

## 在 F# 语言中使用 NAG .NET 算法库求解优化问题

NAG 最近推出新产品 NAG .NET 算法库 (<http://cn.nag-gc.com/numeric/DT/DTdescription.asp>)。这是我们第一个支持 .NET 环境的版本，其中包含了 400 多个重要的数学与统计方法 (method)，有小波转换、积分、插值与估计、随机数生成、时间序列分析与优化。优化的部分包含了许多求解 LP、QP、LS 与 NLP 问题的函数，也同时提供有约束式与无约束式的问题求解函数。同时也包含了全局优化函数。

在以下的示例中，我们在 F# 中使用 .NET 算法库求解 LS 优化问题。我们求解的问题是： $\frac{1}{2}\|b - Ax\|^2$ ，其中 A 是 10 x 9 的矩阵，b 是 10 x 1 的向量。变数 (x<sub>1</sub>,...,x<sub>9</sub>) 各有边界，且有三个约束式：

$$\begin{aligned}0 &\leq x_1 \leq 2 \\0 &\leq x_2 \leq 2 \\-\infty &\leq x_3 \leq 2 \\0 &\leq x_4 \leq 2 \\0 &\leq x_5 \leq 2 \\0 &\leq x_6 \leq 2 \\0 &\leq x_7 \leq 2 \\0 &\leq x_8 \leq 2 \\0 &\leq x_9 \leq 2\end{aligned}$$

$$\begin{aligned}2.0 &\leq x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + 4x_9 \leq \infty \\-\infty &\leq x_1 + 2x_2 + 3x_3 + 4x_4 - 2x_5 + x_6 + x_7 + x_8 + x_9 \leq 2.0 \\1.0 &\leq x_1 - x_2 + x_3 - x_4 + x_5 + x_6 + x_7 + x_8 + x_9 \leq 4.0\end{aligned}$$

程序片段：

```
open System
open NagLibrary

//number of variables
let n: int = 9;;

//number of rows in the matrix A
let m: int = 10;;

//number of general constraints
let nclin: int = 3;;

//the vector of observations
let b = [| 1.0; 1.0; 1.0; 1.0; 1.0; 1.0; 1.0; 1.0; 1.0; 1.0|];;

//lower bounds on the variables and on the constraints
let bl = [| 0.0; 0.0; -1.0e25; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0; 2.0; -1.0e25; 1.0|];;

//upper bounds on the variables and on the constraints
let bu = [| 2.0; 2.0; 2.0; 2.0; 2.0; 2.0; 2.0; 2.0; 2.0; 1.0e25; 2.0; 4.0 |];;

//starting point
let x = [|0.0; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0; 0.0|];;

//matrix A
let a = array2D [ [1.0; 1.0; 1.0; 1.0; 1.0; 1.0; 1.0; 1.0; 1.0; 1.0];
                  [1.0; 2.0; 1.0; 1.0; 1.0; 1.0; 2.0; 0.0; 0.0];
                  [1.0; 1.0; 3.0; 1.0; 1.0; 1.0; -1.0; -1.0; -3.0];
                  [1.0; 1.0; 1.0; 4.0; 1.0; 1.0; 1.0; 1.0; 1.0];
                  [1.0; 1.0; 1.0; 3.0; 1.0; 1.0; 1.0; 1.0; 1.0];
```

```

        [1.0; 1.0; 2.0; 1.0; 1.0; 0.0; 0.0; 0.0; -1.0];
        [1.0; 1.0; 1.0; 1.0; 0.0; 1.0; 1.0; 1.0; 1.0];
        [1.0; 1.0; 1.0; 0.0; 1.0; 1.0; 1.0; 1.0; 1.0];
        [1.0; 1.0; 0.0; 1.0; 1.0; 1.0; 2.0; 2.0; 3.0];
        [1.0; 0.0; 1.0; 1.0; 1.0; 1.0; 0.0; 2.0; 2.0]]

// 3 general constraints
let c = array2D [[1.0; 1.0; 1.0; 1.0; 1.0; 1.0; 1.0; 1.0; 4.0];
                [1.0; 2.0; 3.0; 4.0; -2.0; 1.0; 1.0; 1.0; 1.0];
                [1.0; -1.0; 1.0; -1.0; 1.0; 1.0; 1.0; 1.0; 1.0]]

//on exit: it includes the Lagrange multipliers for each constraint
let clamda = Array.create (n+nclin) 0.0;;

//includes the explicit linear term of the objective function. Not referenced in this
//example
let cvec = Array.create n 0.0;;

//on exit: defines the order of the columns of a with respect to the ordering of x,. //Not
referenced in this problem
let kx = Array.create n 0;;

//specifies the status of the constraints at the start of the feasibility phase. only
//referenced for a warm start
let istate = Array.create (n+nclin) 0;;

//on exit: Errors or warnings detected by the method
let ifail = ref -1;;

//on exit: the value of the objective function at x
let objf = ref 0.0;;

//on exit: the total number of iterations performed
let iter = ref 0;;

//an object used to configure optional parameters to this method
let options = new E04.e04ncOptions();;

options.Set("List");;
options.Set("Print level = 10");;

//method used to solve the problem
E04.e04nc(m, n, nclin, c, bl, bu, cvec, istate, kx, x, a, b, iter, objf, clamda,
options, ifail)

```

输出:

Calls to E04NEF  
 -----

List  
 Print level = 10

\*\*\* E04NCF

Parameters  
 -----

Problem type.....	LS1	Hessian.....	NO
Linear constraints....	3	Feasibility tolerance..	1.05E-08
Variables.....	9	Crash tolerance.....	1.00E-02
Objective matrix rows..	10	Rank tolerance.....	1.11E-14
Infinite bound size....	1.00E+20	COLD start.....	
Infinite step size....	1.00E+20	EPS (machine precision)	1.11E-16
Print level.....	10	Feasibility phase itns.	60

Monitoring file..... -1 Optimality phase itns. 60

Workspace provided is IWORK( 100), WORK( 1000).  
To solve problem we need IWORK( 9), WORK( 261).

Rank of the objective function data matrix = 6

Itn	Step	Ninf	Sinf/Objective	Norm Gz
0	0.0E+00	1	2.145500E+00	0.0E+00
1	2.5E-01	1	1.145500E+00	0.0E+00
2	3.8E-01	0	6.595685E+00	2.3E+01
3	1.0E-01	0	5.342505E+00	1.9E+01
4	7.1E-02	0	4.616975E+00	2.2E+00
5	1.0E-01	0	4.558492E+00	1.3E+00
6	1.0E+00	0	4.523485E+00	9.8E-16
7	3.5E-01	0	1.934106E+00	6.9E+00
8	2.1E-01	0	1.323283E+00	5.1E+00
9	1.4E-02	0	1.307479E+00	0.0E+00
10	1.0E+00	0	9.153991E-01	5.3E-15
11	6.1E-01	0	2.190278E-01	5.9E-01
12	1.0E+00	0	1.652065E-01	2.2E-15
13	1.0E+00	0	9.605160E-02	2.2E-15
14	3.0E-02	0	9.236999E-02	4.5E-01
15	1.0E+00	0	8.134082E-02	8.3E-16

Exit from LS problem after 15 iterations.

Varbl	State	Value	Lower Bound	Upper Bound	Lagr Mult	Slack
V 1	LL	0.00000	.	2.00000	0.1572	.
V 2	FR	4.152607E-02	.	2.00000	.	4.1526E-02
V 3	FR	0.587176	None	2.00000	.	1.413
V 4	LL	0.00000	.	2.00000	0.8782	.
V 5	FR	9.964323E-02	.	2.00000	.	9.9643E-02
V 6	LL	0.00000	.	2.00000	0.1473	.
V 7	FR	4.905781E-02	.	2.00000	.	4.9058E-02
V 8	LL	0.00000	.	2.00000	0.8603	.
V 9	FR	0.305649	.	2.00000	.	0.3056

L Con	State	Value	Lower Bound	Upper Bound	Lagr Mult	Slack
L 1	LL	2.00000	2.00000	None	0.3777	-4.4409E-16
L 2	UL	2.00000	None	2.00000	-5.7914E-02	2.2204E-16
L 3	LL	1.00000	1.00000	4.00000	0.1075	4.4409E-16

Exit E04NCF - Optimal LS solution.

Final LS objective value = 0.8134082E-01

我们还提供了更多的例子，说明如何在 F# 中调用 NAG .NET 算法库。您可参考 <http://cn.nag-gc.com/numeric/DT/fsharp/>。

欲进一步了解本技术文章，请联系：

刘泰兴 - NAG 大中华区分公司 技术总监 [ted@nag-gc.com](mailto:ted@nag-gc.com)

Tel: +886 (02)2509 3288