OBJECT ORIENTED IMPLEMENTATION OF A SECOND-ORDER OPTIMIZATION METHOD

Luís F. D. Brás
Alvaro F. M. Azevedo

Faculty of Engineering
University of Porto
PORTUGAL
OPTIMIZATION APPROACH

- Nonlinear program
- Second-order approximation
- Integrated formulation
- All the problem variables are present in the nonlinear program
- No sensitivity analysis
OPTIMIZATION SOFTWARE

- General purpose code
- Lagrange-Newton method
- Symbolic manipulation of all the functions
- Exact 1\textsuperscript{st} and 2\textsuperscript{nd} derivatives
- Object oriented approach
- Language: C++
OBJECT ORIENTED PROGRAMMING

• What we gain
  ♦ Higher abstraction level
  ♦ Encapsulation of lower level complexities
  ♦ Code maintenance and reuse is facilitated

• What we lose
  ♦ Performance
  ♦ Straightforward coding
OBJECT ORIENTED FEATURES

- Classes
- Function and operator overloading
- Inheritance
- Polymorphism
- Templates
- Exception handling
NONLINEAR PROGRAMMING

Minimize $f(x)$

subject to

$g(x) \leq 0 \quad \rightarrow \quad g_i(x) + s_i^2 = 0$

$h(x) = 0$

- Variables / functions real and continuous
- Generic functions can be treated
LAGRANGIAN

\[ L(\hat{X}) = f(\hat{x}) + \sum_{k=1}^{m} \lambda_k^g \left[ g_k(\hat{x}) + s_k^2 \right] + \sum_{k=1}^{p} \lambda_k^h h_k(\hat{x}) \]

VARIABLES

\[ \hat{X} = (\hat{x}, \hat{s}, \lambda^g, \lambda^h) \]

SOLUTION

• Stationary point of the Lagrangian
SYSTEM OF NONLINEAR EQUATIONS

\[ \nabla L(X) = 0 \quad \Rightarrow \quad \]
\[ 2s_i \lambda_i^g = 0 \quad (i = 1, \ldots, m) \]
\[ g_i + s_i^2 = 0 \quad (i = 1, \ldots, m) \]
\[ \frac{\partial f}{\partial x_i} + \sum_{k=1}^m \lambda_k^g \frac{\partial g_k}{\partial x_i} + \sum_{k=1}^p \lambda_k^h \frac{\partial h_k}{\partial x_i} = 0 \quad (i = 1, \ldots, n) \]
\[ h_i = 0 \quad (i = 1, \ldots, p) \]

- The solution of the system is a KKT solution when

\[ \lambda_i^g \geq 0 \]
LAGRANGE-NEWTON METHOD

• The system of nonlinear equations

\[ \nabla L(X) = 0 \]

is solved by the Newton method

• In each iteration the following system of linear equations has to be solved

\[ H(X^{q-1}) \Delta X^q + \nabla L(X^{q-1}) = 0 \]

• H is the Hessian of the Lagrangian
• Second derivatives of all the functions are required
SYSTEM OF LINEAR EQUATIONS

• Gaussian elimination
  • adapted to the sparsity pattern of the Hessian matrix

• Conjugate gradients
  • diagonal preconditioning
  • adapted to an indefinite Hessian matrix
LINE SEARCH

\[ X^q = X^{q-1} + \alpha \Delta X^q \]

• The value of \( \alpha \) minimizes the error in \( \Delta X^q \) direction
  ♦ the value of \( \alpha \) is often close to one
  ♦ faster convergence
  ♦ process may fail

• The value of \( \alpha \) is made considerably smaller (e.g. \( \alpha = 0.1 \))
  ♦ stable convergence
  ♦ more iterations - slower
AUTOMATIC DIFFERENTIATION

- Expression evaluation
- Partial derivative calculation (first, second, ...)
- Each function is parsed and stored as a tree of tokens (constants, variables and operators)
- Automatic differentiation is based on Rall numbers
RALL NUMBERS

- A Rall number is a class that encapsulates the numerical value of the function, its gradient vector and its Hessian matrix.
- All the operators are overloaded in order to apply the differentiation rules.
- With Rall numbers automatic differentiation can be efficiently performed.
RALL NUMBERS

Example:

Functions \( f(x_1, x_2) \) and \( g(x_1, x_2) \)

Derivatives of the product:

\[
\frac{\partial}{\partial x_1} (f \cdot g) = \frac{\partial f}{\partial x_1} g + f \frac{\partial g}{\partial x_1}
\]

\[
\frac{\partial^2}{\partial x_1 \partial x_2} (f \cdot g) = \frac{\partial^2 f}{\partial x_1 \partial x_2} g + \frac{\partial f}{\partial x_2} \frac{\partial g}{\partial x_1} + \frac{\partial f}{\partial x_1} \frac{\partial g}{\partial x_2} + f \frac{\partial^2 g}{\partial x_1 \partial x_2}
\]
class CRall {
    double x; // Operand value
    double v[2]; // df/dx1, df/dx2
    double m[2][2]; // d2f/dxi dxj

public:
    CRall CRall::operator* (const CRall & g) const {
        CRall t;
        t.x = x * g.x;

        t.v[0] = v[0]*g.x + x*g.v[0];
        t.v[1] = v[1]*g.x + x*g.v[1];

        t.m[0][0] = m[0][0]*g.x+v[0]*g.v[0]+v[0]*g.v[0]+x*g.m[0][0];
        t.m[0][1] = m[0][1]*g.x+v[0]*g.v[1]+v[1]*g.v[0]+x*g.m[0][1];
        t.m[1][0] = m[1][0]*g.x+v[1]*g.v[0]+v[0]*g.v[1]+x*g.m[1][0];
        t.m[1][1] = m[1][1]*g.x+v[1]*g.v[1]+v[1]*g.v[1]+x*g.m[1][1];

        return t;
    }
};
RALL NUMBERS

\[ x = \text{constant value}; \]
\[ v = [0,0]; \]
\[ m = [[0,0],[0,0]] \]

\[ x = \text{value of } x_1; \]
\[ v = [1,0]; \]
\[ m = [[0,0],[0,0]] \]

\[ x = \text{value of } x_2; \]
\[ v = [0,1]; \]
\[ m = [[0,0],[0,0]] \]
EXPRESSION PARSER

- A binary tree is constructed according to the operator precedence
- Each tree node is a Rall number
- A symbol table is initialized with the values of the variables and constants
- The tree traversal causes an evaluation of the function, gradient and Hessian
**EXPRESSION PARSER**

- **Example:**

\[
 f(x_1, x_2, x_3) = \frac{(x_1 + 8x_2)}{(6 - x_3^2)}
\]
SCALING

- Variable substitution: \( x_i = c \overline{x_i} \)
- Constraint normalization: \( g_i = c \overline{g_i} \)

**Min. 2000 \( x_1 \)**

*subject to*

\[- x_1 + 200 + x_3^2 = 0 \]
\[ x_2 - 0.2 + x_4^2 = 0 \]
\[-10 x_1 x_2 + 500 = 0 \]

**Min. \( y_1 \)**

*subject to*

\[-0.640 y_1 + 0.256 + 0.384 y_3^2 = 0 \]
\[ 0.447 y_2 - 0.894 + 0.447 y_4^2 = 0 \]
\[-0.707 y_1 y_2 + 0.707 = 0 \]
NUMERICAL EXAMPLE

\[ F = (f_1, f_2) \]

Variables: \( A, x \)

Svanberg’s solution confirmed

\[ \| \vec{F} \| = 200 \text{ kN} \]
\[ f_2 = 8 \, f_1 \]
\[ \sigma_{\text{max}} = 100 \, \text{kN/cm}^2 \]
NONLINEAR PROGRAM

\[ \text{Min. } w(x_1, x_2) = C_1 x_1 \sqrt{1 + x_2^2} \]

subject to

\[ \sigma_1(x_1, x_2) = C_2 \sqrt{1 + x_2^2} \left( \frac{8}{x_1} + \frac{1}{x_1 x_2} \right) \leq 1 \]

\[ \sigma_2(x_1, x_2) = C_2 \sqrt{1 + x_2^2} \left( \frac{8}{x_1} - \frac{1}{x_1 x_2} \right) \leq 1 \]

\[ 0.2 \leq x_1 \leq 4.0 \; ; \; 0.1 \leq x_2 \leq 1.6 \]
DATA FILE

# Main title
Shape optimization of a two bar truss

# N. of eq. constr.; N. of ineq. constr.
0                    6

# Objective Function
C1 * x1 * sqrt(1+x2^2);

# Allowable stress - bar 1
C2 * sqrt(1+x2^2) * (8/x1+1/x1/x2) - 1;

# Allowable stress - bar 2
C2 * sqrt(1+x2^2) * (8/x1-1/x1/x2) - 1;

# Minimum x1
-x1 + 0.2;

# Maximum x1
x1 - 4.0;

# Minimum x2
-x2 + 0.1;

# Maximum x2
x2 - 1.6;

# N. of variables
4

SUBSTITUTED, 1.000, C1;
SUBSTITUTED, 0.124, C2;
INDEPENDENT, 1.5, x1;
INDEPENDENT, 0.5, x2;
38 bar truss - member sizing

Stress and displacement constraints
CONCLUSIONS

• Code maintenance

• Efficiency and accuracy in the evaluation of derivatives

• Easy inclusion of alternative numerical techniques

• Not efficient in the OO manipulation of the Hessian matrix

• Friendly user interface is still required