COMPGV19: Tutorial 4

Table of Contents

Marta Betcke and Kiko Rul#lan

Exercise 2

Implement the Fletcher-Reeves method and the Polak-Ribiere method using the strong Wolfe conditions for the line search.

```
clear all, close all;
```

Rosenbrock function

```
% For computation define as function of 1 vector variable
F.f = @(x) x(1)^2 + 5*x(1)^4 + 10*x(2)^2;
F.df = @(x) [2*x(1) + 20*x(1)^3; 20*x(2)];
F.d2f = @(x) [2 + 60*x(1)^2, 0; 0, 20];

% For visualisation proposes define as a function of 2 variables (x,y)
FV.f = @(x,y) x.^2 + 5*x.^4 + 10.*y.^2;
FV.dfx = @(x,y) 2*x + 20*x.^3;
FV.dfy = @(x,y) 20*y;
FV.d2fxx = @(x,y) 2 + 60*x.^2;
FV.d2fxy = @(x,y) 0;
FV.d2fyx = @(x,y) 0;
FV.d2fyy = @(x,y) 20;
rosenbrock = FV.f;
```

```
% Initialisation
alpha0 = 1;
c1 = 1e-4;
tol = 1e-12;
maxIter = 500;

%==============================
% Points x0 = [1.2; 1.2], -[1.2,1.2]
%==============================
%x0 = [4; 4];
x0 = [1.2; 1.2];
```

Conjugate gradient with backtracking

```
lsOptsCG_BT.rho = 0.1;
lsOptsCG_BT.c1 = c1;
lsFun = @(x_k, p_k, alpha0) backtracking(F, x_k, p_k, alpha0,
 lsOptsCG_BT);
[xCG_FR_BT, fCG_FR_BT, nIterCG_FR_BT, infoCG_FR_BT] =
 nonlinearConjugateGradient(F, lsFun, 'FR', alpha0, x0, tol, maxIter)
[xCG_PR_BT, fCG_PR_BT, nIterCG_PR_BT, infoCG_PR_BT] =
 nonlinearConjugateGradient(F, lsFun, 'PR', alpha0, x0, tol, maxIter)
```

```
xCG_FR_BT =

   1.0e-12 *

    0.2724
    0.0428


fCG_FR_BT =

   9.2467e-26


nIterCG_FR_BT =

   380


infoCG_FR_BT =

        xs: [2x381 double]
    alphas: [1x381 double]
     betas: [1x380 double]


xCG_PR_BT =

   1.0e-12 *

    0.3914
   -0.0183


fCG_PR_BT =

   1.5651e-25


nIterCG_PR_BT =

    92


infoCG_PR_BT =
        xs: [2x93 double]
    alphas: [1x93 double]
     betas: [1x92 double]
```

Conjugate gradient with line search satisfying strong Wolfe condition

```
lsOptsCG_LS.c1 = c1;
lsOptsCG_LS.c2 = 0.1;
lsFun = @(x_k, p_k, alpha0) lineSearch(F, x_k, p_k, alpha0,
 lsOptsCG_LS);
[xCG_FR_LS, fCG_FR_LS, nIterCG_FR_LS, infoCG_FR_LS] =
 nonlinearConjugateGradient(F, lsFun, 'FR', alpha0, x0, tol, maxIter)
[xCG_PR_LS, fCG_PR_LS, nIterCG_PR_LS, infoCG_PR_LS] =
 nonlinearConjugateGradient(F, lsFun, 'PR', alpha0, x0, tol, maxIter)
```

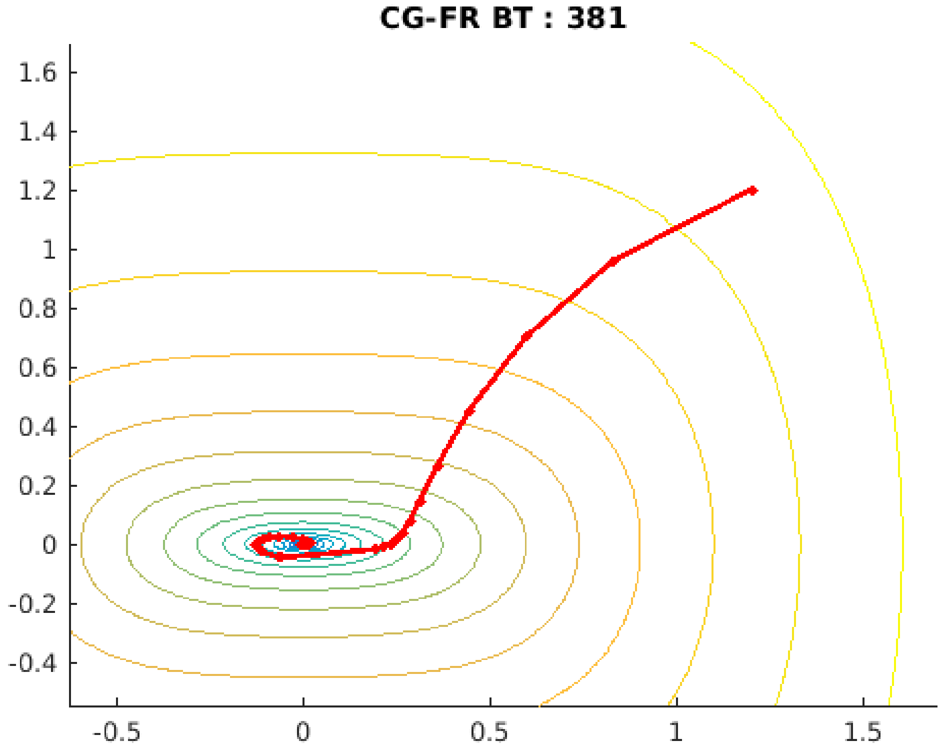
```
xCG_FR_LS =

   1.0e-12 *

   -0.3081
   -0.0173


fCG_FR_LS =

   9.7922e-26


nIterCG_FR_LS =

    32


infoCG_FR_LS =

        xs: [2x33 double]
    alphas: [1x33 double]
     betas: [1x32 double]


xCG_PR_LS =

   1.0e-13 *

   -0.5003
    0.1204


fCG_PR_LS =
   3.9531e-27


nIterCG_PR_LS =

    22


infoCG_PR_LS =

        xs: [2x23 double]
    alphas: [1x23 double]
     betas: [1x22 double]
```

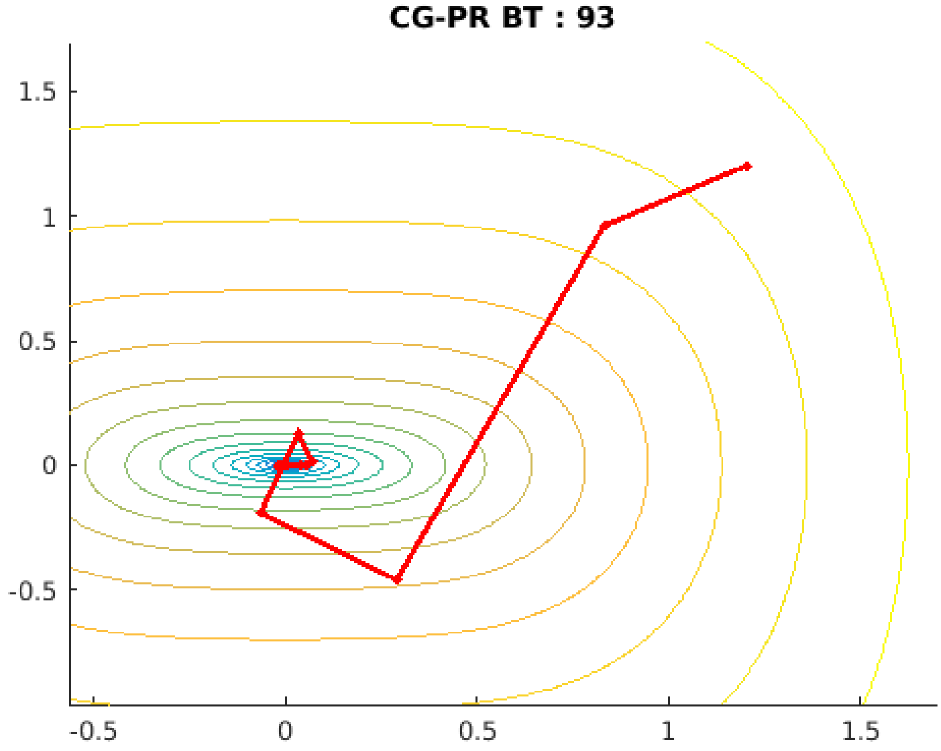
```
%==============================
```

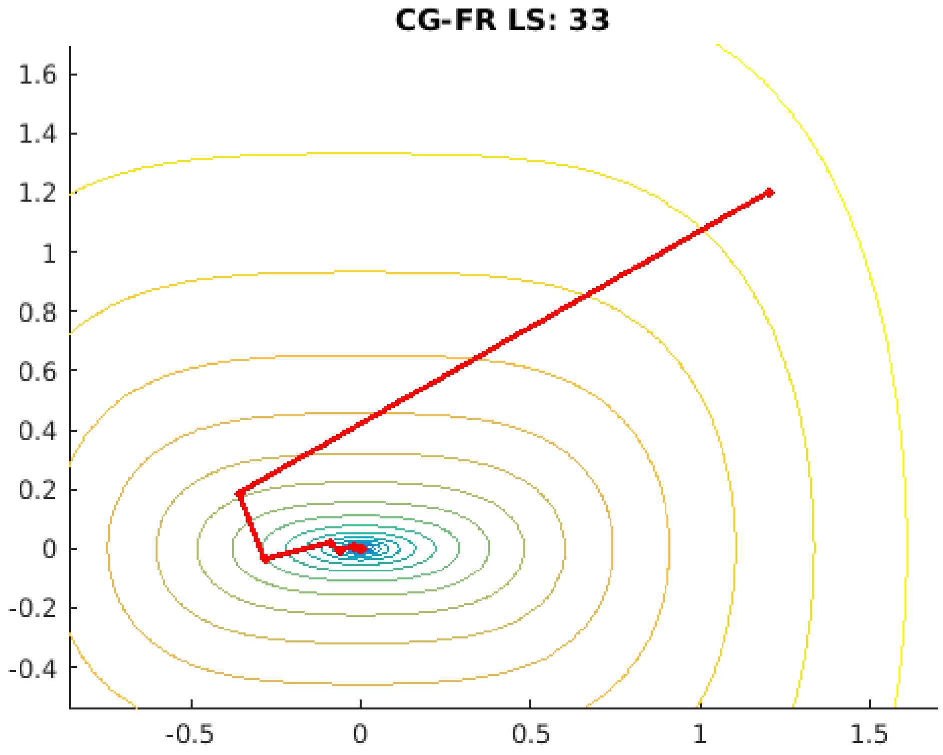
Visualisation

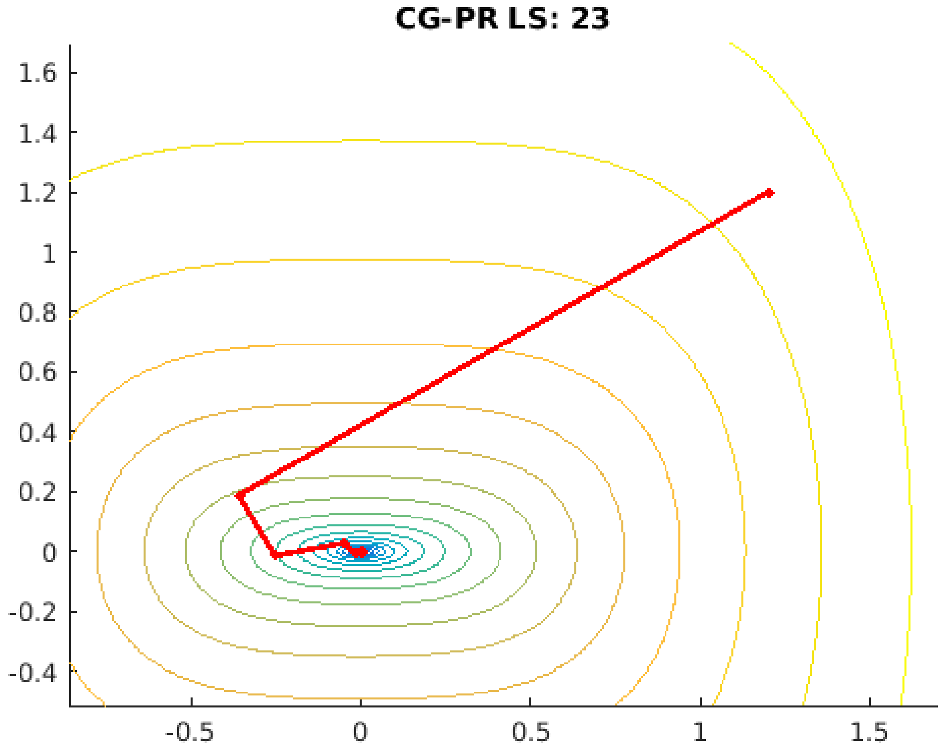
```
%==============================
% Backtracking - FR
n = 300;
x =
 linspace(min(infoCG_FR_BT.xs(1,:))-0.5,max(infoCG_FR_BT.xs(1,:))+0.5,n
+1);
y =
 linspace(min(infoCG_FR_BT.xs(2,:))-0.5,max(infoCG_FR_BT.xs(2,:))+0.5,n
+1);
[X,Y] = meshgrid(x,y);
Z = rosenbrock(X,Y);
% Iterate plot
visualizeConvergence(infoCG_FR_BT,X,Y,log(Z),'final'); title(['CG-FR
 BT : ' num2str(size(infoCG_FR_BT.xs,2))])
saveas(gcf, '../figs/02_01_CG-FR-BT', 'png');

% Backtracking - PR
n = 300;
x =
 linspace(min(infoCG_PR_BT.xs(1,:))-0.5,max(infoCG_PR_BT.xs(1,:))+0.5,n
+1);
y =
 linspace(min(infoCG_PR_BT.xs(2,:))-0.5,max(infoCG_PR_BT.xs(2,:))+0.5,n
+1);
[X,Y] = meshgrid(x,y);
Z = FV.f(X,Y);
% Iterate plot
visualizeConvergence(infoCG_PR_BT,X,Y,log(Z),'final'); title(['CG-PR
 BT : ' num2str(size(infoCG_PR_BT.xs,2))])
saveas(gcf, '../figs/02_02_CG-PR-BT', 'png');

% Line search satisfying strong Wolfe condition - FR
n = 300;
```

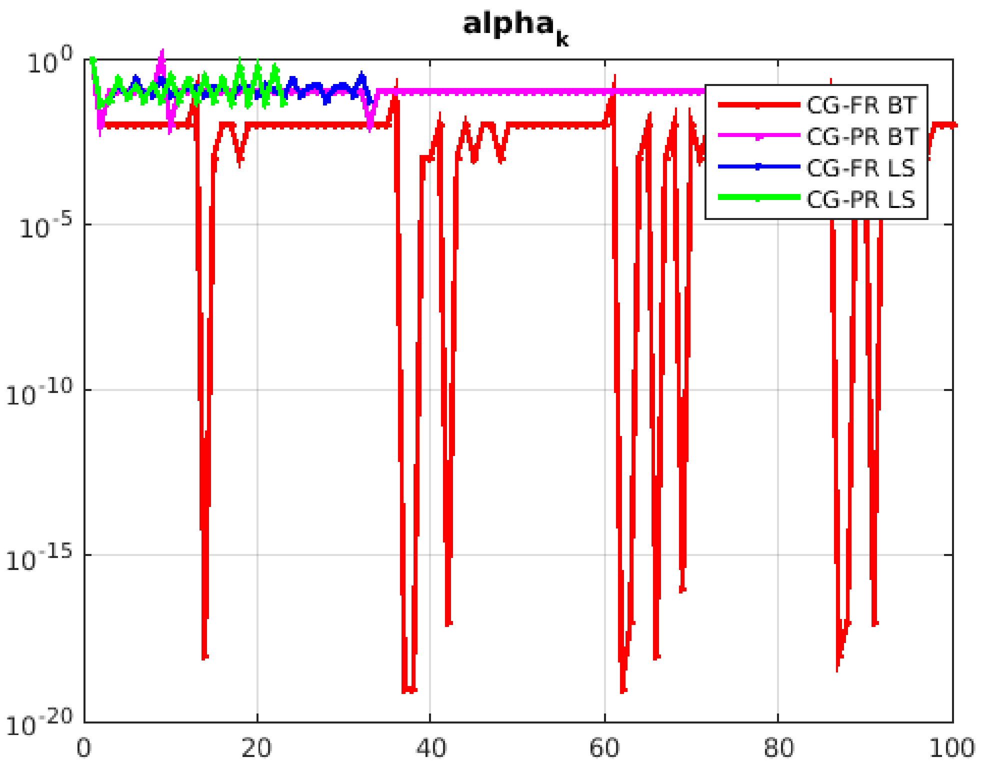
```
x =
 linspace(min(infoCG_FR_LS.xs(1,:))-0.5,max(infoCG_FR_LS.xs(1,:))+0.5,n
+1);
y =
 linspace(min(infoCG_FR_LS.xs(2,:))-0.5,max(infoCG_FR_LS.xs(2,:))+0.5,n
+1);
[X,Y] = meshgrid(x,y);
Z = FV.f(X,Y);
% Iterate plot
visualizeConvergence(infoCG_FR_LS,X,Y,log(Z),'final'); title(['CG-FR
 LS: ' num2str(size(infoCG_FR_LS.xs,2))])
saveas(gcf, '../figs/02_03_CG-FR-LS', 'png');

% Line search satisfying strong Wolfe condition - PR
n = 300;
x =
 linspace(min(infoCG_PR_LS.xs(1,:))-0.5,max(infoCG_PR_LS.xs(1,:))+0.5,n
+1);
y =
 linspace(min(infoCG_PR_LS.xs(2,:))-0.5,max(infoCG_PR_LS.xs(2,:))+0.5,n
+1);
[X,Y] = meshgrid(x,y);
Z = FV.f(X,Y);
% Iterate plot
visualizeConvergence(infoCG_PR_LS,X,Y,log(Z),'final'); title(['CG-PR
 LS: ' num2str(size(infoCG_PR_LS.xs,2))])
saveas(gcf, '../figs/02_04_CG-PR-LS', 'png');

% Step length plot
figure;
semilogy(infoCG_FR_BT.alphas(1:100), '-or', 'LineWidth',
 2, 'MarkerSize', 2); hold on;
semilogy(infoCG_PR_BT.alphas, '-om', 'LineWidth', 2, 'MarkerSize', 2);
semilogy(infoCG_FR_LS.alphas, '-ob', 'LineWidth', 2, 'MarkerSize', 2);
semilogy(infoCG_PR_LS.alphas, '-og', 'LineWidth', 2, 'MarkerSize', 2);
saveas(gcf, '../figs/02_05_Steplength', 'png');

grid on;
title('alpha_k');
legend('CG-FR BT', 'CG-PR BT', 'CG-FR LS', 'CG-PR LS');
```


CG-FR BT : 381
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.5
0
0.5
1
1.5


CG-PR BT : 93
1.5
1
0.5
0
-0.5
-0.5
0
0.5
1
1.5


CG-FR LS: 33
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.5
0
0.5
1
1.5


CG-PR LS: 23
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.5
0
0.5
1
1.5



Published with MATLAB® R2015a