universität ulm

Numerical Optimization
Winter semester 2025/26

Exercise sheet 1 for Numerical Optimization

Manuel Caipo
Degree program: M.Sc. Computational Science Engineering
Enrollment number: 30783212

Instructors: Dr. Michael Lehn, Tobias Born

22. October 2025 - 89081 Ulm

Chapter 1

Solutions Exercise sheet

1.1 Exercise 1

1.1.1 Part a)

Initial Definitions

Let

$$F : \Omega \subset \mathbb{R}^n \longrightarrow \mathbb{R}^m, \qquad F(x) = \begin{pmatrix} F_1(x) \\ F_2(x) \\ \vdots \\ F_m(x) \end{pmatrix}. \tag{1.1}$$

We define

$$s(x) := \|F(x)\|_2 = \sqrt{F_1(x)^2 + F_2(x)^2 + \cdots + F_m(x)^2} \geq 0. \tag{1.2}$$

The quantity $s(x)$ is a scalar (a non-negative number).

Introducing an Auxiliary Function $\phi(t)$

Let

$$\phi : [0, \infty) \longrightarrow \mathbb{R} \tag{1.3}$$

be a strictly increasing function, that is,

$$\forall\, a, b \geq 0, \quad a < b \Rightarrow \phi(a) < \phi(b). \tag{1.4}$$

A typical and convenient choice (for differentiability reasons) is

$$\phi(t) = \tfrac{1}{2}t^2. \tag{1.5}$$

Definition of a New Function $g(x)$

We compose ϕ with $s(x)$:

$$g(x) := \phi(s(x)) = \phi(\|F(x)\|_2) = \tfrac{1}{2}\|F(x)\|_2^2. \tag{1.6}$$

Since $\|F(x)\|_2^2 = F(x)^\top F(x)$, we can also write

$$g(x) = \tfrac{1}{2}F(x)^\top F(x). \tag{1.7}$$

Formal Proof of Equivalence Using Inequalities)

We want to show that

$$\arg\min_x s(x) = \arg\min_x g(x). \tag{1.8}$$

The minimizing points are the same, even though the minimum values are different.

Explicit Substitution: from t to $s(x)$

During the proof, t is only an auxiliary variable representing a non-negative real number. Then we replace t with $s(x) = \|F(x)\|_2$, which depends on x.
Thus, $\phi(t)$ becomes a function of x:

$$\phi(t) = \tfrac{1}{2}t^2 \quad \Rightarrow \quad \phi(s(x)) = \tfrac{1}{2}(s(x))^2 = \tfrac{1}{2}\|F(x)\|_2^2. \tag{1.9}$$

Therefore, the substitution is

$$t \mapsto s(x) = \|F(x)\|_2. \tag{1.10}$$

Alternative Form (Using $F(x)^\top F(x)$)

Since the squared norm can be written as an inner product,

$$\|F(x)\|_2^2 = F(x)^\top F(x), \tag{1.11}$$

the equivalence can also be expressed directly as

$$\boxed{\arg\min_{x\in\Omega} \|F(x)\|_2 = \arg\min_{x\in\Omega} \tfrac{1}{2}F(x)^\top F(x).} \tag{1.12}$$

1.1.2 Part b)

We are given

$$g(x) = \tfrac{1}{2}F(x)^\top F(x), \tag{1.13}$$

where

$$F(x) = \begin{pmatrix} F_1(x) \\ F_2(x) \\ \vdots \\ F_m(x) \end{pmatrix}. \tag{1.14}$$

Hence,

$$g(x) = \tfrac{1}{2}\sum_{i=1}^{m} F_i(x)^2. \tag{1.15}$$

Gradient of $g(x)$

To compute the gradient, we differentiate $g(x)$ componentwise with respect to each variable x_j:

$$\frac{\partial g}{\partial x_j} = \tfrac{1}{2}\sum_{i=1}^{m} \frac{\partial}{\partial x_j}\left[F_i(x)^2\right] = \sum_{i=1}^{m} F_i(x)\,\frac{\partial F_i(x)}{\partial x_j}. \tag{1.16}$$

Since $g(x)$ is a sum of m squared terms, the derivative of the sum is the sum of the derivatives.
Let $F'(x)$ denote the Jacobian of F:

$$F'(x) = \begin{pmatrix} \frac{\partial F_1(x)}{\partial x_1} & \cdots & \frac{\partial F_1(x)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_m(x)}{\partial x_1} & \cdots & \frac{\partial F_m(x)}{\partial x_n} \end{pmatrix} \in \mathbb{R}^{m\times n}. \tag{1.17}$$

Then, the gradient can be written compactly as

$$g'(x) = F'(x)^\top F(x). \tag{1.18}$$

Expanding this expression componentwise, each entry of $g'(x)$ is

$$[g'(x)]_j = \sum_{i=1}^{m} F_i(x)\,\frac{\partial F_i(x)}{\partial x_j}, \qquad j = 1, \ldots, n. \tag{1.19}$$

Hence, the gradient vector can be expressed as the following matrix:

$$g'(x) = \begin{pmatrix} \dfrac{\partial g(x)}{\partial x_1} & \cdots & \dfrac{\partial g(x)}{\partial x_n} \end{pmatrix} = \begin{pmatrix} \displaystyle\sum_{i=1}^{m} F_i(x)\,\frac{\partial F_i(x)}{\partial x_1} & \cdots & \displaystyle\sum_{i=1}^{m} F_i(x)\,\frac{\partial F_i(x)}{\partial x_n} \end{pmatrix} \in \mathbb{R}^{1\times n}. \tag{1.20}$$

Hessian of $g(x)$

Differentiating once more, we obtain the Hessian matrix $g''(x)$. Starting from $g'(x) = F'(x)^\top F(x)$, we apply the product rule:

$$g''(x) = [F'(x)^\top F(x)]' = F'(x)^\top F'(x) + \sum_{i=1}^{m} F_i(x)\,F_i''(x), \tag{1.21}$$

where $F_i''(x)$ is the Hessian matrix of the scalar function $F_i(x)$:

$$F_i''(x) = \begin{pmatrix} \frac{\partial^2 F_i(x)}{\partial x_1^2} & \cdots & \frac{\partial^2 F_i(x)}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 F_i(x)}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 F_i(x)}{\partial x_n^2} \end{pmatrix} \in \mathbb{R}^{n\times n}. \tag{1.22}$$

In componentwise form, the (j,k)-th entry of $g''(x)$ is given by

$$\frac{\partial^2 g(x)}{\partial x_j \partial x_k} = \sum_{i=1}^{m} \frac{\partial F_i(x)}{\partial x_j}\frac{\partial F_i(x)}{\partial x_k} + \sum_{i=1}^{m} F_i(x)\,\frac{\partial^2 F_i(x)}{\partial x_j \partial x_k}. \tag{1.23}$$

Hence, the full Hessian matrix can be represented as

$$\begin{aligned} g''(x) &= \begin{pmatrix} \dfrac{\partial^2 g(x)}{\partial x_1^2} & \cdots & \dfrac{\partial^2 g(x)}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial^2 g(x)}{\partial x_n \partial x_1} & \cdots & \dfrac{\partial^2 g(x)}{\partial x_n^2} \end{pmatrix} \\ &= \begin{pmatrix} \displaystyle\sum_{i=1}^{m} \frac{\partial F_i(x)}{\partial x_1}\frac{\partial F_i(x)}{\partial x_1} + F_i(x)\frac{\partial^2 F_i(x)}{\partial x_1^2} & \cdots & \displaystyle\sum_{i=1}^{m} \frac{\partial F_i(x)}{\partial x_1}\frac{\partial F_i(x)}{\partial x_n} + F_i(x)\frac{\partial^2 F_i(x)}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \displaystyle\sum_{i=1}^{m} \frac{\partial F_i(x)}{\partial x_n}\frac{\partial F_i(x)}{\partial x_1} + F_i(x)\frac{\partial^2 F_i(x)}{\partial x_n \partial x_1} & \cdots & \displaystyle\sum_{i=1}^{m} \frac{\partial F_i(x)}{\partial x_n}\frac{\partial F_i(x)}{\partial x_n} + F_i(x)\frac{\partial^2 F_i(x)}{\partial x_n^2} \end{pmatrix}. \end{aligned} \tag{1.24}$$

Final Result

Therefore, the gradient and Hessian of $g(x)$ can be summarized compactly as

$$\boxed{\begin{aligned} g'(x) &= F'(x)^\top F(x), \\ g''(x) &= F'(x)^\top F'(x) + \sum_{i=1}^{m} F_i(x)\,F_i''(x). \end{aligned}} \tag{1.25}$$

1.2 Exercise 2

1.2.1 (a) Gauss–Newton step in MATLAB is $s_k = -(J_k^\top J_k)^{-1} J_k^\top r_k$

Let $g(z) := \frac{1}{2}\|F(z)\|_2^2$ with residual vector $r(z) := F(z) \in \mathbb{R}^m$ and Jacobian $J(z) := F'(z) \in \mathbb{R}^{m\times n}$. The gradient and the exact Hessian of g are

$$\nabla g(z) = J(z)^\top r(z), \qquad \nabla^2 g(z) = J(z)^\top J(z) + \sum_{i=1}^{m} r_i(z)\,\nabla^2 F_i(z). \tag{1.26}$$

Gauss–Newton neglects the second term (which requires second derivatives) and uses the approximation

$$\nabla^2 g(z) \;\approx\; J(z)^\top J(z). \tag{1.27}$$

A (quasi-)Newton step then solves

$$J(z_k)^\top J(z_k)\, s_k \;=\; -\,J(z_k)^\top r(z_k), \tag{1.28}$$

which, when $J(z_k)^\top J(z_k)$ is invertible, yields the update

$$s_k \;=\; -\,(J(z_k)^\top J(z_k))^{-1} J(z_k)^\top r(z_k), \qquad z_{k+1} = z_k + s_k. \tag{1.29}$$

Equivalently, s_k is the least–squares solution of $\min_s \| r(z_k) + J(z_k)s \|_2$ obtained by the normal equations.

1.2.2 (b) Model F for circle fitting and its derivative

Given data points $(x_i, y_i) \in \mathbb{R}^2$, $i = 1, \dots, m$, and parameters $z = [x_c, y_c, r]^\top$, define the residuals

$$F_i(z) \;=\; (x_i - x_c)^2 + (y_i - y_c)^2 - r^2, \qquad i = 1, \dots, m, \tag{1.30}$$

and $F(z) = [F_1(z), \dots, F_m(z)]^\top$. The Jacobian $J(z) = F'(z) \in \mathbb{R}^{m\times 3}$ has rows

$$\frac{\partial F_i}{\partial x_c} = -2(x_i - x_c), \qquad \frac{\partial F_i}{\partial y_c} = -2(y_i - y_c), \qquad \frac{\partial F_i}{\partial r} = -2r. \tag{1.31}$$

In matrix form,

$$J(z) \;=\; \begin{bmatrix} -2(x_1 - x_c) & -2(y_1 - y_c) & -2r \\ \vdots & \vdots & \vdots \\ -2(x_m - x_c) & -2(y_m - y_c) & -2r \end{bmatrix}. \tag{1.32}$$

Remark. The "derivative" supplied to Gauss–Newton is the *Jacobian* $J = F'$, i.e., all partial derivatives of the vector map F. The gradient would refer to the scalar objective $g(z) = \frac{1}{2}\|F(z)\|^2$ and equals $\nabla g(z) = J(z)^\top F(z)$.

1.2.3 (c) Numerical results (plots)

Figure 1.1 shows the data points and the fitted circle obtained by Gauss–Newton. Figure 1.2 displays $\|F(x_c, y_c, \tilde{r})\|_2$ versus a swept radius $\tilde{r}$, illustrating the symmetric minima at $\tilde{r} = \pm r$.

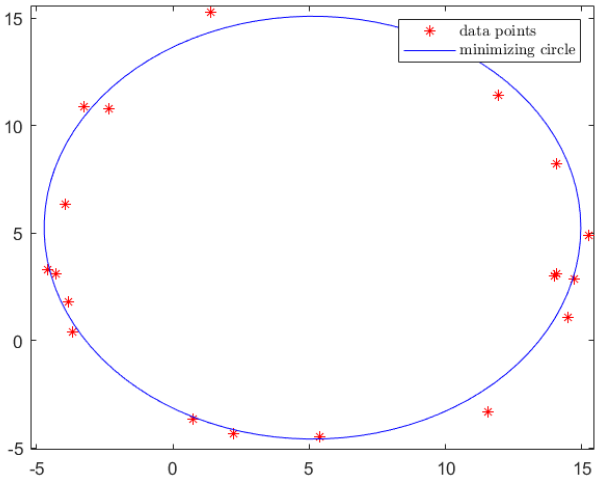

Figure 1.1: Data points and minimizing circle returned by Gauss–Newton.

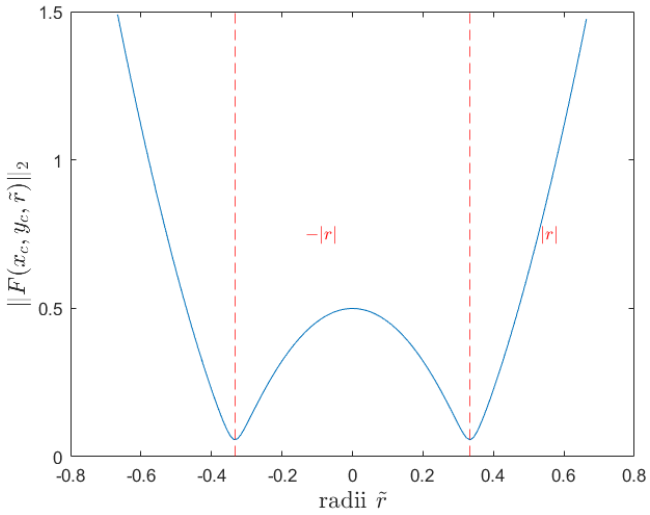

Figure 1.2: Behavior of $\|F(x_c, y_c, \tilde{r})\|_2$ as a function of the trial radius $\tilde{r}$.

1.2.4 (d) Observation with $z_0 = [1, 1, 1]^\top$

Because the residuals use r^2, the objective is invariant under $r \mapsto -r$. Starting from $z_0 = [1, 1, 1]^\top$, the iteration may converge to a solution with $r < 0$ even though the geometry is unchanged. In our implementation we therefore post-process the estimate by

$$\hat{r} \;:=\; |\, r \,|, \tag{1.33}$$

which enforces a nonnegative radius without affecting the fit.

```
clear
clc
close all


% number of data points
m = 20;

% create random data points roughly arranged in circle
c_r = rand(2,1)*10;
r_r = rand(1,1)*10;
ang = rand(m,1)*2*pi;
dev = rand(m,1)*0.2+0.9;


fprintf("Numer of points: %d \n",m);
fprintf('Center [x y]: [');
for i = 1:length(c_r)
    fprintf('%.2f ', c_r(i));
end
fprintf("]\n");

fprintf('Radius: %2f \n',r_r);

fprintf('Dev(i)*Radius: \n');
for i = 1:length(dev)
    fprintf('%.2f ', dev(i)*r_r);
end
fprintf("\n");

fprintf('Angle of position(i)\n');
for i = 1:length(ang)
    fprintf('%.2f ', ang(i));
end
fprintf("\n");

x = c_r(1)+r_r*cos(ang).*dev;
y = c_r(2)+r_r*sin(ang).*dev;

% set up function handles F and dF; z(1) = x_c, z(2) = y_c, z(3) = r
F  = @(z) (x - z(1)).^2 + (y - z(2)).^2 - z(3).^2;

% Jacobian (mŒ3): each row = [-2(x_i - x_c), -2(y_i - y_c), -2r]
dF = @(z) [-2*(x - z(1)), -2*(y - z(2)), -2*z(3)*ones(m,1)];


% set inputs for Gauss-Newton method
x0 = [5;5;5];
tol = 10^(-10);
maxit = 100;

test=3;
i=0;
while i<test
    fprintf("solution:  %d \n",i);
    x0 = [1;1;1];

    % perform Gauss-Newton method
    sol = gauss_newton(x0, F, dF, tol, maxit);
    sol = sol(:,end);
    x_c = sol(1);
    y_c = sol(2);
    %radius can be negative or positive based on the starting point
```

```
    r = abs(sol(3));

    fprintf('Calculated outputs:\n');
    fprintf('x_c = %.4f  (size: %s)\n', x_c, mat2str(size(x_c)));
    fprintf('y_c = %.4f  (size: %s)\n', y_c, mat2str(size(y_c)));
    fprintf('r   = %.4f  (size: %s)\n', r, mat2str(size(r)));



    % plot data points and computed circle
    figure();
    x_circ = x_c + r * cos(linspace(0,2*pi,100));
    y_circ = y_c + r * sin(linspace(0,2*pi,100));
    plot(x,y,'*red');
    hold on;
    plot(x_circ,y_circ,'blue');
    xlim([x_c-r-0.5, x_c+r+0.5]);
    ylim([y_c-r-0.5, y_c+r+0.5]);
    legend('data points', 'minimizing circle', 'Interpreter', 'latex');

    % plot ||F(x_c,y_c,t)||_2 with varying radius t
    figure();
    R = -2*abs(r):0.01:2*abs(r);
    F_r = zeros(length(R),1);
    k = 1;
    for t = R
        F_r(k) = norm(F([x_c, y_c, t]));
        k = k+1;
    end
    plot(R, F_r);
    ylabel('$$\|F(x_c,y_c,\tilde{r})\|_2$$', 'Interpreter', 'latex', 'FontSize', 14)
    ;
    xlabel('radii $$\tilde{r}$$', 'Interpreter', 'latex', 'FontSize', 14);
    xline(r, '--r');
    xline(-r, '--r');
    text(-abs(r)+0.2, max(F_r)/2, '$$-\vert r\vert$$', 'Interpreter', 'latex', '
    Color', 'red');
    text(abs(r)+0.2, max(F_r)/2, '$$\vert r\vert$$', 'Interpreter', 'latex', 'Color'
    , 'red');

    pause(4);
    i=i+1;

end



function X = gauss_newton(x0, F, dF, tol, maxit)
% GAUSS_NEWTON  Nonlinear least-squares solution
% Inputs:
%   x0    initial guess for parameter vector
%   F     function handle returning residual vector
%   dF    function handle returning Jacobian matrix
%   tol   tolerance for stopping criterion
%   maxit maximum number of iterations
%
% Output:
%   X     matrix whose columns are the iterates

    z = x0(:);
    X = z;

    fprintf('\n--- Starting GaussNewton iterations ---\n');
```

```
124     fprintf('Initial guess: z0 = [%8.4f, %8.4f, %8.4f]\n', z(1), z(2), z(3));
125     fprintf('Tolerance (tol): %.2e | Max iterations: %d\n\n', tol, maxit);
126 
127     for k = 1:maxit
128         r = F(z);      % residual vector (mŒ1)
129         J = dF(z);     % Jacobian matrix (mŒn)
130 
131         % Solve normal equations (least-squares step)
132         %s = -J \ r;
133         s = - (J' * J) \ (J' * r);
134 
135         norm_r = norm(r);
136         norm_s = norm(s);
137 
138         fprintf('Iter %2d:  z = [%8.4f, %8.4f, %8.4f]  |s| = %.3e  |r| = %.3e\n',
    ...
139                 k, z(1), z(2), z(3), norm_s, norm_r);
140 
141 
142         % Update iterate
143         z = z + s;
144         X(:,end+1) = z;
145 
146         % Check convergence
147         if norm(s) <= tol * (1 + norm(z))
148             fprintf('Converged at iteration %d\n', k);
149             fprintf('Final estimate: z = [%8.4f, %8.4f, %8.4f]\n', z(1), z(2), z(3))
    ;
150             fprintf('-----------------\n\n');
151             break;
152         end
153 
154         if k == maxit
155             fprintf('Max iterations (%d) reached without convergence.\n', maxit);
156             fprintf('Last estimate: z = [%8.4f, %8.4f, %8.4f]\n', z(1), z(2), z(3));
157             fprintf('-----------------\n\n');
158         end
159     end
160 end
```

1.3 Exercise 3: The GaussNewton Method and Taylors Expansion

Consider the nonlinear least squares (NoLS) problem

$$g(x) = \tfrac{1}{2}\|F(x)\|_2^2 = \tfrac{1}{2}\sum_{i=1}^{m} F_i(x)^2, \qquad F : \Omega \subset \mathbb{R}^n \to \mathbb{R}^m. \tag{1.34}$$

Let the Jacobian of F be denoted by $J(x) = F'(x) \in \mathbb{R}^{m\times n}$.
The gradient and Hessian of $g(x)$ are given by

$$\nabla g(x) = J(x)^T F(x), \qquad \nabla^2 g(x) = J(x)^T J(x) + \sum_{i=1}^{m} F_i(x)\,\nabla^2 F_i(x). \tag{1}$$

1.3.1 (a) Derivation of the Gauss Newton method from Taylor expansion

Starting from the firstorder optimality condition $\nabla g(x_{k+1}) = 0$, we perform a Taylor expansion of ∇g around x_k:

$$\nabla g(x_k + s_k) \approx \nabla g(x_k) + \nabla^2 g(x_k)\, s_k. \tag{1.35}$$

Imposing $\nabla g(x_{k+1}) \approx 0$ yields the Newton step

$$\nabla^2 g(x_k)\, s_k = -\nabla g(x_k). \tag{2}$$

Substituting equation (1) into (2), we obtain

$$\Big[J(x_k)^T J(x_k) + \underbrace{\sum_{i=1}^{m} F_i(x_k)\,\nabla^2 F_i(x_k)}_{\text{second-order terms}}\Big] s_k = -\,J(x_k)^T F(x_k). \tag{1.36}$$

When the residuals $F_i(x_k)$ are small or the problem is nearly linear, the secondorder term can be neglected. This simplification gives the GaussNewton system

$$\boxed{J(x_k)^T J(x_k)\, s_k = -\,J(x_k)^T F(x_k)}. \tag{GN}$$

The iteration formula is then

$$x_{k+1} = x_k + s_k. \tag{1.37}$$

Hence, the GaussNewton method can be seen as Newtons method applied to $g(x) = \frac{1}{2}\|F(x)\|^2$, neglecting the terms involving second derivatives of F.

1.3.2 (b) Normal equations and linear least squares formulation

Consider the linear least squares problem in the variable s:

$$\min_{s\in\mathbb{R}^n} \ \|J(x_k)\, s + F(x_k)\|_2^2. \tag{LS}$$

The normal equations associated with LS are

$$J(x_k)^T\big(J(x_k)\, s + F(x_k)\big) = 0 \quad \Longleftrightarrow \quad J(x_k)^T J(x_k)\, s = -\,J(x_k)^T F(x_k), \tag{1.38}$$

which coincide exactly with the GaussNewton equation (GN). Thus, problems (6) and (7) in the assignment are indeed equivalent.

What does the GaussNewton method do? At each iteration k, the GaussNewton method *linearizes* the nonlinear residual function $F(x)$ around the current point x_k:

$$F(x_k + s) \approx F(x_k) + J(x_k)\, s. \tag{1.39}$$

Then, it solves the linear least squares problem for the step s_k:

$$s_k = \arg\min_s \|J(x_k)\, s + F(x_k)\|_2, \tag{1.40}$$

whose normal equations are $J^T J s = -J^T F$. Finally, the update is performed as $x_{k+1} = x_k + s_k$.
In summary, the GaussNewton method replaces the nonlinear least squares problem with a linear least squares problem on the firstorder approximation of F.

1.4 Exercise 04

1.4.1 b) Corresponding function F and its derivative F'

To apply the GauSS–Newton method, we express the Rosenbrock function in the form

$$r(x,y) = \|F(x,y)\|_2, \qquad F(x,y) = \begin{pmatrix} 1-x \\ 10(y-x^2) \\ \sqrt{x^2+y^2} \end{pmatrix}. \tag{1.41}$$

Hence, in vector notation with $x = (x_1, x_2)^\top$, we can define

$$F(x) = \begin{pmatrix} 1-x_1 \\ 10(x_2-x_1^2) \\ \sqrt{x_1^2+x_2^2} \end{pmatrix}. \tag{1.42}$$

The Jacobian matrix $F'(x)$ of $F(x)$ is defined as

$$F'(x) = \begin{pmatrix} \dfrac{\partial F_1}{\partial x_1} & \dfrac{\partial F_1}{\partial x_2} \\ \dfrac{\partial F_2}{\partial x_1} & \dfrac{\partial F_2}{\partial x_2} \\ \dfrac{\partial F_3}{\partial x_1} & \dfrac{\partial F_3}{\partial x_2} \end{pmatrix}, \tag{1.43}$$

where each entry represents the partial derivative of F_i with respect to x_j.
Computing these derivatives explicitly yields

$$\begin{aligned} \frac{\partial F_1}{\partial x_1} &= -1, & \frac{\partial F_1}{\partial x_2} &= 0, \\ \frac{\partial F_2}{\partial x_1} &= -20x_1, & \frac{\partial F_2}{\partial x_2} &= 10, \\ \frac{\partial F_3}{\partial x_1} &= \frac{x_1}{\sqrt{x_1^2+x_2^2}}, & \frac{\partial F_3}{\partial x_2} &= \frac{x_2}{\sqrt{x_1^2+x_2^2}}. \end{aligned}$$

Hence, the Jacobian matrix becomes

$$F'(x) = \begin{pmatrix} -1 & 0 \\ -20x_1 & 10 \\ \dfrac{x_1}{\sqrt{x_1^2+x_2^2}} & \dfrac{x_2}{\sqrt{x_1^2+x_2^2}} \end{pmatrix}. \tag{1.44}$$

1.4.2 c) Gauss Step

The Gauss–Newton method is designed to solve nonlinear least-squares problems of the form

$$\min_{x\in\mathbb{R}^n} \ \tfrac{1}{2}\|F(x)\|_2^2, \tag{1.45}$$

where $F : \mathbb{R}^n \to \mathbb{R}^m$ and $m \geq n$.
At each iteration k, we linearize $F(x)$ around the current point x_k by its first-order Taylor expansion:

$$F(x_k + s) \approx F(x_k) + F'(x_k)s = r_k + J_k s. \tag{1.46}$$

Then, we minimize the local quadratic model

$$\min_s \ \tfrac{1}{2}\|r_k + J_k s\|_2^2.$$

The corresponding normal equations are obtained by setting the gradient to zero:

$$J_k^\top J_k s = -J_k^\top r_k. \tag{1.47}$$

Solving this linear system gives the Gauss–Newton step

$$s = -(J_k^\top J_k)^{-1} J_k^\top r_k. \tag{1.48}$$

Exact vs. approximate computation in MATLAB. In MATLAB, one can compute the same step equivalently by solving the least-squares problem

$$J_k s \approx -r_k, \tag{1.49}$$

using the backslash operator:

$$s = -J_k \backslash r_k.$$

This command internally computes the least-squares solution via a QR decomposition of J_k, which is numerically more stable than explicitly forming $(J_k^\top J_k)^{-1} J_k^\top$.
Both formulations yield the same theoretical step, since

$$(J_k^\top J_k)^{-1} J_k^\top = J_k^+,$$

where J_k^+ is the MoorePenrose pseudoinverse of J_k. Hence, in exact arithmetic they produce identical results. However, the version using the backslash operator is typically preferred for better numerical stability, while the explicit formula $(J^\top J)^{-1} J^\top r$ is useful for didactic or symbolic derivations.

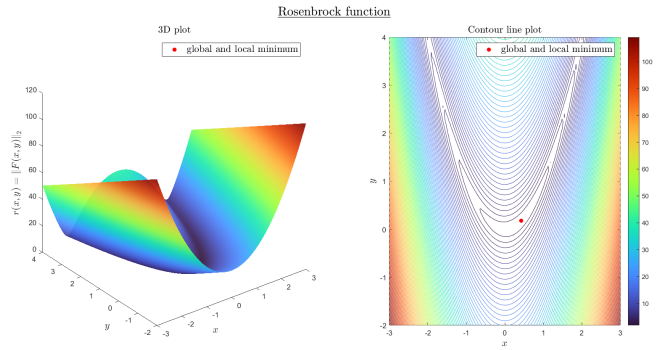

Figure 1.3: Visualization of the Rosenbrock residual function $r(x, y) = \|F(x, y)\|_2$. The global minimum lies in a narrow curved valley, which makes optimization challenging for gradient-based methods.

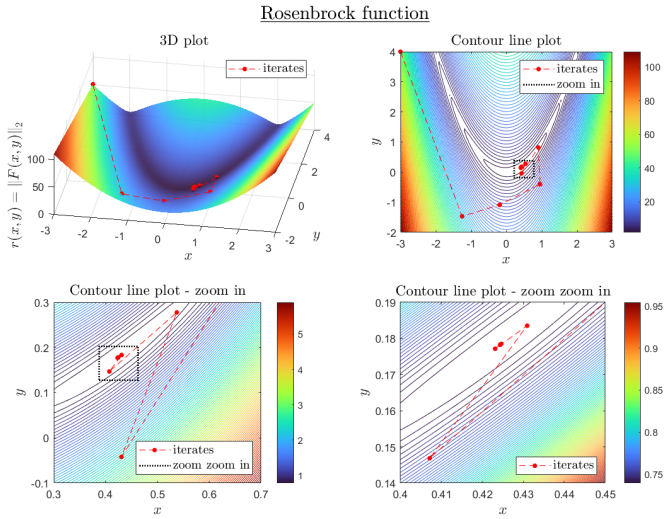

Figure 1.4: Trajectory of the Gauss–Newton iterates on the Rosenbrock function. The iterates follow the curved valley toward the minimizer $(x^*, y^*) \approx (0.4243,\ 0.1783)$. The contour plots and 3D surface confirm convergence within the narrow region.

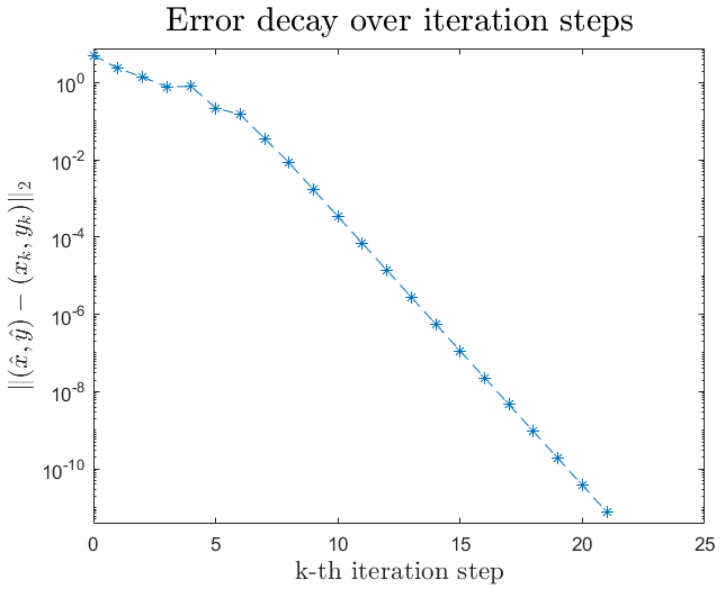

Figure 1.5: Error decay of the Gauss–Newton method on the Rosenbrock problem. The semilogarithmic plot shows almost quadratic convergence once the iterates enter the neighborhood of the minimizer.

```
1
2  function X = gauss_newton(x0, F, dF, tol, maxit)
3  % GAUSS_NEWTON  Nonlinear least-squares via GaussNewton.
4
5  z = x0(:);
6  X = z;
7
8      for k = 1:maxit
9          r = F(z);           % m€1
10         J = dF(z);          % m€n
11
12         % step GaussNewton: J*s  -r  (LS); avoid (J'*J)
13         %s = - J \ r;
14         s = - (J' * J) \ (J' * r);
15         znew = z + s;
16         X(:,end+1) = znew;
17
18         if norm(s) <= tol * (1 + norm(znew))
19             break;
20         end
21         z = znew;
22     end
23 end
```

Gauss Newton

```
1
2  close all
3  clc
4  clear
5
6
7  % F and dF
8  F = @(x) [ 1-x(1); 10*(x(2)-x(1).^2); sqrt(x(1).^2+x(2).^2) ];
9  dF = @(x) [-1, 0; -20*x(1), 10; x(1)/sqrt(x(1).^2+x(2).^2), x(2)/sqrt(x(1).^2+x(2)
       .^2)];
10
11 % apply Gauss-Newton
12 X = gauss_newton([-3;4], F, dF, 10^(-10), 100);
```

```

% exact solution
exact = zeros(2,1);
exact(1) = 0.1 * ( (101/2*(45+sqrt(3237)))^(1/3)/3^(2/3) - 101^(2/3)*(2/3/(45+sqrt
    (3237)))^(1/3) );
exact(2) = 100/101*exact(1)^2;

% compute errors in euclidean 2-Norm
numIt = size(X,2);
err = zeros(numIt,1);
for i = 1:numIt
    err(i) = norm(X(:,i)-exact);
end

% plot errors
semilogy(0:numIt-1, err, '--*');
xlabel('k-th iteration step', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$$\|(\hat{x},\hat{y}) - (x_k,y_k)\|_2$$', 'Interpreter', 'latex', 'FontSize'
    , 14);
ylim([err(end)*0.5, err(1)*1.5]);
title('Error decay over iteration steps', 'Interpreter', 'latex', 'FontSize', 18);
```

Error rosenbrock

```

clear
close all
clc

% F y J=F' para r(x,y)=||F||_2 con F=[1-x; 10(y-x^2); sqrt(x^2+y^2)]
F  = @(x) [ 1 - x(1);
            10*(x(2) - x(1)^2);
            sqrt(x(1)^2 + x(2)^2) ];
dF = @(x) [ -1,                  0;
            -20*x(1),           10;
             x(1)/sqrt(x(1)^2 + x(2)^2),  x(2)/sqrt(x(1)^2 + x(2)^2) ];


% apply Gauss-Newton method
X = gauss_newton([-3;4], F, dF, 10^(-6), 100);

% compute values of Rosenbrock function
[x,y] = meshgrid(linspace(-3,3,100), linspace(-2,4,100));
z =  sqrt( (1-x).^2 + (10*(y-x.^2)).^2 + x.^2+y.^2 );

% set figure
figure('Position', [100,100,1000,700]);
sgtitle('\underline{Rosenbrock function}', 'Interpreter', 'latex', 'FontSize', 18);

% 3D plot
subplot(2,2,1);
h = surf(x,y,z);
h.FaceAlpha = 0.9;
shading interp;
colormap(turbo(256));
hold on;
p3 = plot3(X(1,:),X(2,:), sqrt( (1-X(1,:)').^2 + (10*(X(2,:)'-X(1,:)'.^2)).^2 + X
    (1,:)'.^2+X(2,:)'.^2 ), '--.r', 'MarkerSize',15);
xlabel('$$x$$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$$y$$', 'Interpreter', 'latex', 'FontSize', 14);
zlabel('$$r(x,y)=\left\|F(x,y)\right\|_2$$', 'Interpreter', 'latex', 'FontSize', 14)
    ;
legend(p3,'iterates', 'Interpreter', 'latex', 'Location', 'northeast', 'FontSize',
    12);
```

```
title('3D plot', 'Interpreter', 'latex', 'FontSize', 14);
view(10,60);

% contour lines plot
subplot(2,2,2);
contour(x,y,z,100,'HandleVisibility','off');
hold on;
plot(X(1,:),X(2,:),'--.r','MarkerSize',15);
xlabel('$$x$$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$$y$$', 'Interpreter', 'latex', 'FontSize', 14);
%caxis([0,110]);
colorbar();
rectangle('Position', [0.3-0.4/5,-0.1-0.4/5,0.4+2*0.4/5,0.4+2*0.4/5], 'EdgeColor','
    black', 'LineStyle',':','LineWidth',1.5);
plot(nan, nan, 'Color','black', 'LineStyle',':','LineWidth',1.5, 'DisplayName', '
    zoom in');
legend('iterates', 'zoom in', 'Interpreter', 'latex', 'Location', 'northeast', '
    FontSize', 12);
title('Contour line plot', 'Interpreter', 'latex', 'FontSize', 14);

% contour lines plot - zoom in
subplot(2,2,3);
[x,y] = meshgrid(linspace(0.3,0.7,100), linspace(-0.1,0.3,100));
z =  sqrt( (1-x).^2 + (10*(y-x.^2)).^2 + x.^2+y.^2 );
contour(x,y,z,100,'HandleVisibility','off');
hold on;
plot(X(1,:),X(2,:), '--.r', 'MarkerSize',15);
rectangle('Position', [0.4-0.0125,0.14-0.0125,0.05+2*0.0125,0.05+2*0.0125], '
    EdgeColor','black', 'LineStyle',':','LineWidth',1.5);
plot(nan, nan, 'Color','black', 'LineStyle',':','LineWidth',1.5);
xlabel('$$x$$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$$y$$', 'Interpreter', 'latex', 'FontSize', 14);
xlim([0.3,0.7]);
ylim([-0.1,0.3]);
%caxis([0,110]);
colorbar();
legend('iterates', 'zoom zoom in', 'Interpreter', 'latex', 'Location', 'southeast',
    'FontSize', 12);
title('Contour line plot - zoom in', 'Interpreter', 'latex', 'FontSize', 14);

% contour lines plot - zoom zoom in
subplot(2,2,4);
[x,y] = meshgrid(linspace(0.4,0.45,100), linspace(0.14,0.19,100));
z =  sqrt( (1-x).^2 + (10*(y-x.^2)).^2 + x.^2+y.^2 );
contour(x,y,z,100);
hold on;
p4=plot(X(1,:),X(2,:), '--.r', 'MarkerSize',15);
xlabel('$$x$$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$$y$$', 'Interpreter', 'latex', 'FontSize', 14);
xlim([0.4,0.45]);
ylim([0.14,0.19]);
%caxis([0,110]);
colorbar();
legend(p4,'iterates', 'Interpreter', 'latex', 'Location', 'southeast', 'FontSize',
    12);
title('Contour line plot - zoom zoom in', 'Interpreter', 'latex', 'FontSize', 14);
```

test rosenbrock

Chapter 2

Supporting Manual Notes

Exercises: 01

Part b): $g(x) = \frac{1}{2} \sum_{i=1}^{m} [F_i(x)]^2$

$$\frac{\partial g}{\partial x_j} = \frac{1}{2} \sum_{i=1}^{m} \frac{\partial (F_i(x))^2}{\partial x_j}$$

$$\frac{\partial}{\partial x_j} \sum f_i(x) = \sum \frac{\partial f_i(x)}{\partial x_j}$$

If $F(x) = \begin{bmatrix} F_1(x) \\ F_2(x) \\ \vdots \\ F_m(x) \end{bmatrix}$

$$g(x) = \frac{1}{2} (F(x))^T \cdot F(x))$$

$$g(x) = \frac{1}{2} \sum_{i=1}^{m} (F(x))^2$$

$$\frac{\partial g}{\partial x_j} = \sum_{i=1}^{m} \frac{2}{2} F(x) \frac{\partial F(x)}{\partial x_j}$$

$$\nabla g(x) = \begin{bmatrix} \frac{\partial g}{\partial x_1} \\ \frac{\partial g}{\partial x_2} \\ \vdots \\ \frac{\partial g}{x_n} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{m} F_i(x) \frac{\partial F_i(x)}{\partial x_1} \\ \sum_{i=1}^{m} F_i(x) \frac{\partial F_i(x)}{\partial x_2} \\ \vdots \\ \sum_{i=1}^{m} F(x) \frac{\partial F(x)}{\partial x_n} \end{bmatrix}$$

Knowing

$$Dg(x)[h] = \nabla g(x)^T \cdot h$$

$$DF(x)[h] = J(x) \cdot h$$

$$g(x+h) \approx g(x) + Dg(x)[h]$$

$$Dg(x)[h] = \lim_{\epsilon \to 0} \frac{g(x+\epsilon h) - g(x)}{\epsilon}$$

$$Dg(x)[h] = (DF(x)[h])^T F(x) = h^T \cdot J(x)^T F(x)$$

$$\nabla g(x)^T h = h^T J(x)^T F(x) \Rightarrow \nabla g(x) = J(x)^T \cdot F(x)$$

movement
h: direction
F(x): vector $\mathbb{R}^m$
g(x): scalar

$F(x) = [F_1 \dots F_n]^T$

Example: 01.

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad F(x) = \begin{bmatrix} F_1(x) \\ F_2(x) \end{bmatrix} = \begin{bmatrix} x_1 + 2x_2 \\ x_1 \cdot x_2 \end{bmatrix}$$

$$F'(x) = \underbrace{\begin{pmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial x_1} & \cdots & \frac{\partial F_m}{\partial x_n} \end{pmatrix}}_{J(x)}$$

$$g(x) = \frac{1}{2}\left[\underbrace{(x_1 + 2x_2)}_{u}^2 + (x_1 \cdot x_2)^2\right]$$

$$\frac{\partial g}{\partial x_1} = \frac{2}{2}(x_1 + 2x_2)(1) + \underbrace{\frac{2}{2}(x_1 x_2)(x_2)}_{\leq F_i(x) \cdot \frac{\partial F_i(x)}{\partial x_1}}$$

$$\frac{\partial g}{x_2} = \frac{2}{2}(x_1 + 2x_2)(2) + \underbrace{\frac{2}{2}(x_1 \cdot x_2)(x_1)}_{\leq F_j(x)\,\frac{\partial F_i(x)}{\partial x_i}}$$

$$\nabla g = \begin{bmatrix} x_1 + 2x_2 + x_1 x_2^2 \\ 2x_1 + 4x_2 + x_1^2 x_2 \end{bmatrix}$$

Let $F'(x) = J(x)$

$$J(x) = \begin{bmatrix} \frac{\partial F_1}{x_1} & \frac{\partial F_1}{x_2} \\ \frac{\partial F_2}{x_1} & \frac{\partial F_2}{x_2} \end{bmatrix}$$

$$J(x) = \begin{bmatrix} 1 & 2 \\ x_2 & x_1 \end{bmatrix}$$

$$\nabla g = J(x)^T F(x)$$

$$\nabla g = \begin{bmatrix} 1 & x_2 \\ 2 & x_1 \end{bmatrix} \begin{bmatrix} x_1 + 2x_2 \\ x_1 \cdot x_2 \end{bmatrix}$$

$$\nabla g = \begin{bmatrix} x_1 + 2x_2 + x_2^2 \cdot x_1 \\ 2x_1 + 4x_2 + x_1^2 \cdot x_2 \end{bmatrix}$$

Hessian:

$$\nabla g(x) = J(x)^T \cdot F(x)$$

$$\frac{\partial g}{\partial x_j} = \sum_{i=}^{m} J_{ij}(x) \cdot F_i(x)$$

$$F(x) = \begin{bmatrix} F_1(x) \\ F_2(x) \end{bmatrix} \cdot J(x) = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_2}{\partial x_2} \\ \frac{\partial F_1}{\partial x_1} & \frac{\partial F_2}{\partial x_2} \end{bmatrix}$$

$$J(x)^T \cdot F(x) = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_1} \\ \frac{\partial F_2}{\partial x_2} & \frac{\partial F_2}{\partial x_2} \end{bmatrix} \begin{bmatrix} F_1(x) \\ F_2(x) \end{bmatrix}$$

(2x2) (2x1)

$$J(x)^T \cdot F(x) = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} F_1(x) + \frac{\partial F_1}{\partial x_1} F_2(x) \\ \frac{\partial F_2}{\partial x_2} F_1(x) + \frac{\partial F_2}{\partial x_2} F_2(x) \end{bmatrix}$$

$$J(x)^T F(x) = \begin{bmatrix} \sum_i F_i \frac{\partial F_i}{\partial x_1} \\ \sum_i F_i \frac{\partial F_i}{\partial x_2} \end{bmatrix}$$

to determine: $(\nabla^2 g)_{kj} = \frac{\partial^2 g}{\partial x_k x_j}$

$$\frac{\partial^2 g}{\partial x_k x_j} = \sum_{i=1}^{m} \frac{\partial}{\partial x_k} \left[F_i(x) \frac{\partial F_i(x)}{\partial x_j} \right]$$

$$= \underbrace{\frac{\partial F_i}{\partial x_k} \frac{\partial F_i}{\partial x_j}}_{J_{ik} J_{ij}} + \underbrace{F_i(x) \frac{\partial^2 F_i}{\partial x_k x_j}}_{(k,j)\text{ Hessian}}$$

Notes
linearity
$\frac{\partial}{\partial x_k}\left(\sum_i a_i\right) = \sum_i \frac{\partial a_i}{\partial x_k}$
Product
$\frac{\partial}{\partial x_k}[u \cdot v] = \frac{\partial u}{\partial x_k} v + u \frac{\partial v}{\partial x_k}$
Diff dot product
$d(a^T \cdot b) = (da)^T b + a^T (db)$
$dF(x) = J(x)\, dx$
Directional derivativ
$Dg(x)[h] = \nabla g(x)^T h$
Hessian
$(\nabla^2 g(x))_{kj} = \frac{\partial^2 g}{\partial x_k \partial x_j}$
simetry
$\frac{\partial^2 g}{\partial x_k \partial x_j} = \frac{\partial^2 g}{\partial x_j \partial k}$

$$(\nabla^2 g)_{kj} = \sum_{i=1}^{m} (J_{ik} J_{ij} + F_i (\nabla^2 F_i)_{kj}).$$

$$(J^T \cdot J)_{kj} = \sum_{i=1}^{m} J_{ik} \cdot J_{ij}$$

$$S(x) = \sum_{i=1}^{m} F_i(x) \nabla^2 F_i(x)$$

$$S_{kj} = \Sigma_i F_i (\nabla^2 F_i)_{kj}$$

$$\nabla^2 g(x) = J(x)^T J(x) + \sum_{i=1}^{m} F_i(x) \nabla^2 F_i(x)$$

Example 02: $\nabla g(x) = \begin{bmatrix} x_1 + x_2 + x_1 \cdot x_2^2 \\ x_1 + x_2 + x_1^2 x_2 \end{bmatrix}$

i) standard: $\frac{\partial^2 g}{\partial x_1^2} = 1 + x_2^2$ $\quad \frac{\partial^2 g}{\partial x_2^2} = 1 + x_1^2$

$\frac{\partial^2 g}{\partial x_2 x_1} = 1 + 2x_1$ $\quad \frac{\partial^2 g}{\partial x_1 x_2} = 1 + 2x_1$

$$\nabla^2 g(x) = \begin{bmatrix} \frac{\partial^2 g}{\partial x_1^2} & \frac{\partial^2 g}{\partial x_2 x_1} \\ \frac{\partial^2 g}{\partial x_1 x_2} & \frac{\partial^2 g}{\partial x_2^2} \end{bmatrix} = \begin{bmatrix} 1 + x_2^2 & 1 + 2x_1 \\ 1 + 2x_1 & 1 + x_1^2 \end{bmatrix}$$

ii) using: $\nabla^2 g(x) = J^T \cdot J + \sum_{i=1}^{m} F_i(x) \nabla^2 F_i(x)$

remembering $F(x) = \begin{bmatrix} x_1 + x_2 \\ x_1 \cdot x_2 \end{bmatrix}$

For $F_1(x)$: $\frac{\partial F_1}{\partial x_1} = 1$ $\quad \frac{\partial F_1}{\partial x_2} = 1$

$\frac{\partial^2 F_1}{\partial x_1^2} = 0$ $\quad \frac{\partial^2 F_1}{\partial x_2^2} = 0$ $\quad \frac{\partial^2 F_1}{\partial x_1 \partial x_2} = 0$

$$\nabla^2 F_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

For $F_2(x)$: $\frac{\partial F_2}{\partial x_1} = x_2$ $\quad \frac{\partial F_1}{\partial x_2} = x_1$

$\frac{\partial^2 F_2}{\partial x_1^2} = 0$ $\quad \frac{\partial^2 F_1}{\partial x_2^2} = 0$ $\quad \frac{\partial^2 F_1}{\partial x_2 x_1} = 1$

$$\nabla^2 F_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$J = \begin{bmatrix} 1 & 1 \\ x_2 & x_1 \end{bmatrix}$$

$$\nabla^2 g(x) = \begin{bmatrix} 1 & x_2 \\ 1 & x_1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ x_2 & x_1 \end{bmatrix} + [x_1 + x_2] \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} + [x_1 \cdot x_2] \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\nabla^2 g(x) = \begin{bmatrix} 1 + x_2^2 & 1 + x_1 x_2 \\ 1 + x_1 x_2 & 1 + x_1^2 \end{bmatrix} + \begin{bmatrix} 0 & x_1 \cdot x_2 \\ x_1 \cdot x_2 & 0 \end{bmatrix} = \begin{bmatrix} 1 + x_2^2 & 1 + 2x_1 x_2 \\ 1 + 2x_1 x_2 & 1 + x_1^2 \end{bmatrix}$$

Exercise 02.

tests to remember: one variable Newton

$$g(x) = x^2 - 2x + 1$$

to resolve: $g'(x) = 0$

$$g'(x) = 2x - 2$$

$$g'(x) = g'(x_k) + g''_{(x_k)}(x - x_k)$$

Point where $g'(x) = 0$

$$g(x_{k+1})' = 0$$

$$0 = g'(x_k) + g''_{(x_k)}(x_{k+1} - x_k)$$

$$g''(x_k)\, x_{k+1} = g''(x_k)\, x_k - g'(x_k)$$

$$x_{k+1} = x_k - \frac{g'(x_k)}{g''(x_k)}$$

Example 03 $g(x) = x^4 - 3x^2 + 2$ Determine $g'(x) = 0$

$$g'(x) = 4x^3 - 6x$$
$$g''(x) = 12x^2 - 6$$

Start $x_0 = 2$ $\quad x_1 = 2 - \frac{20}{42} = 1.52$

$x_1 = 1.52$ $\quad x_2 = 1.52 - \frac{5.0}{21.8} \approx 1.29$

several variables Newton

Taylor approximation:

$$g(x_k + s) \approx g(x_k) + \nabla g(x_k)^T \cdot s + \frac{1}{2} s^T H(x_k) s$$

where $H(x_k) = \nabla^2 g(x_k)$

Derivate and equal $= 0$

$$\nabla_s \left[g(x_k) + \nabla g(x_k)^T s + \frac{1}{2} s^T H(x_k) s \right]$$
$$= \nabla g(x_k) + H(x_k) s = 0$$

$$H(x_k)\, s_k = -\nabla g(x_k)$$

update $\quad x_{k+1} = x_k + s_k$

For our case:

$$g(z) = \frac{1}{2}\|F(z)\|_2^2 = \frac{1}{2}\sum_{i=1}^{m} F_i(z)^2$$

$$F: \mathbb{R}^n \to \mathbb{R}^m$$

$$\nabla g(z) = J(z)^T F(z)$$

$$H(z) = \nabla^2 g(z) = \underbrace{J(z)^T J(z)}_{\text{Gaussian term}} + \underbrace{\sum_{i=1}^{m} F_i(z)\nabla^2 F_i(z)}_{2^{nd}\text{ deriv.}}$$

Exact Newton $\quad \left[J^T J + \sum_i F_i \nabla^2 F_i\right] s = -J^T \cdot F$

Gauss Newton $\quad \sum_i F_i \nabla F \approx$ insignificant

$$H(z) \approx J(z)^T J(z) \Rightarrow \boxed{-J^T \cdot F = J^T J \cdot s}$$

$$s = -(J^T J)^{-1}(J^T \cdot F)$$

MATLAB FUNCTION

Exercise 03

a) Remembering: $\nabla g(x) = J(x)^T . F(x)$

$$\nabla^2 g(x) = J(x)^T . J(x) + \sum_{i=1}^{m} F_i(x) \nabla^2 F_i(x)$$

$$\nabla g(x_{k+1}) = 0$$

$$\nabla g(x_k + s_k) \approx \nabla g(x_k) + \nabla^2 g(x_k) s_k$$

given $\nabla g(x_{k+1}) = 0$

$$\underbrace{\nabla^2 g(x_k) s_k}_{} = \underbrace{-\nabla g(x_k)}_{}$$

$$\left[J(x_k)^T J(x_k) + \underbrace{\sum_{i=1}^{m} F_i(x_k) \nabla^2 F_i(x_k)}_{\text{insignificant}} \right] s_k = -J(x_k)^T F(x_k)$$

$$J(x_k)^T J(x_k) s_k = -J(x_k)^T F(x_k)$$

b) considering $\min_{s \in \mathbb{R}^n} \| J(x_k) s + F(x_k) \|_2^2$

$$J(x_k)^T [J(x_k) s_k + F(x_k)] = 0$$

is equivalent

$$J(x_k)^T J(x_k) s = -J(x_k)^T F(x_k)$$