EN.553.761: Nonlinear Optimization I

Homework Assignment #2

Yutao Tang (ytang67) / Wenkai Luo (wluo14)

Starred exercises require the use of MATLAB.

Exercise 2.1: Let $f_i : \mathbb{R}^n \to \mathbb{R}$ be a *convex* function and $0 \leq \alpha_i \in \mathbb{R}$ for $i = 1, \ldots, k$.

(a) Prove that

$$f(x) = \sum_{i=1}^{k} \alpha_i f_i(x)$$

is a convex function.

Answer:
Based on the question, $f_i : \mathbb{R}^n \to \mathbb{R}$ is a *convex* function and $0 \leq \alpha_i \in \mathbb{R}$ for $i = 1, \ldots, k$. Thus,

$$f_i(\beta x_1 + (1-\beta)x_2) \leq \beta f_i(x_1) + (1-\beta) f_i(x_2) \qquad \text{for } i = 1, \ldots, k$$

Where β is a fixed value $\in (0, 1)$, x_1 and x_2 are fixed values $\in D(f)$.

By summing following inequations i from 1 to k:

$$\begin{aligned}
\alpha_1 f_1(\beta x_1 + (1-\beta)x_2) &\leq \alpha_1 \beta f_1(x_1) + \alpha_1(1-\beta) f_1(x_2) \\
\alpha_2 f_2(\beta x_1 + (1-\beta)x_2) &\leq \alpha_2 \beta f_2(x_1) + \alpha_2(1-\beta) f_2(x_2) \\
&\vdots \\
\alpha_k f_k(\beta x_1 + (1-\beta)x_2) &\leq \alpha_k \beta f_k(x_1) + \alpha_k(1-\beta) f_k(x_2)
\end{aligned}$$

We can get this inequations:

$$\begin{aligned}
&\sum_{i=1}^{k} \alpha_i f_i(\beta x_1 + (1-\beta)x_2) \leq \sum_{i=1}^{k} \alpha_i \beta f_i(x_1) + \sum_{i=1}^{k} \alpha_i (1-\beta) f_i(x_2) \\
\Rightarrow &\sum_{i=1}^{k} \alpha_i f_i(\beta x_1 + (1-\beta)x_2) \leq \beta \sum_{i=1}^{k} \alpha_i f_i(x_1) + (1-\beta) \sum_{i=1}^{k} \alpha_i f_i(x_2) \\
&\Rightarrow f(\beta x_1 + (1-\beta)x_2) \leq \beta f(x_1) + (1-\beta) f(x_2)
\end{aligned}$$

Thus, it can be proved that $f(x) = \sum_{i=1}^{k} \alpha_i f_i(x)$ is a convex function.

(b) Prove that

$$f(x) = \max\big(f_1(x), f_2(x), \ldots, f_k(x)\big)$$

is a convex function.

Answer:
Based on the question, $f_i : \mathbb{R}^n \to \mathbb{R}$ is a *convex* function and $0 \leq \alpha_i \in \mathbb{R}$ for $i = 1, \ldots, k$. Thus,

$$f_i(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f_i(x_1) + (1-\alpha) f_i(x_2) \qquad \text{for } i = 1, \ldots, k$$

Where α is a fixed value $\in (0,1)$, x_1 and x_2 are fixed values $\in D(f)$.

For the reason that $f(x) = \max\big(f_1(x), f_2(x), \ldots, f_k(x)\big)$, each $f_i(x)$ has the same inequality:

$$\begin{aligned} f_1(\alpha x_1 + (1-\alpha)x_2) &\le \alpha f_1(x_1) + (1-\alpha) f_1(x_2) \\ &\le \alpha f(x_1) + (1-\alpha) f(x_2) \\ f_2(\alpha x_1 + (1-\alpha)x_2) &\le \alpha f_2(x_1) + (1-\alpha) f_2(x_2) \\ &\le \alpha f(x_1) + (1-\alpha) f(x_2) \\ &\vdots \\ f_k(\alpha x_1 + (1-\alpha)x_2) &\le \alpha f_k(x_1) + (1-\alpha) f_k(x_2) \\ &\le \alpha f(x_1) + (1-\alpha) f(x_2) \end{aligned}$$

Thus We can get this inequations:

$$\begin{aligned} f(\alpha x_1 + (1-\alpha)x_2) &= \max\big(f_1(\alpha x_1 + (1-\alpha)x_2), f_2(\alpha x_1 + (1-\alpha)x_2), \ldots, f_k(\alpha x_1 + (1-\alpha)x_2)\big) \\ &\le \alpha f(x_1) + (1-\alpha) f(x_2) \end{aligned}$$

Thus, it can be proved that $f(x) = \max\big(f_1(x), f_2(x), \ldots, f_k(x)\big)$ is a convex function.

(c) Let $T(x) = Ax + b$ be any affine function from $\mathbb{R}^n \to \mathbb{R}^m$ and let $g : \mathbb{R}^m \to \mathbb{R}$ be a convex function. Prove that $f(x) = g(T(x))$ is a convex function.

Answer:

From $T(x) = Ax + b$ is an affine function, then we can find following property:

$$\begin{aligned} T(\alpha x_1 + (1-\alpha)x_2) &= A(\alpha x_1 + (1-\alpha)x_2) + b \\ &= \alpha A x_1 + \alpha b + (1-\alpha) A x_2 + (1-\alpha) b \\ &= \alpha T(x_1) + (1-\alpha) T(x_2) \\ T(\alpha x_1 + (1-\alpha)x_2) &= \alpha T(x_1) + (1-\alpha) T(x_2) \end{aligned}$$

Based on this property, we cam simplify $f(\alpha x_1 + (1-\alpha)x_2)$ as following:

$$\begin{aligned} f(\alpha x_1 + (1-\alpha)x_2) &= g(T(\alpha x_1 + (1-\alpha)x_2)) \\ &= g(\alpha T(x_1) + (1-\alpha) T(x_2)) \\ g(\alpha T(x_1) + (1-\alpha) T(x_2)) &\le \alpha g(T(x_1)) + (1-\alpha) g(T(x_2)) \quad \text{(Convex function property)} \\ f(\alpha x_1 + (1-\alpha)x_2) &\le \alpha f((x_1) + (1-\alpha) f(x_2) \end{aligned}$$

Thus, it can be proved that $f(x) = g(T(x))$ is a convex function.

(d) Prove that the function $f : \mathbb{R}^n_> \to \mathbb{R}$ given by $f(x) = \sum_{i=1}^n x_i \log(x_i)$ is convex, where $\mathbb{R}^n_>$ denotes the set of vectors with strictly positive ccoordinates, so that $\log(x_i)$ is well-defined.

Answer:

To prove f is a convex function, we can prove $\langle \nabla f(\mathbf{y}) - \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle \geq 0$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ instead.

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \log x_1 + 1 \\ \log x_2 + 1 \\ \vdots \\ \log x_n + 1 \end{bmatrix}$$

Thus,

$$\nabla f(\mathbf{y}) - \nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial y_1} \\ \frac{\partial f}{\partial y_2} \\ \vdots \\ \frac{\partial f}{\partial y_n} \end{bmatrix} - \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \log \frac{y_1}{x_1} \\ \log \frac{y_2}{x_2} \\ \vdots \\ \log \frac{y_n}{x_n} \end{bmatrix}$$

Then $\langle \nabla f(\mathbf{y}) - \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle$ can be computed as:

$$\langle \nabla f(\mathbf{y}) - \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle = \begin{bmatrix} \log \frac{y_1}{x_1} \\ \log \frac{y_2}{x_2} \\ \vdots \\ \log \frac{y_n}{x_n} \end{bmatrix}^T \cdot \begin{bmatrix} y_1 - x_1 \\ y_2 - x_2 \\ \vdots \\ y_n - x_n \end{bmatrix} = \sum_{i=1}^{n} (y_i - x_i) \log \frac{y_i}{x_i}$$

For each term in the equation:

$$y_i - x_i = \begin{cases} \geq 0, & \text{if} \quad y_i \geq x_i > 0 \\ < 0, & \text{if} \quad 0 < y_i < x_i \end{cases}$$

$$\log \frac{y_i}{x_i} = \begin{cases} \geq 0, & \text{if} \quad y_i \geq x_i > 0 \\ < 0, & \text{if} \quad 0 < y_i < x_i \end{cases}$$

Thus,

$$(y_i - x_i) \log \frac{y_i}{x_i} \geq 0 \quad \forall x_i \text{ and } y_1 > 0$$

$$\sum_{i=1}^{n} (y_i - x_i) \log \frac{y_i}{x_i} \geq 0$$

$$\langle \nabla f(\mathbf{y}) - \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle \geq 0, \quad \forall \mathbf{x} \text{ and } \mathbf{y} \text{ are strictly positive coordinates}$$

Thus, it can be proved that $f(x) = \sum_{i=1}^{n} x_i \log(x_i)$ is a convex function.

Note:

Actually we can also prove $g(x) = xlogx$ to be convex using the second derivative condition, i.e. because every $x_i > 0$, $g''(x) = \frac{1}{x} > 0$, , thus $g(x)$ is convex. Then $f(x) = \sum_{i=1}^{n} g_i(x)$. Using the conclusion from Ex2.1(a), we can prove that $f(x)$ is also convex.

EX 2.2

(1). We can totally use the conclusion from HW1 Ex 1.2 (a), by assuming that $\phi(\alpha) = f(x_k + \alpha p_k)$, then we get:

$$f(x_k+\alpha p_k) - f(x_k) - \alpha g_k^T p_k \le \left| f(x_k+\alpha p_k) - f(x_k) - \alpha g_k^T p_k \right| \le \frac{\gamma}{2}\alpha^2 \|p_k\|^2$$

Thus, $f(x_k+\alpha p_k) = f(x_{k+1}) \le f(x_k) + \alpha g_k^T p_k + \frac{\gamma}{2}\alpha^2 \|p_k\|^2$ (Q.E.D.)

Then, replace $p_k = -B_k^{-1} g_k$, we get

$$f(x_{k+1}) \le f(x_k) - \alpha g_k^T B_k^{-1} g_k + \frac{\gamma}{2}\alpha^2 g_k^T B_k^{-2} g_k$$

λ_{max} and λ_{min} are global bounds on eigen-values of B_k, then

$$g_k^T B^{-1} g_k \ge \frac{1}{\lambda_{max}} \|g_k\|^2, \quad g_k^T B_k^{-2} g_k \le \frac{1}{\lambda_{min}^2} \|g_k\|^2$$

$$\therefore f(x_{k+1}) \le f(x_k) - \alpha \frac{\|g_k\|^2}{\lambda_{max}} + \frac{\gamma\alpha^2}{2\lambda_{min}^2} \|g_k\|^2, \text{ (Q.E.D.)}$$

(2). Let $h(\alpha) = -\alpha \frac{\|g_k\|^2}{\lambda_{max}} + \frac{\gamma\alpha^2}{2\lambda_{min}^2} \|g_k\|^2$

Because $h(\alpha)$ is a convex quadratic function, let $h'(\alpha)=0$ can get the minimizer!

$$h'(\alpha) = -\frac{\|g_k\|^2}{\lambda_{max}} + \frac{\gamma\alpha}{\lambda_{min}^2}\|g_k\|^2 = 0 \Rightarrow \alpha_m = \frac{\lambda_{min}^2}{\lambda_{max}\cdot\gamma}$$

Because $\gamma > 0$, λ_{min} & λ_{max} are eigen-values of positive definite matrix B_k.

So $\alpha_m > 0$.

$$h(\alpha_m) = -\frac{\lambda_{min}^2}{\lambda_{max}\gamma}\cdot\frac{\|g_k\|^2}{\lambda_{max}} + \frac{\gamma\|g_k\|^2}{2\lambda_{min}^2}\cdot\frac{\lambda_{min}^4}{\lambda_{max}^2\gamma^2} = -\frac{1}{2}\frac{\lambda_{min}^2}{\lambda_{max}^2\gamma}\|g_k\|^2 < 0$$

$\therefore$ At $\alpha_m = \frac{\lambda_{min}^2}{\lambda_{max}\gamma}$, $h(\alpha)$ is minimized, the minimizer α_m is positive, the minimum value of $h(\alpha)$ is negative.

(3). From (2) we know that

$$f(x_{k+1}) - f(x_k) \le -\frac{\lambda_{min}^2}{2\lambda_{max}^2\gamma}\|g_k\|^2$$
$$f(x_k) - f(x_{k-1}) \le -\frac{\lambda_{min}^2}{2\lambda_{max}^2\gamma}\|g_{k-1}\|^2$$
$$\vdots$$
$$f(x_1) - f(x_0) \le -\frac{\lambda_{min}^2}{2\lambda_{max}^2\gamma}\|g_0\|^2$$

add them together $\Longrightarrow$ $f(x_{k+1}) - f(x_0) \le -\frac{\lambda_{min}^2}{2\lambda_{max}^2\gamma}\sum_{k=0}^{k}\|g_k\|$

Because $f(x_{k+1}) \ge l$ and $-\frac{\lambda_{min}^2}{2\lambda_{max}^2\gamma} < 0$

$$\therefore \sum_{i=0}^{k}\|g_i\|^2 \le \frac{2\lambda_{max}^2\gamma}{\lambda_{min}^2}(f(x_0) - f(x_{k+1})) \le \frac{2\lambda_{max}^2\gamma}{\lambda_{min}^2}(f(x_0) - l)$$

And, $\sum_{i=0}^{k}\|g_i\|^2 \ge (k+1)\min_{i=0,1,\cdots k}\|g_i\|^2$

$$\therefore (k+1)\min_{i=0,1,\cdots k}\|g_i\|^2 \le \frac{2\lambda_{max}^2\gamma}{\lambda_{min}^2}(f(x_0) - l)$$

$$\therefore \min_{i=0,1,\cdots k}\|g_i\|^2 \le \frac{2\lambda_{max}^2\gamma}{\lambda_{min}^2}\cdot\frac{f(x_0)-l}{k+1} \quad \therefore \min_{i=0,1,\cdots k}\|g_i\| \le \sqrt{\frac{2\lambda_{max}^2\gamma}{\lambda_{min}^2}(f(x_0)-l)}\cdot\frac{1}{\sqrt{k+1}}$$

Denote $\sqrt{\frac{2\lambda_{max}^2\gamma}{\lambda_{min}^2}(f(x_0)-l)} = M$ and replacing i, k to k, T, we get

$$\min_{k=0,1,\cdots T}\|g_k\| = \min_{k=0,1,\cdots T}\|\nabla f(x_k)\| \le \frac{M}{\sqrt{T+1}}, \quad (Q.E.D)$$

(4). When $B_k = I$, $\lambda_{min} = \lambda_{max} = 1$, $\therefore \alpha = \frac{\lambda_{min}^2}{\lambda_{max}\gamma} = \frac{1}{\gamma}$

$$M = \sqrt{\frac{2\lambda_{max}^2\gamma}{\lambda_{min}^2}(f(x_0)-l)} = \sqrt{2\gamma(f(x_0)-l)}$$

EX 2.4 b

We can see that using modified newton direction with Armijo backtracking only needs 33 iterations to get to the solution. But using steepest descent direction with Armijo backtracking can't even reach the solution within 50 iterations. We can see that modified newton has an advantage over steepest descent in terms of convergence speed because modified newton uses the second derivatives information while steepest descent only uses gradient.

```
tol      = 1.0e-8  ; % VERY tight stopping tolerance
maxit    = 50       ; % maximum number of iterations allowed
x0       = [0; 0]  ; % initial guess at a solution
fun      = 'fgh';

printlevel = 1;

step = 0; %%%
[x,F,G,H,iter,status] = uncMIN(fun,x0,step,maxit,printlevel,tol);
disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%')
disp('Steepest descent:')
disp('x is:')
disp(x)
disp('How many iterations were done:')
disp(iter)
disp('If the final stopping tolerance was met:')
disp(status)
disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%')


disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%')
disp('Modified Newton method:')
step = 1; %%%
[x,F,G,H,iter,status] = uncMIN(fun,x0,step,maxit,printlevel,tol);
disp('x is:')
disp(x)
disp('How many iterations were done:')
disp(iter)
disp('If the final stopping tolerance was met:')
disp(status)
disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%')
```

```
 iter       |F|            |G|           |H|
    1   6.015625e-01   1.525819e+00  2.604436e+01
    2   4.856232e-01   1.246565e+00  2.817797e+01
    3   4.071306e-01   1.088531e+00  3.313294e+01
    4   3.474143e-01   9.961289e-01  3.464624e+01
    5   3.035430e-01   1.004085e+00  3.982003e+01
    6   2.797938e-01   7.681838e-01  3.952526e+01
    7   2.215572e-01   9.710776e-01  4.681663e+01
    8   2.100249e-01   6.439027e-01  4.615487e+01
    9   1.888613e-01   6.877641e-01  4.701548e+01
   10   1.779430e-01   5.763676e-01  4.941484e+01
   11   1.615602e-01   6.528679e-01  5.003522e+01
   12   1.559922e-01   5.027863e-01  5.126148e+01
   13   1.303171e-01   7.112414e-01  5.654288e+01
   14   1.246252e-01   4.307734e-01  5.581347e+01
   15   9.102121e-02   9.546165e-01  6.028813e+01
   16   8.538361e-02   5.746816e-01  6.143639e+01
   17   8.322336e-02   4.078985e-01  6.212513e+01
   18   8.109167e-02   3.218592e-01  6.300304e+01
```

```
   19   4.076713e-02   5.495567e-01  7.464139e+01
   20   3.899702e-02   2.965706e-01  7.407811e+01
   21   3.841391e-02   2.235133e-01  7.390399e+01
   22   3.716036e-02   2.326966e-01  7.385344e+01
   23   3.644391e-02   1.992393e-01  7.437212e+01
   24   2.217539e-02   6.437834e-01  7.957655e+01
   25   1.989986e-02   2.737824e-01  8.053223e+01
   26   1.945137e-02   1.636735e-01  8.093863e+01
   27   1.910941e-02   1.392449e-01  8.133220e+01
   28   1.710721e-02   2.488438e-01  8.196534e+01
   29   1.673774e-02   1.486397e-01  8.233879e+01
   30   1.645264e-02   1.282341e-01  8.269387e+01
   31   1.477645e-02   2.377048e-01  8.325717e+01
   32   1.444356e-02   1.377279e-01  8.361917e+01
   33   1.420100e-02   1.185559e-01  8.395336e+01
   34   1.282058e-02   2.373920e-01  8.442662e+01
   35   1.249524e-02   1.299532e-01  8.479406e+01
   36   1.228527e-02   1.100813e-01  8.512031e+01
   37   1.117956e-02   2.473463e-01  8.548551e+01
   38   1.083503e-02   1.251210e-01  8.587406e+01
   39   1.064976e-02   1.027775e-01  8.620349e+01
   40   1.011051e-02   1.484641e-01  8.632196e+01
   41   9.972060e-03   1.027874e-01  8.654194e+01
   42   9.701340e-03   1.135200e-01  8.696705e+01
   43   9.542504e-03   9.643128e-02  8.687946e+01
   44   9.056630e-03   1.355475e-01  8.755187e+01
   45   8.940227e-03   9.594923e-02  8.744892e+01
   46   8.697818e-03   1.048253e-01  8.745064e+01
   47   8.559169e-03   9.072950e-02  8.771250e+01
   48   8.122767e-03   1.250929e-01  8.784582e+01
   49   8.022883e-03   9.003745e-02  8.802955e+01
   50   7.664374e-03   1.537554e-01  8.874317e+01
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Steepest descent:
x is:
    0.9151
    0.8306

How many iterations were done:
    50

If the final stopping tolerance was met:
     1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Modified Newton method:

 iter       |F|            |G|           |H|
    1   6.015625e-01   1.525819e+00  2.604436e+01
    2   3.751206e-01   1.194635e+00  3.581462e+01
    3   2.245904e-01   9.590600e-01  4.645622e+01
    4   1.253837e-01   8.055402e-01  5.773955e+01
    5   6.441517e-02   6.658475e-01  6.863324e+01
```

```
     6   2.999793e-02   5.200024e-01  7.831432e+01
     7   1.249005e-02   3.756023e-01  8.621860e+01
     8   4.613482e-03   2.488149e-01  9.211399e+01
     9   1.517515e-03   1.515554e-01  9.611478e+01
    10   4.523481e-04   8.598354e-02  9.859393e+01
    11   1.254336e-04   4.630971e-02  1.000161e+02
    12   3.320298e-05   2.412348e-02  1.007872e+02
    13   8.555145e-06   1.232555e-02  1.011903e+02
    14   2.172281e-06   6.231779e-03  1.013966e+02
    15   5.473705e-07   3.133547e-03  1.015010e+02
    16   1.373875e-07   1.571241e-03  1.015536e+02
    17   3.441545e-08   7.867437e-04  1.015799e+02
    18   8.612464e-09   3.936536e-04  1.015931e+02
    19   2.154193e-09   1.968973e-04  1.015997e+02
    20   5.386830e-10   9.846630e-05  1.016030e+02
    21   1.346876e-10   4.923756e-05  1.016047e+02
    22   3.367401e-11   2.461988e-05  1.016055e+02
    23   8.418765e-12   1.231022e-05  1.016059e+02
    24   2.104724e-12   6.155178e-06  1.016061e+02
    25   5.261852e-13   3.077606e-06  1.016062e+02
    26   1.315468e-13   1.538807e-06  1.016063e+02
    27   3.288677e-14   7.694048e-07  1.016063e+02
    28   8.221700e-15   3.847027e-07  1.016063e+02
    29   2.055426e-15   1.923514e-07  1.016063e+02
    30   5.138566e-16   9.617572e-08  1.016063e+02
    31   1.284642e-16   4.808786e-08  1.016063e+02
    32   3.211605e-17   2.404393e-08  1.016063e+02
    33   8.029013e-18   1.202197e-08  1.016063e+02
x is:
    1.0000
    1.0000

How many iterations were done:
    33

If the final stopping tolerance was met:
     0

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Published with MATLAB® R2018b

```
function [F,G,H] = fgh(x)
    x1 = x(1);
    x2 = x(2);
    F = 10*(x2-x1^2)^2 + (x1-1)^2;

    G = [-40*(x2*x1-x1^3) + 2*(x1-1);
         20*(x2-x1^2)];

    H = [-40*(x2-3*x1^2)+2, -40*x1;
        -40*x1, 20];
end
```

```
Not enough input arguments.

Error in fgh (line 2)
    x1 = x(1);
```

Published with MATLAB® R2018b

```
function [ B, flag ] = modNewton( H, beta )

    assert(issymmetric(H),'Input H needs to be symmetric!')
    assert(beta > 1,'Input beta needs to be larger than 1')

    flag = 0;
    [V,D] = eig(H); % H*V = V*D, H = V*D*V'
    if all(diag(D) > 0)
        B = H;
    else
        flag = 1;

        if H == 0
            epsilon = 1;
        else
            epsilon = norm(H)/beta;
        end

        for i = 1:length(D)
            if D(i,i) <= (-1)*epsilon
                D(i,i) = (-1)*D(i,i);
%                 flag = 1;
            elseif D(i,i) >= epsilon
                D(i,i) = D(i,i);
            else
                D(i,i) = epsilon;
%                 flag = 1;
            end

        end

        B = V*D*V';
    end
end
```

```
Not enough input arguments.

Error in modNewton (line 3)
    assert(issymmetric(H),'Input H needs to be symmetric!')
```

Published with MATLAB® R2018b

```
function [x,F,G,H,iter,status] =
 uncMIN(fun,x0,step,maxit,printlevel,tol)

    func = str2func(fun);
    %
    iter     = 0        ;
    [F0,G0,H0]  = func(x0);
    F        = F0       ;
    G        = G0       ;
    H        = H0       ;
    x        = x0       ;

    if printlevel == 1
        fprintf('\n iter       |F|            |G|           |H|\n');
    end

    while norm(G) > tol*max(1,norm(G0)) && iter < maxit

        iter = iter + 1;

        if step == 0 % steepest descent
            p = -G;
        else % modified newton
            beta = 2;
            [B, flag] = modNewton(H, beta);
            p = -B\G;
        end
        % compute alpha

        alpha0 = 1;
        tao = 0.5;
        eta = 0.7;

        alpha = alpha0;
        x_new = x + alpha * p;
        [F_new, G_new, H_new] = func(x_new);

        while F_new > F + eta*alpha*G'*p
            alpha = tao*alpha;
            x_new = x + alpha * p;
            [F_new, G_new, H_new] = func(x_new);
        end
        % compute alpha
        x = x + alpha*p;
        [F,G,H] = func(x);

        if printlevel == 1
           fprintf(' %4g  %13.6e  %13.6e %13.6e \n', iter, norm(F),
 norm(G), norm(H))
       end

    end
```

```
    status = 0;
    if iter == maxit
        status = 1;
    end

end
```

```
Not enough input arguments.

Error in uncMIN (line 3)
    func = str2func(fun);
```

Published with MATLAB® R2018b