## 最优化理论与算法 作业 2

**姓名：华浩宇 学号：23020211153893**

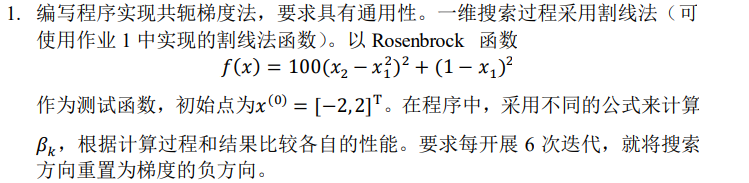
要求：

(1) 在文档中说明解题思路、方法实现、求解结果等，必要时进行分析和讨论。

(2) 将源代码作为附录粘贴于作业文档末尾，并同时作为提交的附件与作业文档 一起放在压缩包内备查。

(3) 推荐编程语言为 MATLAB，可选用其他熟悉的编程语言。

(4) 独立完成，严禁抄袭。

共轭梯度法（Conjugate Gradient）是介于最速下降法与牛顿法之间的一个方法，它仅需利用一阶导数信息，但克服了最速下降法收敛慢的缺点，又避免了牛顿法需要存储和计算Hesse矩阵并求逆的缺点，共轭梯度法不仅是解决大型线性方程组最有用的方法之一，也是解大型非线性最优化最有效的算法之一。

实验代码

首先，我们实现共轭梯度法：

function [fv, bestx, iter\_num] = conjungate\_gradient(f, x, x0, epsilon, show\_detail)

syms lambdas

n = length(x);

nf = cell(1, n);

for i = 1 : n

    nf{i} = diff(f, x{i});

end

nfv = subs(nf, x, x0);

nfv\_pre = nfv;

count = 0;

k = 0;

xv = x0;

d = - nfv;

if show\_detail

    fprintf('Initial:\n');

    fprintf('f = %s, x0 = %s, epsilon = %f\n\n', char(f), num2str(x0), epsilon);

end

while (norm(nfv) > epsilon)

    xv = xv+lambdas\*d;

    phi = subs(f, x, xv);

    nphi = diff(phi);

    lambda = solve(nphi);

    lambda = double(lambda);

    if length(lambda) > 1

        lambda = lambda(abs(imag(lambda)) < 1e-5);

        lambda = lambda(lambda > 0);

        lambda = lambda(1);

    end

    if lambda < 1e-5

        break;

    end

    xv = subs(xv, lambdas, lambda);

    xv = double(xv);

    nfv = subs(nf, x, xv);

    count = count + 1;

    k = k + 1;

    alpha = sum(nfv(:).\*nfv(:)) / sum(nfv\_pre(:).\*nfv\_pre(:));

    if show\_detail

        fprintf('epoch: %d\n', count);

        fprintf('x(%d) = %s, lambda = %f\n', count, num2str(xv), lambda);

        fprintf('nf(x) = %s, norm(nf) = %f\n', num2str(double(nfv)), norm(double(nfv)));

        fprintf('d = %s, alpha = %f\n', num2str(double(d)), double(alpha));

        fprintf('\n');

    end

    d = -nfv + alpha .\* d;

    nfv\_pre = nfv;

    if k >= 6

        k = 0;

        d = - nfv;

    end

end % while

fv = double(subs(f, x, xv));

bestx = double(xv);

iter\_num = count;

end

对实现的共轭梯度法进行测试：

syms x1 x2;

f = 100 \* (x2-x1.^2).^2 + (1 - x1).^2;

x = {x1, x2};

% initial value

x0 = [-2 2];

% tolerance

epsilon = 1e-5;

%% call conjungate gradient method

show\_detail = true;

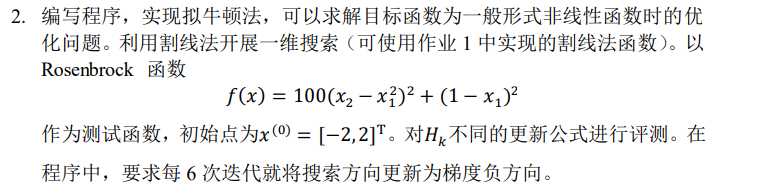
[bestf, bestx, count] = conjungate\_gradient(f, x, x0, epsilon, show\_detail);

% print result

fprintf('bestx = %s, bestf = %f, count = %d\n', num2str(bestx), bestf, count);

实验结果

bestx = 1, bestf = 0.000000



拟牛顿法和最速下降法(Steepest Descent Methods)一样只要求每一步迭代时知道目标函数的梯度。通过测量梯度的变化，构造一个目标函数的模型使之足以产生超线性收敛性。这类方法大大优于最速下降法，尤其对于困难的问题。

实验代码

BFGS算法，即使用BFGS矩阵作为拟牛顿法中的对称正定迭代矩阵的方法。

function [f, xk, k] = BFGS(x0, fun, grid, eps, kmax)

  k = 0;

  n = length(x0);

  H0 = eye(n); % 初始选取单位阵作为Hessen矩阵的逆的近似阵

  Hk = H0;

  xk = x0;

  gk = feval(grid, xk);

  while k <= kmax

      if norm(gk) < eps

          break;

      end

      dk = -Hk \* gk; % 拟牛顿下降方向

      alpha = Armjio(fun, grid, xk, dk);

      x\_ = xk; % x\_ 保存上一个点坐标

      xk = x\_ + alpha \* dk; % 更新 xk

      gk\_ = gk; % gk\_ 保存上一个点的梯度值

      gk = feval(grid, xk); % 更新 gk

      sk = xk - x\_; % 记 xk - x\_ 为 sk

      yk = gk - gk\_; % 记 gk - gk\_ 为 yk

      if sk' \* yk > 0

          v = yk' \* sk;

          % BFGS公式

          Hk = Hk + (1 + (yk' \* Hk \* yk) / v) \* (sk \* sk') / v - (sk \* yk' \* Hk + Hk \* yk \* sk') / v;

      end

      k = k + 1;

  end

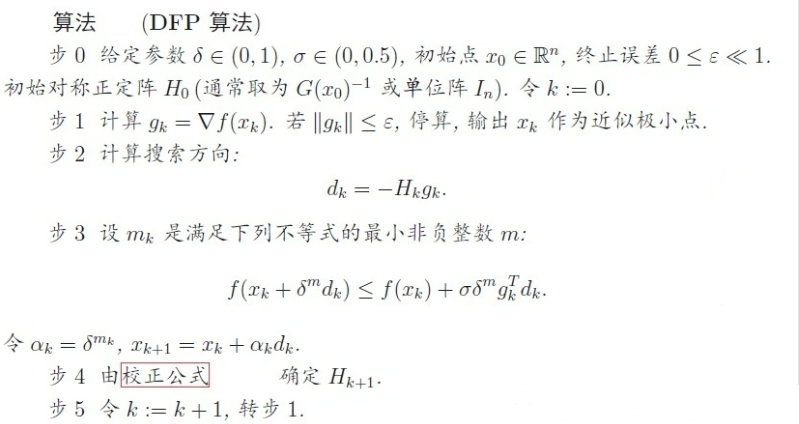
  f = feval(fun, xk);

end

实验结果



DFP算法：



function [f, xk, k] = DFP(x0, fun, grid, eps, kmax)

  k = 0;

  n = length(x0);

  H0 = eye(n); % 初始选取单位阵作为Hessen矩阵的逆的近似阵

  Hk = H0;

  xk = x0;

  gk = feval(grid, xk);

  while k <= kmax

      if norm(gk) < eps

          break;

      end

      dk = -Hk \* gk; % 拟牛顿下降方向

      alpha = Armjio(fun, grid, xk, dk);

      x\_ = xk; % x\_ 保存上一个点坐标

      xk = x\_ + alpha \* dk; % 更新 xk

      gk\_ = gk; % gk\_ 保存上一个点的梯度值

      gk = feval(grid, xk); % 更新 gk

      sk = xk - x\_; % 记 xk - x\_ 为 sk

      yk = gk - gk\_; % 记 gk - gk\_ 为 yk

      if sk' \* yk > 0

          v = Hk \* yk;

          % DFP公式

          Hk = Hk + (sk \* sk') / (sk' \* yk) - (v \* v') / (yk' \* v);

      end

      k = k + 1;

  end

  f = feval(fun, xk);

end

实验结果

