Constrained NLP: Project description

Salil Sharma
IE 538 Nonlinear Optimization Algorithms and Models
Spring 2017
Purdue University

Contents

1 Introduction

To solve the constrained nonlinear optimization problem with equality constraints, Sequential Quadratic Programing (SQP) is implemented in the MATLAB.

2 Sequential Quadratic Programing

Algorithm (Local SQP Algorithm)
1: Choose an initial pair (x^0);
2: set k $\leftarrow$ 0 and λ^0;
3: **repeat** until a convergence test is satisfied
4: Evaluate $f_k, \bigtriangledown f_k, \bigtriangledown^2_{xx}\mathcal{L}, c_k$ and A_k;
5: Solve for p^k and λ^k;
6: Set $x^{k+1} \leftarrow x^k + p^k$ and $\lambda^{k+1} \leftarrow \lambda^k$
7: **end (repeat)**

2.1 Analysis of algorithm

Line 1: We choose initial guess of the solution in the form of x^0.
Line 2: Start with $k = 0$ iteration.
Line 3: Iterate the loop body until a convergence is reached. If $\bigtriangledown\mathcal{L}$ is within our specified tolerance, we terminate the loop an state that we have reached a critical point.
Line 4: To compute $\bigtriangledown^2_{xx}\mathcal{L}$, the Lagrange is separated in to two parts: objective function and constraint functions. Then both parts are added up.
Line 5: p^k and λ^k are solved using Gaussian Elimination. The equation is as follows:

$$\begin{bmatrix} \bigtriangledown^2_{xx}\mathcal{L} & -A_k^T \\ A_k & 0 \end{bmatrix} \begin{bmatrix} p^k \\ \lambda^{k+1} \end{bmatrix} = \begin{bmatrix} -\bigtriangledown f_k \\ -c_k \end{bmatrix}$$

Line 6: Update the iterates.
Line 7: Repeat until the loop satisfies the termination criterion.

3 MATLAB code

This section describes all the m-files which were developed to implement SQP algorithm. The zip file contains following files.

1. cnlp.m
   It solves non-linear optimization with equality constraints using Sequential Quadratic Programing. It requires three inputs: initial guess (column vector), objective function, and the equality constraints separated by ';' in nx1 matrix. It outputs the both the optimal solution and the function value at the optimal point

2. solveHessian.m
   It produces the Hessian matrix for a function. It requires the input vector as well as the function for which Hessian to be computed.
3. LagHessian.m
   It produces a structure array of Hessian matrices computed from the equality constraints.

4. gradientfd.m
   It produces the gradient of a function using forward difference method.

5. LagGrad.m
   Since equality constraints forms a column vector during the constraint input, this function evaluates the gradient vector separately for all the constraints and arranges them in a column of the matrix.

6. GE1.m
   It solves the system of equation of the form $Ax = b$ using Gaussian Elimination with back substitution.

4 Execution of MATLAB code

The MATLAB code is called from the command line by calling:

```
>> [optsol, optofv] = cnlp(x0, @(x) objfn(x), @(x) consfn(x))
```

5 Results

Choose $n = 3$.
$x^0 = [0.2; 0.2; 0.1]$ (Initial guess)

5.1 r=2

The following function is called at the terminal:

```
>> [optsol,optofv]=cnlp([0.2;0.2;0.1],@(x) -x(1)^2+x(2)^2-x(3)^2,@(x)[([x(1),x(2),x(3)]...
*[x(1);x(2);x(3)])-4;([x(1)-1,x(2),x(3)]*[1,0,0;0,2,0;0,0,1]*[x(1)-1;x(2);x(3)])-4])

optsol =

    0.5000
   -0.0000
    1.9365


optofv =

   -4.0000

Elapsed time is 0.006066 seconds.
```

5.2 r=3

```
optsol =

    0.5000
   -0.0001
    1.9365


optofv =

   -7.3868

Elapsed time is 0.015943 seconds.
```

5.3 r=4

```
optsol =

    0.5000
   -0.0001
    1.9365


optofv =

  -14.1250

Elapsed time is 0.011325 seconds.
```

5.4 r=5

```
optsol =

    1.4495
```

```
   -1.3780
   -0.0001


optofv =

  -11.3678

Elapsed time is 0.012842 seconds.
```

5.5 r=6

```
optsol =

    1.4495
   -1.3780
   -0.0001


optofv =

   -2.4266

Elapsed time is 0.282015 seconds.
```

5.6 r=k

5.6.1 n=3

```
optsol =

    0.5000
   -0.0000
    1.9365


optofv =

   -7.7618

Elapsed time is 0.035374 seconds.
```

5.6.2 n=4

$x^0 = [0.2; 0.2; 0.1; 0.1]$ (Initial guess)

```
optsol =

    0.5000
   -0.0001
    1.9365
```

```
   -0.0001


optofv =

   -7.7618

Elapsed time is 0.020302 seconds.
```

5.6.3 n=5

$x^0 = [0.2; 0.2; 0.1; 0.1; 0.1]$ (Initial guess)

```
optsol =

    1.3165
   -0.0001
   -0.7961
    1.2779
   -0.0000


optofv =

    1.8548

Elapsed time is 0.028902 seconds.
```

6 Discussion

The algorithm works up to $r = 5$.
For the cases, $r = k$, the algorithm works up to n=4.

7 Reference

Mikosch, Thomas V., Wright Stephen J, Nocedal Jorge, and Springerlink (Online Service). *Numerical Optimization*. N.p.: New York, NY : Springer New York, 2006. Print.