Mathematical Programming
Models and Algorithms
for Engineering Design Optimization

J. Herskovits

COPPE - Federal University of Rio de Janeiro
Mechanical Engineering Program
Caixa Postal 68503, 21945 970 Rio de Janeiro, Brazil

Abstract

Mathematical Programming provides general tools for Engineering Design Optimization. We present numerical models for Simultaneous Analysis and Design Optimization (SAND) and Multidisciplinary Design Optimization (MDO) represented by Mathematical Programs and numerical techniques to solve these models. These techniques are based on the Feasible Arc Interior Point Algorithm (FAIPA) for Nonlinear Constrained Optimization. Even if MDO is a very large optimization problem, our approach reduces considerably the computer effort. Several tools for very large problems are also presented. The present approach is very strong and efficient for real industrial applications and can easily interact with existing simulation engineering codes.

Key words: Engineering Design Optimization, MDO - Multidisciplinary Design Optimization, SAND - Simultaneous Analysis and Design, Nonlinear Programming, Interior Point Algorithms

1 Introduction

The Feasible Arc Interior Point Algorithm, FAIPA, is a new technique for nonlinear inequality and equality constrained optimization, [28]. FAIPA requires an initial point at the interior of the inequality constraints, and generates a
sequence of interior points. When the problem has only inequality constraints, the objective function is reduced at each iteration. An auxiliary potential function is employed when there are also equality constraints.

The fact of giving interior points, even when the constraints are nonlinear, makes of FAIPA an efficient tool for engineering design optimization, where functions evaluation is in general very expensive. Since any intermediate design can be employed, the iterations can be stopped when the objective reduction per iteration becomes small enough. Interior point algorithms are essential to solve problems that deal with an objective function, or constraints, that are not defined at infeasible points. This occurs in several applications, in particular, size and shape structural optimization. When applying interior points algorithms to real time process optimization, as the feasibility is maintained and the cost reduced, the controls can be activated at each iteration, [10,11].

FAIPA, that is an extension of the Feasible Directions Interior Point Algorithm [17–19,21,42], integrates ideas coming from the modern Interior Point Algorithms for Linear Programming with Feasible Directions Methods. At each point, FAIPA defines a ”Feasible Descent Arc”. Then, it finds on the arc a new interior point with a lower objective. Newton, quasi - Newton and first order versions of FAIPA can be obtained. FAIPA is supported by strong theoretical results. In particular, the search along an arc ensures superlinear convergence for the quasi - Newton version, even when there are highly non-linear constraints, avoiding the so called ”Maratos’ effect”, [38]. The present method, that is simple to code, does not require the solution of quadratic programs and it is not a penalty neither a barrier method. It merely requires the solution of three linear systems with the same matrix per iteration. Several practical applications of the present and previous versions of FAIPA, as well as several numerical results, show that the present is a very strong and efficient technique for engineering design optimization, [1,2,4–8,25,24,34,36,47]. In a book by Laporte and Le Tallec, [33], can be found a complete description of the application of FAIPA to shape optimization in aeronautics industry. The present approach was also applied to stress analysis problems involving mathematical programming, as contact problems, [3,35,53], and nonlinear limit analysis, [55]. In [23,51] algorithms for Nonlinear Least Squares Problems and Linear and Convex Quadratic Programming are described.

The classical model for design optimization requires, at each iteration, the solution of the state equation. When this equation is solved iteratively the whole process can be very painful. The SAND technique solves simultaneously the state equation and the optimization problem, [26]. For a given initial set of design variables and state variables the optimal design and the corresponding state variables are obtained iteratively. We remark that the intermediate state variables do not necessarily verify the state equations.
Multidisciplinary Design Optimization, MDO, “can be described as a methodology for design of complex engineering systems that are governed by mutually interacting physical phenomena and made up of distinct interacting subsystems”, [48].

Modern design techniques require numerical models of each of the parts of the system and each of the interacting physical phenomena. These models were generally developed independently, as well as the simulation codes based on them. From a practical point of view, to be successful, MDO must be based on existing codes, as they are. It is not reasonable to ask engineers and scientists working in the different disciplines to modify their mathematical and numerical models and the corresponding computer codes to adapt them to MDO.

MDO problems are naturally very large. They normally deal with a large number of design variables and include the state variables and constraints coming from all disciplines as well as the interaction between disciplines. Several techniques were developed to overcome this difficulty, [12,15,30,32,46,50,54]. Most of them try to decompose the problem into smaller sub problems or to work with reduced models for analysis and/or optimization. A recent survey can be found in [49].

FAIPA_MDO is a numerical optimization algorithm for MDO that works with a model that considers the complete problem without reductions, decompositions or simplifications. This goal is very ambitious due to the size and complexity of the problems, but it can be a way to obtain strong and efficient tools for MDO, [27].

Our model works with linking variables and equality constraints to introduce the interaction between disciplines. State equations can be treated implicitly as in the classical optimization model or, alternatively, included in the mathematical program as in SAND optimization.

FAIPA_SAND and FAIPA_MDO reduce the state variables and equality constraints given by the state equations. An original point of our technique is that this reductions are carried out without need of solving of the state equations. This formulation requires the solution of systems of equations given by a linearized equilibrium equation and can be performed very efficiently employing the solvers of engineering simulations codes.

FAIPA includes several techniques for very large size problems, as an extension for constrained optimization of Limited Memory quasi-Newton Method and iterative numerical techniques for the solution of the internal systems of FAIPA.

In the next session, Mathematical Programming models for classical engi-
neering design optimization and for SAND optimization are described. We also introduce our model for MDO. The basic ideas involved in FAIPA are described in Section 3. The subsequent sections are devoted to describe line search techniques, procedures to solve the internal systems of FAIPA and First Order, Newton and quasi-Newton versions of FAIPA. In section 7 a basic version FAIPA is stated. Sections 8 and 9 are dedicated to FAIPA versions for SAND and MDO respectively.

2 Numerical Models for Engineering Design Optimization

We consider the optimal design of engineering systems described by a state equation \( e(x, u) = 0 \), \( e \in \mathbb{R}^r \), where \( u \in \mathbb{R}^r \) represents the state variables. In most applications of structural optimization, the state variables are the nodal displacements and the state equation is given by the equilibrium. The state equation also depends on the design variables vector \( x \in \mathbb{R}^n \), that are the unknowns of the design problem.

The Optimal Design Problem consists on finding the value of \( x \) that minimizes a cost function \( f(x, u(x)) \) submitted to inequality constraints \( g(x, u(x)) \leq 0 \) and equality constraints \( h(x, u(x)) = 0 \), where \( g \in \mathbb{R}^m \) and \( h \in \mathbb{R}^p \) and \( u(x) \) verifies \( e(x, u(x)) = 0 \). This problem is represented by the following Mathematical Program:

\[
\begin{align*}
\min_x f(x, u(x)) \\
\text{s. t. } g(x, u(x)) &\leq 0 \\
h(x, u(x)) &= 0
\end{align*}
\]

(1)

We assume that \( f, g, h \) and \( e \) are continuous, as well as their first derivatives.

In engineering applications, Problem (1) is a Nonlinear Program and is solved making iterations on the design variables. At each of the iterations, the state equation must be solved and the sensitivity of the state variables must be computed for the current design.

There is an extensive literature about numerical methods for Problem (1). An engineering view is given in [13,16,31,52] and a mathematical approach in [9,29,37,40,43]. A condensed description of nonlinear optimization methods can be found in [20].

To solve simultaneously analysis and optimization problems, the state variables are included within the unknowns of the optimization problem and the
The state equation is considered as a set of additional equality constraints. The Mathematical Program for SAND Optimization is stated as follows:

$$\begin{align*}
\min_{x,u} f(x,u) \\
\text{s. t. } g(x,u) &\leq 0 \\
&h(x,u) = 0 \\
&c(x,u) = 0
\end{align*}$$

This model for engineering optimization is very advantageous in the case of nonlinear systems but, on the other hand, the size of the Mathematical Program is greatly increased. In general, the number of state variables is much larger than those of design variables.

We consider now the MDO of an engineering system integrated by ne subsystems and/or disciplines. The state variables of the subsystems are $u \equiv (u_1, u_2, \ldots, u_{ne})$, $u_i \in \mathbb{R}^{r_i}$. We define a vector of linking variables $z \in \mathbb{R}^s$ that represents the physical interactions between all the disciplines. The state equations of the disciplines can then be written as $e_1(x, z, u_1)$, $e_2(x, z, u_2)$, ..., $e_{ne}(x, z, u_{ne})$, $e_i \in \mathbb{R}^{r_i}$. We include conditions that impose compatibility of the interactions between disciplines in the set of equality constraints $h(x, z, u) = 0$ and call them "Compatibility Conditions".

Then, we propose the following Mathematical Program as a model for Multi-disciplinary Design Optimization.

$$\begin{align*}
\min_{x,z,u} f(x, z, u) \\
\text{s. t. } g(x, z, u) &\leq 0 \\
&h(x, z, u) = 0 \\
&e_1(x, z, u_1) = 0 \\
&e_2(x, z, u_2) = 0 \\
....................
&e_{ne}(x, z, u_{ne}) = 0
\end{align*}$$

As an example, we consider the case of structural and aerodynamics multidisciplinary optimization of airplanes and their components. The aerodynamics efforts acting on the structure can be included within the linking variables. These efforts are computed by the aerodynamics analysis code as functions of the aerodynamics state variables, that represent velocities. In the present
model the equality constraints include conditions imposing that, at the optimal design, the efforts employed as inputs for the structural analysis are equal to those computed by the aerodynamics analysis.

We remark that one or more disciplines can be treated implicitly in the MDO Problem (3) and the state equation solved at each iteration, as in the classical model for optimal design (1).

3 FAIPA - The Feasible Arc Interior Point Algorithm

FAIPA is an iterative algorithm to solve the Nonlinear Programming Problem

\[
\begin{array}{l}
\min f(x) \\
\text{s. t. } g(x) \leq 0 \\
h(x) = 0,
\end{array}
\]

where \( x \in \mathbb{R}^n \) and \( f \in \mathbb{R}, g \in \mathbb{R}^m, h \in \mathbb{R}^p \).

Let be \( \Omega \equiv \{x \in \mathbb{R}^n / g(x) \leq 0\} \). The following assumption on the problem are required:

**Assumption 3.1.** The functions \( f(x), g(x) \) and \( h(x) \) are continuous in \( \Omega \), as well as their first derivatives.

**Assumption 3.2.** (Regularity Condition) For all \( x \in \Omega \) the vectors \( \nabla g_i(x) \), for \( i = 1, 2, \ldots, m \) such that \( g_i(x) = 0 \) and \( \nabla h_i(x) \) for \( i = 1, 2, \ldots, p \) are linearly independent.

This condition must be checked in practical applications. There are several examples when the previous assumption is not true. In the case when there are sets of constraints that take the same values due to symmetries of the problem, only one constraint of each set must be considered.

We also introduce the definitions:

**Definition 3.1.** \( d \in \mathbb{R}^n \) is a Descent Direction for a smooth function \( \phi: \mathbb{R}^n \rightarrow \mathbb{R} \) at \( x \in \mathbb{R}^n \) if for some \( \psi > 0 \) it is \( \phi(x + td) < \phi(x) \) for all \( t \in (0, \psi] \). It can be proved that this condition is true if \( d^T \nabla \phi(x) < 0 \).

**Definition 3.2.** \( d \in \mathbb{R}^n \) is a Feasible Direction (with respect the inequality constrains) for the problem (4), at \( x \) such that \( g(x) \leq 0 \), if for some \( \theta > 0 \) we
have \( g(x + td) \leq 0 \) for all \( t \in [0, \theta] \). This condition is true if \( d^t \nabla g_i(x) < 0 \) for \( i = 1, 2, ..., m \).

At each point FAIPA defines a ”feasible descent arc”. A search is then performed along this arc to get a new interior point with a lower potential function.

We denote \( \nabla g(x) \in \mathbb{R}^{n \times m} \) and \( \nabla h(x) \in \mathbb{R}^{n \times p} \) the matrix of derivatives of \( g \) and \( h \) respectively and call \( \lambda \in \mathbb{R}^m \) and \( \mu \in \mathbb{R}^p \) the corresponding vectors of Lagrange multipliers. The Lagrangian is \( l(x, \lambda, \mu) = f(x) + \lambda^t g(x) + \mu^t h(x) \) and \( L(x, \lambda, \mu) = \nabla^2 f(x) + \sum_{i=1}^m \lambda_i \nabla^2 g_i(x) + \sum_{i=1}^p \mu_i \nabla^2 h_i(x) \) its Hessian. \( G(x) = \text{diag}(g(x)) \) denotes a diagonal matrix such that \( G_{ii}(x) = g_i(x) \).

Let us consider Karush-Kuhn-Tucker, (KKT), first order optimality conditions:

\[
\nabla f(x) + \nabla g(x) \lambda + \nabla h(x) \mu = 0 \tag{5}
\]

\[
G(x) \lambda = 0 \tag{6}
\]

\[
h(x) = 0 \tag{7}
\]

\[
\lambda \geq 0 \tag{8}
\]

\[
g(x) \leq 0 \tag{9}
\]

A point \( x^* \) is a Stationary Point if it exists \( \lambda^* \) and \( \mu^* \) such that (5)-(7) are true and is a KKT Point if KKT conditions (5)-(9) hold.

KKT conditions constitute a nonlinear system of equations and inequations on the unknowns \( (x, \lambda, \mu) \). It can be solved by computing the set of solutions of the nonlinear system of equations (5)-(7) and then, looking for those solutions such that (8) and (9)are true. However, this procedure is useless in practice.

FAIPA makes Newton-like iterations to solve the nonlinear equations (5)-(7) in the primal and the dual variables. With the object of ensuring convergence to KKT points, the system is solved in such a way as to have the inequalities (8) and (9) satisfied at each iteration.
Let be $S = L(x, \lambda, \mu)$. A Newton iteration for the solution of (5)-(7) is defined by the following linear system:

$$
\begin{bmatrix}
S & \nabla g(x) & \nabla h(x) \\
\Lambda \nabla g^T(x) & G(x) & 0 \\
\nabla h^T(x) & 0 & 0
\end{bmatrix}
\begin{bmatrix}
x_0 - x \\
\lambda_0 - \lambda \\
\mu_0 - \mu
\end{bmatrix}
= \begin{bmatrix}
\nabla f(x) + \nabla g(x) \lambda + \nabla h(x) \mu \\
G(x) \lambda \\
h(x)
\end{bmatrix}
$$

(10)

where $(x, \lambda, \mu)$ is the current point and $(x_0, \lambda_0, \mu_0)$ is a new estimate. We call $\Lambda = \text{diag}(\lambda)$.

We can also take $S \equiv B$, a quasi-Newton approximation of $L(x, \lambda, \mu)$, or $S \equiv I$. Depending on how $S$ is defined, (10) is a Newton, a quasi-Newton, or a First Order Newton-like iteration, see [21].

Iterative methods for nonlinear problems in general include a local search procedure to force global convergence to a solution of the problem. This is the case of Line Search and Trust Region Algorithms for nonlinear optimization, [37,40]. The present method includes a line search procedure, in the space of the primal variables $x$ only, that enforces the new iterate to be closer from the solution.

Let be $d_0 \in \mathbb{R}^n$ such that $d_0 = x_0 - x$. From (10), we have

$$
\begin{cases}
Sd_0 + \nabla g(x) \lambda_0 + \nabla h(x) \mu_0 = -\nabla f(x) \\
\Lambda \nabla g^t(x)d_0 + G(x) \lambda_0 = 0 \\
\nabla h^t(x)d_0 = -h(x)
\end{cases}
$$

(11)

which is independent of the current value of $\mu$. Then, (11) gives a direction in the space of primal variables $x$ and new estimates of the Lagrange multipliers.

Let be the potential function

$$
\phi(c, x) = f(x) + \sum_{i=1}^{p} c_i |h_i(x)|
$$

(12)

where, at the iteration $k$, $c_i^k$ is such that

$$
sg[h_i(x^k)](c_i + \mu_{0i}^k) < 0; i = 1, 2, ..., p,
$$

(13)
where \( sg(.) = \frac{(.)}{\|(.)\|} \). It is proved in [19,21] that \( d^k_0 \) is a descent direction of \( \phi(c^k, x) \).

However, \( d_0 \) is not useful as a search direction since it is not necessarily feasible. This is due to the fact that as any constraint goes to zero, it follows from (11) that \( d_0 \) goes to a direction tangent to the feasible set, [21].

To obtain a feasible direction, a negative vector \( -\rho\lambda \) is added in the right hand side of (11). A perturbed linear system in \( d \) and \( \bar{\lambda} \) is then obtained,

\[
\begin{cases}
Sd + \nabla g(x)\bar{\lambda} + \nabla h(x)\bar{\mu} = -\nabla f(x) \\
\Lambda \nabla g^t(x)d + G(x)\bar{\lambda} = -\rho\lambda \\
\nabla h^t(x)d = 0
\end{cases}
\]  

(14)

where \( \rho \in \mathbb{R} \) is positive. The new direction is \( d_1 \) and \( \bar{\lambda} \) and \( \bar{\mu} \) are the new estimate of the Lagrange multipliers. We have now that \( d \) is a feasible direction, since \( \nabla g^t(x)d = -\rho < 0 \) for the active constraints.

The addition of a negative number in the right hand side of (11) produces a deflection on \( d_0 \), proportional to \( \rho \), in the sense of the interior of the feasible region. To ensure that \( d \) is also a descent direction, we establish an upper bound on \( \rho \) in order to have

\[
d^t \nabla \phi(c, x) \leq \alpha d^t_0 \nabla \phi(c, x)
\]

(15)

with \( \alpha \in (0, 1) \), that implies \( d^t \nabla \phi(c, x) < 0 \). Thus, \( d \) is a descent direction of the potential function.

In general, the rate of descent of \( \phi \) along \( d \) will be smaller than along \( d_0 \). This is a price that we pay to get a feasible descent direction.

To obtain the upper bound on \( \rho \), we solve the auxiliary linear system in \((d_1, \lambda_1)\)

\[
\begin{cases}
Sd_1 + \nabla g(x)\lambda_1 + \nabla h(x)\mu_1 = 0 \\
\Lambda \nabla g^t(x)d_01 + G(x)\lambda_1 = -\lambda \\
\nabla h^t(x)d_1 = 0
\end{cases}
\]

(16)

It follows from (11), (14) and (16) that \( d = d_0 + \rho d_1 \). Then, we have that (15) is true for any \( \rho > 0 \), if \( d^t_1 \nabla \phi(c, x) < 0 \). Otherwise, we take

\[
\rho < \frac{(\alpha - 1)d^t_0 \nabla \phi(c, x)}{d^t_1 \nabla \phi(c, x)}
\]

(17)
and (15) holds.

The Feasible Directions Interior Point Algorithm, FDIPA, described in [21], employs $d$ as a search direction. The line search procedure looks for a step-length $t$ that ensures that the new point $(x + td)$ satisfies the inequality constraints with a “reasonable” decrease of the potential function $\phi(x, \lambda_0, \mu_0)$.

FDIPA has global convergence to a Karush-Kuhn-Tucker Point for any way of updating $S$ and $\lambda$ provided the following assumptions are true.

**Assumption 3.3.** There exist positive numbers $\lambda^I_1, \lambda^S$ and $\bar{g}$ such that $0 < \lambda_i \leq \lambda^S$, $i = 1, ..., m$, and $\lambda_i \geq \lambda^I_1$ for any $i$ such that $g_i(x) \geq -\bar{g}$.

**Assumption 3.4.** There exist positive numbers $\sigma_1$ and $\sigma_2$ such that

$$\sigma_1 \| d \|^2 \leq d^T B d \leq \sigma_2 \| d \|^2$$

for any $d \in \mathbb{R}^n$.

It was also proved superlinear asymptotic convergence of the quasi-Newton version of FDIPA, provided that the step-length obtained in the line search goes to one as the iterates go the solution of the problem.

However, when there are highly nonlinear constraints, the length of the feasible segment supported by the feasible descent search direction $d$ can be not enough to accept a step equal to one. This fact is similar to the Maratos’ Effect observed in [38] when employing the Sequential Quadratic Programming Method, [20,44,45].

Maratos’ effect was rarely observed in numerical tests carried out with FDIPA. But, when it occurred, the number of iterations required to solve the problem with a given precision was greatly improved. A strong optimization technique for real engineering applications must avoid Maratos’ Effect, since superlinear convergence can be broken with only one highly nonlinear constraint.

The basic idea to avoid this problem consists on making the line search along a second order arc, tangent to the feasible descent direction $d$ and with a curvature ”close” to the curvature of the feasible set boundary. The arc at $x$ is given by the following expression:

$$x(t) := x + td + t^2 \bar{d}$$
where $\tilde{d}$ is obtained by solving

$$
\begin{cases}
S \tilde{d} + \nabla g(x) \tilde{\lambda} + \nabla h(x) \tilde{\mu} = 0 \\
\Lambda \nabla g^t(x) \tilde{d} + G(x) \tilde{\lambda} = -\Lambda \tilde{\omega}^I \\
\nabla h^t(x) \tilde{d} = -\tilde{\omega}^E
\end{cases}
$$

(18)

This linear system is similar to (16), being $\tilde{\omega}_I \approx 2d^t \nabla^2 g_i(x) d$ and $\tilde{\omega}_E \approx 2d^t \nabla^2 h_i(x) d$ computed as follows:

$$
\begin{align*}
\tilde{\omega}_I^i &= g_i(x + d) - g_i(x) - \nabla g_i^t(x)d; \quad i = 1, \ldots, m \\
\tilde{\omega}_E^i &= h_i(x + d) - h_i(x) - \nabla h_i^t(x)d; \quad i = 1, \ldots, p
\end{align*}
$$

The arc search was proposed in [39] and [41] for the Sequential Quadratic Programming algorithm and it was also employed in an algorithm based on FDIPA, described in [42]. However, in both references the computation of $\tilde{d}$ requires the solution of a Quadratic Programming Problem while FAIPA merely solves an additional linear system with the same matrix.

4 Line Search Techniques for Interior Point Algorithms

The Feasible Arc Interior Point algorithms requires at each iteration a constrained line search looking for a step-length corresponding to a feasible point with a lower potential. The first idea consists on solving the following optimization problem on $t$:

$$
\begin{cases}
\min_t \phi(x^k + td^k + t^2 \tilde{d}^k) \\
\text{s. t. } g(x^k + td^k + t^2 \tilde{d}^k) \leq 0
\end{cases}
$$

(19)

Instead of making an exact minimization on $t$, it is more efficient to employ inexact line search techniques,[20,37,29,40]. These include acceptance criteria for the step length, compatible with a formal proof of global convergence, and a numerical procedure to find a value of $t$ that verifies these criteria. We extend to the constrained line search well known procedures employed for unconstrained optimization.
4.1 Armijo’s Line Search

Armijo’s line search defines a procedure to find a step-length ensuring a reasonable decrease of the potential function. In our case, we add the condition of feasibility of the inequality constrains. Armijo’s search is stated as follows:

**Algorithm 1. Armijo’s Constrained Line Search**

Define the step-length $t$ as the first number of the sequence $\{1, \nu, \nu^2, \nu^3, \ldots\}$, satisfying

$$\phi(x + td + t^2\tilde{d}) \leq \phi(x) + t\eta_1 \nabla \phi^t(x)d$$ \hspace{1cm} (20)

and

$$g(x + td + t^2\tilde{d}) \leq 0,$$ \hspace{1cm} (21)

where $\eta_1 \in (0,1)$ and $\nu \in (0,1)$ also.

Condition (20) imposes an upper bound on $t$. It ensures a reasonable decrease of the potential function, at least $\eta_1$ times the reduction obtained on a linear function tangent to $\phi$ at $x$. Let us call $t_{\text{max}}$ the greatest $t$ that satisfies (20) and $t_{\text{min}} = \inf(1, \nu t_{\text{max}})$. It is easy to prove that $t \in [t_{\text{min}}, t_{\text{max}}]$.

4.2 Wolfe’s and Goldstein’s Constrained Line Search Criteria:

Wolfe’s and Goldstein’s Criteria define intervals of acceptance of the step-length. According to our extension of Wolfe’s criterion, the step length $t$ is accepted if (20) and (21) are true and at least one of the following $m + 1$ conditions hold:

$$\nabla \phi^t(x + td + t^2\tilde{d})d \geq \eta_2 \nabla \phi^t(x)d$$ \hspace{1cm} (22)

and

$$g_i(x + td + t^2\tilde{d}) \geq \gamma g_i(x); i = 1, 2, \ldots, m$$ \hspace{1cm} (23)

where $\eta_1 \in (0,1/2)$, $\eta_2 \in (\eta_1, 1)$ and $\gamma \in (0,1)$. \hfill \square
Condition (22) imposes a decrease of the directional derivative of the potential function. However, if the exact minimum of Problem (19) is not interior, the existence of \( t \) such that (22) holds is not guaranteed. This is the motivation to include (23). We have that (20) and (21) define upper bounds on the step-length, while (22) and (23) define lower bounds.

Wolfe’s condition is very strong because guarantees a reduction of the potential function and of its directional derivative. However it requires the computation the directional derivative of the potential function. This is avoided by an extension of Goldstein’s criterion that, instead of (22), employs the following condition:

\[
\phi(x + td + t^2 \tilde{d}) \geq \phi(x) + t\eta_2 \nabla \phi^i(x)d, \tag{24}
\]

with \( \eta_2 \in (\eta_1, 1) \).

A step-length satisfying Wolfe’s or Goldstein’s criteria for interior point algorithms can be obtained iteratively in a similar way as in [29]. Given an initial \( t \), if it is too short, extrapolations are carried out until a good or a too long step is obtained. Once a too long step is obtained, interpolations based on the longest known short step and the shortest known long step are carried out, until the criterion is satisfied. The interpolations and extrapolation are based on polynomial approximations of the potential function and the inequality constraints. As the criterion of acceptance is quite wide, the process generally requires few iterations. A line search procedure based on Wolfe’s and Goldstein’s Criteria is described now.

**Algorithm 2. Wolfe’s and Goldstein’s Constrained Line Search**

**Parameters.** \( \eta_1 \in (0, 0.5) \), \( \eta_2 \in (\eta_1, 1) \) and \( \gamma \in (0, 1) \).

**Data.** Define an initial estimate of the step length, \( t > 0 \). Set \( t_R = t_L = 0 \).

**Step 1.** Test for the upper bound on \( t \).

If

\[
\phi(x + td + t^2 \tilde{d}) \leq \phi(x) + t\eta_1 \nabla \phi^i(x)d
\]

and

\[
g(x + td + t^2 \tilde{d}) \leq 0,
\]

Go to **Step 2**. Else go to **Step 4**.

**Step 2.** Test for the lower bound on \( t \).
If
\[ \nabla \phi'(x + td + t^2 \tilde{d})d \geq \eta_2 \nabla \phi'(x)d, \]
for Wolfe’s Criterium. Or
\[ \phi(x + td + t^2 \tilde{d}) \geq \phi(x) + t\eta_2 \nabla \phi'(x)d, \]
for Goldstein’s Criterium.

Or any
\[ g_i(x + td + t^2 \tilde{d}) \geq \gamma g_i(x) \text{ for } i = 1, 2, ..., m, \]
then \( t \) verifies Line Search Criterium, STOP.

Step 3. Get a longer \( t \).

Set \( t_L = t \).

i) If \( t_R = 0 \), find a new \( t \) by extrapolation based on \((0, t)\).

ii) If \( t_R > 0 \), find a new \( t \) by interpolation in \((t, t_R)\).

Return to Step 1. Step 4. Get a shorter \( t \).

Set \( t_R = t \).

Find a new \( t \) by interpolation in \((t_L, t)\) Return to Step 1.

5 Solving FAIPA’s Internal Linear Systems

The matrix corresponding to the linear systems (11), (16) and (18), called "Primal-Dual Matrix", is not symmetric, neither positive definite. However, in [42] it was proved that the solution is unique, provided the problem satisfies Regularity Condition.

It follows from (11) that
\[ d_0 = -S^{-1}[\nabla f(x) + \nabla g(x)\lambda_0 + \nabla h(x)\mu_0] \]

and
\[ [\Lambda \nabla g^t(x)S^{-1}\nabla g(x) - G(x)]\lambda_0 + \nabla g^t(x)S^{-1}\nabla h(x)\mu_0 = -\Lambda \nabla g^t(x)S^{-1}\nabla f(x) \quad (26) \]
\[ \nabla h^t(x)S^{-1}\nabla g(x)\lambda_0 + \nabla h^t(x)S^{-1}\nabla h(x)\mu_0 = -\nabla h^t(x)S^{-1}\nabla f(x) + h(x) \]
Then, instead of (11), we can obtain \( \lambda_0 \) and \( \mu_0 \) by solving (26), and \( d_0 \) by substitution in (25). Similar expressions, involving linear systems with the same matrix as in (26), called "Dual Matrix", can be obtained to solve (16) and (18).

The Dual Matrix is symmetric and positive definite when the regularity condition is true, [17,19]. However the condition number becomes worst as some of the components of \( \lambda \) grows. To compute the matrices we need \( S^{-1} \) or the products \( S^{-1}\nabla g(x) \) and \( S^{-1}\nabla f(x) \). In quasi-Newton algorithms, this can be easily obtained by working with an approximation of \( L^{-1}(x, \lambda, \mu) \).

In a similar way as before, we deduce from (11) that

\[
\lambda_0 = -G(x)^{-1}\Lambda\nabla g \cdot d_0
\]  

(27)

and

\[
\begin{cases}
[S - \nabla g(x)G^{-1}(x)\Lambda\nabla g'(x)]d_0 + \nabla h(x)\mu_0 = -\nabla f(x) \\
\nabla h'(x)d_0 = -h(x)
\end{cases}
\]  

(28)

The system above is symmetric and positive definite. This is true since \( S \) is positive definite and \( \nabla g(x)G^{-1}(x)\Lambda\nabla g'(x) \) is negative definite at \( x \) such that \( g(x) \leq 0 \). When there are only inequality constraints, (28) gives \( d_0 \) and the corresponding matrix is called "Primal Matrix". Also in this case similar expressions are obtained to solve (16) and (18).

6 First Order, Newton and Quasi-Newton Iterations

In the present method \( S \) can be taken equal to the second derivative of the Lagrangian, to a quasi-Newton estimate or to the Identity Matrix.

6.1 A First Order Algorithm

Taking \( S = I \) the present method becomes an extension of the gradient method. In particular, when all the constraints are active, the search direction is equal to the projected gradient, [20]. Up to now, we have no theoretical results about the speed of convergence, but probably the rate of convergence is no better than linear. Since the computer effort and memory storage are smaller, this algorithm can be efficient in engineering applications that do not
need a very precise solution or when the computation of the involved functions and derivatives is not expensive.

6.2 A Newton Algorithm

To have a Newton algorithm we take \( S = L(x^k, \lambda^k, \mu^k) \) as in the iteration (10). However, as mentioned above, \( S \) must be positive definite. Since this is not always true, the Newton version of the present method can be obtained for particular applications only. This is the case of the algorithm for Nonlinear Limit Analysis and of the algorithm for Linear Stress Analysis with Contact described in references [55] and [3].

A study of the rate of convergence in this case is still required. Newton method for nonlinear system has quadratic local convergence. But, we make iterations in the primal and dual variables, and, in practice we are interested on the rate of convergence in terms of the design variables \( x \).

6.3 A Quasi - Newton Algorithm

In quasi - Newton methods for constrained optimization, \( S = B \), an approximation of the Hessian of the Lagrangian \( L(x, \lambda, \mu) \). Then, it should be possible to obtain \( B \) with the same updating rules as in unconstrained optimization, but taking \( \nabla_x l(x, \lambda, \mu) \) instead of \( \nabla f(x) \). However, since \( L(x, \lambda, \mu) \) is not necessarily positive definite at a KKT point, it is not always possible to get \( B \) positive definite, as required by the present technique. Powell, [44,45], proposed a modification of Broyden - Fletcher - Goldfarb and Shanno (BFGS) Updating Rule that gives positive definite quasi - Newton matrices.

**BFGS Updating Rule for Constrained Optimization**

Take

\[
\delta = x^{k+1} - x^k
\]

and

\[
\gamma = \nabla_x l(x^{k+1}, \lambda_0^k, \mu_0^k) - \nabla_x l(x^k, \lambda_0^k, \mu_0^k)
\]

If

\[
\delta^T \gamma < 0.2 \delta^T B^k \delta,
\]

then compute

\[
\phi = \frac{0.8 \delta^T B^k \delta}{\delta^T B^k \delta - \delta^T \gamma}
\]
and take

\[ \gamma := \phi \gamma + (1 - \phi)B^k \delta. \]

Set

\[ B^{k+1} := B^k + \frac{\gamma \gamma^t}{\delta^t \gamma} B^k \delta^t \delta - \frac{B^k \delta^t B^k}{\delta^t B^k \delta} \]

In [21] it was proved that the convergence of the present algorithm is two-step superlinear, provided that a unit step length is obtained after a finite number of iterations.

It is also interesting to build a quasi - Newton approximation of the inverse of the Hessian Matrix, \( H \approx L^{-1}(x, \lambda_0, \mu_0) \). An Updating Rule for \( H \) is obtained by employing Sherman-Morrison Formula, [40], to get the inverse of \( B^{k+1} \):

\[ H^{k+1} := H^k + \left( 1 + \frac{\gamma^t H^k \gamma}{\gamma^t \delta} \right) \delta \delta^t - \frac{\gamma^t H^k \gamma + H^k \gamma \delta^t}{\gamma^t \delta} \]  

(29)

6.4 Limited Memory quasi-Newton Method

Quasi - Newton methods, when applied to constrained optimization, construct an approximation of the second derivative of the Lagrange function [20,37,40]. Quasi - Newton method, when applied to constrained optimization, are very efficient since it avoids the calculus of the second derivative of the objective function and the constraints. However they require the storage of the quasi - Newton matrix, that is full. Limited memory quasi - Newton methods avoid the storage of this matrix, [14,40]. They were first developed for unconstrained optimization and then extended to problems with box constraints. FAIPA is particularly adapted to employ this technique in problems with any kind of nonlinear constraints.

Limited Memory Method computes efficiently the product of the quasi-Newton Matrix \( H^{k+1} \) by a vector \( v \in \mathbb{R}^n \), or by a matrix, without the explicit assembly and storage of \( H^{k+1} \), requiring only the storage of the \( q \) most recent pairs of vectors \( \delta \) and \( \gamma \).

Updating rule (29) for \( H \) can be expressed as follows:

\[ H^{k+1} = H^{k-q} + [\Delta \ H^{k-q} \Gamma] E[\Delta \ H^{k-q} \Gamma]^t \]  

(30)

where

\[ \Delta = [\delta^{k-q}, \delta^{k-q+1}, \delta^{k-q+2}, ..., \delta^{k-1}]; \ \Delta \in \mathbb{R}^{n \times q} \]
\[ \Gamma = [\gamma^{k-q}, \gamma^{k-q+1}, \gamma^{k-q+2}, ..., \gamma^{k-1}]; \quad \Gamma \in \mathbb{R}^{n \times q} \]

\[ E = \begin{bmatrix} \mathbb{R}^{-t}(D + \Gamma' H^{k-q}\Gamma)R^{-1} - R^{-t} \\ -R^{-1} \\ 0 \end{bmatrix}; \quad E \in \mathbb{R}^{2q \times 2q} \]

\[ R = \text{upper}(\Delta' \Gamma); \quad R \in \mathbb{R}^{q \times q} \]

\[ D = \text{diag}(R) \]

We write \( A = \text{upper}(B) \) when \( A_{ij} = B_{ij} \) for \( j \geq i \) and \( A_{ij} = 0 \) otherwise.

Limited Memory Method takes \( H^{k-q} = I \). Then, the following expression for \( H^{k+1}v \) is obtained:

\[ H^{k+1}v = v + [\Delta \quad \Gamma][E][\Delta \quad \Gamma]'v. \quad (31) \]

Now, the coefficient matrices and right sides of the Dual System (26) can be obtained without need of computing and storing the quasi - Newton Matrix. A Limited Memory formulation for the BFGS matrix \( B \) can be obtained in a similar way and employed to solve the Primal System (28).

Even in very large problems, taking \( q \approx 10 \), the number of iterations with Limited Memory and with the original Quasi - Newton Algorithm remains similar.

Iterative methods for linear systems in general compute the product of the coefficient matrix by a vector at each iteration. Limited Memory formulation can also be applied to solve iteratively the internal linear systems of FAIPA without storing the quasi - Newton matrix. This can be extremely efficient when the constraints derivatives are sparse.

7 Statement of FAIPA

Algorithm 3. Feasible Arc Interior point Algorithm

Parameters. \((\alpha, \eta, \nu_1, \nu_2, \gamma) \in (0, 1), \varphi > 0 \text{ and } c \in \mathbb{R}^p, c > 0.\)

Data. Initial values for \( x \in \mathbb{R}^n \) such that \( g(x) < 0, \lambda \in \mathbb{R}^m, \lambda > 0, \text{ and } S \in \mathbb{R}^{n \times n} \) symmetric and positive definite.

Step 1. Computation of a feasible descent direction.
(i) Solve the linear systems:

\[
\begin{align*}
Sd_0 + \nabla g(x)\lambda_0 + \nabla h(x)\mu_0 &= -\nabla f(x) \\
\Lambda \nabla g'(x)d_0 + G(x)\lambda_0 &= 0 \\
\nabla h'(x)d_0 &= -h(x)
\end{align*}
\]  \hfill (32)

and

\[
\begin{align*}
Sd_1 + \nabla g(x)\lambda_1 + \nabla h(x)\mu_1 &= 0 \\
\Lambda \nabla g'(x)d_1 + G(x)\lambda_1 &= -\lambda \\
\nabla h'(x)d_1 &= 0
\end{align*}
\]  \hfill (33)

Let be the potential function

\[
\phi_c(x) = f(x) + \sum_{i=1}^{p} c_i |h_i(x)|,
\]  \hfill (34)

(ii) If \(c_i < -1.2\mu_0(i)\), then set \(c_i = -2\mu_0(i); i = 1, ..., p\).

(iv) If \(d_1^T \nabla \phi_c(x) > 0\), set

\[
\rho = \min[\varphi \parallel d_0 \parallel^2_2 ; (\alpha - 1)d_0^T \nabla \phi_c(x)/d_1^T \nabla \phi_c(x)]
\]  \hfill (35)

Otherwise, set

\[
\rho = \varphi \parallel d_0 \parallel^2_2.
\]  \hfill (36)

(iii) Compute the feasible descent direction: \(d = d_0 + \rho d_1\)

**Step 2.** Computation of a ”restoring direction” \(\tilde{d}\)

Compute:

\[
\begin{align*}
\tilde{\omega}_i^f &= g_i(x + d) - g_i(x) - \nabla g_i'(x)d; i = 1, ..., m \\
\tilde{\omega}_i^E &= h_i(x + d) - h_i(x) - \nabla h_i'(x)d; i = 1, ..., p
\end{align*}
\]
Solve:

\[
\begin{align*}
S\dd + \nabla g(x)\tilde{\lambda} + \nabla h(x)\tilde{\mu} &= 0 \\
\Lambda\nabla g^l(x)\dd + G(x)\tilde{\lambda} &= -\Lambda\tilde{\omega}^f \\
\nabla h^l(x)\dd &= -\tilde{\omega}^E
\end{align*}
\]

(37)

**Step 3.** Arc search.

Employ a line search procedure to get a step-length \( t \) based on the Potential Function \( \phi_c(x + td + t^2\dd) \)

**Step 4.** Updates.

(i) Set the new point:

\[ x := x + td + t^2\dd \]

(ii) Define new values for \( \lambda > 0 \) and \( S \) symmetric and positive definite.

(iii) Go back to Step 1.

The present algorithm is very general in the sense that it converges to a Karush-Kuhn-Tucker point of the problem for any initial interior point. This is true no matter how \( \lambda \) and \( S \) are updated in Step 4, taking care of Assumptions 1 and 2. We work with the following updating rule for \( \lambda \).

**Updating Rule for \( \lambda \)**

Set, for \( i = 1, \ldots, m \),

\[ \lambda_i := \max \left[ \lambda_0; \epsilon \| d_0 \|_2^2 \right]. \]  

(38)

If \( g_i(x) \geq -\bar{g} \) and \( \lambda_i < \lambda^I \), set \( \lambda_i = \lambda^I \).

The parameters \( \epsilon, \bar{g} \) and \( \lambda^I \) are taken positive. In this rule, \( \lambda_i \) is a second order perturbation of \( \lambda_0 \), given by Newton iteration (10). If \( \bar{g} \) and \( \lambda^I \) are taken small enough, then after a finite number of iterations, \( \lambda_i \) becomes equal to \( \lambda_0 \) for the active constraints.

8 A Quasi-Newton Method for Simultaneous Analysis and Design

In the case when FAIPA is applied to solve the SAND Problem (2), the sizes of the internal linear systems and of the quasi-Newton matrix are much increased
since the number of degrees of freedom of the state equation in general is much larger that the number of design variables.

We present a quasi-Newton algorithm, FAIPA SAND, that reduces these sizes to the same values as in the classical design optimization model. Our technique can be considered as a generalization of the Reduced Gradient method. Existing Reduced Algorithms restore the equality constraints at each iteration. The present formulation avoids this procedure, that is equivalent to solve the state equation at each iteration.

Let be the the Mathematical Program for SAND problem:

$$\begin{align*}
\min_{x,u} & \quad f(x, u) \\
\text{s. t.} & \quad g(x, u) \leq 0 \\
& \quad h(x, u) = 0 \\
& \quad e(x, u) = 0
\end{align*}$$

(39)

The first linear system (11) of FAIPA becomes

$$\begin{bmatrix}
B_{xx} d_{0x} + B_{xu} d_{0u} + \nabla_x g(x, u) \lambda_0 + \nabla_x h(x, u) \mu_0 + \nabla_x e(x, u) \nu_0 = -\nabla_x f(x, u) \\
B_{ux} d_{0x} + B_{uu} d_{0u} + \nabla_u g(x, u) \lambda_0 + \nabla_u h(x, u) \mu_0 + \nabla_u e(x, u) \nu_0 = -\nabla_u f(x, u) \\
\Lambda \nabla_x g'(x, u) d_{0x} + \Lambda \nabla_u g'(x, u) d_{0u} + G(x, u) \lambda_0 = 0 \\
\nabla_x h'(x, u) d_{0x} + \nabla_u h'(x, u) d_{0u} = -h(x, u) \\
\nabla_u e'(x, u) d_{0x} + \nabla_u e'(x, u) d_{0u} = -e(x, u)
\end{bmatrix}$$

(40)

where

$$B = \begin{bmatrix}
B_{xx} & B_{xu} \\
B_{ux} & B_{uu}
\end{bmatrix}, \quad d_0 = \begin{bmatrix}
d_{0x} \\
d_{0u}
\end{bmatrix}$$

and $\nu_0 \in \mathbb{R}^r$ represents the Lagrange Multipliers corresponding to the state equation.

Now, we assume that $\nabla_u e'(x, u)$ has an unique inverse and call

$$\delta u \equiv [\nabla_u e'(x, u)]^{-1} e(x, u); \quad \delta u \in \mathbb{R}^r$$

(41)

and

$$\Delta u \equiv [\nabla_u e'(x, u)]^{-1} \nabla_x e'(x, u); \quad \Delta u \in \mathbb{R}^{r \times n}.$$
We remark that \( \nabla u(x, u) \) is the so called "Tangent Matrix" and it has a particular structure depending on the application. Numerical analysis codes in different disciplines usually include techniques to solve the tangent equation that take advantage of the structure and that can be employed to compute \( \delta u \) and \( \Delta u \).

Let us define \( M = [I - \Delta u^t], M \in \mathbb{R}^{n \times (n+r)} \), \( I_x = [I^{n \times n}], I_x \in \mathbb{R}^{n \times (n+r)} \), and \( I_u = [I^{r \times r}], I_u \in \mathbb{R}^{r \times (n+r)} \). By elimination of \( d_0 \) and \( \nu_0 \) from (40), the following reduced system is obtained:

\[
\begin{align*}
\bar{B}d_{0x} + [\nabla x g(x, u) - \Delta u^t \nabla a g(x, u)] \lambda_0 + [\nabla x h(x, u) - \Delta u^t \nabla a h(x, u)] \mu_0 &= b \\
\Lambda[\nabla x g(x, u) - \Delta u^t \nabla a g(x, u)]^t d_{0x} + G(x, u) \lambda_0 &= -\Lambda \nabla x g(x, u) \delta u \\
[\nabla x h(x, u) - \Delta u^t \nabla a h(x, u)]^t d_{0x} &= -[h(x, u) + \nabla x h(x, u) \delta u]
\end{align*}
\]

where \( \bar{B} = MBM^t, \bar{B} \in \mathbb{R}^{n \times n} \), is a reduced quasi-Newton matrix and

\[
b = -\nabla x f(x, u) + \Delta u^t \nabla a f(x, u) - (I_x + \Delta u^t I_u)B I_u^t \delta u.
\]

We employ the Limited Memory quasi-Newton formulation to compute the matrix \( \bar{B} \) and the vector \( b \) without need of computing and storing the full quasi-Newton matrix \( B \). With the present formulation the internal linear systems has the same size as in the classical model for engineering optimization.

9 FAIPA_MDO: A Mathematical Programming Algorithm for MDO

We consider now the Problem (3), that represents a model for Multidisciplinary Design Optimization. This one is a particular case of Problem (2), where \( u \equiv (u_1, u_2, ..., u_{ne}) \) and

\[
e(x, z, u) \equiv [e_1(x, z, u_1), e_2(x, z, u_2), ..., e_{ne}(x, z, u_{ne})].
\]

It follows that \( \delta u = (\delta u_1, \delta u_2, ..., \delta u_{ne}) \) and \( \Delta u = (\Delta u_1, \Delta u_2, ..., \Delta u_{ne}) \), where

\[
\delta u_i = [\nabla u_i e^i_t(x, u_i)]^{-1} e_i(x, u_i); \delta u_i \in \mathbb{R}^{r_i}
\]

and

\[
\Delta u_i = [\nabla u_i e^i_t(x, u_i)]^{-1} \nabla x e^i_t(x, u_i); \Delta u_i \in \mathbb{R}^{r_i \times n}.
\]

Then, the tangent state equations corresponding to the subsystems have to be solved at each iteration of FAIPA_MDO.
Given initial values for the Design Variables \( x \), the State Variables \( u_i \) corresponding to all the subsystems and the Linking Variables \( z \), FAIPA_MDO generates a sequence converging to an optimal point that verifies the State Equations and the Compatibility Equations. The initial point only needs to verify the inequality constraints.

The stopping criterion of the main algorithm includes the verification, with a given tolerance, of Karush - Kuhn - Tucker optimality conditions and of the state equations of all the subsystems. KKT conditions are easily checked since FAIPA produces estimates of the Lagrange Multipliers.

Following, a simplified description of the iterative process of FAIPA_MDO is presented.

Algorithm 4. \( FAIPA\_MDO \)

**Data:** Initial values for \( x \in \mathbb{R}^n \), \( z \in \mathbb{R}^s \) and \( u \equiv (u_1, u_2, ..., u_{ne}) \), \( u_i \in \mathbb{R}^{r_i} \) such that \( g(x, z, u) < 0 \) and \( \lambda \in \mathbb{R}^m \), \( \lambda > 0 \), and \( S \in \mathbb{R}^{nxn} \) symmetric and positive definite.

**Step 1.** For each discipline \( i \) compute:

The State Equation

\[ e_i(x, z, u) \]

The State Equation Sensitivities

\[ \nabla_x e_i(x, z, u), \nabla_z e_i(x, z, u), \nabla_u e_i(x, z, u) \]

**Step 2.** For each discipline \( i \) solve:

\[ [\nabla_{u_i} e_i^t(x, u_i)] \delta u_i = e_i(x, u_i) \]

and

\[ [\nabla_{u_i} e_i^t(x, u_i)] \Delta u_i = \nabla_x e_i^t(x, u_i) \]

**Step 3.** Compute a Feasible Descent Direction

\[ d \equiv (d_x, d_z, d_{u1}, d_{u2}, ..., d_{une}) \]

and estimates of the Lagrange Multipliers

\[ \lambda_0, \mu_0, \nu_{01}, \nu_{02}, ..., \nu_{0ne} \]
**Step 4.** For each discipline $i$ compute:

The State Equation

$$e_i(x + d_x, z + d_z, u + d_u)$$

**Step 5.** Compute a Restoring Direction

$$\tilde{d} \equiv (\tilde{d}_x, \tilde{d}_z, \tilde{d}_{u1}, \tilde{d}_{u2}, \ldots, \tilde{d}_{une})$$

**Step 6.** Arc search.

Find a step-length $t$ based on the Potential Function $\phi_c(x + td + t^2\tilde{d})$

**Step 7.** Updates.

(i) Set the new point:

$$x := x + td_x + t^2\tilde{d}_x$$

$$z := z + td_z + t^2\tilde{d}_z$$

$$u_i := u_i + td_{u_i} + t^2\tilde{d}_{u_i}; \text{ for } i = 1, 2, \ldots, ne$$

(ii) Define new values for $\lambda > 0$ and $S$ symmetric and positive definite.

(iii) Go back to Step 1.

FAIPA_MDO solves three linear systems with the same matrix per iteration. The size of these ones depends only on the number of design variables and constraints. The additional systems required to compute $\delta u_i$ and $\Delta u_i$ can have their matrix computed by the engineering simulations codes, as well as, be solved by these codes. In this way, existing analysis codes and their solvers can be employed without changes, for MDO. We remark that FAIPA and FAIPA_MDO have strong theoretical bases that ensure global convergence to KKT points of Problem (3).

**References**


