CSC 2305 Numerical Methods for Optimization Problems (Winter 2016)
<u>Assignment # 3</u>

1002119049 Hao Wang (UTORid: `wangh110`)

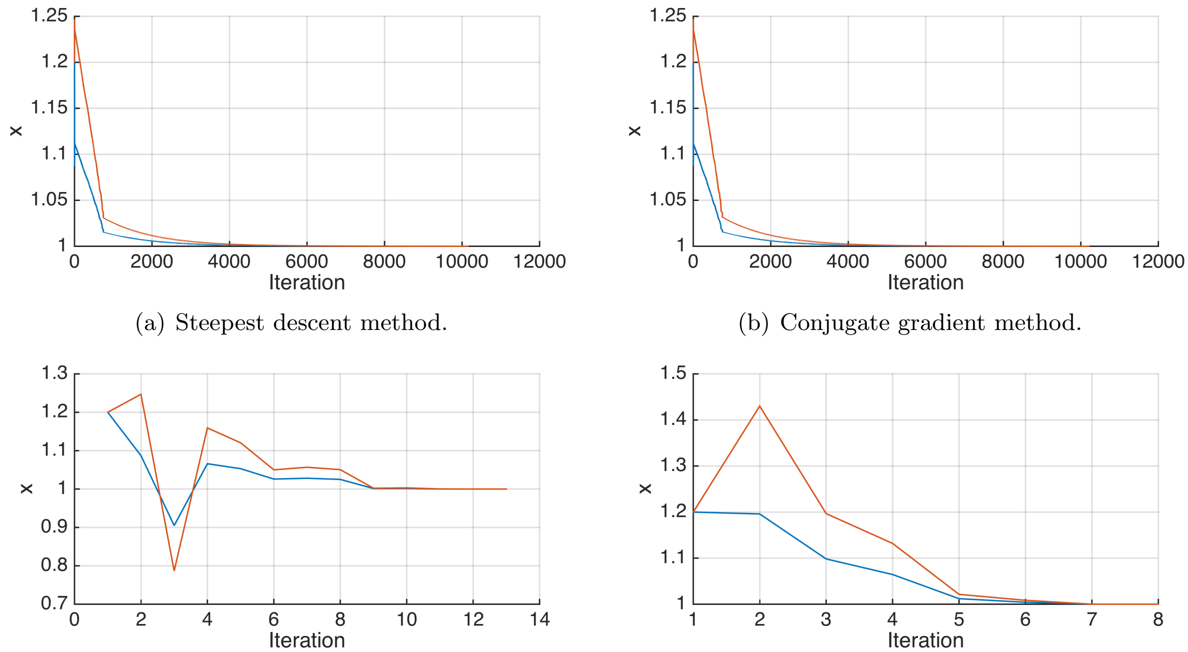
February 28, 2016

1 Problem 1

1.1 Comments

For starting points $x_0 = (1.2, 1.2)^T$ and $x_0 = (-1.2, 1)^T$, four algorithms all converges to the minimizer $x = (1, 1)$. The BFGS and Newton methods reach the convergence very soon, while it takes longer time for the steepest descent method and conjugate method. The reason is BFGS and Newton methods exploit the second order information (*i.e.* Hessian), which accounts for the "velocity" of the gradient, and takes the appropriate step size in that direction.

1.2 Plots



(a) Steepest descent method.

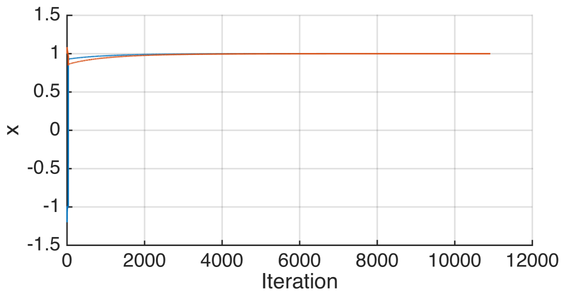
(b) Conjugate gradient method.

(c) BFGS method.

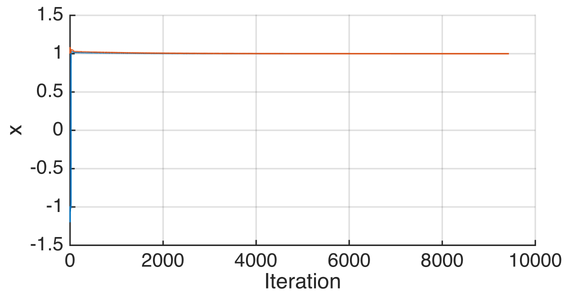
(d) Newton method.

Figure 1: $\{x_k\}$ iteration with initial point $x_0 = (1.2, 1.2)^T$.

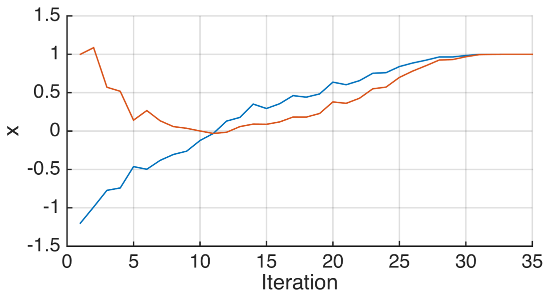
1.3 Output



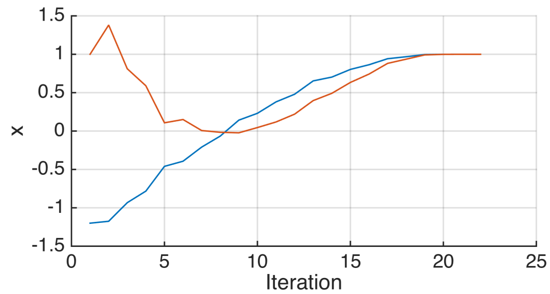
(a) Steepest descent method.



(b) Conjugate gradient method.



(c) BFGS method.



(d) Newton method.

Figure 2: $\{x_k\}$ iteration with initial point $x_0 = (-1.2, 1)^T$.

```
% Steepest descent methods with start = [1.2, 1.2]';
% i            x(1)          x(2)          ||x_{k+1} - x_k||_2
  1            1.2           1.2              0
  2          1.08711       1.24687          0.122236
  3          1.11457       1.23417          0.0302593
  4          1.11082       1.23575          0.00407096
  5          1.1114       1.23539          0.000678414
  6          1.11113       1.23532          0.000280713
  7          1.11131       1.23504          0.000337664
  8          1.1109       1.23503          0.000419141
  9          1.11109       1.23485          0.000266203
 10          1.1105       1.23433          0.000782034
 ...
 10163          1.00001          1.00002          2.52871e-08
 10164          1.00001          1.00002          2.4512e-08
 10165          1.00001          1.00002          2.37806e-08
 10166          1.00001          1.00002          2.30909e-08
 10167          1.00001          1.00002          2.24411e-08
 10168          1.00001          1.00002          2.18294e-08
 10169          1.00001          1.00002          2.1254e-08
 10170          1.00001          1.00002          2.07133e-08
 10171          1.00001          1.00002          2.02055e-08
 10172          1.00001          1.00002          1.97291e-08
```

```
% Steepest descent methods with start = [-1.2, 1]';
% i             x(1)            x(2)           ||x_{k+1} - x_k||_2
   1           -1.2               1              0
   2         -0.989453         1.08594          0.22741
   3         -1.06433         1.04417          0.0857393
   4         -1.02345         1.06148          0.0443947
   5         -1.02677          1.056          0.00640425
   6         -1.02026         1.05532          0.00654475
   7         -1.02384         1.04969          0.00666594
   8         -1.01709         1.04913          0.00676867
   9         -1.02085         1.0434          0.00684839
  10         -1.01397         1.04291          0.00690576
  ...
 10908          0.999992         0.999984         2.58026e-08
 10909          0.999992         0.999984         2.49943e-08
 10910          0.999992         0.999984         2.4231e-08
 10911          0.999992         0.999984         2.35109e-08
 10912          0.999992         0.999984         2.28321e-08
 10913          0.999992         0.999984         2.21926e-08
 10914          0.999992         0.999984         2.15908e-08
 10915          0.999992         0.999984         2.10248e-08
 10916          0.999992         0.999984         2.04931e-08
 10917          0.999992         0.999984         1.99938e-08
```

```
% Conjugate gradient methods with start = [1.2, 1.2]';
% i             x(1)            x(2)           ||x_{k+1} - x_k||_2
   1            1.2             1.2              0
   2          1.08711         1.24687         0.122236
   3          1.08711         1.24687            0
   4          1.11229         1.23471         0.027968
   5          1.11099         1.23519         0.00138719
   6          1.11133         1.23485         0.000479725
   7          1.11102         1.23489         0.000305256
   8          1.11105         1.23449         0.000402873
   9          1.11066         1.23446         0.000386174
  10          1.11083         1.23429         0.000243104
  ...
 10213          1.00001         1.00002         2.54752e-08
 10214          1.00001         1.00002         2.46817e-08
 10215          1.00001         1.00002         2.39326e-08
 10216          1.00001         1.00002         2.32258e-08
 10217          1.00001         1.00002         2.25596e-08
 10218          1.00001         1.00002         2.19322e-08
 10219          1.00001         1.00002         2.13417e-08
 10220          1.00001         1.00002         2.07864e-08
 10221          1.00001         1.00002         2.02647e-08
 10222          1.00001         1.00002         1.9775e-08
```

```
% Conjugate gradient methods with start = [-1.2, 1]';
% i              x(1)          x(2)          ||x_{k+1} - x_k||_2
  1             -1.2            1             0
  2          -0.989453       1.08594        0.22741
  3          -0.989453       1.08594            0
  4          -1.06013       1.04339        0.0824913
  5          -1.02278       1.05911        0.0405224
  6          -1.02639       1.04952        0.0102459
  7          -1.01531       1.05107        0.0111914
  8          -1.02321       1.0432        0.0111493
  9          -1.01231       1.04467        0.011002
 10          -1.01994       1.03693        0.0108714
 ...
 9424          1.00001       1.00002        2.55843e-08
 9425          1.00001       1.00002        2.47822e-08
 9426          1.00001       1.00002        2.40247e-08
 9427          1.00001       1.00002        2.33099e-08
 9428          1.00001       1.00002        2.26361e-08
 9429          1.00001       1.00002        2.20012e-08
 9430          1.00001       1.00002        2.14036e-08
 9431          1.00001       1.00002        2.08415e-08
 9432          1.00001       1.00002        2.03134e-08
 9433          1.00001       1.00002        1.98174e-08
```

```
% BFGS methods with start = [1.2, 1.2]';
% i              x(1)          x(2)          ||x_{k+1} - x_k||_2
  1              1.2           1.2            0
  2          1.08711       1.24687        0.122236
  3          0.90482       0.786471        0.495178
  4          1.06594       1.15932        0.406168
  5          1.0531       1.12056        0.0408245
  6           1.026       1.05009        0.0755019
  7          1.02814       1.05676        0.00700632
  8          1.02509       1.05058        0.00689535
  9          1.0014       1.00217        0.0538974
 10          1.00151        1.003        0.000836001
 11          1.00013       1.00026        0.00306171
 12                1             1       0.000294169
 13                1             1       1.02671e-06
```

```
% BFGS methods with start = [-1.2, 1]';
% i              x(1)          x(2)          ||x_{k+1} - x_k||_2
  1             -1.2            1             0
```

```
  2         -0.989453        1.08594        0.22741
  3         -0.772346        0.570592        0.55921
  4         -0.741285        0.518426        0.0607132
  5         -0.463158        0.141711        0.468262
  6         -0.497591        0.267292        0.130215
  7         -0.381971        0.133909        0.176519
  8         -0.304751        0.0582936        0.108077
  9         -0.261659        0.0361369        0.0484548
 10         -0.12335        0.0013522        0.142616
 11         -0.0311547        -0.0311259        0.0977486
 12         0.130051        -0.0154993        0.161962
 13         0.177649        0.0583964        0.0878984
 14         0.351569        0.0905307        0.176863
 15         0.293915        0.0882444        0.0576994
 16         0.356436        0.119517        0.0699063
 17         0.461592        0.182783        0.122721
 18         0.441625        0.182158        0.0199774
 19         0.483813        0.229011        0.0630482
 20         0.637325        0.379702        0.215114
 21         0.603506        0.36134        0.0384828
 22         0.656782        0.42751        0.0849515
 23         0.75299        0.550488        0.15614
 24         0.760526        0.572652        0.0234097
 25         0.840085        0.698663        0.149025
 26         0.8864        0.781153        0.0946026
 27         0.923076        0.851441        0.079281
 28         0.96517        0.925438        0.0851321
 29         0.965304        0.931125        0.00568878
 30         0.983979        0.967864        0.0412131
 31         0.997896        0.995347        0.0308055
 32         0.999294        0.998613        0.0035529
 33         1.00001        1.00002        0.00158085
 34              1        0.999999        2.5459e-05
 35              1              1        7.83587e-07
```

```
% Newton methods with start = [1.2, 1.2]';
% i            x(1)           x(2)          ||x_{k+1} - x_k||_2
  1             1.2            1.2              0
  2         1.19592        1.4302        0.23024
  3         1.09828        1.19669        0.253105
  4         1.06449        1.13199        0.0729909
  5         1.01199        1.02137        0.122445
  6         1.00426        1.00848        0.0150321
  7         1.00005        1.00008        0.00939417
  8              1              1        9.66249e-05
```

```
% Newton methods with start = [-1.2, 1]';
% i            x(1)            x(2)          ||x_{k+1} - x_k||_2
  1           -1.2              1              0
  2         -1.17528         1.38067        0.381476
  3         -0.932981         0.811211        0.618868
  4         -0.78254         0.589736        0.267738
  5         -0.459997         0.107563        0.580108
  6         -0.393046         0.150002        0.079269
  7         -0.209412         0.00677013        0.232888
  8         -0.065719         -0.0163287        0.145538
  9         0.142043         -0.0229888        0.207868
 10         0.231107         0.045478         0.11234
 11         0.379743         0.118146         0.165448
 12         0.479595         0.220041         0.142664
 13         0.653406         0.396729         0.247849
 14         0.702624         0.491258         0.106574
 15         0.802786         0.633221         0.173741
 16         0.863491         0.741931         0.124511
 17         0.942079         0.881336         0.160031
 18         0.967992         0.936337         0.0607992
 19         0.99621         0.991639         0.0620854
 20         0.999479         0.998948         0.00800735
 21         0.999999         0.999998         0.00117074
 22             1               1         2.72682e-06
```

1.4 Source code list

```
function a3_1()
    % assignment 3 Q1 main function
    TOL = 10e-6;
    objFunc = @rosenbrock;
    start = [1.2, 1.2]';
%     start = [-1.2, 1]';

    algs = {@stpDescent, @conjGrd, @BFGS, @newton};
    alg_names = {'stpDescent', 'conjGrd', 'BFGS', 'newton'};

    for i = 1:length(algs)
        [xiter, output] = algs{i}(objFunc, start, TOL);
        fig = draw(xiter);
        fig.PaperUnits = 'inches';
        fig.PaperPosition = [0 0 6 3];
        print(fig, '-depsc', '-r0', sprintf('../figs/%s-1', alg_names{i}));

        dlmwrite(sprintf('../output/1-%s-0.txt', alg_names{i}), ...
            output, 'delimiter','\t', 'precision', '%6.6g');
```

```
    end
end
```

```
function [ xiter, outputlist ] = stpDescent( func, x, tol )
    % steepest descent methods
    [~, gx, ~] = feval(func, x);
    gxnorm = norm(gx,2);

    it = 1;
    xiter{it} = x;
    outputlist(it, :) = [it, x(1), x(2), 0];
%     outputlist(it, :) = [it, 0];

    while( gxnorm > tol )
        s = -gx;
        alphak = backtracking(func, s, x, 0.5, 1e-4);
        x = x + alphak * s;
        disp(x);
        [~, gx, ~] = feval(func, x);
        it = it + 1;
        xiter{it} = x;
        diff = norm(xiter{it} - xiter{it-1}, 2);
        outputlist(it, :) = [it, x(1), x(2), diff];
%         outputlist(it, :) = [it, diff];

        gxnorm = norm(gx,2);
    end
```

```
function [ xiter, outputlist ] = conjGrd( func, x, tol )
    % conjugate gradient methods
    [~, gx, ~] = feval(func, x);
    gxnorm = norm(gx,2);

    old_gx = gx;
    p = -gx;
    it = 1;
    xiter{it} = x;
    outputlist(it, :) = [it, x(1), x(2), 0];
%    outputlist(it, :) = [it, 0];


    while( gxnorm > tol )

        b = (gx' * (gx - old_gx)) / (old_gx' * old_gx);
        b = max(b, 0);
```

```
        p = -gx + b * p;
        alphak = backtracking(func, p, x, 0.5, 1e-4);
        x = x + alphak * p;
        disp(x);
        it = it + 1;
        xiter{it} = x;
        diff = norm(xiter{it} - xiter{it-1}, 2);
        outputlist(it, :) = [it, x(1), x(2), diff];
%         outputlist(it, :) = [it, diff];

        [~, gx, ~] = feval(func, x);
        gxnorm = norm(gx,2);
    end
end
```

```
function [ xiter, outputlist ] = BFGS( func, x, tol )
    % BFGS methods
    [n, ~] = size(x);
    H = eye(n);
    dist = 2*tol;

    it = 1;
    xiter{it} = x;
    outputlist(it, :) = [it, x(1), x(2), 0];
%     outputlist(it, :) = [it, 0];

    while dist > tol
        [~, grad] = func(x);
        p = -H * grad;

        alpha = backtracking(func, p, x, 0.5, 1e-4);

        x = x + alpha * p;

        it = it + 1;
        xiter{it} = x;
        diff = norm(xiter{it} - xiter{it-1}, 2);
        outputlist(it, :) = [it, x(1), x(2), diff];
%         outputlist(it, :) = [it, diff];

        s = alpha * p;
        dist = norm(s);
        [~, newgrad] = func(x);
        y = newgrad - grad;
        rho = 1 / (y' * s);
        H = (eye(n) - rho * s * y') * H * (eye(n) - rho*y*s') ...
            + rho * (s * s');
```

```
    end
end
```

```
function [ xiter, outputlist ] = newton( func, x, tol )
    % Newton methods
    [~, gx, hx] = feval(func, x);
    gxnorm = norm(gx,2);
    it = 1;
    xiter{it} = x;
    outputlist(it, :) = [it, x(1), x(2), 0];
     outputlist(it, :) = [it, 0];

    while( gxnorm > tol )
        p = - hx \ gx;
        alphak = backtracking(func, p, x, 0.5, 1e-4);
        x = x + alphak * p;

        it = it + 1;
        xiter{it} = x;
        diff = norm(xiter{it} - xiter{it-1}, 2);
        outputlist(it, :) = [it, x(1), x(2), diff];
%        outputlist(it, :) = [it, diff];

        [~, gx, hx] = feval(func, x);
        gxnorm = norm(gx,2);
    end
end
```

```
function [y, gy, hy] = rosenbrock( x )
    % Rosenbrock function
    y = 100.0*(x(2) - x(1).^2).^2 + (1 - x(1)).^2;
    gy = [-400.0*(x(2) - x(1)^2)*x(1) - 2*(1 - x(1)); 200*(x(2) - x(1)^2)];
    hy = [-400.0*(x(2) - 3*x(1)^2) + 2, -400*x(1); -400*x(1), 200];

end
```

```
function [ fig ] = draw( x )
    figure;

    set(gcf, 'Position', [0 0 500 200]);
    set(gca,'FontSize',14);

    hold on;
    grid on;
```

```

    X = NaN(length(x), 2);
    Y = NaN(length(x), 2);
    for i = 1:length(x)
        X(i,:) = [i, x{i}(1)];
        Y(i,:) = [i, x{i}(2)];
    end

    plot(X(:,1), X(:,2));
    plot(Y(:,1), Y(:,2));
    xlabel('Iteration');
    ylabel('x');

    fig = gcf;
end
```

2 Problem 2

2.1 Comments

I spent a lot of time in developing the gradient and hessian calculation function. Since I think hard coding all those functions is ugly and redundant. With symbolic toolbox in MATLAB, I implemented the following function to calculate gradient and hessian in flight. Unfortunately, this implementation is too slow, it takes more than hours to finish steepest descent method with $n = 4$. Eventually, I hard coded gradient and hessian functions as `extRosenbrock4`, `extRosenbrock8` and `extRosenbrock16`, which are pre-calculated by the MATLAB `hessian` function and symbolic toolbox.

```
function [ y, gy, hy ] = extRosenbrock( x )

    [n, ~] = size(x);
    r = sym(zeros(n, 1));
    v = sym('v', [n, 1]);

    for i = 1 : 2 : n - 1
        r(i) = 10.0 * (v(i + 1) - v(i)^2);
        r(i + 1) = 1.0 - v(i);
    end

    y = r' * r;
    gy = gradient(y, v);
    hy = hessian(y, v);

    y = vpa(subs(y, v, x), 5);
    gy = vpa(subs(gy, v, x), 5);
    hy = vpa(subs(hy, v, x), 5);

end
```

The iteration number increases when n is getting larger. Newton and BFGS methods still have better performance than steepest descent method and conjugate gradient method. The Newton method has slightly better performance than BFGS. BFGS applies rank-one updates specified by gradient evaluations (or approximate gradient evaluations) to approximate the Hessian matrix.

2.2 Output:

2.2.1 $n = 4$.

```
% Steepest descent methods
% i          ||x_{k+1} - x_k||_2
  1               0
  2          0.172867
  3          0.0427932
```

```
  4        0.0057572
  5        0.000959422
  6        0.000396988
  7        0.00047753
  8        0.000592755
  9        0.000376468
 10        0.00110596
 ...
0594       2.5424e-08
0595       2.46391e-08
0596       2.38982e-08
0597       2.31995e-08
0598       2.25411e-08
0599       2.19211e-08
0600       2.13378e-08
0601       2.07895e-08
0602       2.02746e-08
0603       1.97913e-08
```

```
% Conjugate gradient methods
% i        ||x_{k+1} - x_k||_2
  1             0
  2        0.172867
  3             0
  4        0.0395527
  5        0.00196178
  6        0.000678434
  7        0.000431697
  8        0.000569749
  9        0.000546132
 10        0.000343801
 ...
0644       2.56139e-08
0645       2.48107e-08
0646       2.40521e-08
0647       2.33363e-08
0648       2.26615e-08
0649       2.20258e-08
0650       2.14274e-08
0651       2.08645e-08
0652       2.03356e-08
0653       1.9839e-08
```

```
% BFGS methods
% i        ||x_{k+1} - x_k||_2
```

```
  1             0
  2         0.172867
  3         0.700287
  4         0.574409
  5         0.0577346
  6         0.106776
  7         0.00992511
  8         0.0027258
  9         0.0122801
 10         0.0310893
 11         0.0274482
 12         0.0176052
 13         0.0034513
 14         0.00176654
 15         0.00213944
 16         0.000422755
 17         9.62826e-05
 18         8.74168e-06
```

```
% Newton methods
% i         ||x_{k+1} - x_k||_2
  1             0
  2         0.325609
  3         0.357944
  4         0.103225
  5         0.173163
  6         0.0212586
  7         0.0132854
  8         0.000136648
```

2.2.2 $n = 8$.

```
% Steepest descent methods
% i         ||x_{k+1} - x_k||_2
  1             0
  2         0.244471
  3         0.0605187
  4         0.00814191
  5         0.00135683
  6         0.000561426
  7         0.000675329
  8         0.000838282
  9         0.000532406
 10         0.00156407
...
```

```
1025        2.55135e-08
1026        2.47215e-08
1027        2.39739e-08
1028        2.32686e-08
1029        2.2604e-08
1030        2.1978e-08
1031        2.1389e-08
1032        2.08352e-08
1033        2.0315e-08
1034        1.98267e-08
```

```
% Conjugate gradient methods
% i         ||x_{k+1} - x_k||_2
   1            0
   2        0.244471
   3            0
   4        0.055936
   5        0.00277438
   6        0.00095945
   7        0.000610512
   8        0.000805747
   9        0.000772347
  10        0.000486207
  ...
1075        2.57049e-08
1076        2.48945e-08
1077        2.41292e-08
1078        2.34069e-08
1079        2.27259e-08
1080        2.20841e-08
1081        2.148e-08
1082        2.09117e-08
1083        2.03775e-08
1084        1.98759e-08
```

```
% BFGS methods
% i         ||x_{k+1} - x_k||_2
   1            0
   2        0.244471
   3        0.990356
   4        0.812336
   5        0.081649
   6        0.151004
   7        0.0140853
   8        0.00672896
```

```
  9         0.0201702
 10         0.0321362
 11         0.00383713
 12         0.000393179
 13         0.0214331
 14         0.0434435
 15         0.0200948
 16         0.00026547
 17         0.000354085
 18         0.000357901
 19         1.84209e-05
 20         4.11455e-06
```

```
% Newton methods
% i         ||x_{k+1} - x_k||_2
  1              0
  2         0.460481
  3         0.50621
  4         0.145982
  5         0.24489
  6         0.0300642
  7         0.0187883
  8         0.00019325
```

2.2.3 $n = 16$.

```
% Steepest descent methods
% i         ||x_{k+1} - x_k||_2
  1              0
  2         0.345735
  3         0.0855863
  4         0.0115144
  5         0.00191884
  6         0.000793976
  7         0.000955059
  8         0.00118551
  9         0.000752935
 10         0.00221192
 ...
1456        2.55691e-08
1457        2.47719e-08
1458        2.40194e-08
1459        2.33094e-08
1460        2.26402e-08
1461        2.20098e-08
```

```
1462        2.14166e-08
1463        2.08589e-08
1464        2.03348e-08
1465        1.98429e-08
```

```
 % Conjugate gradient methods
 % i         ||x_{k+1} - x_k||_2
   1              0
   2        0.345735
   3              0
   4        0.0791054
   5        0.00392356
   6        0.00135687
   7        0.000863394
   8        0.0011395
   9        0.00109226
  10        0.000687601
  ...
1482        2.5175e-08
1483        2.4405e-08
1484        2.36784e-08
1485        2.29933e-08
1486        2.2348e-08
1487        2.17405e-08
1488        2.11691e-08
1489        2.06323e-08
1490        2.01282e-08
1491        1.96552e-08
```

```
 % BFGS methods
 % i         ||x_{k+1} - x_k||_2
   1              0
   2        0.345735
   3        1.40057
   4        1.14882
   5        0.115469
   6        0.213552
   7        0.0199354
   8        0.00988384
   9        0.014455
  10        0.0367697
  11        0.0171865
  12        0.093136
  13        0.0314112
  14        0.0105094
```

```
15         0.0158041
16         0.000949521
17         0.00291164
18         6.51753e-06
```

```
% Newton methods
% i           ||x_{k+1} - x_k||_2
  1                0
  2          0.651218
  3          0.715889
  4          0.206449
  5          0.346327
  6          0.0425171
  7          0.0265707
  8          0.000273297
```

2.3 Source code list

```
function a3_2()

    TOL = 10e-6;
    start1(1:4) = 1.2;
    start2(1:8) = 1.2;
    start3(1:16) = 1.2;

    input = {start1, start2, start3};
    algs = {@stpDescent, @conjGrd, @BFGS, @newton};
    obj = {@extRosenbrock4, @extRosenbrock8, @extRosenbrock16};
    alg_names = {'stpDescent', 'conjGrd', 'BFGS', 'newton'};

    for j = 1:length(input)
        start = input{j}';
        for i = 1:length(algs)

            [~, output] = algs{i}(obj{j}, start, TOL);
            disp(alg_names{i});
            dlmwrite(sprintf('../output/2-%s-%d.txt', alg_names{i}, j), ...
                output, 'delimiter','\t', 'precision', '%6.6g');
        end
    end
end
```

```
function [ y, gy, hy ] = extRosenbrock4( x )
    t = num2cell(x);
```

```
    [v1, v2, v3, v4] = t{:};

    y = (conj(v1) - 1)*(v1 - 1) + (conj(v3) - 1)*(v3 - 1)...
        + (10*conj(v2) - 10*conj(v1)^2)*(- 10*v1^2 + 10*v2)...
        + (10*conj(v4) - 10*conj(v3)^2)*(- 10*v3^2 + 10*v4);

    gy = [v1 + conj(v1) - 20*v1*(10*conj(v2) - 10*conj(v1)^2) - 20*conj(v1)*(- 10*v
        100*v2 + 100*conj(v2) - 100*conj(v1)^2 - 100*v1^2;
        v3 + conj(v3) - 20*v3*(10*conj(v4) - 10*conj(v3)^2) - 20*conj(v3)*(- 10*v3
        100*v4 + 100*conj(v4) - 100*conj(v3)^2 - 100*v3^2];

    hy = [200*conj(v1)^2 - 200*conj(v2) - 200*v2 + 800*v1*conj(v1) + 200*v1^2 + 2,
        - 200*v1 - 200*conj(v1), 200, 0, 0;
        0,0,200*conj(v3)^2 - 200*conj(v4) - 200*v4 + 800*v3*conj(v3) + 200*v3^2 + 2
        0,0,- 200*v3 - 200*conj(v3),200];
end
```

To save space, function `extRosenbrock8` and function `extRosenbrock16` are not presented.

3 Problem 3

3.1 Comments

Beale's function is selected as our objective function:

$$f(x, y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2$$

Output by three initial points ($x_0 = (2.5, 0.4)^T$, $x_0 = (1.2, 1.2)^T$ and $x_0 = (-1.2, 1)^T$) are presented. All the four algorithms successfully converged and found the minimizer with the initial points $x_0 = (2.5, 0.4)^T$ and $x_0 = (1.2, 1.2)^T$, while none of them reached the minimizer.

Figure 6 and the output of x_1 and x_2 show that methods of steepest descent, conjugate gradient keep zig-zag on the valley (as Figure 3), which is extremely slow and only part of the output is presented. I think steepest descent method and conjugate gradient method will eventually find a local minimizer and stop there. BFGS method converges to a stationary point $(-2.6825, 0)^T$ and Newton method converges to $(0, 1)^T$.

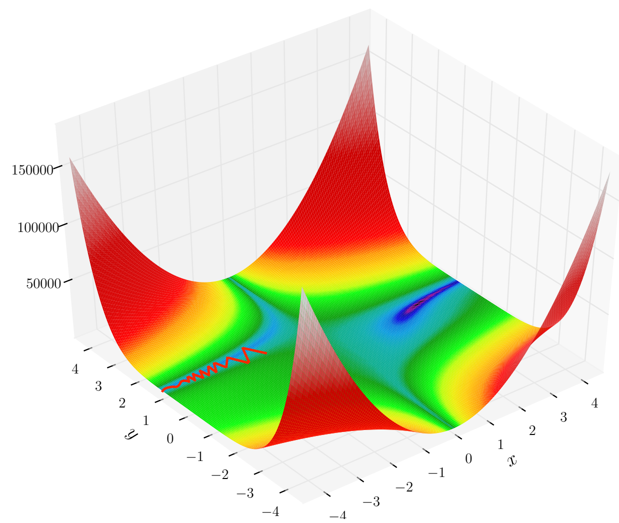


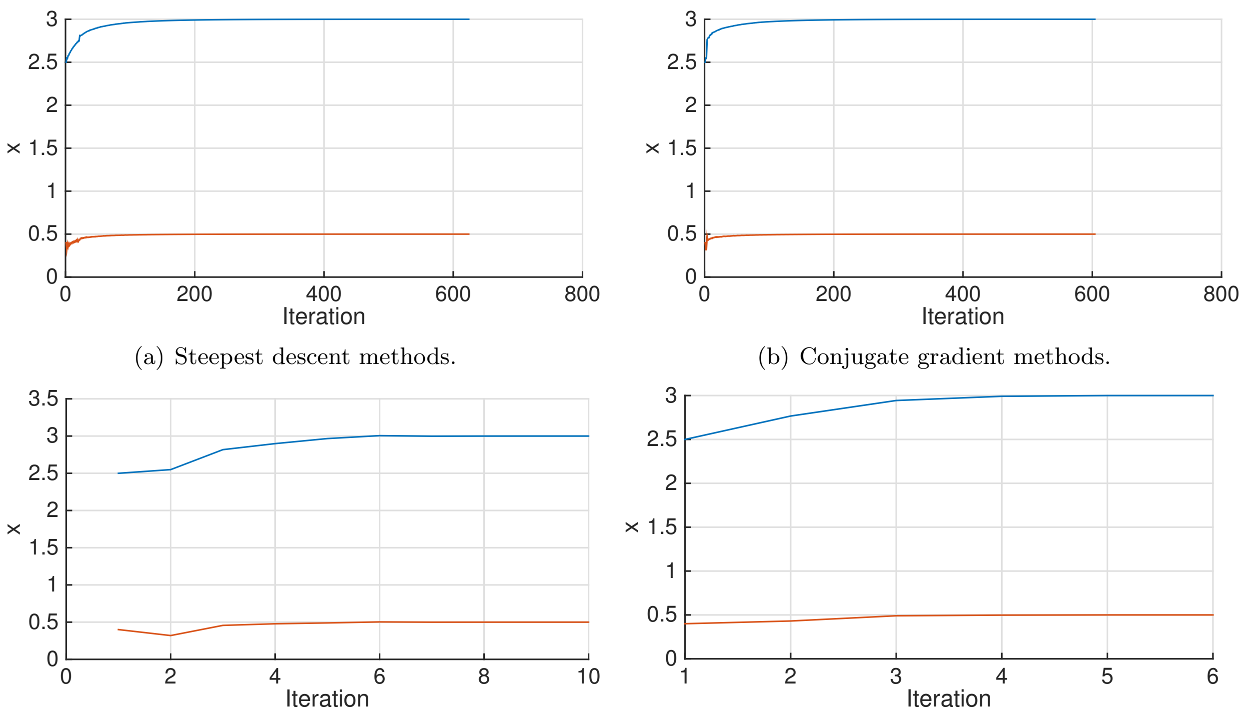
Figure 3: $\{x_k\}$ iteration with initial point $x_0 = (2.5, 0.4)^T$.

3.2 Plots

3.3 Output

With initial point at $x_0 = [2.5, 0.4]^T$.

```
% Steepest descent methods with start = [2.5, 0.4]';
% i          x(1)          x(2)          ||x_{k+1} - x_k||_2
  1          2.5           0.4             0
  2       2.54909        0.31975        0.0940765
  3       2.54413        0.386595        0.0670293
  4       2.57548        0.352199        0.0465373
  5       2.58056        0.388361        0.036517
  6       2.60381        0.370396        0.0293844
```



(a) Steepest descent methods.

(b) Conjugate gradient methods.

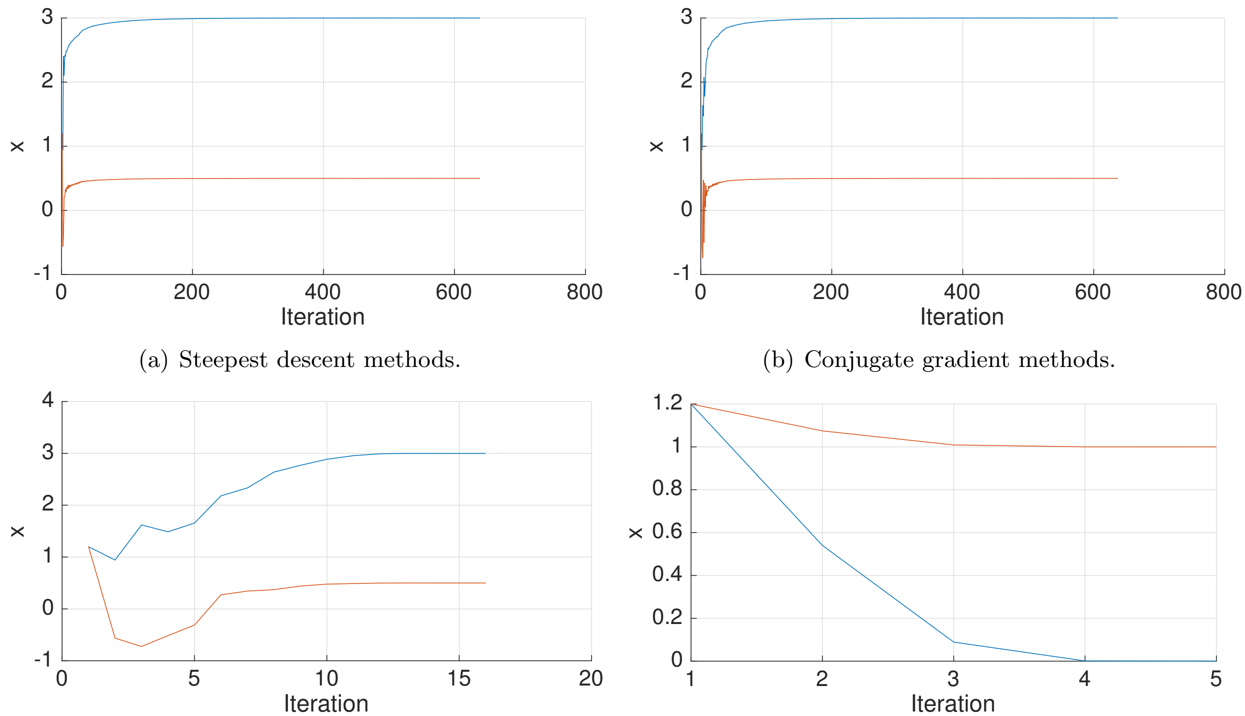
(c) BFGS methods.

(d) Newton methods.

Figure 4: $\{x_k\}$ iteration with initial point $x_0 = (2.5, 0.4)^T$.

```
  7         2.61142       0.394369        0.0251537
  8         2.63035       0.383013        0.0220785
  9         2.63827       0.40142        0.0200364
 10         2.65461       0.392695        0.0185216
 ...
615         2.99997       0.499993        5.40255e-07
616         2.99997       0.499993        7.25272e-07
617         2.99997       0.499994        5.75613e-07
618         2.99997       0.499993        7.48008e-07
619         2.99997       0.499994        6.16034e-07
620         2.99997       0.499993        7.76807e-07
621         2.99998       0.499994        6.61839e-07
622         2.99998       0.499993        8.12149e-07
623         2.99998       0.499994        7.1338e-07
624         2.99998       0.499994        4.27255e-07
```

```
% Conjugate gradient with start = [2.5, 0.4]';
% i              x(1)          x(2)         ||x_{k+1} - x_k||_2
  1               2.5           0.4                 0
  2         2.54909       0.31975        0.0940765
  3         2.54909       0.31975              0
  4         2.76238       0.474171        0.263321
  5         2.77852       0.430273        0.0467711
```



(a) Steepest descent methods.

(b) Conjugate gradient methods.

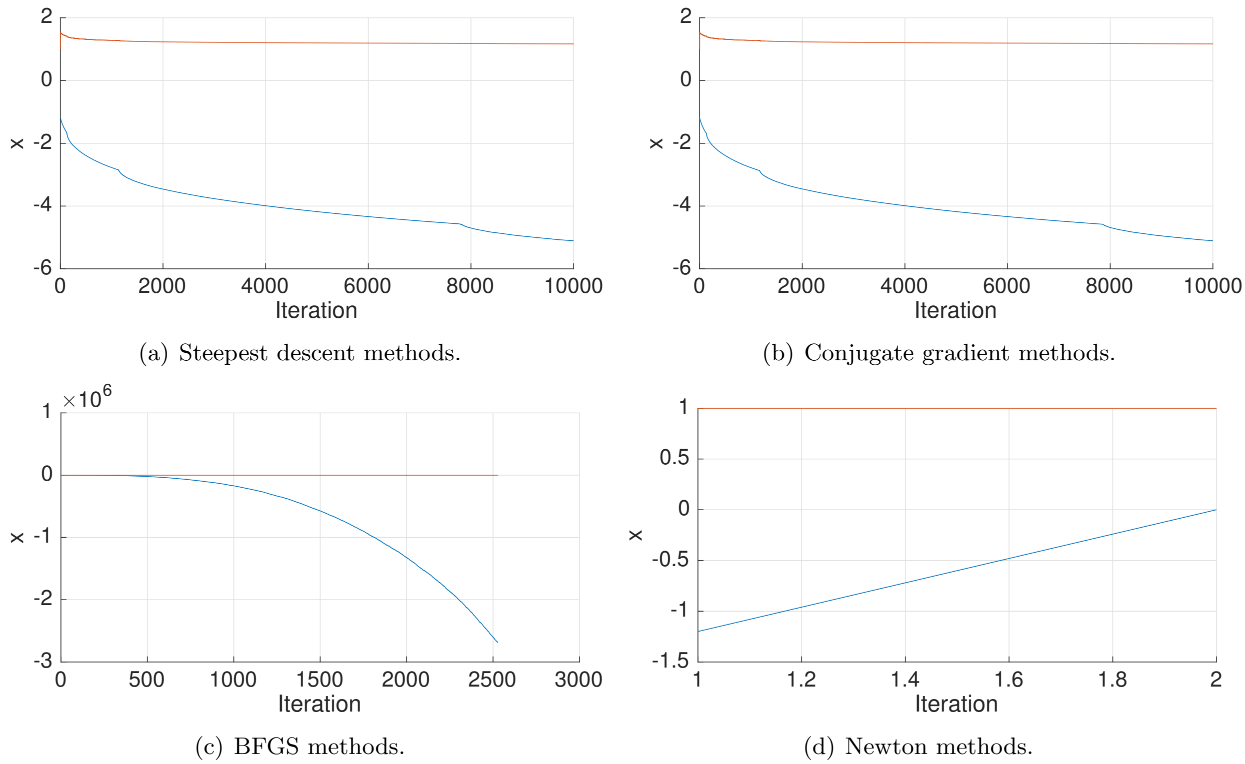
(c) BFGS methods.

(d) Newton methods.

Figure 5: $\{x_k\}$ iteration with initial point $x_0 = (1.2, 1.2)^T$.

```
 6          2.78619        0.430608        0.00767723
 7          2.78676        0.442708        0.0121136
 8          2.81231        0.439416        0.025766
 9          2.81245        0.450516        0.0111008
10          2.82372        0.44728        0.0117244
...
 595          2.99997        0.499993        5.14323e-07
 596          2.99997        0.499993        7.10408e-07
 597          2.99997        0.499993        5.45684e-07
 598          2.99997        0.499993        7.28461e-07
 599          2.99997        0.499994        5.81854e-07
 600          2.99997        0.499993        7.52168e-07
 601          2.99997        0.499994        6.23136e-07
 602          2.99998        0.499993        7.82015e-07
 603          2.99998        0.499994        6.69854e-07
 604          2.99998        0.499994        4.0924e-07
```

```
% BFGS methods with start = [2.5, 0.4]';
% i             x(1)          x(2)          ||x_{k+1} - x_k||_2
 1              2.5           0.4             0
 2          2.54909        0.31975        0.0940765
 3          2.81737        0.456467        0.301103
```



(a) Steepest descent methods.

(b) Conjugate gradient methods.

(c) BFGS methods.

(d) Newton methods.

Figure 6: $\{x_k\}$ iteration with initial point $x_0 = (-1.2, 1)^T$.

```
   4          2.89849          0.477988          0.0839215
   5          2.9663          0.489385          0.06877
   6          3.00569          0.502971          0.0416631
   7          2.99821          0.499546          0.0082237
   8          2.99971          0.499934          0.00154413
   9                3             0.5          0.000297382
  10                3             0.5          1.73945e-06
```

```
% Newton methods with start = [2.5, 0.4]';
% i           x(1)           x(2)           ||x_{k+1} - x_k||_2
   1          2.5            0.4                 0
   2          2.76661          0.430869          0.268387
   3          2.94357          0.490473          0.186733
   4          2.99153          0.497874          0.0485273
   5          2.99989          0.499979          0.00861654
   6                3             0.5          0.000115568
```

With initial point at $x_0 = (1.2, 1.2)^T$.

```
% Steepest descent methods with start = [1.2, 1.2]';
% i           x(1)           x(2)           ||x_{k+1} - x_k||_2
```

```
  1          1.2          1.2          0
  2        0.942669      -0.564086       1.78276
  3        2.40297      -0.402277      1.46924
  4        2.1087      0.206745       0.676389
  5        2.41606       0.215649       0.307492
  6        2.39978       0.332126       0.117609
  7        2.48361       0.285803       0.0957701
  8        2.47998       0.361086       0.0753706
  9        2.51456       0.334633       0.0435412
 10        2.53879       0.393072       0.0632642
 ...
630        2.99997       0.499994       5.30772e-07
631        2.99997       0.499993       7.10887e-07
632        2.99997       0.499994       5.65698e-07
633        2.99997       0.499993       7.33527e-07
634        2.99997       0.499994       6.05598e-07
635        2.99998       0.499993       7.62131e-07
636        2.99998       0.499994       6.50789e-07
637        2.99998       0.499994       7.9717e-07
638        2.99998       0.499994       7.01616e-07
639        2.99998       0.499994       4.19553e-07
```

```
% Conjugate gradient methods with start = [1.2, 1.2]';
% i              x(1)          x(2)        $||x_{k+1} - x_k||_2$
  1          1.2          1.2          0
  2        0.942669      -0.564086       1.78276
  3        1.63513      -0.741576       0.714841
  4        1.47093       0.473935       1.22655
  5        2.07428      -0.504601       1.1496
  6        1.77897       0.424608       0.975006
  7        2.00774       0.0510181       0.438069
  8        2.2746      0.383558       0.426377
  9        2.37166       0.228042       0.183318
 10        2.37705       0.309104       0.0812412
 ...
628        2.99997       0.499993       5.19994e-07
629        2.99997       0.499993       7.08701e-07
630        2.99997       0.499994       5.52813e-07
631        2.99997       0.499993       7.28662e-07
632        2.99997       0.499994       5.90503e-07
633        2.99997       0.499993       7.54402e-07
634        2.99998       0.499994       6.3337e-07
635        2.99998       0.499993       7.864e-07
636        2.99998       0.499994       6.81748e-07
637        2.99998       0.499994       4.12563e-07
```

```
% BFGS methods with start = [1.2, 1.2]';
% i            x(1)           x(2)          ||x_{k+1} - x_k||_2
   1            1.2            1.2              0
   2         0.942669       -0.564086         1.78276
   3         1.61932       -0.725104         0.695545
   4         1.4872       -0.515602         0.247683
   5         1.65541       -0.31168         0.264348
   6         2.18018       0.271298         0.784378
   7         2.33323       0.343672         0.1693
   8         2.63866       0.371026         0.306649
   9         2.76922       0.440192         0.147747
  10         2.88573       0.477628         0.122378
  11         2.95532       0.488673         0.0704597
  12         2.99129       0.49821        0.0372132
  13         2.99921       0.49979        0.00808016
  14         3.00001       0.500008         0.000830284
  15               3            0.5       1.56344e-05
  16               3            0.5       6.53115e-07
```

```
% Newton methods with start = [1.2, 1.2]';
% i            x(1)           x(2)          ||x_{k+1} - x_k||_2
   1            1.2            1.2              0
   2         0.540199        1.07454         0.671622
   3         0.0890643        1.00898         0.455874
   4         0.00129436        1.00007         0.0882206
   5         1.37612e-07              1         0.00129627
```

With initial point at $x_0 = (-1.2, 1)^T$.

```
% Steepest descent methods with start = [-1.2, 1]';
% i            x(1)           x(2)          ||x_{k+1} - x_k||_2
   1           -1.2              1              0
   2           -1.2        1.52031        0.520312
   3         -1.21118         1.5284        0.0137969
   4         -1.21186         1.51649         0.0119342
   5         -1.21748         1.52091         0.00715315
   6         -1.22362         1.51284         0.0101336
   7         -1.22923         1.51752         0.00730539
   8         -1.23519         1.50914         0.0102777
   9         -1.24086         1.5143        0.00766699
  10         -1.24657         1.50536         0.0106101
  ...
9991         -5.106        1.1673        0.00042718
9992         -5.10605         1.16711         0.00019731
9993         -5.10625         1.16725         0.000246758
9994         -5.10629         1.16711         0.000152507
```

```
9995         -5.10649        1.16722        0.0002257
9996         -5.10653        1.16711        0.000119595
9997         -5.10693        1.16727        0.000425461
9998         -5.10697        1.16708        0.000193325
9999         -5.10717        1.16723        0.000244851
10000         -5.10721        1.16708        0.00014983
```

```
 % Conjugate gradient methods with start = [-1.2, 1]';
 % i              x(1)            x(2)          ||x_{k+1} - x_k||_2
   1            -1.2              1              0
   2            -1.2          1.52031        0.520312
   3          -1.21118          1.5284        0.0137969
   4          -1.21213          1.51668        0.0117595
   5          -1.22315          1.52478        0.0136739
   6          -1.22407          1.51301        0.0118074
   7          -1.22956          1.51727        0.00694877
   8          -1.23581          1.50945        0.0100091
   9          -1.24128          1.51392        0.00706556
  10          -1.24738          1.50585        0.0101172
  ...
  9991         -5.10274         1.16726         0.000145182
  9992         -5.10293         1.16714         0.00022226
  9993         -5.10303         1.16734         0.000225513
  9994         -5.10308         1.1672         0.000149726
  9995         -5.10327         1.16731         0.000223441
  9996         -5.10336         1.16709         0.000230521
  9997         -5.10342         1.16724         0.00015355
  9998         -5.10361         1.16712         0.00022573
  9999         -5.10371         1.16733         0.000237675
  10000         -5.10376         1.16718         0.000158596
```

```
 % BFGS methods with start = [-1.2, 1]';
 % i              x(1)            x(2)          ||x_{k+1} - x_k||_2
   1            -1.2              1              0
   2            -1.2          1.52031        0.520312
   3          -1.24541          1.52182        0.0454398
   4          -1.31342          1.49776        0.0721334
   5          -1.67489          1.38386        0.378989
   6          -1.5312         1.43668        0.153091
   7          -1.62524          1.41533        0.0964328
   8          -2.11356          1.31651        0.498226
   9          -1.90094          1.37048        0.219367
  10          -2.02833          1.35306        0.12858
  ...
2520         -2.6658e+06              1         0.00792383
```

```
2521         -2.6658e+06             1        0.316947
2522         -2.66919e+06             1        3385.36
2523         -2.67372e+06             1        4531.23
2524         -2.67706e+06             1        3342.14
2525         -2.68435e+06             1        7291.67
2526         -2.68026e+06             1        4092.93
2527         -2.68251e+06             1        2247.88
2528         -2.68251e+06             1        2.0647e-05
2529         -2.68251e+06             1        3.62424e-06
```

```
% Newton methods with start = [-1.2, 1]';
% i           x(1)          x(2)          ||x_{k+1} - x_k||_2
   1          -1.2            1              0
   2            0             1            1.2
```

3.4 Source code list

```
function a3_3()

    % assignment 3 main function
    TOL = 10e-6;
    objFunc = @bealsFunction;
%     start = [1.2, 1.2]';
    start = [-1.2, 1]';

    algs = {@stpDescent, @conjGrd, @BFGS, @newton};
    alg_names = {'stpDescent', 'conjGrd', 'BFGS', 'newton'};

    for i = 1:length(algs)
        [xiter, output] = algs{i}(objFunc, start, TOL);
        disp(alg_names{i});

        fig = draw(xiter);
        fig.PaperUnits = 'inches';
        fig.PaperPosition = [0 0 6 3];
        print(fig, '-depsc', '-r0', ...
            sprintf('../figs/3-%s-1', alg_names{i}));

        dlmwrite(sprintf('../output/3-%s-1.txt', alg_names{i}), ...
            output, 'delimiter','\t', 'precision', '%6.6g');
    end
end
```

```
function [ y, gy, hy ] = bealsFunction( x )

```

```
    v1 = x(1);
    v2 = x(2);

    y = (1.5 - v1 + v1*v2)^2 + (2.25 - v1 + v1*v2^2)^2 + ...
        (2.625 - v1 + v1*v2^3)^2;
    gy = [2*(v2^2 - 1)*(v1*v2^2 - v1 + 9/4)...
        + 2*(v2^3 - 1)*(v1*v2^3 - v1 + 21/8)...
        + 2*(v2 - 1)*(v1*v2 - v1 + 3/2);
        2*v1*(v1*v2 - v1 + 3/2) + 4*v1*v2*(v1*v2^2 - v1 + 9/4)...
        + 6*v1*v2^2*(v1*v2^3 - v1 + 21/8)];
    hy = [2*(v2 - 1)^2 + 2*(v2^2 - 1)^2 + 2*(v2^3 - 1)^2,...
        6*v2^2*(v1*v2^3 - v1 + 21/8) - 2*v1 + 2*v1*v2...
        + 2*v1*(v2 - 1) + 4*v2*(v1*v2^2 - v1 + 9/4)...
        + 4*v1*v2*(v2^2 - 1) + 6*v1*v2^2*(v2^3 - 1) + 3;...
        6*v2^2*(v1*v2^3 - v1 + 21/8) - 2*v1 + 2*v1*v2...
        + 2*v1*(v2 - 1) + 4*v2*(v1*v2^2 - v1 + 9/4)...
        + 4*v1*v2*(v2^2 - 1) + 6*v1*v2^2*(v2^3 - 1) + 3,...
        8*v1^2*v2^2 + 18*v1^2*v2^4 + 4*v1*(v1*v2^2 - v1 + 9/4)...
        + 2*v1^2 + 12*v1*v2*(v1*v2^3 - v1 + 21/8)];

end
```