Gradient, Newton and conjugate direction methods for unconstrained nonlinear optimization

Consider the gradient method (steepest descent), with exact unidimensional search, the Newton method and the conjugate direction methods with the Fletcher-Reeves (FR) and Polak-Ribière (PR) updates.

Let $\varepsilon > 0$ be the given tolerance value. As a stopping criterion, use the condition $||\nabla f(\underline{x}_k)||_2 < \varepsilon$, where $\nabla f(\underline{x}_k)$ is the gradient of the function f computed in $\underline{x}_k$.

a) Implement the gradient method.

b) Implement the Newton method.

c) Implement the conjugate direction method with the Fletcher-Reeves (FR) and Polak-Ribière (PR) updates.

d) Homework: implement the quasi-Newton DFP method and compare it to the other methods on the two functions.

Implementation sketch

- File `descentmethod.m`: "stub" of a generic descent method. Find a local minimum $\underline{x}^*$ (with value $f(\underline{x})^*$) of the function f starting from an initial point $\underline{x}_0$, given a tolerance $\varepsilon > 0$ and an iteration limit. Return the number of iterations (`counter`) and the norm of the gradient $||\nabla f(\underline{x}^*)||_2$ in the last solution (`error`). The variables `xks` and `fks` contain the list of solutions found at each iteration and their corresponding objective function

values (useful to represent graphically the results).

```
% Stub of a descent method
function [xk, fk, counter, error, xks, fks] = ...
    descentmethod(f, x0, epsilon, maxiterations)

  xks = [x0'];
  fks = [feval(f,x0)];

  xk = x0;
  counter = 0;
  error = 1e300;

  while error > epsilon && counter < maxiterations

    counter = counter + 1;
    % d =
    alpha = fminsearch(@(a) feval(f,xk + a*d), 0.0); %exact 1-d opt
    xk = xk + alpha*d;

    error = norm(grad(f,xk));
    fk = feval(f,xk);

    xks = [xks; xk'];
    fks = [fks; fk];

  end

end %of function
```

- FILE `grad.m`: numerical estimation of the gradient $\nabla f(\underline{x})$ of f in the point $\underline{x}$.

```
% Gradient of a function at a point
function gradf = grad(f, x)
  h = 0.0001;
  n = length(x);
  gradf = zeros(n,1);
  for i = 1:n
    delta = zeros(n, 1); delta(i) = h;
    gradf(i) = ( feval(f, x+delta) - feval(f,x) ) / h;
  end
end %of function
```

FILE `hes.m`: numerical estimation of the Hessian $\nabla^2 f(\underline{x})$ of f in $\underline{x}$.

```
% Hessian of a function at a point
function H = hes(f, x)

  h = 0.0001;
  n = length(x);
  H = zeros(n,n);

  for i = 1:n
    delta_i = zeros(n, 1); delta_i(i) = h;
    for j = 1:n
      delta_j = zeros(n, 1); delta_j(j) = h;
      H(i,j) = (feval(f,x+delta_i+delta_j) - feval(f,x+delta_i) ...
          - feval(f,x+delta_j) + feval(f,x)) / (h^2);
    end
  end

end %end of function
```

- FILE `contour_stub.m`: plots the level curves of the bivariate function `myFunction` on the domain $[x_{lb}, x_{ub}] \times [y_{lb}, y_{ub}]$.

```
f = @myFunction;
%plot the contour of the function
[X,Y] = meshgrid(x_lb:x_step:x_ub, y_lb:y_step:y_ub);
Z = zeros(size(X,1),size(X,2));
for i = 1:size(X,1)
    for j = 1:size(X,2)
       Z(i,j) = feval(f,[X(i,j);Y(i,j)]);
    end
end
figure; contour(X,Y,Z,50);
hold on;
```

Exercise 1: quadratic strictly convex function

Consider the problem:

$$\min f(x_1, x_2) = \frac{1}{2}x_1^2 + \frac{a}{2}x_2^2$$

with $a \geq 1$.

a) What do you know about the two conjugate direction variants on this kind of problem?

b) Solve the problem with $a = 4$ and $a = 16$, starting from $\underline{x}_0 = (a, 1)$, and compare the sequence of points generated by the four methods.

The code for the function with $a = 4$ is in `f_ex_4.m`

```
function f = f_ex1_4(x)
  f = 0.5*x(1)^2 + 4/2*x(2)^2;
end
```

and with $a = 16$ in `f_ex_16.m`

```
function f = f_ex1_16(x)
  f = 0.5*x(1)^2 + 16/2*x(2)^2;
end
```

Exercise 2: Rosenbrock function

Consider the following Rosenbrock function

$$f(\underline{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2.$$

It is a quadratic function, nonconvex, often used to test the convergence of nonlinear optimization algorithms.

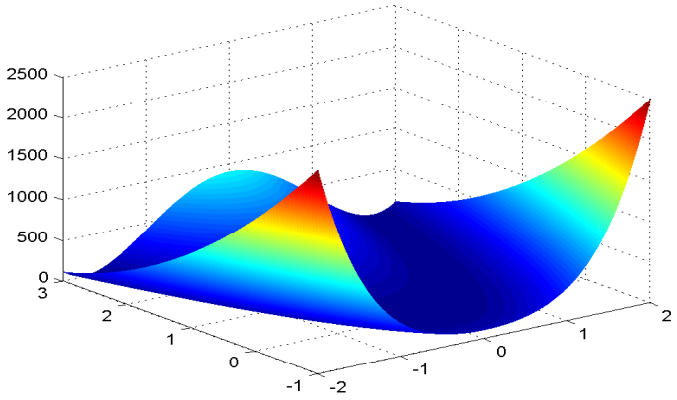


Figure 1: Rosenbrock function

a) Find analytically the only global optimum.

b) Observe the behavior of the gradient method, starting from the initial points: $\underline{x}'_0 = (-2, 2)$, $\underline{x}''_0 = (0, 0)$ and $\underline{x}'''_0 = (-1, 0)$, using a maximum number of 50, 500 and 1000 iterations.

c) Compare the sequence of solutions found by the four methods.

- FILE `f_rosenbrock.m`

```
function f = f_rosenbrock(x)
  f = 100*(x(2)-x(1)^2)^2 + (1-x(1))^2;
end
```

Solution

Implementation of the four methods

a) File `steepestdescent.m`: gradient method.

```
% Steepest descent method
function [xk, fk, counter, error, xks, fks] = ...
    steepestdescent(f, x0, epsilon, maxiterations)

  xks = [x0'];
  fks = [feval(f,x0)];

  xk = x0;
  counter = 0;
  error = 1e300;

  while error > epsilon && counter < maxiterations
    counter = counter + 1;
    d = -grad(f, xk);
    alpha = fminsearch(@(a) feval(f,xk + a*d), 0.0);
    xk = xk + alpha * d;

    error = norm(grad(f,xk));
    fk = feval(f,xk);

    xks = [xks; xk'];
    fks = [fks; fk];
  end

end %of function
```

b) File `newton.m`: Newton method

```
% Newton method
function [xk, fk, counter, error, xks, fks] = ...
    newton(f, x0, epsilon, maxiterations)

  xks = [x0'];
  fks = [feval(f,x0)];
  xk = x0;
  counter = 0;
  error = 1e300;

  while error > epsilon && counter < maxiterations

    counter = counter + 1;
    gradF = grad(f, xk);
    H = hes(f, xk);
    d = -inv(H)*gradF;
    alpha = 1;
    xk = xk + alpha * d;

    error = norm(grad(f,xk));
    fk = feval(f,xk);

  xks = [xks; xk'];
  fks = [fks; fk];
  end

end
```

c) File `cg_fr.m`: conjugate direction method, FR version.

```
% Fletcher-Reeves conjugate gradient method
function [xstar, fstar, counter, error, xks, fks] = ...
    cg_fr(f, x0, epsilon, maxiterations)

  x = x0;

  counter = 0;  error = 1e300;

  xks = [x'];
  fks = [feval(f, x)];

  gradF = grad(f, x);

  d = -gradF;

  while error > epsilon && counter < maxiterations

    counter = counter + 1;

    alpha = fminsearch(@(a) feval(f,x + a*d), 0.0);

    x = x + alpha * d;

    xks = [xks; x'];
    fks = [fks; feval(f, x)];

    error = norm(d);

    gradFp = gradF;

    gradF = grad(f, x);

    beta = (gradF'*gradF)/(gradFp'*gradFp);

    d = -gradF + beta * d;

  end

  xstar = x;
  fstar = feval(f, x);

end
```

File `cg_pr.m`: conjugate direction method, PR version.

```
% Polak-Ribiere conjugate gradient method
function [xstar, fstar, counter, error, xks, fks] = ...
    cg_pr(f, x0, epsilon, maxiterations)

  x = x0;

  counter = 0;  error = 1e300;

  xks = [x'];
  fks = [feval(f, x)];

  gradF = grad(f, x);

  d = -gradF;

  while error > epsilon && counter < maxiterations

    counter = counter + 1;

    alpha = fminsearch(@(a) feval(f,x + a*d), 0.0);

    x = x + alpha * d;

    xks = [xks; x'];
    fks = [fks; feval(f, x)];

    gradFp = gradF;

    gradF = grad(f, x);

    error = norm(gradF);

    beta = gradF'*(gradF-gradFp)/(gradFp'*gradFp);
    %beta = max(gradF'*(gradF-gradFp)/(gradFp'*gradFp),0);
    %beta = abs(gradF'*(gradF-gradFp))/(gradFp'*gradFp);

    d = -gradF + beta * d;

  end

  xstar = x;
  fstar = feval(f, x);


end
```

Exercize 1: minimization of a convex quadratic function

a) For convex quadratic functions, with an exact 1-d search, the two conjugate direction methods (FR, PR) generate the same sequence of Q-conjugate direction, being equivalent to the conjugate gradient method.

b) Let us run the methods with the script `run_ex1_16.m` for $a = 16$:

```
f = @f_ex1_4;

[X,Y] = meshgrid(-4:.01:4, -1:.01:1);
Z = zeros(size(X,1),size(X,2));

for i = 1:size(X,1)
    for j = 1:size(X,2)
       Z(i,j) = feval(f,[X(i,j);Y(i,j)]);
    end
end

figure; contour(X,Y,Z,30);

x0 = [4; 1];
eps = 0.01;
max_iter = 100;

[xstar, fstar, counter, error, xks, fks ] = steepestdescent(f, x0, eps, max_iter);

hold on; plot(xks(:,1), xks(:,2), 'r');
hold on; plot(xks(:,1), xks(:,2), 'r*');

[xstar, fstar, counter, error, xks, fks ] = cg_fr(f, x0, eps, max_iter);

hold on; plot(xks(:,1), xks(:,2), 'g');
hold on; plot(xks(:,1), xks(:,2), 'g*');

[xstar, fstar, counter, error, xks, fks ] = newton(f, x0, eps, max_iter);

hold on; plot(xks(:,1), xks(:,2), 'b');
hold on; plot(xks(:,1), xks(:,2), 'b*');
```

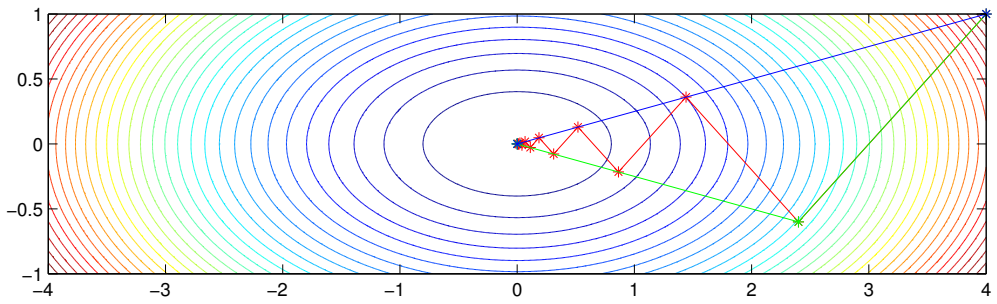


Figure 2: Gradient method (red), conjugate gradient (green) and Newton (blue) for $a = 4$. **Note:** the axis have different scales!

Let us run the methods with the script `run_ex1_16.m` for $a = 16$:

```
f = @f_ex1_16;

[X,Y] = meshgrid(-5:.01:20, -1:.01:1);
Z = zeros(size(X,1),size(X,2));

for i = 1:size(X,1)
    for j = 1:size(X,2)
        Z(i,j) = feval(f,[X(i,j);Y(i,j)]);
    end
end

figure; contour(X,Y,Z,30);

x0 = [16; 1];
eps = 0.01;
max_iter = 100;

[xstar, fstar, counter, error, xks, fks ] = steepestdescent(f, x0, eps, max_iter);

hold on; plot(xks(:,1), xks(:,2), 'r');
hold on; plot(xks(:,1), xks(:,2), 'r*');

[xstar, fstar, counter, error, xks, fks ] = cg_fr(f, x0, eps, max_iter);

hold on; plot(xks(:,1), xks(:,2), 'g');
hold on; plot(xks(:,1), xks(:,2), 'g*');

[xstar, fstar, counter, error, xks, fks ] = newton(f, x0, eps, max_iter);

hold on; plot(xks(:,1), xks(:,2), 'b');
hold on; plot(xks(:,1), xks(:,2), 'b*');
```

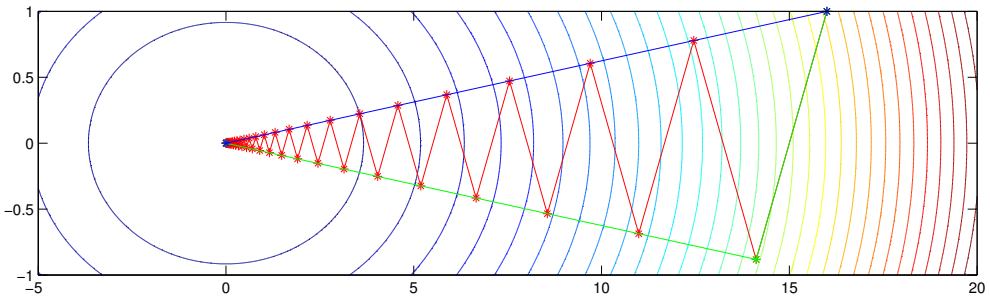


Figure 3: Gradient method (red), conjugate gradient (green) and Newton (blue) for $a = 16$. **Note:** the axis have different scales!

c) The method of conjugate direction converges in at most n iterations for convex quadratic forms. In this case, 2 iterations are sufficient. The quadratic form has eigenvalues $(\frac{1}{2}, \frac{a}{2})$. Recall that, for strictly convex quadratic function with matrix Q, the gradient method has linear convergence with rate $\left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}\right)$, where λ_n and λ_1 are the largest and smallest eigenvalues of Q. In this case, the convergence rate is $\left(\frac{a-1}{a+1}\right)$ which approaches 1 asyntotically as $a \to \infty$, justifying the slow convergence we observe.

Exercise 2: Rosenbrock function

a) The global optimum of the function is in $\underline{x}^* = (1, 1)$, as we can observe computing the points where the gradient is 0.

b) The script `run_ex2_grad.m` runs (for the point $(-2, 2)$) the gradient method on the considered function for 500 iterations, indicates the distance in norm 2 (*optimality gap*) from the optimal solution $\underline{x}^*$ and prints the sequence $\{\underline{x}_k\}$ of points on the level curve plot. Modify the script to run all the indicated experiments.

```
clear all
close all
f = @f_rosenbrock;

[X,Y] = meshgrid(-3:.01:3, -5:.01:6);
Z = zeros(size(X,1),size(X,2));

for i = 1:size(X,1)
    for j = 1:size(X,2)
        Z(i,j) = feval(f,[X(i,j);Y(i,j)]);
    end
end

%plot the function
figure;
surf(X,Y,Z,'FaceColor','interp','EdgeColor','none')
camlight left; lighting phong
hold on;
plot3(1,1,0,'*y');

%plot the contour of the function
figure; contour(X,Y,Z,50);
hold on;

%optimize --for 500 iterations, starting from x0
eps = 0.001;
x0 = [-2; 2];
max_iter = 500;

plot(x0(1),x0(2),'o');
plot(1,1,'k*');
[xstar, xstar, counter, error, xks, fks ] = steepestdescent(f, x0, eps, max_iter);

gap = norm(xstar-[1; 1]);
plot(xks(:,1), xks(:,2), 'r');
```

Table 1 reports the solution found with the gradient method, the number of iterations and the gap for the initial points. The experimental results show that the algorithm converges to a local optimum for any $\underline{x}_0$. The convergence speed, however, is highly sensitive to the choice of the initial solution.

Table 1: Results of the gradient method applied to the Rosenbrock function.

$\underline{x}_0$	it	$\underline{x}^*$	$\|\nabla f(\underline{x}^*)\|$	$\|\underline{x}^* - (1,1)\|$	$\|f(\underline{x}^*)\|$
(-2,2)	50	$(-0.0138, -0.0057)$	2.3709	1.4280	1.0313
(0,0)	50	$(0.4771, 0.2280)$	1.1124	0.9324	0.2734
(-1,0)	50	$(0.5297, 0.2795)$	0.7320	0.8604	0.2213
(-2,2)	500	$(0.9744, 0.9493)$	0.0317	0.0568	0.0007
(0,0)	500	$(0.9983, 0.9966)$	0.0406	0.0038	0.0000
(-1,0)	500	$(1.0082, 1.0165)$	0.0395	0.0184	0.0001

c) The script `run_ex2_newton.m` runs the Newton method on the considered problem:

```
f = @f_rosenbrock;

%plot the contour of the function
[X,Y] = meshgrid(-3:.01:3, -10:.01:5);
Z = zeros(size(X,1),size(X,2));

for i = 1:size(X,1)
    for j = 1:size(X,2)
        Z(i,j) = feval(f,[X(i,j);Y(i,j)]);
    end
end

figure; contour(X,Y,Z,50);
hold on;

%optimize
x0 = [0; 0];
eps = 0.001;
max_iter = 500;

%newton
[xstar, fstar, counter, error, xks, fks ] = newton(f, x0, eps, max_iter);

gap = norm(xstar-[1; 1])
plot(xks(:,1), xks(:,2), 'b');
plot(xks(:,1), xks(:,2), 'b*');
```

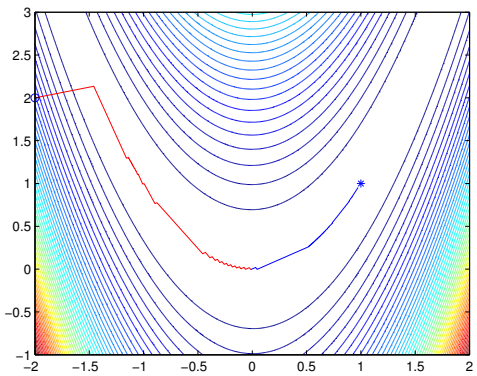
Table 2 reports the solution found with the Newton method, the number of iterations and the gap for each given initial points.

Table 2: Results of the Newton method applied to Rosenbrock function.

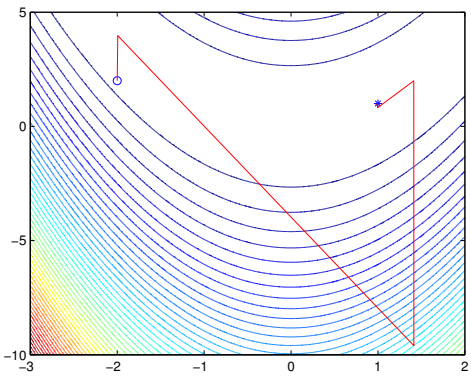
$\underline{x}_0$	it	$\underline{x}^*$	$\|\nabla f(\underline{x}^*)\|$	$\|\underline{x}^* - (1,1)\|$	$\|f(\underline{x}^*)\|$
(-2,2)	7	$(0.9715, 0.9437)$	$0.5030 \cdot 10^{-3}$	0.0631	$0.8143 \cdot 10^{-3}$
(0,0)	4	$(0.9715, 0.9438)$	$0.7094 \cdot 10^{-3}$	0.0630	$0.8111 \cdot 10^{-3}$
(-1,0)	7	$(0.9714, 0.9435)$	$0.0236 \cdot 10^{-3}$	0.0633	$0.8200 \cdot 10^{-3}$

The function is quadratic but not convex, so it requires more than one iteration. The convergence is still must faster than the gradient method.

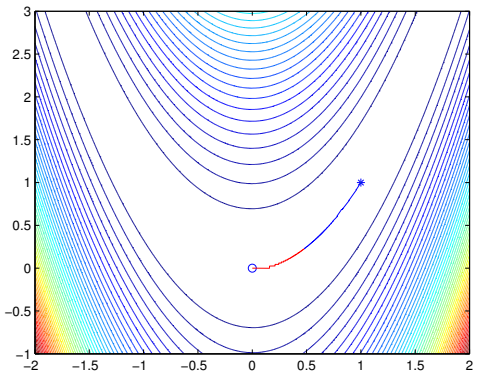
The succession $\{\underline{x}_k\}$ generated by the gradient and Newton methods are depitcted in Figure 4. Note how, considering the same initial solution $\underline{x}_0$, the successions $\{\underline{x}_k\}$ are significantly different. The gradient method requires many iterations and converges to $\underline{x}^*$ very slowly. Newton method converges to $\underline{x}^*$ in very few iterations. For points $\underline{x}_0 = (-2, 2)$ and $\underline{x}_0 = (-1, 0)$, observe that in some iterations, the direction $\underline{d}_k$ is not a *descent* direction.



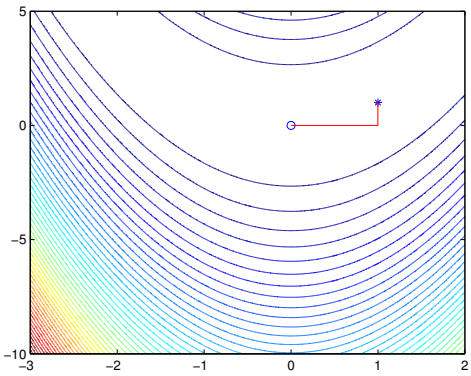
(a) $\underline{x}_0 = (-2, 2)$, gradient method.



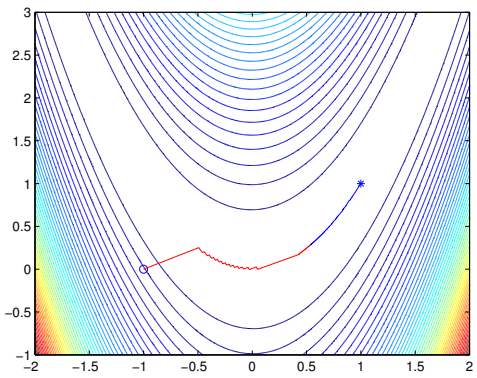
(b) $\underline{x}_0 = (-2, 2)$, Newton method.



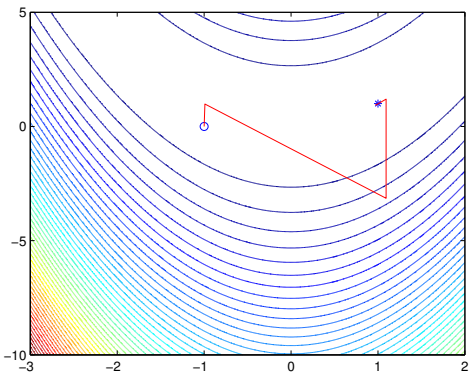
(c) $\underline{x}_0 = (0, 0)$, gradient method.



(d) $\underline{x}_0 = (0, 0)$, Newton method.



(e) $\underline{x}_0 = (-1, 0)$, gradient method.



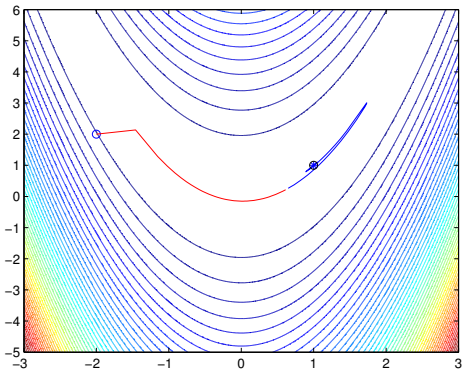
(f) $\underline{x}_0 = (-1, 0)$, Newton method.

Figure 4: Gradient and Newton method on Rosenbrock function: in red, points generated up to the 50-th iteration; in blue, points generated at iterations with index between 50 and 500.

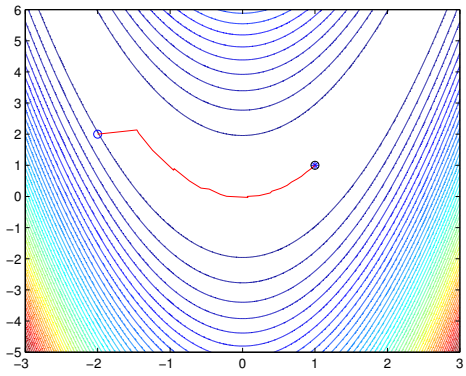
Let us run the conjugate direction methods with the script `run_ex2_cg.m`:

```
clear all
close all
f = @f_rosenbrock;

[X,Y] = meshgrid(-3:.01:3, -5:.01:6);

Z = zeros(size(X,1),size(X,2));

for i = 1:size(X,1)
    for j = 1:size(X,2)
        Z(i,j) = feval(f,[X(i,j);Y(i,j)]);
    end
end

%plot the function
%figure;
%surf(X,Y,Z,'FaceColor','interp','EdgeColor','none')
%camlight left; lighting phong
%hold on;
%plot3(1,1,0,'*y');

%plot the contour of the function
figure; contour(X,Y,Z,50);
hold on;

%optimize --for 500 iterations, starting rfom [-2;2]. Change the code for
%           the other cases
eps = 0.001;

x0 = [-1; 0];
max_iter = 500;

plot(x0(1),x0(2),'o');
plot(1,1,'b*');
[xstar, fstar, counter, error, xks, fks ] = cg_fr(f, x0, eps, max_iter);
%[xstar, fstar, counter, error, xks, fks ] = cg_pr(f, x0, eps, max_iter);

xstar
gap = norm(xstar-[1; 1])
counter
plot(xks(1:50,1), xks(1:50,2), 'r');
plot(xks(51:end,1), xks(51:end,2), 'b');
%plot(xks(:,1), xks(:,2), 'r*');


plot(1,1,'ko');
```

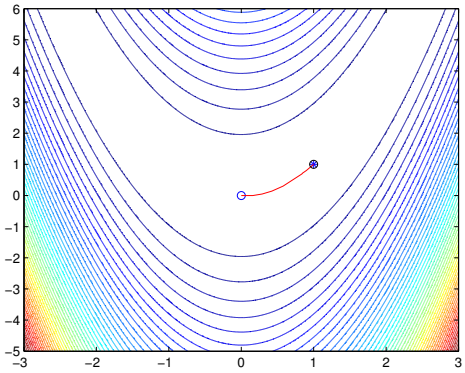
The succession $\{\underline{x}_k\}$ generated by the two variants are depicted in Figure 4. The behavior of the two methods appears similar for $\underline{x}_0 = (0, 0)$ and $\underline{x}_0 = (-1, 0)$, but there is a significant difference for $\underline{x}_0 = (-2, 2)$. Looking at the value of the gradient of the solutions found with the same number of iterations (try, e.g., 100 iterations), the one obtained with PR is significantly smaller in all the considered cases.



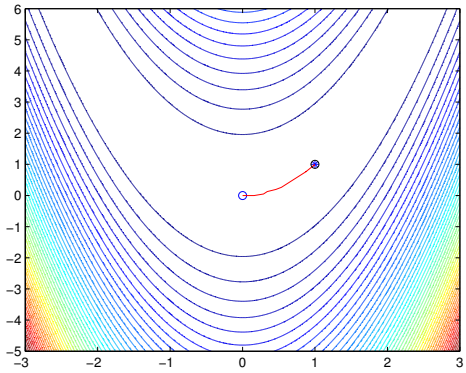
(a) $\underline{x}_0 = (-2, 2)$, FR conjugate direction method.



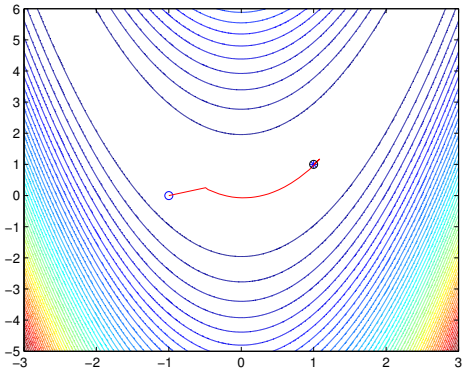
(b) $\underline{x}_0 = (-2, 2)$, PR conjugate direction method.



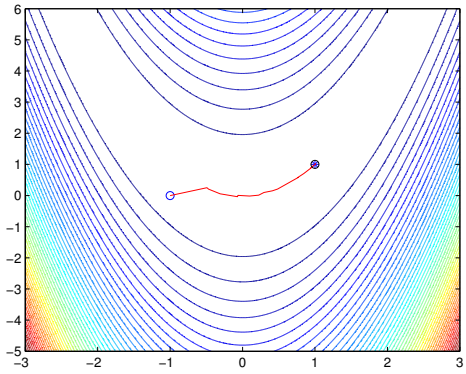
(c) $\underline{x}_0 = (0, 0)$, FR conjugate direction method.



(d) $\underline{x}_0 = (0, 0)$, PR conjugate direction method.



(e) $\underline{x}_0 = (-1, 0)$, FR conjugate direction method.



(f) $\underline{x}_0 = (-1, 0)$, PR conjugate direction method.

Figure 5: Conjugate direction method on Rosenbrock function with FR and PR update: in red, points generated up to the 50-th iteration; in blue, points generated at iterations with index between 50 and 500.