Prof. Dr. Karsten Urban
M. Sc. Lewin Ernst

Winter term 2024/2025
Sheet 02, 31 Points

Numerical Optimization
Solution to exercise sheet

review on 30.10.2024 during the exercise class

1. *(Levenberg-Marquardt Method)*

   Consider again for a parameter $\lambda \in \mathbb{R}$ the parametrized function

$$F_\lambda : \mathbb{R} \to \mathbb{R}^2, \qquad F_\lambda(x) = \begin{pmatrix} x+1 \\ \lambda x^2 + x - 1 \end{pmatrix}$$

   and the nonlinear least squares problem

$$g(x) := \frac{1}{2}\|F_\lambda(x)\|_2^2 \to \min, \tag{1}$$

   which we have already seen on the last sheet.

   a) Prove that for every $\lambda < 1$ there exists a penalty-parameter $\mu > 0$, such that the Levenberg-Marquardt-Method converges locally to $x^* = 0$.

   b) In the material you will find the MATLAB-function

```
x = LevenbergMarquardt(F, dF, x0, mu, beta0, beta1, maxIt, tol),
```

   which computes the solution `x` $\in \mathbb{R}^n$ of the nonlinear least squares problem by using the Levenberg-Marquardt-Method. The damping parameter μ is controlled via the expression

$$\varepsilon_\mu := \frac{\|F(x_k)\|_2^2 - \|F(x_k + s_k)\|_2^2}{\|F(x_k)\|_2^2 - \|F(x_k) + F'(x_k)s_k\|_2^2}$$

   and the parameter $0 < \beta_0 < \beta_1 < 1$. Adjust the function such that the error in the Hessian approximation, i.e.

$$\frac{\left\| (\nabla^2 g)(x^{(k)}) - \left[\left(F'(x^{(k)})\right)^T F'(x^{(k)}) + \mu_k^2 I \right] \right\|_2}{\|(\nabla^2 g)(x^{(k)})\|_2}$$

   gets returned for all iterations as well as all damping parameter μ_k.

   c) Apply the method to the problem from above by adjusting the script from sheet 1 task 3 d) (it will be also in the material or you can use your own). You should monitor the convergence, the error of the hessian and the damping parameters μ_k for the Levenberg-Marquardt method. What do you observe for $\lambda < -1$?

   d) Copy your script from c) and adjust it, so that the circle regression from the lecture notes is solved and analysed. You can use ChatGPT to calculate the Hessian of F_i. It can provide you also a MATLAB implementation, but be careful, it can make mistakes when calculating the Hessian.

$(4 + 2 + 4 + 5 = 15$ Points$)$

Solution:

a) The fixpoint function for the Levenberg-Marquardt method is

$$\Phi_{\mu}^{LM}(x) = x - [F'(x)^T F'(x) + \mu^2 I]^{-1}\ F'(x)^T F(x) = x - [F'(x)^T F'(x) + \mu^2 I]^{-1}\ \nabla g(x) \quad (2)$$

We plug the function F into (2) and we get:

$$\Phi_{\mu,\lambda}^{LM}(x) = \frac{2\lambda(4\lambda+\mu^2)x + \mu^2 + 2}{4\lambda(4\lambda^2x^2 + 4\lambda x + \mu^2 + 2)} + \frac{x}{2} - \frac{1}{4\lambda}.$$

Let now $\lambda < 1$. We notice, that $\Phi_{\mu,\lambda}^{LM}$ is at least continuously differentiable in a neighborhood of $x^* = 0$ with

$$\frac{d}{dx}\Phi_{\mu,\lambda}^{LM}(x) = \frac{1}{2} - \frac{4\lambda^2(4\lambda+\mu^2)x^2 + 4\lambda(\mu^2+2)x - (4\lambda+\mu^2-2)(\mu^2+2)}{2(4\lambda^2x^2 + 4\lambda x + \mu^2 + 2)^2}$$

and therefore

$$\left|\frac{d}{dx}\Phi_{\mu,\lambda}^{LM}(0)\right| = \frac{|2\lambda+\mu^2|}{\mu^2+2}.$$

Let now $\lambda < 1$ be fix. Choose e.g. $\mu^2 := 2|\lambda| + 1$. Then $\left|\frac{d}{dx}\Phi_{\mu,\lambda}^{LM}(0)\right| < 1$ holds. Because of the continuity of $\frac{d}{dx}\Phi_{\mu,\lambda}^{LM}$, there is a neighbourhood of $x^* = 0$ such that $\left|\frac{d}{dx}\Phi_{\mu,\lambda}^{LM}(x)\right| < 1$ holds for all $x \in B_\varepsilon(0)$ with $\varepsilon > 0$ small enough. Hence, there is $\delta > 0$ with

$$\max_{x \in \overline{B_\delta(0)}} \left|\frac{d}{dx}\Phi_{\mu,\lambda}^{LM}(x)\right| = L < 1.$$

With the choice $\mu^2 := 2|\lambda| + 1$ for given $\lambda < 1$ the convergence of the Levenberg-Marquardt method follows from Banach's fixpoint theorem.

b) The function could look like:

```
function [x,iter,error_hess, mu] = LevenbergMarquardt(F, dF, hess, ...
                                                       x0, muk, ...
                                                       beta0, beta1, ...
                                                       maxIt, tol)

% Initialise iterates
x     = x0;
[M,N] = size(x0);

% iteration count
k = 1;

F_xk  = F(x0);
dF_xk = dF(x0);

% set up the matrix and right hand side for LSQP
A = [dF_xk;muk^2*eye(M)];
b = [-F_xk;zeros(M,N)];

% linear least squares problem solver
s     = lscov(A,b);
F_xks = F(x0+s);

% estimator for mu
```

```
eps_mu = (F_xk'*F_xk - F_xks'*F_xks) / ...
    (F_xk'*F_xk - (F_xk+dF_xk*s)'*(F_xk+dF_xk*s));

% Norm of (J' * J)
error_hess(k) = norm((dF_xk' * dF_xk + muk^2*eye(M))-...
                    hess(x(:,k)),2)/norm(hess(x(:,k)),2);

% for monitoring mu
mu(k) = muk;

% Iteration
while norm(s) > tol && k < maxIt

    % while steps are not accepted try to double the penalty parameter
    while eps_mu <= beta0
        % double mu if step is not accepted
        muk = 2*muk;

        % set up the matrix and right hand side for LSQP
        A = [dF_xk;muk^2*eye(M)];
        b = [-F_xk;zeros(M,N)];

        % linear least squares problem solver
        s = lscov(A,b);

        % calculate the residual at x_k+1
        F_xks = F(x(:,k)+s);

        % estimator for mu
        eps_mu = (F_xk'*F_xk - F_xks'*F_xks) / ...
          (F_xk'*F_xk - (F_xk+dF_xk*s)'*(F_xk+dF_xk*s));

        if norm(s) < tol
            break
        end
    end

    % if mu is too large half it
    if eps_mu >= beta1
        muk = muk/2;
    end

    % take a step
    x(:,k+1) = x(:,k) + s;

    % eval residual and derivative
    F_xk  = F(x(:,k+1));
    dF_xk = dF(x(:,k+1));

    % store mu
    mu(k+1) = muk;

    % error of hessian
    error_hess(k+1) = norm((dF_xk' * dF_xk + muk^2*eye(M))-hess(x(:,k+1)),2)...
                     /norm(hess(x(:,k+1)),2);

    % set up the matrix and right hand side for LSQP
    A = [dF_xk;muk^2*eye(M)];
    b = [-F_xk;zeros(M,N)];

    % linear least squares problem solver
    s = lscov(A,b);
```

```
    % update residual
    F_xks = F(x(:,k)+s);

    % estimator for mu
    eps_mu = (F_xk'*F_xk - F_xks'*F_xks) / ...
        (F_xk'*F_xk - (F_xk+dF_xk*s)'*(F_xk+dF_xk*s));

    % update iteration count
    k = k + 1;
end

iter = k;

end
```

c) The script could look like:

```
% Gauss-Newton Method on a Nonlinear Oscillatory Problem
clc; clear; close all;

% Define the parameter
lambda = -2;

% Define the residual function
f = @(x) [ x + 1; lambda*x.^2 + x - 1 ];

% Define the Jacobian of the residual function
J = @(x) [ 1; 2*lambda*x + 1 ];

% Function for newtons method
g = @(x) 0.5*((x+1).^2+(lambda*x.^2+x-1).^2);
nabla_g = @(x) x*(2*lambda.^2*x.^2+3*lambda*x-2*(lambda-1));
Hess_g  = @(x) 6*lambda.^2*x.^2+6*lambda*x-2*(lambda-1);

% Define the initial value and termination parameters
x0    = 10;
maxIt = 1e2;
tol   = 1e-8;

% Apply the Gauss-Newton method
[x_GN, res, iter_GN, error_hess] = gaussNewton_ChatGPT(f, J, Hess_g, ...
                                                       x0, maxIt, tol);
% Apply the Newton method
[x_N, iter_N] = newtonsMethod(nabla_g, Hess_g, x0, maxIt, tol);
% Apply the Levenberg-Marquardt method
beta0 = 0.3;
beta1 = 0.9;
mu    = 1;
[x_LM, iter_LM, error_hess_LM, mu_LM] = LevenbergMarquardt(f, J, Hess_g, x0,...
                                                           mu, beta0, beta1, ...
                                                           maxIt, tol);

% Plot the path taken by the Gauss-Newton method
figure(1);
subplot(3,1,1);
semilogy(1:iter_GN+1, abs(x_GN), 'r-o', 'MarkerSize', 5, ...
     'LineWidth', 1.5);
hold on
semilogy(1:iter_N+1, abs(x_N), 'b-x', 'MarkerSize', 5, ...
     'LineWidth', 1.5);
```

```
semilogy(1:iter_LM, abs(x_LM), 'g-v', 'MarkerSize', 5, ...
     'LineWidth', 1.5);

title("Gaus-Netwon, Newton and Levenberg-Marquardt Convergence");
xlabel('Iterations');
ylabel('Absolute Error $\Vert x_{sol}-x^{(k)} \Vert_2$',Interpreter='latex');
legend('Gaus-Newton','Newton', 'Levenberg-Marquardt');
grid on;
hold off
fontsize(16,"points")

subplot(3,1,2);
plot(1:iter_GN, error_hess, 'r-x', 'MarkerSize', 5, ...
     'LineWidth', 1.5);
hold on
plot(1:iter_LM, error_hess_LM, 'g-x', 'MarkerSize', 5, ...
     'LineWidth', 1.5);
title("Error of Hessian approximation");
xlabel('Iterations');
ylabel('Error [%]');
legend('error of hessian GN', 'error of hessian LM');
grid on;
hold off;
fontsize(16,"points")

subplot(3,1,3);
plot(1:iter_LM, mu_LM, 'g-x', 'MarkerSize', 5, ...
     'LineWidth', 1.5);
title("Evolution of penalty parameter \mu");
xlabel('Iterations');
ylabel('Value');
legend('\mu');
xlim([0,max(iter_LM,iter_GN)])
grid on;
hold off;
fontsize(16,"points")
```

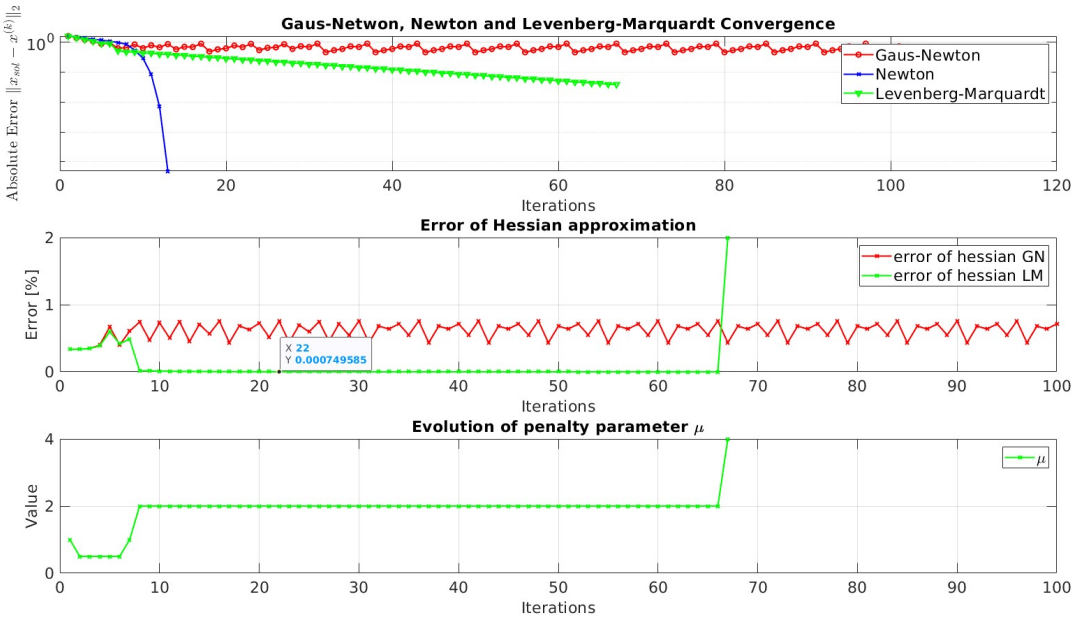
d) The script could look like:

```
clear, close all
clc
%----prescribe the data ----------------------------------------
sol = [3,2,2]';
N=100;                          % no of points
dr = 0.1;                      % max deviation from radius 2
                                % severelv impacts convergence
rad_L=(2*dr).*randn(N,1)+2-dr;  % radius with random distortions
zx_L=3;                         % 'exact' center coordinate
zy_L=2;                         %
x_L=zx_L*ones(N,1) + cos(linspace(0,2*pi,N)').*rad_L;
y_L=zy_L*ones(N,1) + sin(linspace(0,2*pi,N)').*rad_L;
%-------------------------------------------------------------

x=x_L;
y=y_L;
m=length(x);

%--- provide F and Jacobian as functions ----------------------
F=@(p) sqrt((x-p(1)).^2+(y-p(2)).^2)-p(3);
DF=@(p) [(p(1)-x)./sqrt((x-p(1)).^2+(y-p(2)).^2), ...
         (p(2)-y)./sqrt((x-p(1)).^2+(y-p(2)).^2), ...
```

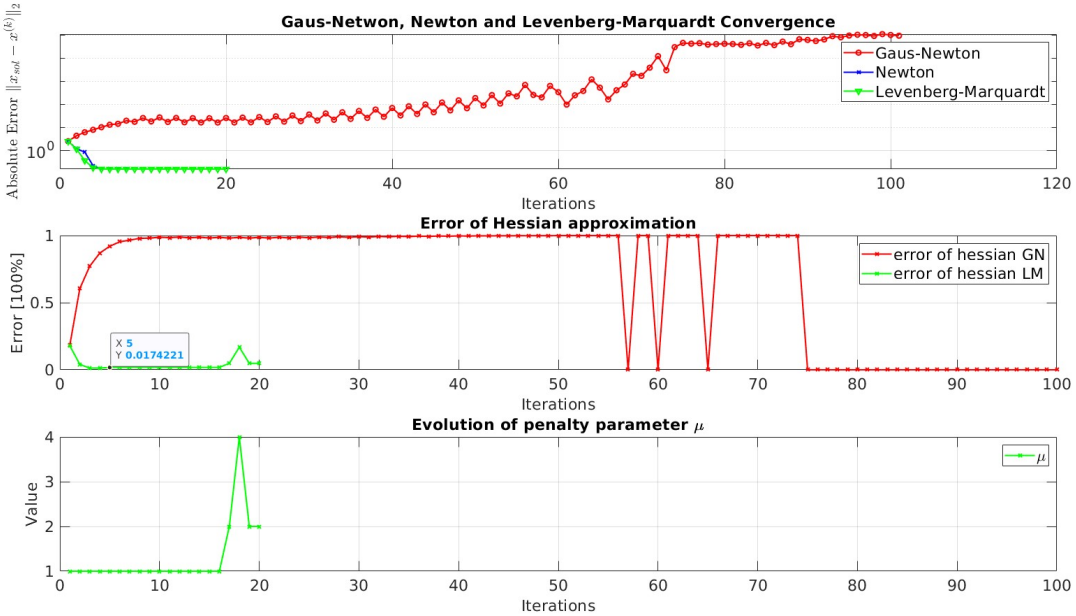
```
         -ones(m,1)];
%-------------------------------------------------------------


%--- provide derivatives of g as functions ----------------------
% Function for newtons method
g       = @(p) 0.5*F(p)'*F(p);
nabla_g = @(p) DF(p)'*F(p);
Hess_g  = @(p) HessianCircle(p, x, y, F, DF);

% Define the initial value and termination parameters
 %-- initial value - might severely influence convergence
x0    = [1;1;1];
maxIt = 1e2;
tol   = 1e-10;

% Apply the Gauss-Newton method
[x_GN, res, iter_GN, error_hess] = gaussNewton_ChatGPT(F, DF, Hess_g, ...
                                                       x0, maxIt, tol);
% Apply the Newton method
[x_N, iter_N] = newtonsMethod(nabla_g, Hess_g, x0, maxIt, tol);
% Apply the Levenberg-Marquardt method
beta0 = 0.3;
beta1 = 0.9;
mu    = 1;
[x_LM, iter_LM, error_hess_LM, mu_LM] = LevenbergMarquardt(F, DF, Hess_g, ...
                                                           x0, mu, ...
                                                           beta0, beta1, ...
                                                           maxIt, tol);

% Plot the path taken by the Gauss-Newton method
figure(1);
subplot(3,1,1);
semilogy(1:iter_GN+1, vecnorm(x_GN-sol,2,1), 'r-o', 'MarkerSize', 5, ...
     'LineWidth', 1.5);
hold on
semilogy(1:iter_N+1, vecnorm(x_N-sol,2,1), 'b-x', 'MarkerSize', 5, ...
     'LineWidth', 1.5);
semilogy(1:iter_LM, vecnorm(x_LM-sol,2,1), 'g-v', 'MarkerSize', 5, ...
     'LineWidth', 1.5);

title("Gaus-Netwon, Newton and Levenberg-Marquardt Convergence");
xlabel('Iterations');
ylabel('Absolute Error $\Vert x_{sol}-x^{(k)} \Vert_2$',Interpreter='latex');
legend('Gaus-Newton','Newton', 'Levenberg-Marquardt');
grid on;
hold off
fontsize(16,"points")

subplot(3,1,2);
plot(1:iter_GN, error_hess, 'r-x', 'MarkerSize', 5, ...
     'LineWidth', 1.5);
hold on
plot(1:iter_LM, error_hess_LM, 'g-x', 'MarkerSize', 5, ...
     'LineWidth', 1.5);
title("Error of Hessian approximation");
xlabel('Iterations');
ylabel('Error [100%]');
legend('error of hessian GN', 'error of hessian LM');
grid on;
hold off;
fontsize(16,"points")
```

```
subplot(3,1,3);
plot(1:iter_LM, mu_LM, 'g-x', 'MarkerSize', 5, ...
     'LineWidth', 1.5);
title("Evolution of penalty parameter \mu");
xlabel('Iterations');
ylabel('Value');
legend('\mu');
xlim([0,max(iter_LM,iter_GN)])
grid on;
hold off;
fontsize(16,"points")
```

Exercise c) with $\lambda = -2$



Exercise d)



2. *(Convex functions)*

a) Let $X \subset \mathbb{R}^n$ be convex. Prove:
If $f_i : X \to \mathbb{R}$ is convex and $\alpha_i \geq 0$ for $i = 1, \ldots, m$, then

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

is convex on X. If at least one f_i is strictly convex and the corresponding $\alpha_i > 0$, then f is strictly convex on X.

b) Let $X \subset \mathbb{R}^n$ be convex. Prove:
If $g : X \to \mathbb{R}^m$ is affine and $f : \mathrm{Im}(g) \to \mathbb{R}$ is convex, then $(f \circ g)$ is convex on X, whereas $\mathrm{Im}(g)$ denotes the image of g. (Realise why $\mathrm{Im}(g)$ is convex!)

$(4 + 4 = 8$ Points)

Solution:

a) Let $\lambda \in (0, 1)$, then $\lambda x + (1 - \lambda)y \in X$ for all $x, y \in X$, because X is convex. If in addition there holds

$$f_i(\lambda x + (1 - \lambda)y) \leq \lambda f_i(x) + (1 - \lambda) f_i(y),$$

for all $i = 1, \ldots, m$, $\lambda \in (0, 1)$ and all $x, y \in X$, then f_i is convex. With $\alpha_i \geq 0$ if follows

$$\alpha_i f_i(\lambda x + (1 - \lambda)y) \leq \alpha_i \left(\lambda f_i(x) + (1 - \lambda) f_i(y)\right),$$

and therefore

$$\begin{aligned} f(\lambda x + (1 - \lambda)y) &= \sum_{i=1}^{m} \alpha_i f_i(\lambda x + (1 - \lambda)y) \leq \sum_{i=1}^{m} \alpha_i \left(\lambda f_i(x) + (1 - \lambda) f_i(y)\right) \\ &= \lambda \sum_{i=1}^{m} \alpha_i f_i(x) + (1 - \lambda) \sum_{i=1}^{m} \alpha_i f_i(y) = \lambda f(x) + (1 - \lambda) f(y). \end{aligned}$$

If at least one f_i is strictly convex and $\alpha_i > 0$, then the strict convexity of f follows analogously, because for i it holds

$$\alpha_i f_i(\lambda x + (1 - \lambda)y) < \alpha_i \left(\lambda f_i(x) + (1 - \lambda) f_i(y)\right).$$

b) For an affine function $g : X \to \mathbb{R}^m$, $X \subseteq \mathbb{R}^n$ there is a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $b \in \mathbb{R}^m$ such that g can be represented as

$$g(x) = Ax + b.$$

The convexity of $\mathrm{Bild}(g)$ follows from the linearity of g: let $r, s \in \mathrm{Bild}(g)$, i.e. there exists $x, y \in \mathbb{R}^n$ with $r = g(x)$ and $s = g(y)$. Then for all $0 \leq \alpha \leq 1$:

$$\begin{aligned} \alpha r + (1 - \alpha)s &= \alpha(Ax + b) + (1 - \alpha)(Ay + b) = A(\alpha x + (1 - \alpha)y) + b \\ &= g(\alpha x + (1 - \alpha)y) \in \mathrm{Bild}(g). \end{aligned}$$

Therefore $\mathrm{Bild}(g)$ is convex, and it is left to show, that the function $h := f \circ g$ is convex on C. To do so let $x, y \in X$ and $\lambda \in (0, 1)$ be arbitrary, then it follows

$$\begin{aligned} \lambda h(x) + (1 - \lambda)h(y) &= \lambda f\left(g(x)\right) + (1 - \lambda) f(g(y)) \\ &\geq f\left(\lambda g(x) + (1 - \lambda) g(y)\right) \; = f\left(\lambda A x + \lambda b + (1 - \lambda) A y + (1 - \lambda) b\right) \\ &= f\left(A(\lambda x + (1 - \lambda)y) + b\right) = f\left(g(\lambda x + (1 - \lambda)y)\right) \\ &= h(\lambda x + (1 - \lambda)y). \end{aligned}$$

3. *(Convexity and Minimizers)*

a) Let $f : X \to \mathbb{R}$ be convex on some convex set $X \subset \mathbb{R}^n$. Then the set $S := \{x \in X : f(x) \leq f(y)\ \forall y \in X\}$ is convex, i.e. the set of global minimizers is convex.

b) A point $x^* \in \mathcal{U} \subset \mathbb{R}^n$ is called an isolated local minimizer of $f : \mathcal{U} \to \mathbb{R}$, if there is a neighborhood $U(x^*)$ such that x^* is the only local minimizer in $U(x^*)$. Prove that isolated local minimizers are strict local minimizers.
Hint: Prove the contraposition.
Remark: Not every strict local minimizer is isolated, consider e.g. $x^* = 0$ of the function

$$f(x) = \begin{cases} x^4 \cos(1/x) + 2x^4, & x \neq 0 \\ 0, & \text{else} \end{cases} \in C^2(\mathbb{R}).$$

(4 + 4 = 8 Points)

Solution:

a) We shall prove that S is convex. If S has only one element, then it is certainly convex. So lets assume it has more than one element. Then for all $x, y \in S$ and $\lambda \in [0, 1]$,

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda) f(y),$$

since f is convex. We know that x, y are global minimizers, so that $f(x) = f(y)$ and therefore

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda) f(x) = f(x).$$

The latter inequality can only hold with equality, because x is a global minimizer and it follows $f(\lambda x + (1-\lambda)y) = f(x)$ and that $\lambda x + (1-\lambda)y$ is a global minimizer. Hence $\lambda x + (1-\lambda)y \in S$ and S is convex.

b) We prove the contraposition: If x^* is not a strict local minimizer, then x^* is not an isolated local minimizer. There are two cases:
Case I: x^* is not even a local minimizer, then it can not be an isolated local minimizer.
Case II: x^* is a local minimizer but not strict, i.e. let $U(x^*)$ be a neighborhood, such that $f(x^*) \leq f(x) \forall x \in U(x^*)$. Because the minimum is not strict, there must be $\tilde{x} \in U(x^*) \setminus \{x^*\}$ such that $f(\tilde{x}) = f(x^*)$. Hence $\tilde{x}$ is also a local minimizer in $U(x^*)$. Since we can do this for every neighborhoods $U(x^*)$ with $f(x^*) \leq f(x) \forall x \in U(x^*)$, x^* cannot be an isolated local minimizer.