{pmatrix} d_k \\ u_k - v_k \end{pmatrix} < 0 \qquad (8)$$

Finally one has to approximate the Hessian matrix of the Lagrangian function in a suitable way. To avoid calculation of second derivatives and to obtain a final super linear convergence rate, the standard approach is to update B_k, by the BFGS quasi-Newton formula, cf. Powell [37] or Stoer [58].

The implementation of a line search algorithm is a critical issue when implementing a nonlinear programming algorithm, and has significant effect on the overall efficiency of the resulting code. On the one hand we need a line search to stabilize the algorithm, on the other hand it is not desirable to waste too many function calls. Moreover, the behaviour of the merit function becomes irregular in case of constrained optimization because of very steep slopes at the border caused by large penalty terms. Even the implementation is more complex than shown above, if linear constraints and bounds of the variables are to be satisfied during the line search.

Usually, the step-length parameter α_k is chosen to satisfy the Armijo [1] condition

$$\phi_r(\sigma\beta^i) \le \phi_r(0) + \sigma\beta^i \mu \phi'_{r_k}(0) \qquad (9)$$

see for example Ortega and Rheinboldt [33], The constants are from the ranges $0 < \mu < 0.5$, $0 < \beta < 1$, and $0 < \sigma \le 1$. We start with $i = 0$ and increase i until (9) is satisfied for the first time, say at i_k. Then the desired steplength is $\alpha_k = \sigma\beta^{i_k}$.

Fortunately, SQP methods are quite robust and accept the steplength one in the neighbourhood of a solution. Typically the test parameter μ for the Armijo-type sufficient descent property (9) is

very small. Nevertheless the choice of the reduction parameter β must be adopted to the actual slope of the merit function. If β is too small, the line search terminates very fast, but on the other hand the resulting stepsizes are usually too small leading to a higher number of outer iterations. On the other hand, a larger value close to one requires too many function calls during the fine search. Thus, we need some kind of compromise, which is obtained by first applying a polynomial interpolation, typically a quadratic one, and use (9) only as a stopping criterion. Since $\phi_r(0)$, $\phi_r'(0)$, and $\phi_r(\alpha_i)$ are given, α_i the actual iterate of the line search procedure, we easily get the minimiser of the quadratic interpolation. We accept then the maximum of this value and the Armijo parameter as a new iterate, as shown by the subsequent code fragment implemented in NLPQLP.

Algorithm 2.1

Let β, μ *with* $0 < \beta < 1, \quad 0 < \mu < 0.5$ *be given*

Start: $\alpha_0 := 1$

For $i = 0, 1, 2, \bullet\bullet\bullet,$ *do*:

1) *If* $\phi_r(\alpha_i) < \phi_r(0) + \mu \alpha_i \phi_{r_k}'(0),$ *then stop.*

2) *Compute* $\alpha_i := \dfrac{0.5\, \alpha_i^2\ \phi_r'(0)}{\alpha_i \phi_{r_k}'(0) - \phi_r(\alpha_i) + \phi_r(0)}$

3) *Let* $\alpha_{i+1} := \max(\beta \alpha_i\ \overline{\alpha}_i).$

Corresponding convergence results are found in Schittkowski [45]. $\overline{\alpha}_i$ is the minimiser of the quadratic interpolation, and we use the Armijo descent property for checking termination. Step 3 is required to avoid irregular values, since the minimiser of the quadratic interpolation could be outside of the feasible domain $(0,1]$. The search algorithm is implemented in NLPQLP together with additional safeguards, for example to prevent violation of bounds. Algorithm 4.1 assumes that $\phi_r(1)$ is known before calling the procedure, i.e., that the corresponding function values are given. We have to stop the algorithm, if sufficient descent is not observed after a certain number of iterations, say 10. If the tested stepsize falls below machine precision or the accuracy by which model function values are computed, the merit function cannot decrease further.

To outline the new approach, let us assume that functions can be computed simultaneously on l different machines. Then l test values $\alpha_i = \beta^{i-1}$ with $\beta = \varepsilon^{1/(l-1)}$ are selected, $i = 1, \bullet\bullet\bullet, l$. where ε is a guess for the machine precision. Next we require l parallel function calls to get the corresponding model function values. The first α_i satisfying a sufficient descent property (9), say for $i = i_k$ is accepted as the new steplength to set the subsequent iterate by $\alpha_k := \alpha_{i_k}$. One has to be sure that existing convergence results of the SQP algorithm are not violated.

The proposed parallel line search will work efficiently, if the number of parallel machines l is sufficiently large, and works as follows, where we omit the iteration index k.

Algorithm 2.2

$Let\ \beta, \mu\ with\ 0 < \beta < 1,\ \ 0 < \mu < 0.5\ \ be\ given$

$Start:\ For\quad \alpha_i = \beta^{i-1}\quad compute\quad \phi_r(\alpha_i)\quad for\ \ i = 0, 1, 2, \bullet\bullet\bullet, l-1$

$For\quad i = 0, 1, 2, \bullet\bullet\bullet\quad do:$

$\qquad If\quad \phi_r(\alpha_i) < \phi_r(0) + \mu\alpha_i\phi_r'(0),\quad then\ stop.$

To precalculate l candidates in parallel at log-distributed points between a small tolerance $\alpha = \tau$ and $\alpha = 1,\ 0 < \tau << 1$, we propose $\beta = \tau^{1/(l-1)}$.

The paradigm of parallelism is SPMD, i.e., Single Program Multiple Data. In a typical situation we suppose that there is a complex application code providing simulation data, for example by an expensive Finite Element calculation in mechanical structural optimisation. It is supposed that various instances of the simulation code providing function values, are executable on a series of different machines, so-called slaves, controlled by a master program that executes NLPQLP. By a message passing system, for example PVM, see Geist et al. [16], only very few data need to be transferred from the master to the slaves. Typically only a set of design parameters of length n must to be passed. On return, the master accepts new model responses for objective function and constraints, at most $m+1$ double precision numbers. All massive numerical calculations and model data, for example the stiffness matrix of a Finite Element

model in a mechanical engineering application, remain on the slave processors of the distributed system.

In both situations, i.e., the serial or parallel version, it is still possible that Algorithm 2.1 or Algorithm 2.2 breaks down because to too many iterations. In this case, we proceed from a descent direction of the merit function, but $\phi_r(0)$ is extremely small. To avoid interruption of the whole iteration process, the idea is to repeat the line search with another stopping criterion. Instead of testing (9), we accept a stepsize α_k as soon as the inequality

$$\phi_{r_k}(\alpha_k) \leq \max_{k-p(k)\leq j\leq k}\left[\phi_{r_j}(0) + \alpha_{i_k}\mu\phi'_{r_k}(0)\right] \qquad (10)$$

is satisfied, where $p(k)$ is a predetermined parameter with $p(k)=\min[k, p]$, p a given tolerance. Thus, we allow an increase of the reference value $\phi_{r_{j_k}}(0)$ in a certain error situation, i.e., an increase of the merit function value. To implement the non-monotone line search, we need a queue consisting of merit function values at previous iterates. In case of $k=0$, the reference value is adapted by a factor greater than 1, i.e., $\phi_{r_{j_k}}(0)$ is replaced by $t\phi_{r_{j_k}}(0)$, $t>1$.

The basic idea to store reference function values and to replace the sufficient descent property by a sufficient 'ascent' property in max-form, is for example described in Dai [11], where a general convergence proof for the unconstrained case is presented. The general idea goes back to Grippo, Lampariello, and Lucidi [18], and was extended to constrained optimization and trust region methods in a series of subsequent papers, see Bonnans et al. [4], Deng et al. [13], Grippo et al. [19, 20], Ke and Han [26], Ke et al. [27], Lucidi et al. [30], Panier and Tits [34], Raydan [39], and Toint [59, 60]. However, there is a difference in the methodology: Our goal is to allow monotone line searches as long as they terminate successfully, and to apply a non-monotone one only in an error situation.

The final step of an SQP method consists of updating the quasi-Nerwton Matrix B_k, e.g., by the BFGS formula

$$B_k := B_k + \frac{q_k q_k^T}{p_k^T q_k} - \frac{B_k p_k p_k^T B_k}{p_k^T B_k p_k} , \qquad (11)$$

where $q_k := \nabla_x L\left(x_{k+1}, u_k\right) - \nabla_x L\left(x_k, u_k\right)$ and $p_k := x_{k+1} - x_k$. Special safeguards guarantee that $p_k^T q_k > 0$ and that thus all matrices B_k r remain positive definite provided that B_0 is positive definite. A possible scaling factor and restart procedure is to replace an actual B_k by $\gamma_k I$ before performing the update (11), where $\gamma_k = \frac{p_k^T q_k}{p_k^T p_k}$ and where I denotes the identity matrix, see for example Liu and Nocedal [29]. Scaled restarts are recommended if, e.g., the convergence turns out to become extremely slow.

3 Performance Evaluation

3.1 The Test Environment

Our numerical tests use the 306 academic and real-life test problems published in Hock and Schittkowski [25] and in Schittkowski [48]. Part of them are also available in the Cute library, see Bongartz et. al [7], and their usage is described in Schittkowski [55].

Since analytical derivatives are not available for all problems, we approximate them numerically. The test examples are provided with exact solutions, either known from analytical precalculations *by hand* or from the best numerical data found so far.

First we need a criterion to decide whether the result of a test run is considered as a successful return or not. Let $\varepsilon > 0$ be a tolerance for defining the relative accuracy, x_k the final iterate of a test run, and x^* the supposed exact solution known from the test problem collection. Then we call the output a successful return, if the relative error in the objective function is less than e and if the maximum constraint violation is less than ε^2 i.e., if

$$f(x_k) - f(x^*) < \varepsilon |f(x^*)|, \quad if\ f(x^*) \neq 0$$

or

$$f(x_k) < \varepsilon, \quad if\ f(x^*) = 0$$

and

$$r(x_k) = \left\| g(x_k)^- \right\|_\infty < \varepsilon^2 ,$$

where $\| \bullet \bullet \bullet \|_\infty$ denotes the maximum norm and $g(x_k)^- = \min[0, g(x_k)]$, $j > m_e$ and $g(x_k)^- = g(x_k)$ otherwise.

We take into account that a code returns a solution with a better function value than the known one, subject to the error tolerance of the allowed constraint violation. However, there is still the possibility that an algorithm terminates at a local solution different from the known one. Thus, we call a test run a successful one, if in addition to the above decision the internal termination conditions are satisfied subject to a reasonably small tolerance (IFAIL = 0), and if

$$f(x_k) - f(x^*) \geq \varepsilon |f(x^*)|, \quad if\ f(x^*) \neq 0$$

or

$$f(x_k) \geq \varepsilon, \quad if\ f(x^*) = 0$$

and

$$r(x_k) < \varepsilon^2 ,$$

For our numerical tests, we use $\varepsilon = 0.01$ to determine a successful return, i.e., we require a final accuracy of one percent. Note that in all cases, NLPQLP is called with a termination tolerance of 10^{-7}.

If gradients are not available in analytical form, they must be approximated in a suitable way. The three most popular difference formulae are the following ones:

1. **Forward differences:**

$$\frac{\partial}{\partial x_i} f(x) \approx \frac{1}{\eta_i} [f(x + \eta_i e_i) - f(x)] \tag{12}$$

2. **Two-sided differences:**

$$\frac{\partial}{\partial x_i} f(x) \approx \frac{1}{2\eta_i} [f(x + \eta_i e_i) - f(x - \eta_i e_i)] \tag{13}$$

3. **Forth-order formula:**

$$\frac{\partial}{\partial x_i} f(x) \approx \frac{1}{4!\,\eta_i} [2f(x - 2\eta_i e_i) - 16f(x - \eta_i e_i) + 16f(x + \eta_i e_i) - 2f(x + 2\eta_i e_i)] \tag{14}$$

Here $\eta_i = \eta \max\left(10^{-5}, |x_i|\right)$ and e_i is the $i-th$ unit vector, $i = 1, \bullet\bullet\bullet, n$. The tolerance η depends on the difference formula and is set to $\eta = \eta_m^{1/2}$ for forward differences, $\eta = \eta_m^{1/3}$ for two-sided differences, and $\eta = \left(\eta_m / 72\right)^{1/4}$ for fourth-order formulae. η_m is a guess for the accuracy by which function values are computed, i.e., either machine accuracy in case of

analytical formulae or an estimate of the noise level in function computations. In a similar way, derivatives of constraints are computed.

The Fortran implementation of the SQP method introduced in the previous section, is called NLPQLP. The code represents the most recent version of NLPQL which is frequently used in academic and commercial institutions. NLPQLP is prepared to run also under distributed systems, but behaves in exactly the same way as the serial version, if the number of simulated processors is set to one. Functions and gradients must be provided by reverse communication and the quadratic programming sub-problems are solve by the primal-dual method of Goldfarb and Idnani [17] based on numerically stable orthogonal decompositions. NLPQLP is executed with termination accuracy ACC = 10^{-7} as mentioned already above, and a maximum number of iterations MAXIT = 500.

In the subsequent tables, we use the notation

- n_{succ} - number of successful test runs (according to above definition)
- n_{func} - average number of function evaluations
- n_{grad} - average number of gradient evaluations or iterations, respectively
- $f(x)$ - final objective function value
- $r(x)$ - final constraint violation
- i_{fail} - failure code

To get n_{func} or n_{grad}, we count each evaluation of a whole set of function or gradient values, respectively, for a given iterate x_k also in the case of several simulated processors, $l > 0$. However, additional function evaluations needed for gradient approximations, are not counted for n_{func}. Their average number is n_{func} for forward differences, $2 \text{ x } n_{func}$ for two-sided differences, and $4 \text{ x } n_{func}$ for fourth-order formulae. One gradient computation corresponds to one iteration of the SQP method.

The Fortran codes are compiled by the Intel Visual Fortran Compiler, Version 10.1, EM64T, under Windows XP64 and Dual Core AMD Opteron Processor 265, 1.81 GHz, with 8 GB of RAM.

3.2 Testing Distributed Function Calls

First we investigate the question, how parallel line searches influence the overall performance. Table 1 shows the number of successful test runs and the average number of iterations or gradient evaluations, n_{it} for an increasing number of simulated parallel calls of model functions denoted by l. The forward difference formula (12) is used for gradient approximations and non-monotone line search is applied with a queue size of $p = 30$. Calculation time is about one second for solving all 306 test problems without random perturbations.

$l = 1$ corresponds to the sequential case, when Algorithm 2.1 is applied to the line search consisting of a quadratic interpolation combined with an Armijo-type bisection strategy and a non-monotone stopping criterion.

l	n_{succ}	n_{grad}	l	n_{succ}	n_{grad}
1	306	23	8	299	39
3	224	177	9	301	32
4	242	170	10	302	31
5	268	114	15	303	23
6	289	75	20	303	22
7	297	45	50	303	22

Table 1: Performance Results for Parallel Line Search

In all other cases, $l > 1$ simultaneous function evaluations are made according to Algorithm 2.2. To get a reliable and robust line search, one should use at least seven parallel processors. No significant improvements are observed, if we evaluate more than ten functions in parallel.

The most promising possibility to exploit a parallel system architecture occurs, when gradients cannot be calculated analytically, but have to be approximated numerically, for example by forward differences, two-sided differences, or even higher order methods. Then we need at least n additional function calls, where n is the number of optimisation variables, or a suitable multiple of n.

3.3 Function Evaluations and Gradient Approximations by a Difference Formulae Under Random Noise

For our numerical tests, we apply the forth-order difference formula (14). To test the stability of the formula and the underlying SQP code, we add randomly generated noise to each function value. Non-monotone line search is applied with a queue length of p = 30 in error situations, and the serial line search calculation by Algorithm 2.1 is used. Moreover, the BFGS quasi-Newton updates are restarted with ρI if a descent direction cannot be computed.

To compare the different stabilisation approaches, we apply three different scenarios:

Table 2 - monotone line search, no restarts

Table 3 - non-monotone line search, no restarts

Table 4 - non-monotone line search and restarts

ε_{err}	n_{succ}	n_{func}	n_{grad}
0	304	35	22
10^{-12}	303	36	22
10^{-10}	297	40	23
10^{-8}	293	43	23
10^{-6}	280	57	24
10^{-4}	243	74	26
10^{-2}	133	133	32

Table 2: Test Results for Monotone Line Search without Restarts

ε_{err}	n_{succ}	n_{func}	n_{grad}
0	306	38	22
10^{-12}	303	37	23
10^{-10}	300	43	25
10^{-8}	300	53	24
10^{-6}	295	71	25
10^{-4}	268	116	30
10^{-2}	294	185	33

Table 3: Test Results for Non-Monotone Line Search without Restarts

ε_{err}	n_{succ}	n_{func}	n_{grad}
0	306	38	22
10^{-12}	305	39	23
10^{-10}	300	47	24
10^{-8}	303	83	27
10^{-6}	302	105	30
10^{-4}	297	318	43
10^{-2}	279	647	64

Table 4: Test Results for Non-Monotone Line Search and Restarts

The corresponding results are evaluated for increasing random perturbations (ε_{err}). More precisely, if ρ denotes a uniformly distributed random number between 0 and 1, we replace $f(x_k)$ by $f(x_k)\left[1+\varepsilon_{err}(2\rho-1)\right]$ at each iterate x_k. In the same way, restriction functions are perturbed. The tolerance for approximating gradients, η_m is set to the machine accuracy in case of $\varepsilon_{err}=0$, and to the random noise level otherwise.

The numerical results are surprising and depend heavily on the new non-monotone line search strategy and the additional stabilisation procedures. We are able to solve about 90% of the test examples in case of extremely noisy function values with at most one correct digit in partial derivative values. However, the stabilization process is costly. The more test problems are successfully solved, the more iterations, especially function evaluations, are needed.

3.4 Testing Scaled Restarts

In some situations, the convergence of an SQP method becomes quite slow for many reasons, e.g., badly scaled variables or functions, inaccurate derivatives, or inaccurate solutions of the quadratic program (4). In these situations, errors in the search direction or the partial derivatives influence the update procedure (11) and the quasi-Newton matrices B_k are getting more and more inaccurate.

A frequently proposed remedy is to restart the update algorithm by replacing the actual matrix B_k by the initial matrix B_0 or any similar one, if more information is available. One possibility is to multiply a special scaling factor with the identity matrix, i.e., to let $B_k := \gamma_k I$ for selected

iterates k, where $\gamma_k := \dfrac{p_k^T q_k}{p_k^T p_k}$ and where I denotes the identity matrix, see for example Liu and Nocedal [29]. $q_k := \nabla_x L\left(x_{k+1}, u_k\right) - \nabla_x L\left(x_k, u_k\right)$ and $p_k := x_{k+1} - x_k$.

Scaled restarts are recommended if convergence turns out to become extremely slow. To illustrate the situation, we consider a few test runs where the examples are generated by discretising a two-dimensional elliptic partial differential equation, see Maurer and Mittelmann [31, 32]. The original formulation is that of an optimal control problem where the state and control variables are both discretised.

From a total of 13 original test cases, we select five problems that could not be solved by NLPQLP as efficiently as expected with standard solution tolerances. Depending on the grid size in our case 20 in each direction, we get problems with $n = 722$ or $n = 798$ variables, respectively, and $m_e = 361$ or $m_e = 437$ nonlinear equality constraints. They are obtained by applying the five-star formula to approximate second partial derivatives.

Tables 5 to 8 contain numerical results first for standard tolerances and MODE = 0, where ACC is set to 10^{-7} in all cases. For the results of the remaining tables, We used MODE = 2, MODE = 7, and MODE = 20. MODE = 2 means that the scaled restart is applied at the very first step. Note that also for all other test cases with MODE > 0, the initial BFGS matrix is $B_0 = \gamma_0 I$.

problem	n m_e	n_{func}	n_{grad}	$f(x)$	$r(x)$	i_{fail}	
EX 1	722	361	64	64	0.45903100E-1	0.33E-10	0
EX 2	722	361	109	109	0.40390974E-1	0.22E-8	0
EX 3	722	361	88	88	0.11009561E+0	0.49E-9	0
EX 4	798	437	113	113	0.75833416E-1	0.12E-9	0
EX 5	798	437	200	200	0.51376012E-1	0.60E-5	1

Table 5: Test Results for Scaled Restarts: MODE = 0

problem	$n\,m_e$	n_{func}	n_{grad}	$f(x)$	$r(x)$	i_{fail}	
EX 1	722	361	64	64	0.45903100E-1	0.64E-11	0
EX 2	722	361	108	108	0.40390974E-1	0.21E-8	0
EX 3	722	361	75	75	0.11009568E+0	0.46E-9	0
EX 4	798	437	108	108	0.75833417E-1	0.63E-9	0
EX 5	798	437	200	200	0.51369466E-1	0.14E-6	1

Table 6: Test Results for Scaled Restarts: MODE = 2

problem	$n\,m_e$	n_{func}	n_{grad}	$f(x)$	$r(x)$	i_{fail}	
EX 1	722	361	20	20	0.45903160E-1	0.19E-7	0
EX 2	722	361	21	21	0.40390974E-1	0.79E-7	0
EX 3	722	361	41	41	0.11009561E+0	0.23E-9	0
EX 4	798	437	58	58	0.75833423E-1	0.44E-8	0
EX 5	798	437	112	112	0.51365403E-1	0.15E-12	0

Table 7: Test Results for Scaled Restarts: MODE = 7

problem	$n\,m_e$	n_{func}	n_{grad}	$f(x)$	$r(x)$	i_{fail}	
EX 1	722	361	50	50	0.45903100E-1	0.68E-8	0
EX 2	722	361	33	33	0.40390974E-1	0.22E-8	0
EX 3	722	361	36	36	0.11009561E+0	0.23E-7	0
EX 4	798	437	61	61	0.75833414E-1	0.17E-8	0
EX 5	798	437	75	75	0.51365398E-1	0.14E-7	0

Table 8: Test Results for Scaled Restarts: MODE = 20

The error codes are the same as produced by NLPQLP through the parameter IFAIL, i.e., IFAIL = 0 for successful termination and IFAIL = 1 for reaching the upper limit of 200 iterations.

We observe a significant speedup for scaled restarts every seven iterations. Initial scaling and more then 7 restarts do not yield any better results.

4 Program Documentation

NLPQLP is implemented in form of a Fortran subroutine. The quadratic programming problem is solved by the code QL, an implementation of the primal-dual method of Goldfarb and Idnani [17] going back to Powell [38], see also Schittkowski [53] for more details about implementation and usage. Model functions and gradients must be provided by reverse communication. The user has to evaluate function and gradient values in the same program that executes NLPQLP, according to the following rules:

1. Choose starting values for the variables to be optimised, and store them in the first column of an array called X.
2. Compute objective and all constraint function values, store them in XF(1) and the first column of G, respectively.
3. Compute gradients of objective function and all constraints, and store them in DF and DG, respectively. The J-th row of DG contains the gradient of the J-th constraint, $J = 1, \bullet \bullet \bullet, M$.
4. Set IFAIL = 0 and execute NLPQLP.
5. If NLPQLP returns with IFAIL = -1, compute objective and constraint function values for all variables found in the first L columns of X, store them in F (first L positions) and G (first L columns), and call NLPQLP again.
6. If NLPQLP terminates with IFAIL = -2, compute gradient values with respect to the variables stored in the first column of X, and store them in DF and DG. Only derivatives for active constraints, ACT(J) = .TRUE., need to be computed. Then call NLPQLP again.
7. If NLPQLP terminates with IFAIL = 0, the internal optimality criteria are satisfied. In the case of IFAIL > 0, an error has occurred.

If analytical derivatives are not available, simultaneous function calls can be used for gradient approximations, for example by forward differences 2N > L two-sided differences 4N > L ≥ 2N, or even higher order formulae L ≥ 4N.

Usage:

```
      CALL NLPQLP(     L,      M,     ME,   MMAX,      N,
     /            NMAX,   MNN2,      X,      F,      G,
     /              DF,     DG,      U,     XL,     XU,
     /               C,      D,    ACC,  ACCQP, STPMIN,
     /          MAXFUN,  MAXIT,  MAXNM,   RHOB, IPRINT,
     /            MODE,   IOUT,  IFAIL,     WA,    LWA,
     /             KWA,   LKWA,    ACT,   LACT,    LQL,
     /           QPSLVE,                               )
```

Definition of the parameters:

L :	Number of parallel systems, i.e., function calls during line search at predetermined iterates.
M :	Total number of constraints.
ME :	Number of equality constraints.
MMAX :	Row dimension of array DG containing Jacobian of constraints. MMAX must be at least one and greater or equal to M.
N :	Number of optimisation variables.
NMAX :	Row dimension of C. NMAX must be at least two and greater than N.
MNN2 :	Must be equal to M + N +N +2 when calling NLPQLP.
X(NMAX, L):	Initially, the first column of X has to contain starting values for the optimal solution. On return, X is replaced by the current iterate. In the driving program the row dimension of X has to be equal to NMAX. X is used internally to store L different arguments for which function values should be computed simultaneously.
F(L):	On return, F(1) contains the final objective function value. F is used also to store L different objective function values to be computed from L sets of arguments stored in X.
G(MMAX, L):	On return, the first column of G contains the constraint function values at the final iterate X. In the driving program, the row dimension of G has to be equal to MMAX. G is used internally to store L different sets of constraint function values to be computed from L sets of argumentsstored in X.
DF(NMAX):	DF contains the current gradient of the objective function. In case of numerical differentiation and a distributed system (L > 1) , it is recommended to apply parallel evaluations of F to compute DF.

U(MNN2): U contains the multipliers with respect to the actual iterate stored in the first column of X. The first M locations contain the multipliers of the M nonlinear constraints, the subsequent N locations the multipliers of the lower bounds, and the final N locations the multipliers of the upper bounds. At an optimal solution, all multipliers with respect to inequality constraints should be nonnegative

XL (N), XU (N): On input, the one-dimensional arrays XL and XU must contain the lower and upper bounds of the variables, respectively.

C(NMAX, NMAX): On return, C contains the last computed approximation of the Hessian matrix of the Lagrangian function. C is stored in form of an Cholesky decomposition, is LQL is set to false, see below. In this case, C contains the lower triangular factor of an LDL factorization of the final quasi-Newton matrix (without diagonal elements, which are always one). In the driving program, the row dimension of C has to be equal to NMAX.

D (NMAX): The elements of the diagonal matrix of the LDL decomposition of the quasi-Newton matrix are stored in the one-dimensional array D, if LQL is false.

ACC : The user has to specify the desired final accuracy (e.g. 1.0D-7). The termination accuracy should not be much smaller than the accuracy by which gradients are computed.

ACCQP : The tolerance is needed for the QP solver to perform several tests, for example whether optimality conditions are satisfied or whether a number is considered as zero or not. If ACCQP is less or equal to zero, then the machine precision is computed by NLPQLP and subsequently multiplied by 1.0D+4.

STPMIN : Minimum steplength in case of L > 1. Recommended is any value in the order of the accuracy by which functions are computed. The value is needed to compute a steplength reduction factor by $\text{STPMIN}**\left(1/\left(\text{L}-1\right)\right)$.

If STPMIN ≤ 0, then STPMIN = ACC is used.

MAXFUN : The integer variable defines an upper bound for the number of function calls during the line search (e.g., 20). MAXFUN is only needed in case of L = 1, and must not be greater than 50.

MAXIT : Maximum number of outer iterations, where one itera\tion corresponds to one formulation and solution of the quadratic programming subproblem, or, alternatively, one evaluation of gradients (e.g., 100).

MAXNM : Stack size for storing merit function values at previous iterations for non-monotone line search (e.g., 10). If MAXNM = 0, a monotone line search is performed. MAXNM should not be greater than 50.

RHOB : Parameter for performing a restart in case of IFAIL = 2 by setting the BFGS-update matrix to RHOB∗I, where I denotes the identity matrix. The number of restarts is bounded by MAXFUN. A value greater than one is recommended. (e.g., 100).

IPRINT : Specification of the desired output level.

0 - No output of the program.

1 - Only final convergence analysis.

2 - One line of intermediate results for each iteration.

3 - More detailed information for each iteration.

4 - More line search data is displayed.

Note that constraint and multiplier values are not displayed for N, M > 1000.

MODE : The parameter specifies the desired version of NLPQLP.

0 - Normal execution (reverse communication!).

1 - Initial guess for multipliers in U and Hessian of the Lagrangian function in C and D provided. In case of LQL = .TRUE., D is ignored. Otherwise, the lower part of C has to contain the lower triangular factor of an LDL decomposition and D the diagonal part.

2 - Initial scaling (Oren-Luenberger) after first step, BFGS updates started from multiple of identity matrix.

3 - Scaled restart, if scaling parameter is less than square root of ACC.

>3 - Initial and repeated scaling every MODE steps, reset of BFGS matrix to multiple of identity matrix.

IOUT : Integer indicating the desired output unit number, i.e., all write statements start with 'WRITE(IOUT, • • • '.

IFAIL : The parameter shows the reason for terminating a solution process. Initially, IFAIL must be set to zero. On return, IFAIL could contain the following values:

-2 - Compute new gradient values.

-1 - Compute new function values.

0 - Optimality conditions satisfied.

1 - Stop after MAXIT iterations

2 - Uphill search direction.

3 - Underflow when computing new BFGS-update matrix.

4 - Line search exceeded MAXFUN iterations.

5 - Length of a working array too short.

6 - False dimensions, if M > MMAX, N ≥ NMAX, or MNN2 ≠ M + N + N + 2.

7 - Search direction close to zero at infeasible iterate.

8 - Starting point violates lower or upper bound.

9 - Wrong input parameter, e.g., MODE, IPRINT, IOUT.

10 - Inconsistency in QP, division by zero.

>100 - Error message of QP solver.

WA(LWA): WA is a double precision working array of length LWA. On return, the first N positions contain the best feasible iterate obtained, WA(N+1) the corresponding objective function value, and the subsequent M positions the constraint values. If no intermediate feasible solution exists, WA(N+1) contains a large value, e.g., 1.0D+72.

LWA : Length of WA, has to be at least at least 23*N+4*M+3*MMAX+150.

NOTE: The standard QP-solver coming together with NLPQLP (QL). needs additional memory for 3*NMAX*NMAX/2+10*NMAX+MMAX+N+1 double precision numbers.

KWA (LKWA): KWA is an integer working array of length LKWA. On return, the first 5 positions contain the following information.

KWA(1) - Number of function evaluations.

KWA(2) - Number of gradient evaluations.

KWA(3) - Iteration count.

KWA(4) - Number of QP's solved.

KWA(5) - Flag for better feasible, but non-stationary iterate (=1) or not (=0), see below.

LKWA : Length of KWA, has to be at least at least 20.

NOTE: The standard QP-solver coming together with NLPQLP (QL) needs additional memory for N double precision numbers.

ACT(LACT K): The logical array indicates constraints, which NLPQLP considers to be active at the last computed iterate, i.e., G (J, 1) is active, if and only if ACT (J) is true for $J = 1, \bullet\bullet\bullet, M$.

LACT : Length of ACT, has to be at least 2*M + 10.

LQL : If LQL is set to true in the calling program, the quadratic programming problem is solved proceeding from a full positive definite quasi-Newton matrix. Otherwise, a Cholesky decomposition (LDL) is performed and updated internally, so that matrix C always consists of the lower triangular factor and D of the diagonal

QPSLVE : External subroutine to solve the quadratic programming subproblem. The calling sequence is

```
   CALL QPSLVE(     M,     ME,   MMAX,      N,  NMAX,
  /              MNN,      C      D,      A,     B,
  /               XL,     XU,     X,      U,   EPS,
  /             MODE,   IOUT,  IFAIL, IPRINT,   WAR,
  /             LWAR,   IWAR,  LIWAR                )
```

For more details about the choice and dimensions of arguments, see [53].

Some of the termination reasons depend on the accuracy used for approximating gradients. If we assume that all functions and gradients are computed within machine precision and that the implementation is correct, there remain only the following possibilities that could cause an error message:

1. The termination parameter ACC is too small, so that the numerical algorithm plays around with round-off errors without being able to improve the solution. Especially the Hessian approximation of the Lagrangian function becomes unstable in this case. A straightforward remedy is to restart the optimization cycle again with a larger stopping tolerance.

2. The constraints are contradicting, i.e., the set of feasible solutions is empty. There is no way to find out, whether nonlinear and nonconvex constraints are feasible or not. Thus, the nonlinear programming algorithms will proceed until running in any of the mentioned error situations. In this case, the correctness of the model must be carefully checked.

3. Constraints are feasible, but active constraints are degenerate, e.g., redundant. One should know that SQP algorithms assume the satisfaction of the so-called linear independency constraint qualification, i.e., that gradients of active constraints are linearly independent at each iterate and in a neighborhood of an optimal solution. In this situation, it is recommended to check the formulation of the model constraints.

However, some of the error situations also occur if, because of wrong or non-accurate gradients, the quadratic programming subproblem does not yield a descent direction for the underlying merit function. In this case, one should try to improve the accuracy of function evaluations, scale the model functions in a proper way, or start the algorithm from other initial values.

Since Version 2.1, NLPQLP returns the best iterate obtained. In case of successful termination (IFAIL = 0), this is always the last one. But it might be possible that in an exceptional situation, an intermediate iterate is feasible with a better objective function value than that one of the final iterate, but the KKT optimality conditions axe not satisfied. In this case, the better feasible solution is stored at the first n positions of the double precision working array and the corresponding objective function value at position $n+1$. Moreover, positions $n+2$ to $n+1+m$ contain the constraint values. Note that feasibility is tested by sum of constrained violations tested against ACC.

On successful return with IFAIL = 0, KWA(5) is set to zero. If, however, a better feasible objective function value has been found during the first five iterations, then KWA(5) is set to 1, the BFGS-update matrix C is set to ρI with $\rho < 1$, where I denotes the identity matrix. The corresponding formal argument of NLPQLP is called RHOB. Moreover, the multiplier approximation vector U is set to 0. Thus, an immediate restart under control of the user is possible with MODE = 1. Some information is printed on the standard IO channel in case of IPRINT > 0. For compatibility reasons with previous versions, RHOB replaces TOLNM and is set to zero for all input values less than one.

The QP solver is defined in form of an external subroutine to allow a replacement in case of exploiting special sparsity patterns. A typical example is the usage of NLPQLP for solving least squares problems, where artificially introduced equality constraints lead to a Jacobian which consist partially of the identity matrix, see Schittkowski [50, 51].

The internal scaling and restart option is borrowed from limited-memory quasi-Newton methods, see for example Liu and Nocedal [29]. If requested by the user, the quasi-Newton matrix is replaced by a scalar multiple of the identity matrix just before updating. Either an initial scaling or a reset of the whole matrix and computation of a new scaling parameter is performed depending on the input parameter MODE. A scaled restart is recommended, if, e.g., the convergence turns out to become extremely slow.

5 Examples

To give an example how to organize the code, we consider Rosenbrock's post office problem, i.e., test problem TP37 of Hock and Schittkowski [25].

$$x_1, x_2 \in IR: \quad \begin{cases} \min -x_1 x_2 x_3 \\ x_1 + 2x_2 + 2x_3 \geq 0 \\ 72 - x_1 - 2x_2 - 2x_3 \geq 0 \\ 0 \leq x_1 \leq 42 \\ 0 \leq x_2 \leq 42 \\ 0 \leq x_3 \leq 42 \end{cases} \tag{15}$$

NLPQLP comes with a couple of demo programs by which the following situations are to be illustrated:

File name	Comments
nlp_demoA.for	numerical differentiation and distributed function calls
nlp_demoB.for	numerical differentiation
nlp_demoC.for	numerical derivatives
nlp_demoD.for	warm and cold restarts
nlp_demoE.for	simultaneous function and gradient evaluation
nlp_demoF.for	active set strategy
nlp_demoG.for	active set strategy

A Fortran source code for a typical situation is listed below. Gradients are approximated by forward differences. The function block inserted in the main program can be replaced by a subroutine call. Also the gradient evaluation is easily exchanged by an analytical one or higher order derivatives

```
      IMPLICIT          NONE
      INTEGER           NMAX, MMAX, LMAX, MNN2X, LWA, LKWA, LACTIV
      PARAMETER (       NMAX = 4,
     /                  MMAX = 2,
     /                  LMAX = 10,
     /                  MNN2X = MMAX + NMAX + NMAX + 2,
     /                  LWA = 1.5*NMAX*NMAX + 33*NMAX + 9*MMAX + 200,
     /                  LKWA = NMAX + 10,
     /                  LACTIV = 2+MMAX + 10)
      INTEGER           KWA(LKWA), N, ME, M, L, MNN2, MAXIT, MAXFUN,
     /                  IPRINT, MAXNM, IOUT, MODE, IFAIL, I, J, K, NFUNC
      DOUBLE PRECISION  X(NMAX,LMAX), F(LMAX), G(MMAX,LMAX), DF(NMAX),
     /                  DG(MMAX,NMAX), U(MNN2X), XL(NMAX), XU(NMAX),
     /                  C(NMAX,NMAX), D(NMAX), WA(LWA), ACC, ACCQP,
```

```
      /                 STPMIN, EPS, EPSREL, FBCK, GBCK(MMAX), XBCK,
      /                 RHOB
       LOGICAL          ACTIVE(LACTIV), LQL
       EXTERNAL         QL
C
C  Set some constants and initial values
C
       IOUT    = 6
       ACC     = 1.0D-8
       ACCQP   = 1.0D-12
       STPMIN  = 1.0D-10
       EPS     = 1.0D-7
       MAXIT   = 100
       MAXFUN  = 10
       MAXNM   = 10
       RHOB    = O.ODO
       LQL     = .TRUE.
       IPRINT  = 2
       N       = 3
       L       = N
       M       = 2
       ME      = 0
       MNN2    = M + N + N + 2
       MODE    = 0
       IFAIL   = 0
       NFUNC   = 0
       DO I=1,N
          DO K=1,L
             K(I,K) = 10.0D0
          ENDDO
             XL(I) = 0.0D0
             XU(I) = 42.0D0
       ENDDO
      1 CONTINUE
C=====================================================================
C  This is the main block to compute all function value
C  simultaneously, assuming that there are L nodes.
C  The block is executed either for computing a steplength
C  or for approximating gradients by forward differences.
C
       DO K=1,L
          F(K)   = -X(1,K)*X(2,K)*X(3,K)
          G(1,K) = X(1,K) + 2.0D0*X(2,K) + 2.0D0*X(3,K)
          G(2,K) = 72.0D0 - X(1,K) - 2.0D0*X(2,K) - 2.0D0*X(3,K)
       ENDDO
C
C=====================================================================
       NFUNC = NFUNC + 1
       IF (IFAIL.EQ.-1) GOTO 4
       IF (NFUNC.GT.l) GOTO 3
      2 CONTINUE
       FBCK = F(l)
       DO J=1,M
          GBCK(J) = G(J,1)
       ENDDO
       XBCK = X(l,l)
       DO I=1,N
          EPSREL = EPS*DMAX1(1.0D0,DABS(X(I,1)))
          DO K=2,L
             X(I,K) = X(I,1)
          ENDDO
          X(I,I) = X(I,1) + EPSREL
       ENDDO
```

```
        GOTO 1
       3 CONTINUE

        X(1,1) = XBCK
        DO 1=1,N
           EPSREL = EPS*DMAX1(1.0D0,DABS(X(I,1)))
           DF(I) = (F(I) - FBCK)/EPSREL
           DO J=1,M
              DG(J,I) = (G(J,I) - GBCK(J))/EPSREL
           ENDDO
        ENDDO
        F(1) = FBCK
        DO J=1,M
           G(J,1) = GBCK(J)
        ENDDO
C
       4 CONTINUE
        CALL NLPQLP (      L,        M,      ME,     MMAX,        N,
       /                 NMAX,     MNN2,       X,        F,        G,
       /                   DF,       DG,       U,       XL,       XU,
       /                    C,        D,     ACC,    ACCQP,   STPMIN,
       /               MAXFUN,    MAXIT,   MAXNM,     RHOB,   IPRINT,
       /                 MODE,     IOUT,   IFAIL,       WA,      LWA,
       /                  KWA,     LKWA,  ACTIVE,   LACTIV,      LQL,
       /                   QL)
        IF (IFAIL.EQ.-1) GOTO 1
        IF (IFAIL.EQ.-2) GOTO 2
C
        WRITE(IOUT,1000) NFUNC
   1000 FORMAT(' *** Number of function calls: ',13)
C
        STOP
        END
```

The following output should appear on screen:

```
-------------------------------------------------------------------
START OF THE SWQUENTIAL QUADRATIC PROGRAMMING ALGORITHM
-------------------------------------------------------------------


Parameters:
   N       =        3
   M       =        2
   ME      =        0
   MODE    =        0
   ACC     =   0.1000D-07
   ACCQP   =   0.1000D-11
   STPMIN  =   0.1000D-09
   MAXFUN  =        3
   MAXNM   =       10
   MAXIT   =      100
   IPRINT  =        2

Output in the following order:
   IT    - iteration number
   F     - objectivefunction value
   SCV   - sum of constraint violations
   NA    - number ofactive constraints
   I     - number of line search iterations
   ALPHA - steplength parameter
   DELTA - additional variable to prevent inconsistency
   KKT   - Karush-Kuhn-Tucker optimality criterion
```

```
  IT         F           SCV     NA  I    ALPHA     DELTA      KKT
 ------------------------------------------------------------------------------
   1 -0.10000000D+04  0.00D+00    2  0  0.00D+00  0.00D+00  0.44D+04
   2 -0.23625000D+04  0.64D-07    1  1  0.10D+01  0.00D+00  0.11D+04
   3 -0.32507304D+04  0.11D-13    1  1  0.10D+01  0.00D+00  0.69D+03
   4 -0.33041403D+04  0.11D-13    1  1  0.10D+01  0.00D+00  0.36D+03
   5 -0.34527380D+04  0.11D-13    1  1  0.10D+01  0.00D+00  0.58D+01
   6 -0.34559625D+04  0.00D+00    1  1  0.10D+01  0.00D+00  0.10D+00
   7 -0.34559625D+04  0.00D+00    1  2  0.10D-04  0.00D+00  0.23D+00
   8 -0.34559625D+04  0.00D+00    1  2  0.10D-04  0.00D+00  0.76D-01
   9 -0.34560000D+04  0.17D-10    1  1  0.10D+01  0.00D+00  0.24D-04
  10 -0.34560000D+04  0.48D-12    1  1  0.10D+01  0.00D+00  0.20D-07
  11 -0.34560000D+04  0.11D-13    1  1  0.10D+01  0.00D+00  0.25D-11

--- Final Convergence Analysis at Last Iterate ---

      Best result at iteration:          ITER =       11
      Objective function value:          F(X) =  -0.34560000D+04
      Solution values:                   X    =
         0.24000000D+02  0.12000000D+02     0.12000000D+02
      Distances from lower bounds:       X-XL =
         0.24000000D+02  0.12000000D+02     0.12000000D+02
      Distances from upper bounds:       XU-X =
         0.18000000D+02  0.30000000D+02     0.30000000D+02
      Multipliers for lower bounds:      U    =
         0.00000000D+00  0.00000000D+00     0.00000000D+00
      Multipliers for upper bounds:      U    =
         0.00000000D+00  0.00000000D+00     0.00000000D+00
      Constraint values:                 G(X) =
         0.72000000D+02 -0.10658141D-13
      Multipliers for constraints:       U    =
         0.00000000D+00  0.14400000D+03
      Number of function calls:          NFUNC =       11
      Number of gradient calls:          NFUNC =       11
      Number of calls of QP solver:      NFUNC =       11

   *** Number of function calls:  22
```

In case of $L = 1$ and analytical derivative computations, the corresponding serial implementation of the main program is as follows:

```
      IMPLICIT          NONE
      INTEGER           NMAX, MMAX, MNN2X, LWA, LKWA, LACTIV
      PARAMETER (       NMAX = 4,
     /                  MMAX = 2,
     /                  MNN2X = MMAX + NMAX + NMAX + 2,
     /                  LWA = 1.5*NMAX*NMAX + 33*NMAX + 9*MMAX + 200,
     /                  LKWA = NMAX + 10,
     /                  LACTIV = 2+MMAX + 10)
      INTEGER           KWA(LKWA), N, ME, M, L, MNN2, MAXIT, MAXFUN,
     /                  IPRINT, MAXNM, IOUT, MODE, IFAIL, I, J, NFUNC
      DOUBLE PRECISION  X(NMAX), F, G(MMAX), DF(NMAX), DG(MMAX,NMAX),
     /                  U(MNN2X), XL(NMAX), XU(NMAX), C(NMAX,NMAX),
     /                  D(NMAX), WA(LWA), ACC, ACCQP, STPMIN, RHOB
      LOGICAL           ACTIVE(LACTIV), LQL
      EXTERNAL          QL
C
C  Set some constants and initial values
C
      IOUT    = 6
      ACC     = 1.0D-10
      ACCQP   = 1.0D-12
      STPMIN  = 0.0
      EPS     = 1.0D-7
      MAXIT   = 100
      MAXFUN  = 10
```

```
         MAXNM    = 0
         RHOB     = O.ODO
         LQL      = .TRUE.
         IPRINT   = 2
         N        = 3
         M        = 2
         ME       = 0
         MNN2     = M + N + N + 2
         MODE     = 0
         IFAIL    = 0
         NFUNC    = 0
         DO I=1,N
            X(I)  = 10.0D0
            XL(I) = 0.0D0
            XU(I) = 42.0D0
         ENDDO
       1 CONTINUE
C=======================================================================
C  This block computes all function values.
C
         F    = -X(1)*X(2)*X(3)
         G(1) =  X(1) + 2.0D0*X(2) + 2.0D0*X(3)
         G(2) = 72.0D0 - X(1) - 2.0D0*X(2) - 2.0D0*X(3)
C
C=======================================================================
         NFUNC = NFUNC + 1
         IF (IFAIL.EQ.-1) GOTO 4
       2 CONTINUE
C=======================================================================
C  This block computes all derivative values.
C
         DF(1)   = -X(2)*X(3)
         DF(2)   = -X(1)*X(3)
         DF(3)   = -X(1)*X(2)
         DG(1,1) =  1.0D0
         DG(1,2) =  2.0D0
         DG(1,3) =  2.0D0
         DG(2,1) = -1.0D0
         DG(2,2) = -2.0D0
         DG(2,3) = -2.0D0
C
C=======================================================================
       4 CONTINUE
         CALL NLPQLP (      L,        M,      ME,     MMAX,        N,
        /                NMAX,     MNN2,       X,        F,        G,
        /                  DF,       DG,       U,       XL,       XU,
        /                   C,        D,     ACC,    ACCQP,   STPMIN,
        /              MAXFUN,    MAXIT,   MAXNM,     RHOB,   IPRINT,
        /                MODE,     IOUT,   IFAIL,       WA,      LWA,
        /                 KWA,     LKWA,  ACTIVE,   LACTIV,      LQL,
        /                  QL)
         IF (IFAIL.EQ.-1) GOTO 1
         IF (IFAIL.EQ.-2) GOTO 2
C
         WRITE(IOUT,1000) NFUNC
    1000 FORMAT(' *** Number of function calls: ',13)
C
         STOP
         END
```

NLPQLP displays the following output:

```
------------------------------------------------------------------
START OF THE SWQUENTIAL QUADRATIC PROGRAMMING ALGORITHM
------------------------------------------------------------------


Parameters:
   N       =        3
   M       =        2
   ME      =        0
   MODE    =        0
   ACC     =   0.1000D-09
   ACCQP   =   0.1000D-11
   STPMIN  =   0.1000D-09
   MAXFUN  =       10
   MAXNM   =        0
   MAXIT   =      100
   IPRINT  =        2

Output in the following order:
   IT    - iteration number
   F     - objectivefunction value
   SCV   - sum of constraint violations
   NA    - number ofactive constraints
   I     - number of line search iterations
   ALPHA - steplength parameter
   DELTA - additional variable to prevent inconsistency
   KKT   - Karush-Kuhn-Tucker optimality criterion

  IT         F           SCV     NA  I    ALPHA     DELTA      KKT
-------------------------------------------------------------------------------
   1 -0.10000000D+04  0.00D+00    2  0  0.00D+00  0.00D+00  0.44D+04
   2 -0.23625000D+04  0.00D+00    1  1  0.10D+01  0.00D+00  0.11D+04
   3 -0.32507304D+04  0.36D-14    1  1  0.10D+01  0.00D+00  0.69D+03
   4 -0.33041403D+04  0.00D+00    1  1  0.10D+01  0.00D+00  0.36D+03
   5 -0.34527380D+04  0.00D+00    1  1  0.10D+01  0.00D+00  0.58D+01
   6 -0.34559629D+04  0.00D+00    1  1  0.10D+01  0.00D+00  0.76D-01
   7 -0.34560000D+04  0.00D+00    1  1  0.10D+01  0.00D+00  0.25D-04
   8 -0.34560000D+04  0.36D-14    1  1  0.10D+01  0.00D+00  0.90D-10

--- Final Convergence Analysis at Last Iterate ---

      Objective function value:         F(X) =  -0.34560000D+04
      Solution values:                  X    =
         0.24000003D+02  0.11999999D+02    0.11999999D+02
      Distances from lower bounds:      X-XL =
         0.24000003D+02  0.11999999D+02    0.11999999D+02
      Distances from upper bounds:      XU-X =
         0.17999997D+02  0.30000001D+02    0.30000001D+02
      Multipliers for lower bounds:     U    =
         0.00000000D+00  0.00000000D+00    0.00000000D+00
      Multipliers for upper bounds:     U    =
         0.00000000D+00  0.00000000D+00    0.00000000D+00
      Constraint values:                G(X) =
         0.72000000D+02 -0.35527137D-14
      Multipliers for constraints:      U    =
         0.00000000D+00  0.14400000D+03
      Number of function calls:         NFUNC =       8
      Number of gradient calls:         NFUNC =       8
      Number of calls of QP solver:     NFUNC =       8

   *** Number of function calls:   8
```

6 Conclusions

We present a modification of an SQP algorithm designed for execution under a parallel computing environment (SPMD) and where a non-monotone line search is applied in error situations. Numerical results indicate stability and robustness for a set of 306 standard test problems. It is shown that not more than 7 parallel function evaluations per iterations are required for performing a sufficiently accurate line search. Significant performance improvement is achieved by the non-monotone line search especially in case of noisy function values and numerical differentiation, and by restarts in a severe error situation. With the new version of NLPQLP, we are able to solve about 90% of a standard set of 306 test examples subject to an termination accuracy of 10^{-7} in case of extremely noisy function values with relative accuracy of 1% and numerical differentiation. In the worst case, at most one digit of a partial derivative value is correct.

References

[1] Armijo L. (1966): Minimization of functions having Lipschitz continuous first partial derivatives, Pacific Journal of Mathematics, Vol. 16, 1-3.

[2] Barzilai J., Borwein J.M. (1988): Two-point stepsize gradient methods, IMA Journal of Numerical Analysis, Vol. 8, 141-148.

[3] Boderke P., Schittkowski K., Wolf M., Merkle H.P. (2000): Modeling of diffusion and concurrent metabolism in cutaneous tissue, Journal on Theoretical Biology, Vol. 204, No. 3, 393-407.

[4] Bonnans J.F., Panier E., Tits A., Zhou J.L. (1992): *Avoiding the Maratos effect by means of a nonmonotone line search, II: Inequality constrained problems – feasible iterates,* SIAM Journal on Numerical Analysis, Vol. 29, 1187-1202.

[5] Birk J., Liepelt M., Schittkowski K., Vogel F. (1999): *Computation of optimal feed rates and operation intervals for tubular reactors,* Journal of Process Control, Vol. 9, 325-336.

[6] Blatt M., Schittkowski K. (1998): *Optimal Control of One-Dimensional Partial Differential Equations Applied to Transdermal Diffusion of Substrates,* in: Optimization Techniques and Applications, L. Caccetta, K.L. Teo, P.F. Siew, Y.H. Leung, L.S. Jennings, V. Rehbock ed., School of Mathematics and Statistics, Curtin University of Technology, Perth, Australia, Vol. 1, 81-93.

[7] Bongartz I., Conn A.R., Gould N., Toint Ph. (1995): *CUTE: Constrained and unconstrained testing environment,* Transactions on Mathematical Software, Vol. 21, No. 1, 123-160.

[8] Bünner M.J., Schittkowski K., van de Braak G. (2004): *Optimal design of electronic components by mixed-integer nonlinear programming,* Optimization and Engineering, Vol. 5, 271-294.

[9] Dai Y.H., Liao L.Z. (1999): *R-Linear Convergence of the Barzilai and Borwein Gradient Method,* Research Reoport 99-039, Institute of Computational Mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences.

[10] Dai Y.H. (2000): *A nonmonotone conjugate gradient algorithm for unconstrained optimization,* Research Report, Institute of Computational Mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences.

[11] Dai Y.H. (2002): *On the nonmonotone line search*, Journal of Optimization Theory and Applications, Vol. 112, No. 2, 315-330.

[12] Dai Y.H., Schittkowski K. (2008): *A sequential quadratic programming algorithm with non-monotone line search,* Pacific Journal of Optimization, Vol. 4, 335-351.

[13] Deng N.Y., Xiao Y., Zhou F.J. (1993): *Nonmonotonic trust-region algorithm,* Journal of Optimization Theory and Applications, Vol. 26, 259-285.

[14] Edgar T.F., Himmelblau D.M. (1988): *Optimization of Chemical Processes,* Mc-Graw Hill.

[15] Frias J.M., Oliveira J.C, Schittkowski K. (2001): *Modelling of maltodextrin DE12 drying process in a convection oven,* Applied Mathematical Modelling, Vol. 24, 449-462.

[16] Geist A., Beguelin A., Dongarra J.J., Jiang W., Manchek R., Sunderam V. (1995): *PVM 3.0. A User's Guide and Tutorial for Networked Parallel Computing,* The MIT Press.

[17] Goldfarb D., Idnani A. (1983): *A numerically stable method for solving strictly convex quadratic programs,* Mathematical Programming, Vol. 27, 1-33.

[18] Grippo L., Lampariello F., Lucidi S. (1986): *A nonmonotone line search technique for Newtons's method,* SIAM Journal on Numerical Analysis, Vol. 23, 707-716.

[19] Grippo L., Lampariello F., Lucidi S. (1989): *A truncated Newton method with nonmonotone line search for unconstrained optimization,* Journal of Optimization Theory and Applications, Vol. 60, 401-419.

[20] Grippo L., Lampariello F., Lucidi S. (1991): *A class of nonmonotone stabilization methods in unconstrained optimization,* Numerische Mathematik, Vol. 59, 779-805.

[21] Han S.-P. (1976): *Superlinearly convergent variable metric algorithms for general nonlinear programming problems,* Mathematical Programming, Vol. 11, 263-282.

[22] Han S.-P. (1977): *A globally convergent method for nonlinear programming*, Journal of Optimization Theory and Applications, Vol. 22, 297 309.

[23] Hartwanger C., Schittkowski K., Wolf H. (2000): *Computer aided optimal design of horn radiators for satellite communication,* Engineering Optimization, Vol. 33, 221-244.

[24] Hock W., Schittkowski K. (1981): *Test Examples for Nonlinear Programming Codes,* Lecture Notes in Economics and Mathematical Systems, Vol. 187, Springer.

[25] Hock W., Schittkowski K. (1983): *A comparative performance evaluation of 21 nonlinear programming codes,* Computing, Vol. 30, 335-358.

[26] Ke X., Han J. (1995): *A nonmonotone trust region algorithm for equality constrained optimization,* Science in China, Vol. 38A, 683-695.

[27] Ke X., Liu G., Xu D. (1996): *A nonmonotone trust-region algorithm for unconstrained optimization,* Chinese Science Bulletin, Vol. 41, 197-201.

[28] Kneppe G., Krammer J., Winkler E. (1987): *Structural optimization of large scale problems using MBB-LAGRANGE,* Report MBB-S-PUB-305, Messerschmitt-Bölkow-Blohm, Munich.

[29] Liu D.C., Nocedal J. (1989): *On the limited memory BFGS method for large scale optimization,* Mathematical Programming, Vol. 45, 503-528.

[30] Lucidi S., Rochetich F, Roma M. (1998): *Curvilinear stabilization techniques for truncated Newton methods in large-scale unconstrained optimization,* SIAM Journal on Optimization, Vol. 8, 916-939.

[31] Maurer H., Mittelmann H.D. (2000): *Optimization techniques for solving elliptic control problems with control and state constraints: Part 1. Boundary control,* Computational Optimization and Applications, Vol. 16, 29-55.

[32] Maurer H., Mittelmann H.D. (2001): *Optimization techniques for solving elliptic control problems with control and state constraints. Part 2: Distributed control,* Computational Optimization and Applications, Vol. 18, 141-160.

[33] Ortega J.M., Rheinbold W.C. (1970): *Iterative Solution of Nonlinear Equations in Several Variables, Academic Press,* New York-San Francisco-London.

[34] Panier E., Tits A. (1991): *Avoiding the Maratos effect by means of a nonmonotone line search, I: General constrained problems,* SIAM Journal on Numerical Analysis, Vol. 28, 1183-1195.

[35] Papalambros P.Y., Wilde D.J. (1988): *Principles of Optimal Design,* Cambridge University Press.

[36] Powell M.J.D. (1978): *A fast algorithm for nonlinearly constraint optimization calculations*, in: Numerical Analysis, G.A. Watson ed., Lecture Notes in Mathematics, Vol. 630, Springer.

[37] Powell M.J.D. (1978): *The convergence of variable metric methods for nonlinearly constrained optimization calculations,* in: Nonlinear Programming 3, O.L. Mangasarian, R.R. Meyer, S.M. Robinson eds., Academic Press.

[38] Powell M.J.D. (1983): *On the quadratic programming algorithm of Goldfarb and Idnani.* Report DAMTP 1983/Na 19, University of Cambridge, Cambridge.

[39] Raydan M. (1997): *The Barzilai and Borwein gradient method for the large-scale unconstrained minimization problem,* SIAM Journal on Optimization, Vol. 7, 26-33.

[40] Schittkowski K. (1980): *Nonlinear Programming Codes,* Lecture Notes in Economics and Mathematical Systems, Vol. 183 Springer.

[41] Schittkowski K. (1981): *The nonlinear programming method of Wilson, Han and Powell. Part 1: Convergence analysis,* Numerische Mathematik, Vol. 38, 83-114.

[42] Schittkowski K. (1981): *The nonlinear programming method of Wilson, Han and Powell. Part 2: An efficient implementation with linear least squares subproblems,* Numerische Mathematik, Vol. 38, 115-127.

[43] Schittkowski K. (1982): *Nonlinear programming methods with linear least squares subproblems,* in: Evaluating Mathematical Programming Techniques, J.M. Mulvey ed., Lecture Notes in Economics and Mathematical Systems, Vol. 199, Springer.

[44] Schittkowski K. (1983): *Theory, implementation and test of a nonlinear programming algorithm,* in: Optimization Methods in Structural Design, H. Eschenauer, N. Olhoff eds., Wissenschaftsverlag.

[45] Schittkowski K. (1983): *On the convergence of a sequential quadratic programming method with an augmented Lagrangian search direction,* Mathematische Operations-forschung und Statistik, Series Optimization, Vol. 14, 197-216.

[46] Schittkowski K. (1985): *On the global convergence of nonlinear programming algorithms,* ASME Journal of Mechanics, Transmissions, and Automation in Design, Vol. 107, 454-458.

[47] Schittkowski K. (1985/86): NLPQL: *A Fortran subroutine solving constrained nonlinear programming problems,* Annals of Operations Research, Vol. 5, 485-500.

[48] Schittkowski K. (1987): *More Test Examples for Nonlinear Programming,* Lecture Notes in Economics and Mathematical Systems, Vol. 182, Springer.

[49] Schittkowski K. (1988): *Solving nonlinear least squares problems by a general purpose SQP-method,* in: Trends in Mathematical Optimization, K. H. Hoffmann, J.- B. Hiriart-Urruty, C. Lemarechal, J. Zowe eds., International Series of Numerical Mathematics, Vol. 84, Birkhäuser, 295-309.

[50] Schittkowski K. (1994): *Parameter estimation in systems of nonlinear equations,* Numerische Mathematik, Vol. 68, 129-142.

[51] Schittkowski K. (2002): *Numerical Data Fitting in Dynamical Systems,* Kluwer Academic Publishers, Dordrecht.

[52] Schittkowski K. (2002): *EASY-FIT: A software system for data fitting in dynamic systems,* Structural and Multidisciplinary Optimization, Vol. 23, No. 2, 153-169.

[53] Schittkowski K. (2003): QL: *A Fortran code for convex quadratic programming - user's guide,* Report, Department of Mathematics, University of Bayreuth, 2003.

[54] Schittkowski K. (2008): *An active set strategy for solving optimization problems with up to 200,000,000 nonlinear constraints,* Applied Numerical Mathematics, Vol. 59, 2999-3007.

[55] Schittkowski K. (2008): *An updated set of 306 test problems for nonlinear programming with validated optimal solutions - user's guide,* Report, Department of Computer Science, University of Bayreuth.

[56] Schittkowski K., Zillober C., Zotemantel R. (1994): *Numerical comparison of nonlinear programming algorithms for structural optimization,* Structural Optimization, Vol. 7, No. 1, 1-28.

[57] Spellucci P (1993): *Numerische Verfahren der nichtlinearen Optimierung,* Birkhäuser.

[58] Stoer J. (1985): *Foundations of recursive quadratic programming methods for solving nonlinear programs,* in: Computational Mathematical Programming, K. Schittkowski, ed., NATO ASI Series, Series F: Computer and Systems Sciences, Vol. 15, Springer.

[59] Toint P.L. (1996): *An assessment of nonmontone line search techniques for unconstrained optimization*, SIAM Journal on Scientific Computing, Vol. 17, 725-739.

[60] Toint P.L. (1997): *A nonmonotone trust-region algorithm for nonlinear optimization subject to convex constraints,* Mathematical Programming, Vol. 77, 69-94.

[61] Wolfe P. (1969*): Convergence conditions for ascent methods,* SIAM Review, Vol. 11, 226-235.

[62] Zhou J.L., Tits A. (1993): *Nonmonotone line search for minimax problems,* Journal of Optimization Theory and Applications, Vol. 76, 455-476.

Appendix B

The Regulator Problem

Reference: Leyland, J. A.: Use of the NLPQLP Sequential Programming Algorithm to Solve Rotorcraft Aeromechanical Constrained Optimisation Problems, Volume 1: Theory and Methodology, Section 2.4, NASA TM-2014-216632, Jan. 2014.

The Regulator Problem

Unlike the non-linear programming problems that are constrained/unconstrained optimisation problems, the regulator problem is a steady-state problem; its solution process seeks maintenance of a steady-state condition with minimal control and, in some cases, minimal control rate of change. Because the regulator problem solution is analytically explicit and known, its solution process is fast and has a relatively small computational load compared to the constrained/unconstrained optimisation solution processes. In some cases, "External Limiting" constraints (i.e., direct maximum limits on control vector elements imposed *after* the analytic explicit solution is obtained) are imposed on control vector elements. Correspondingly, early attempts (i.e., circa the 1950s) to solve constrained/ unconstrained optimisation control problems formulated these problems as regulator problems because the algorithms for constrained/unconstrained optimisation solution processes and computer technology were in their early stages of development and not sufficiently reliable or efficient for this purpose. As computer technology advances occurred and efficient, reliable optimisation techniques were developed, use of numerical optimisation techniques became feasible for actual test applications.

A control vector metric, and a rate of change of the control vector metric if required, are adjoined to a steady-state excursion metric to form the performance index for the regulator problem. By appropriately defining the steady-state excursion metric, which is the first term in this performance index, and carefully tuning the weighting coefficients for all the terms in the performance index, a pseudo-optimal solution can be obtained that satisfies, or nearly satisfies, any required constraints. This tuning must, of course, be accomplished before actual test applications.

As in the case of the General T-Matrix NLP Control Problem described in section 2.2, a T-Matrix linear plant model that relates the measurement $Z-\text{vector}$ to the control $\theta-\text{vector}$ is assumed for the regulator problem described below. The first-term performance index is the steady-state excursion metric and a quadratic function of a T-Matrix plant model, and correspondingly a quadratic function of the control $\theta-\text{vector}$. The second term in the performance index is simply a weighted control $\theta-\text{vector}$ quadratic. If required, the third term in the performance index is a weighted time rate of change of the control $\theta-\text{vector}$ quadratic. The regulator problem is:

Determine the $\theta-$*vector,* θ_{Sol}*, which solves the problem :*

Minimise $\underset{\theta_p \in \theta}{}$ $\quad J = Z^{\mathsf{T}} W_Z Z + \theta^{\mathsf{T}} W_\theta \theta + \dot{\theta}^{\mathsf{T}} W_{\dot{\theta}} \dot{\theta} \qquad$ *for* $\quad p \in I_\theta$

where $\quad Z = Z(\theta) = Z_{\mathsf{A}} + \mathsf{T}(\theta - \theta_0)$

$$Z = Z(\theta) = \left[\bullet \ \bullet \ \bullet \ \left\{ Z_q \,\middle|\, q \in I_Z \right\} \bullet \ \bullet \ \bullet \right]^{\mathsf{T}}$$

and $\quad \theta = \left[\bullet \ \bullet \ \bullet \ \left\{ \theta_p \,\middle|\, p \in I_\theta \right\} \bullet \ \bullet \ \bullet \right]^{\mathsf{T}}$

$$\dot{\theta} = \left[\bullet \ \bullet \ \bullet \ \left\{ \dot{\theta}_p \,\middle|\, p \in I_\theta \right\} \ \bullet \ \bullet \ \bullet \right]^{\mathsf{T}}$$

$$I_Z = \left\{ \bullet \ \bullet \ \bullet \ \left\{ \forall \ q \ \ni \ Z_q \in Z \right\} \bullet \ \bullet \ \bullet \right\}$$

$$I_\theta = \left\{ \bullet \ \bullet \ \bullet \ \left\{ \forall \ p \ \ni \ \theta_p \in \theta \right\} \bullet \ \bullet \ \bullet \right\}$$

Subject to NO constraints per se during the regulator problem solution process to determine θ_{Sol}*.*

The Explicit Solution $\theta_{Sol}-$ *vector is* :

$$\theta_{Sol} = \left(D\, W_{\dot{\theta}} + D\, \mathsf{T}^{\mathsf{T}} W_Z \mathsf{T} \right) \theta_0 - \alpha D\, \mathsf{T}^{\mathsf{T}} W_Z Z_{\mathsf{A}}$$

where $\quad D = \left(\mathsf{T}^{\mathsf{T}} W_Z \mathsf{T} + W_{\dot{\theta}} + W_\theta \right)^{-1}$

and $\quad \alpha \in [0, 1]$

In some cases, "External Limiting" constraints are imposed on θ_{Sol} after the explicit analytic solution for θ_{Sol} is determined. Specifically:

$$\left.\begin{array}{l} \theta_{\mathrm{MIN}_p} \le \theta_{Solp} \le \theta_{\mathrm{MAX}_p} \\ \theta_{\mathrm{MIN}_p} \in (-\infty, +\infty) \\ \theta_{\mathrm{MAX}_p} \in (-\infty, +\infty) \end{array}\right\} \left\{\begin{array}{l} \text{Direct Constraints on the Control} \\ \theta_{Sol}-\text{vector Elements } \theta_{Solp} \text{ for: } p \in I_\theta \end{array}\right.$$

if $\theta_{Sol_p} \le \theta_{\mathrm{MIN}_p}$ *then set* $\theta_{Sol_p} = \theta_{\mathrm{MIN}_p}$

if $\theta_{Sol_p} \ge \theta_{\mathrm{MAX}_p}$ *then set* $\theta_{Sol_p} = \theta_{\mathrm{MIN}_p}$

otherwise there is no change to the value of θ_{Sol_p} *as determined by the equation for the explicit solution for* θ_{Sol_p}.

then

$$\theta_{Sol} = \left[\bullet \ \bullet \ \bullet \ \left\{ \theta_{Sol_p} \,\middle|\, p \in I_\theta \right\} \bullet \ \bullet \ \bullet \right]^{\mathrm{T}}.$$

Appendix C

The DCL Command File Code for Cases Run on the Hewlett-Packard Alpha Mainframe Computer

```
$ ASSIGN  SYS$COMMAND:  SYS$INPUT
$ ASSIGN  SYS$INPUT     FOR005
$ ASSIGN  SYS$OUTPUT    FOR006
$        @SYS$LIBRARY:CXML$SET_LIB VAX
$       SET TERM/WIDTH=80
$       SET VERIFY
$       SET NOVERIFY
$ !
$ ! *****   NLP10x10 COMMAND PROCEDURE:    NLP10x10.COM      *****
$ !
$ !     ON WARNING THEN GOTO _______
$ !     ON ERROR THEN GOTO _______
$ !     ON SEVERE THEN GOTO _______
$ !
$ START:
$ !
$ ! *****   Determine if a NLP10x10 Case is to be RUN   *****
$ !
$ RUN0:
$       INQUIRE RUN00 "RUN a NLP10x10 Case?  (Y/N)"
$       IF RUN00 .EQS. "N" THEN GOTO TERM1
$ !
$ RUN1:
$ !
$       INQUIRE RUNDEMO "Enter NAME of the NLP10x10 System to be RUN"
$ !
$ ! *****   RUN this NLP10x10 Case   *****
$ !
$       ASSIGN  CDATA.DAT SYS$INPUT
$       ASSIGN  EDATA.DAT SYS$OUTPUT
$ !
$       ON ERROR THEN GOTO RUN2
$       COPY CDATA.DAT FOR005.DAT
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT "   *********   INPUT    *****    INPUT   ********   "
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT " "
$       TYPE FOR005.DAT
$ !
$ RUN2:
$ !
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT "   *********   OUTPUT   *****   OUTPUT   ********   "
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT "RUN the NLP10x10 Case."
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT "START RUN."
$       WRITE SYS$OUTPUT " "
$ !
$       SET TERM/WIDTH=132
$ !
$ ! *****   Execute  OPTIMYY   *****
$ !
$       ON ERROR THEN GOTO RUN4
$       RUN 'RUNDEMO'
$ !
$       SET TERM/WIDTH=80
$       GOTO RUN5
$ !
$ RUN4:
$       SET TERM/WIDTH=80
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT "ERROR in Running the NLP10x10 Case."
$       WRITE SYS$OUTPUT " "
$ !
$ RUN5:
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT "END of RUN."
$       WRITE SYS$OUTPUT " "
```

```
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT "   ***********  END  **********   END   *********   "
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT " "
$ !
$       ON ERROR THEN GOTO TERM0
$       DEASSIGN SYS$OUTPUT
$       INQUIRE DSPL00 "Display the INPUT  CDATA.DAT  on screen?  (Y/N)"
$       IF DSPL00 .EQS. "N" THEN GOTO DSPL1
$       SET TERM/WIDTH=132
$       TYPE CDATA.DAT
$       SET TERM/WIDTH=80
$ DSPL1:
$       INQUIRE DSPL01 "Display the OUTPUT  EDATA.DAT  on screen?  (Y/N)"
$       IF DSPL01 .EQS. "N" THEN GOTO TERM0
$       SET TERM/WIDTH=132
$       TYPE EDATA.DAT
$       SET TERM/WIDTH=80
$       GOTO TERM0
$ !
$ ! *****   Determine if a NLP10x10 System is to be LINKed   *****
$ !
$ LINK0:
$       INQUIRE LINK00 "LINK the NLP10x10 System?  (Y/N)"
$       IF LINK00 .EQS. "N" THEN GOTO TERM2
$ !
$ LINK1:
$ !
$       INQUIRE LINKDEMO "Enter NAME of the NLP10x10 System to be LINKed"
$ !
$ ! *****   LINK the NLP10x10 System   *****
$ !
$       INQUIRE LINK0L "LINK with the IMSL Static Library? (Y/N)"
$       IF LINK0L .EQS. "N" THEN GOTO LINK3
$ !
$       INQUIRE LINK01 "LINK with /MAP/CROSS_REFERENCE Qualifiers? (Y/N)"
$       IF LINK01 .EQS. "N" THEN GOTO LINK2
$ !
$ ! *****   LINK Code with the IMSL Static Library and the
$ !                                   /MAP/CROSS_REFERENCE Qualifiers   *****
$ !
$       ON ERROR THEN GOTO LINK5
$       LINK/MAP/CROSS_REFERENCE    'LINKDEMO',       LIBA/LIBRARY,    -
                                     LIBD/LIBRARY,    LIBF/LIBRARY,    -
                    LINK_F90_STATIC_GFLOAT/OPT
$       GOTO LINK6
$ !
$ LINK2:
$ !
$ ! *****   LINK Code with the IMSL Static Library with NO
$ !                                   /MAP/CROSS_REFERENCE Qualifiers   *****
$ !
$       ON ERROR THEN GOTO LINK5
$       LINK                        'LINKDEMO',       LIBA/LIBRARY,    -
                                     LIBD/LIBRARY,    LIBF/LIBRARY,    -
                    LINK_F90_STATIC_GFLOAT/OPT
$       GOTO LINK6
$ !
$ LINK3:
$ !
$       INQUIRE LINK03 "LINK with /MAP/CROSS_REFERENCE Qualifiers? (Y/N)"
$       IF LINK03 .EQS. "N" THEN GOTO LINK4
$ !
$ !
$ ! *****   LINK Code without the IMSL Static Library but with the
$ !                                   /MAP/CROSS_REFERENCE Qualifiers   *****
$ !
$       ON ERROR THEN GOTO LINK5
$       LINK/MAP/CROSS_REFERENCE    'LINKDEMO',       LIBA/LIBRARY,    -
                                     LIBD/LIBRARY,    LIBF/LIBRARY
$       GOTO LINK6
$ !
```

```
$ LINK4:
$ !
$ ! *****   LINK Code without the IMSL Static Library and with NO
$ !                                   /MAP/CROSS_REFERENCE Qualifiers   *****
$ !
$       ON ERROR THEN GOTO LINK5
$       LINK                          'LINKDEMO',       LIBA/LIBRARY,    -
                                       LIBD/LIBRARY,    LIBF/LIBRARY
$       GOTO LINK6
$ !
$ LINK5:
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT "ERROR in Linking the NLP10x10 System."
$       WRITE SYS$OUTPUT " "
$       GOTO TERM5
$ !
$ LINK6:
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT "The NLP10x10 System was Linked Successfully."
$       WRITE SYS$OUTPUT " "
$       GOTO TERM5
$ !
$ !
$ ! *****   Edit Files   *****
$ !
$ EDIT0:
$       INQUIRE EDIT00 "EDIT a File?  (Y/N)"
$       IF EDIT00 .EQS. "N" THEN GOTO TERM3
$ !
$ ! *****   EDIT a File    *****
$ !
$ EDIT1:
$       INQUIRE EDIT01 "ENTER NAME of File to be EDITED."
$       ON ERROR THEN GOTO EDIT2
$       EDT 'EDIT01'
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT "Editing File Completed Successfully."
$       WRITE SYS$OUTPUT " "
$       ON ERROR THEN GOTO EDIT3
$       @XPURGE
$       GOTO EDIT0
$ !
$ EDIT2:
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT "ERROR in Editing a File."
$       WRITE SYS$OUTPUT " "
$       GOTO EDIT0
$ !
$ EDIT3:
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT "ERROR in Purging Excess Files."
$       WRITE SYS$OUTPUT " "
$       GOTO EDIT0
$ !
$ CMPL0:
$       INQUIRE CMPL000 "COMPILE a File?  (Y/N)"
$       IF CMPL000 .EQS. "N" THEN GOTO TERM4
$ !
$ ! *****   COMPILE a File   *****
$ !
$       INQUIRE CFILE "ENTER NAME of File to be COMPILED."
$ !
$ ! *****   FORTRAN Compilation   *****
$ !
$       INQUIRE CMPL01 "Specify the /LIST Qualifier?  (Y/N)"
$       IF CMPL01 .EQS. "N" THEN GOTO CMPL2
$       INQUIRE CMPL02 "Specify the /SHOW=INCLUDE Qualifier?  (Y/N)"
$       IF CMPL02 .EQS. "N" THEN GOTO CMPL1
$       ON ERROR THEN GOTO CMPL4
$       FORTRAN/LIST/CONTINUATIONS=30/SHOW=INCLUDE/NOWARNINGS  'CFILE'.FOR
$       GOTO CMPL5
$ CMPL1:
```

```
$       ON ERROR THEN GOTO CMPL4
$       FORTRAN/LIST/CONTINUATIONS=30/NOWARNINGS  'CFILE'.FOR
$       GOTO CMPL5
$ CMPL2:
$       INQUIRE CMPL02 "Specify the /SHOW=INCLUDE Qualifier?  (Y/N)"
$       IF CMPL02 .EQS. "N" THEN GOTO CMPL3
$       ON ERROR THEN GOTO CMPL4
$       FORTRAN/CONTINUATIONS=30/SHOW=INCLUDE/NOWARNINGS  'CFILE'.FOR
$       GOTO CMPL5
$ CMPL3:
$       ON ERROR THEN GOTO CMPL4
$       FORTRAN/CONTINUATIONS=30/NOWARNINGS  'CFILE'.FOR
$       GOTO CMPL5
$ CMPL4:
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT "ERROR in FORTRAN Compilation."
$       WRITE SYS$OUTPUT " "
$       GOTO CMPL0
$ CMPL5:
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT "FORTRAN Compilation Completed Successfully."
$       WRITE SYS$OUTPUT " "
$       GOTO CMPL0
$ !
$ !
$ ! *****   Test for Termination   *****
$ !
$ !
$ TERM0:
$      INQUIRE TERM00 "Terminate Process?"
$      IF TERM00 .EQS. "N" THEN GOTO RUN0
$      DELETE FOR005.DAT;*
$      GOTO TERMINATE
$ !
$ TERM1:
$      INQUIRE TERM01 "Terminate Process?"
$      IF TERM01 .EQS. "N" THEN GOTO LINK0
$      GOTO TERMINATE
$ !
$ TERM2:
$      INQUIRE TERM02 "Terminate Process?"
$      IF TERM02 .EQS. "N" THEN GOTO EDIT0
$      GOTO TERMINATE
$ !
$ TERM3:
$      INQUIRE TERM03 "Terminate Process?"
$      IF TERM03 .EQS. "N" THEN GOTO CMPL0
$      GOTO TERMINATE
$ !
$ TERM4:
$      INQUIRE LINK000 "LINK the NLP10x10 System System?  (Y/N)"
$      IF LINK000 .EQS. "N" THEN GOTO TERM5
$      GOTO LINK1
$ !
$ TERM5:
$      INQUIRE RUN000 "RUN the NLP10x10 System?  (Y/N)"
$      IF RUN000 .EQS. "N" THEN GOTO TERMINATE
$      GOTO RUN1
$ !
$ ! *****   Termination   *****
$ !
$ TERMINATE:
$      WRITE SYS$OUTPUT " "
$      WRITE SYS$OUTPUT "TERMINATE RUN."
$      WRITE SYS$OUTPUT " "
$ !
$      DEASSIGN SYS$INPUT
$      DEASSIGN SYS$OUTPUT
$ !
$ EXIT
```