Quadratic forms

We consider the quadratic function $f : \mathbb{R}^2 \to \mathbb{R}$ defined by

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x} \quad \text{with } \boldsymbol{x} = (x_1, x_2)^T, \tag{1}$$

where $\boldsymbol{A} \in \mathbb{R}^{2\times 2}$ is symmetric and $\boldsymbol{b} \in \mathbb{R}^2$. We will see that, depending on the eigenvalues of $\boldsymbol{A}$, the quadratic function f behaves very differently. Note that $\boldsymbol{A}$ is the second derivative of f, i.e., the Hessian matrix. To study basic properties of quadratic forms we first consider the case with a positive definite matrix

$$\boldsymbol{A} = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}, \quad \boldsymbol{b} = \boldsymbol{0}. \tag{2}$$

The eigenvectors of $\boldsymbol{A}$ corresponding to the eigenvalues $\lambda_1 = 1, \lambda_2 = 3$ are

$$\boldsymbol{u}^1 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \boldsymbol{u}^2 = \frac{1}{\sqrt{2}}\begin{pmatrix} -1 \\ 1 \end{pmatrix}.$$

Defining the orthonormal matrix $\boldsymbol{U} := [\boldsymbol{u}^1, \boldsymbol{u}^2]$ we obtain the eigenvalue de-

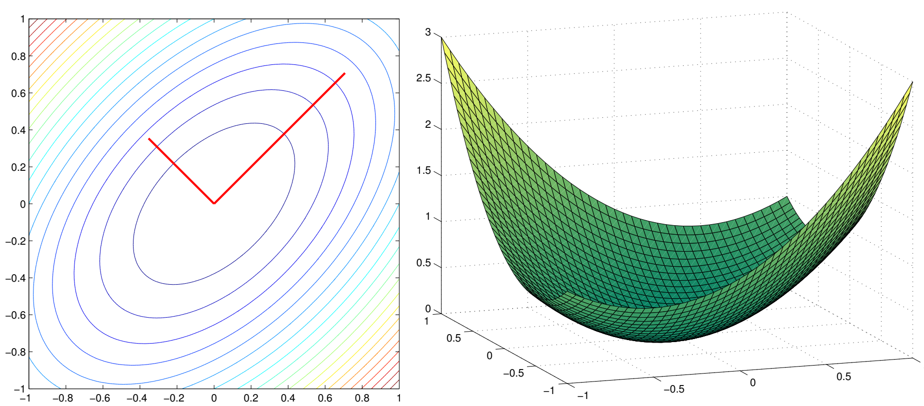

Figure 1: Quadratic form with the positive definite matrix $\boldsymbol{A}$ defined in (2). Left: Contour lines with the red lines indicate the eigenvector directions of $\boldsymbol{A}$. Right: Graph of the function. Note that the function is bounded from below and convex.

composition of $\boldsymbol{A}$, i.e.,

$$\boldsymbol{U}^T\boldsymbol{A}\boldsymbol{U} = \boldsymbol{\Lambda} = \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix}.$$

Note that $\boldsymbol{U} = \boldsymbol{U}^{-1}$. The contour lines for f as well as the eigenvector directions are shown in Figure 1. Defining the new variables $\bar{\boldsymbol{x}} := \boldsymbol{U}^T\boldsymbol{x}$, the quadratic form corresponding to (2) can, in the new variables, be written as

$$\bar{f}(\bar{\boldsymbol{x}}) = \frac{1}{2}\bar{\boldsymbol{x}}\boldsymbol{\Lambda}\bar{\boldsymbol{x}}.$$

Thus, in the variables that correspond to the eigenvector directions, the quadratic form is based on the diagonal matrix $\boldsymbol{\Lambda}$, and the eigenvalue matrix $\boldsymbol{U}$ corresponds to the basis transformation. Thus, to study basic properties of quadratic forms, we can restrict ourselves to diagonal matrices $\boldsymbol{A}$.

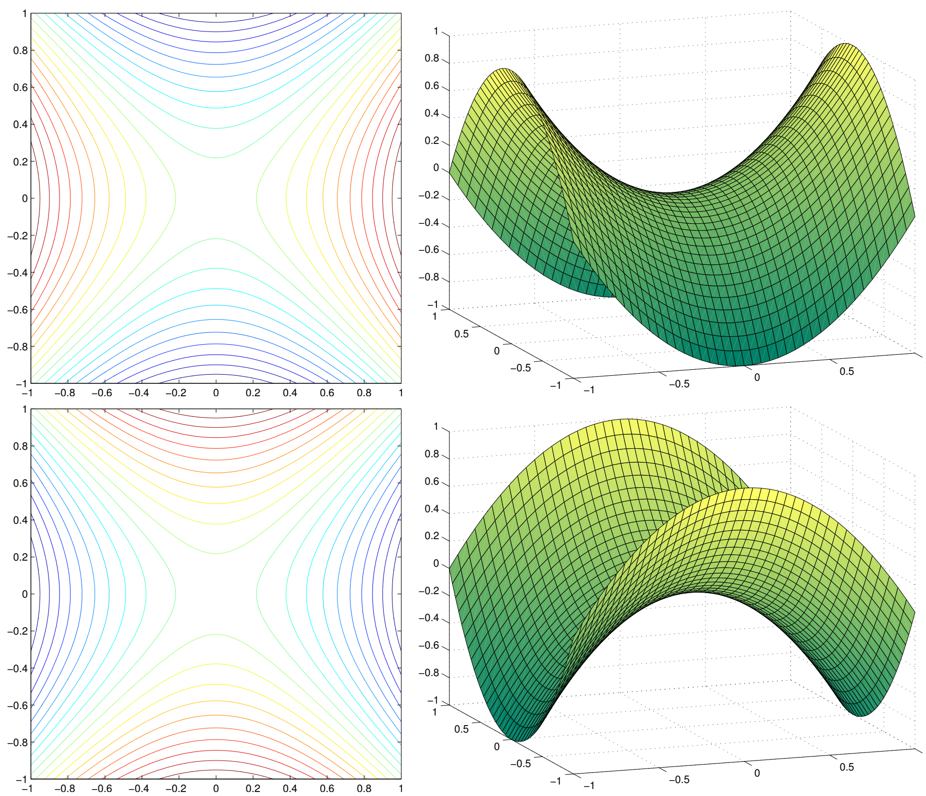

Figure 2: Quadratic form with indefinite matrices $\boldsymbol{A}_1$ (upper row) and $\boldsymbol{A}_2$ (lower row) defined in (3). Left: Contour lines. Right: Graph of the function. Note that the function is unbounded from above and from below.

We next consider the quadratic form corresponding to the indefinite matrices

$$\boldsymbol{A}_1 = \begin{pmatrix} 2 & 0 \\ 0 & -2 \end{pmatrix}, \quad \boldsymbol{A}_2 = \begin{pmatrix} -2 & 0 \\ 0 & 2 \end{pmatrix}, \tag{3}$$

and use $\boldsymbol{b} = \mathbf{0}$. Visualizations of the corresponding quadratic form are shown in Figure 2. Note that the functions corresponding to $\boldsymbol{A}_1$ coincides with the one from $\boldsymbol{A}_2$ after exchanging the coordinate axes. The origin is a maximum in one coordinate direction, and it is a minimum in the other direction, which is a consequence of the indefinite matrices $\boldsymbol{A}_1, \boldsymbol{A}_2$. These functions are neither bounded from above, nor from below and thus do not have a minimum nor a maximum.

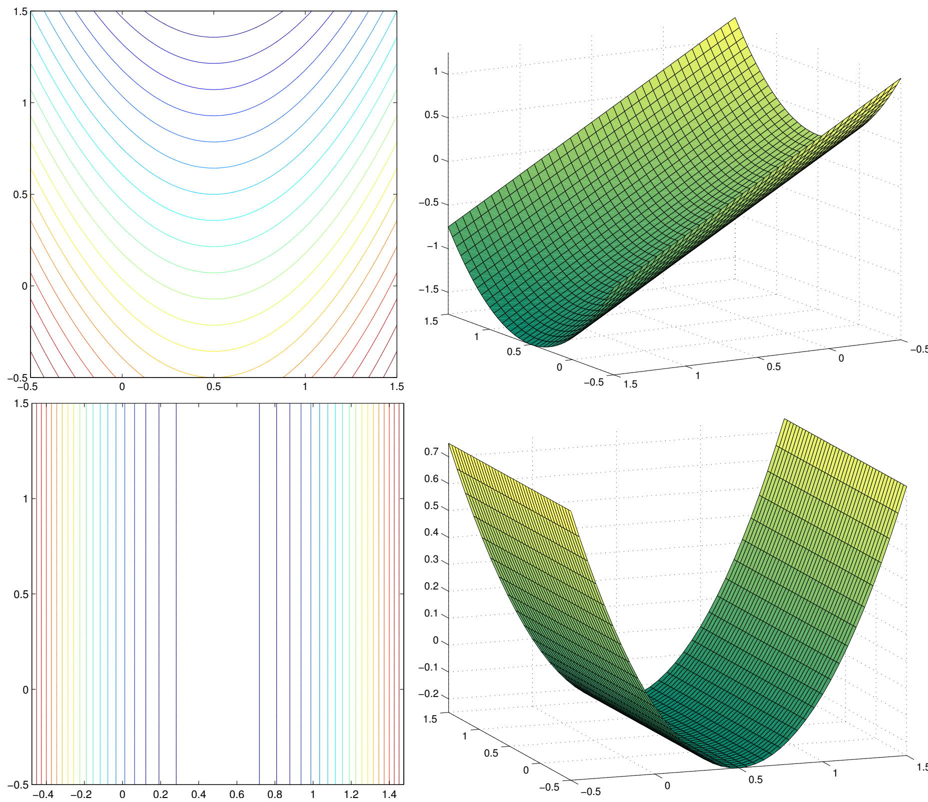

Figure 3: Quadratic form with semi-definite matrix $\boldsymbol{A}$ and $\boldsymbol{b}_1$ (upper row) and $\boldsymbol{b}_2$ (lower row) defined in (4). Left: Contour lines. Right: Graph of the function. Note that depending on $\boldsymbol{b}$, the function does not have a minimum (upper row) or has infinitely many minima (lower row).

Finally, we study quadratic forms with semi-definite Hessian matrices. We consider the cases

$$\boldsymbol{A} = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix}, \ \boldsymbol{b}_1 = \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \ \boldsymbol{b}_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{4}$$

For the indefinite case, the choice of $\boldsymbol{b}$ influences weather there exists a minimum or not. Visualizations of the quadratic forms can be seen in Figure 3. In the direction where the Hessian matrix is singular, the function is dominated by the linear term $\boldsymbol{b}$. The function based on $\boldsymbol{A}$ and $\boldsymbol{b}_1$ is unbounded from below and, thus, does not have a minimum. On the other hand, the function based on $\boldsymbol{A}$ and $\boldsymbol{b}_2$ is independent from x_2, and bounded from below. Thus, all points with $x_2 = 0$ are minima of f.

Convergence of steepest descent for increasingly ill-conditioned matrices

We consider the quadratic function

$$f(x_1, x_2) = \frac{1}{2}(c_1 x_1^2 + c_2 x_2^2) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \tag{5}$$

for various c_1 and c_2, where $\boldsymbol{A} = \mathrm{diag}(c_1, c_2)$ and $x = (x_1, x_2)^T$. The function is convex and has a global minimum at $x_1 = x_2 = 0$. Since $\boldsymbol{A}$ is diagonal, c_1 and c_2 are also the eigenvalues of $\boldsymbol{A}$. We use the steepest descent method with exact line search to minimize (5). A listing of the simple algorithm is given next.

```
% starting point
x1 = c2 / sqrt(c1^2 + c2^2);
x2 = c1 / sqrt(c1^2 + c2^2);
for iter = 1:100
  err = sqrt(x1^2 + x2^2);
  fprintf('Iter: %3d: x1: %+4.8f, x2: %+4.8f, error %4.8f\n',iter,x1,x2,err);
  if (error < 1e-12)
    fprintf('Converged with error %2.12f.\n', error);
    break;
  end
  % exact line search
  alpha = (c1^2*x1^2 + c2^2*x2^2) / (c1^3*x1^2 + c2^3*x2^2);
  g1 = c1 * x1;
  g2 = c2 * x2;
  x1 = x1 - alpha * g1;
  x2 = x2 - alpha * g2;
end
```

Running the above script with $c_1 = c_2 = 1$, the method terminates after a single iteration. This one-step convergence is a property of the steepest descent when the eigenvalues c_1, c_2 coincide and thus the contour lines are circles; see Figure 4.

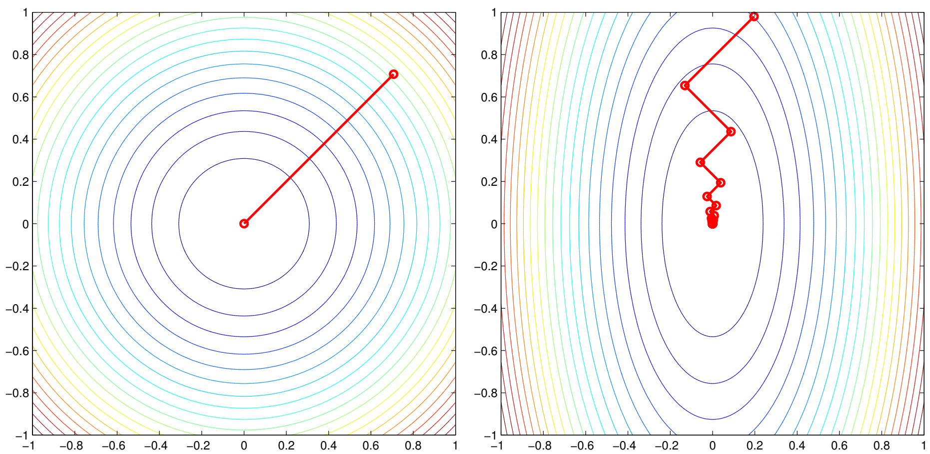

Figure 4: Contour lines and iterates for $c_1 = c_2 = 1$ (left plot) and $c_1 = 5, c_2 = 1$ (right plot).

```
Iteration   1: x1: +0.70710678, x2: +0.70710678, error 1.00000000
Iteration   2: x1: +0.00000000, x2: +0.00000000, error 0.00000000
Converged with error 0.000000000000.
```

For $c_1 = 5, c_2 = 1$, the iteration terminates after 36 iterations; the first iterations are as follows:

```
Iteration   1: x1: +0.19611614, x2: +0.98058068, error 1.000000000000
Iteration   2: x1: −0.13074409, x2: +0.65372045, error 0.666666666667
Iteration   3: x1: +0.08716273, x2: +0.43581363, error 0.444444444444
Iteration   4: x1: −0.05810848, x2: +0.29054242, error 0.296296296296
Iteration   5: x1: +0.03873899, x2: +0.19369495, error 0.197530864198
Iteration   6: x1: −0.02582599, x2: +0.12912997, error 0.131687242798
Iteration   7: x1: +0.01721733, x2: +0.08608664, error 0.087791495199
Iteration   8: x1: −0.01147822, x2: +0.05739110, error 0.058527663466
Iteration   9: x1: +0.00765215, x2: +0.03826073, error 0.039018442311
Iteration  10: x1: −0.00510143, x2: +0.02550715, error 0.026012294874
```

Taking coefficients between errors of two consecutive iterations, we observe that

$$\frac{\mathrm{err}_{k+1}}{\mathrm{err}_k} = \frac{2}{3} = \frac{5-1}{5+1} = \frac{\kappa - 1}{\kappa + 1},$$

where κ denotes the condition number of the matrix $\boldsymbol{A}$, i.e.,

$$\kappa = \mathrm{cond}\begin{pmatrix} c_1 & 0 \\ 0 & c_2 \end{pmatrix} = \frac{\lambda_{\max}(\boldsymbol{A})}{\lambda_{\min}(\boldsymbol{A})} = \frac{c_1}{c_2}.$$

The contour lines of f for $c_1 = c_2 = 1$ and $c_1 = 5, c_2 = 1$ are shown in Figure 4. Now, we study the function (5) with $c_2 = 1$ and with different values for c_1, namely $c_1 = 10, 50, 100, 1000$. The number of iterations required for these cases are $139, 629, 1383$ and 13817, respectively. As can be seen, the number increases significantly with c_1, and thus with increasing κ. The output of the first iterations for $c_1 = 10$ are

```
Iteration  1: x1: +0.09950372, x2: +0.99503719, error 1.000000000000
Iteration  2: x1: -0.08141213, x2: +0.81412134, error 0.818181818182
Iteration  3: x1: +0.06660993, x2: +0.66609928, error 0.669421487603
Iteration  4: x1: -0.05449903, x2: +0.54499032, error 0.547708489857
Iteration  5: x1: +0.04459012, x2: +0.44590117, error 0.448125128065
Iteration  6: x1: -0.03648282, x2: +0.36482823, error 0.366647832053
Iteration  7: x1: +0.02984958, x2: +0.29849582, error 0.299984589862
Iteration  8: x1: -0.02442239, x2: +0.24422386, error 0.245441937160
Iteration  9: x1: +0.01998195, x2: +0.19981952, error 0.200816130403
Iteration 10: x1: -0.01634887, x2: +0.16348870, error 0.164304106694
```

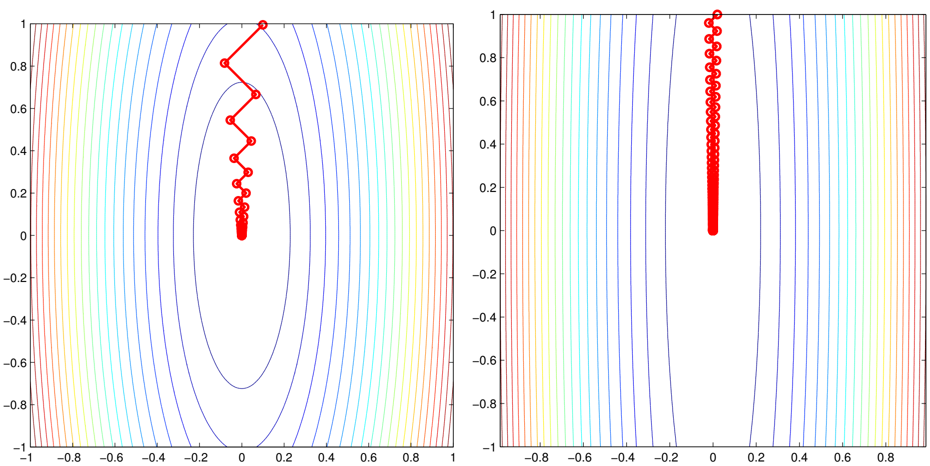

Figure 5: Contour lines and iterates for $c_1 = 10$ (left plot) and $c_1 = 50$ (right plot).

Taking quotients of consecutive errors, we again observe the theoretically expected convergence rate of $(\kappa-1)/(\kappa+1) = 9/11 = 0.8182$. The large number of iterations for the other cases of c_1 can be explained due to the increasingly ill-conditioning of the quadratic form. The theoretically proved convergence rates for $c_1 = 50, 100, 1000$ are $0.9608, 0.9802$ and 0.9980, respectively. These are also exactly the rates we observe for all these cases. Contour lines and iterates for $c_2 = 10, 50$ are shown in Figure 5.

Convergence examples for Newton's method

Here, we study convergence properties of Newton's method for various functions. We start by studying the nonlinear function $f : \mathbb{R} \to \mathbb{R}$ defined by

$$f(x) = \frac{1}{2}x^2 - \frac{1}{3}x^3. \tag{6}$$

The graph of this function, as well as of its derivative are shown in Figure 6.

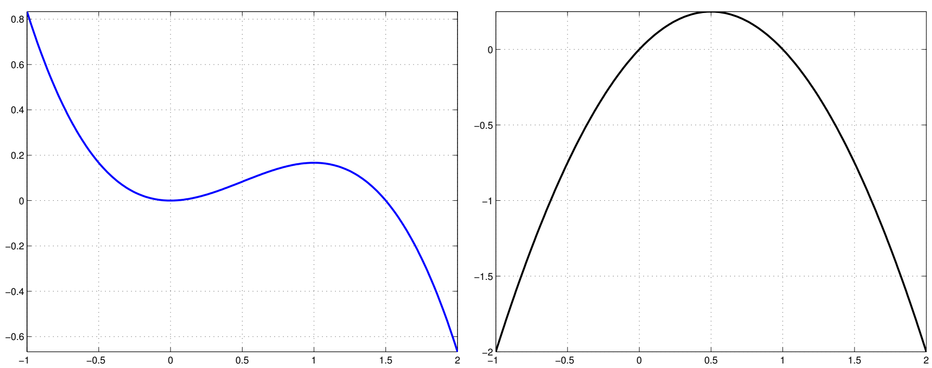

Figure 6: Graph of nonlinear function defined in (6) (left plot) and of its derivative (right plot).

We want to find the (local) minimum of f, i.e., the point $x = 0$. As expected, at the local minimum, the derivative of f vanishes. However, the derivative also vanishes at the local maximum $x = 1$. A Newton method to find the local minimum of f uses the stationarity of the derivative at the extremal points (minimum and maximum). At a given point x_k, the new point x_{k+1} is computes as (where we use a step length of 1)

$$x_{k+1} = \frac{x_k^2}{2x_k - 1} \tag{7}$$

This expression is plotted in Figure 7. First, we consider a Newton iteration starting from $x = 20$. We obtain the iterations

```
Iteration    1:  x:+20.0000000000000000
Iteration    2:  x:+10.2564102564102573
Iteration    3:  x:  +5.3910172175612390
Iteration    4:  x:  +2.9710656645912565
Iteration    5:  x:  +1.7861182949934302
Iteration    6:  x:  +1.2402508292312779
Iteration    7:  x:  +1.0389870964455772
```

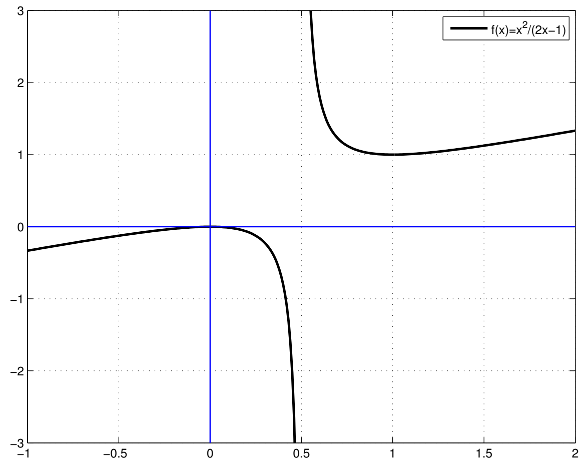

Figure 7: The Newton step (7) plotted as a function.

```
Iteration    8: x:  +1.0014100464549898
Iteration    9: x:  +1.0000019826397768
Iteration   10: x:  +1.0000000000039309
Iteration   11: x:  +1.0000000000000000
```

Thus, the Newton method converges the local maximum $x = 1$. Observe that initially, the convergence rate appears to be linear, but beginning from the 6th iteration we can observe the quadratic convergence of Newton's method, i.e., the number of correct digits doubles in every Newton iteration.

Next we use the initialization $x = -20$, which results in the following iterations:

```
Iteration    1: x:-20.0000000000000000
Iteration    2: x:  -9.7560975609756095
Iteration    3: x:  -4.6402366520692562
Iteration    4: x:  -2.0944362725536236
Iteration    5: x:  -0.8453981595529151
Iteration    6: x:  -0.2656083788656865
Iteration    7: x:  -0.0460730399974074
Iteration    8: x:  -0.0019436273713611
Iteration    9: x:  -0.0000037630593883
Iteration   10: x:  -0.0000000000141605
Iteration   11: x:  -0.0000000000000000
```

Now, the iterates converge to the local minimum $x = 0$. Again we initially observe linear convergence, and as we get close to the minimum, the convergence becomes quadratic.

To show how sensitive Newton's method is to the initial guess, we now compute initialize the method with values that are very close from each other. Initializing with $x = 0.501$ results in convergence to $x = 1$:

```
Iteration    1: x:   +0.5010000000000000
Iteration    2: x:+125.5004999999998745
Iteration    3: x:  +63.0012499959999559
Iteration    4: x:  +31.7526249580009043
Iteration    5: x:  +16.1303121430340468
Iteration    6: x:   +8.3231533526241517
Iteration    7: x:   +4.4275548879571378
Iteration    8: x:   +2.4956038610665341
Iteration    9: x:   +1.5604396125094859
Iteration   10: x:   +1.1480954481351822
Iteration   11: x:   +1.0169205491424664
Iteration   12: x:   +1.0002769332576908
Iteration   13: x:   +1.0000000766495756
Iteration   14: x:   +1.0000000000000058
```

Initialization with $x = 0.499$ leads to convergence to the local minimum $x = 0$:

```
Iteration    1: x:  +0.4990000000000000
Iteration    2: x:  -124.5004999999998887
Iteration    3: x:  -62.0012499959999630
Iteration    4: x:  -30.7526249580009114
Iteration    5: x:  -15.1303121430340521
Iteration    6: x:  -7.3231533526241543
Iteration    7: x:  -3.4275548879571387
Iteration    8: x:  -1.4956038610665345
Iteration    9: x:  -0.5604396125094862
Iteration   10: x:  -0.1480954481351824
Iteration   11: x:  -0.0169205491424666
Iteration   12: x:  -0.0002769332576907
Iteration   13: x:  -0.0000000766495756
Iteration   14: x:  -0.0000000000000059
```

Finally, if the method is initialized with $x = 0.5$, it diverges since the second derivative (i.e., the Hessian) of f is singular at this point.

Next, we study the function

$$f(x) = \frac{x^4}{4}$$

which is shown, together with its derivative in Figure 8. This function has a singular Hessian at its minimum $x = 0$. Using the Newton matrix to find the global minimum results in the with starting guess $x = 1$ results in the following iterations (only the first 15 iterations are shown):

```
Iteration    1: x:  +1.0000000000000000
```

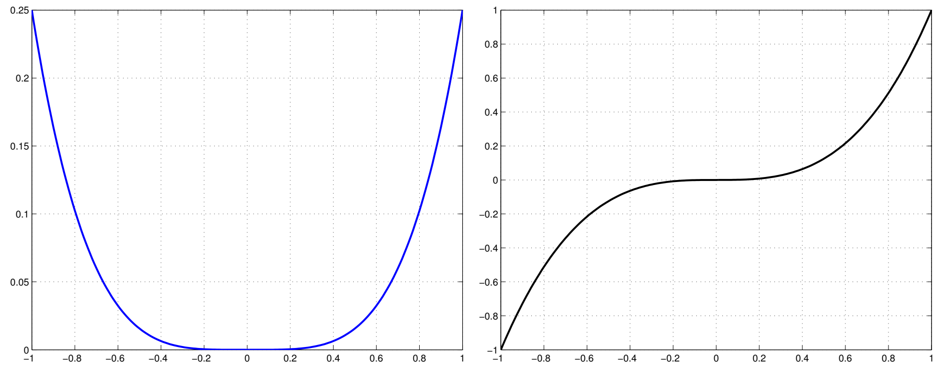

Figure 8: Function $f(x) = x^4/4$ (left) and its derivative (right).

```
Iteration   2: x: +0.6666666666666666
Iteration   3: x: +0.4444444444444444
Iteration   4: x: +0.2962962962962963
Iteration   5: x: +0.1975308641975309
Iteration   6: x: +0.1316872427983539
Iteration   7: x: +0.0877914951989026
Iteration   8: x: +0.0585276634659351
Iteration   9: x: +0.0390184423106234
Iteration  10: x: +0.0260122948737489
Iteration  11: x: +0.0173415299158326
Iteration  12: x: +0.0115610199438884
Iteration  13: x: +0.0077073466292589
Iteration  14: x: +0.0051382310861726
Iteration  15: x: +0.0034254873907817
```

Note that the iterates converge to the solution $x = 0$, but they only converge at a linear rate due to singularity of the Hessian at the solution. For the initial guess $x = -1$, the iterates have the negative values of the ones shown above.

Finally, we consider the negative hyperbolic secant function

$$f(x) = -\mathrm{sech}(x).$$

The graph of the function and its derivative are shown in Figure 9. This function changes its curvature, and thus Newton's method diverges if the initial guess is too far from the minimum. The Newton iteration to find the minimum $x = 0$ of this function computes, at a current iterate x_k the new iterate as

$$x_{k+1} = x_k + \frac{\sinh(2x_k)}{\cosh(2x_k) - 3}. \tag{8}$$

We first study the Newton iterates for starting value $x = 0.1$ and observe quadratic convergence (actually, the convergence is even faster than quadratic,

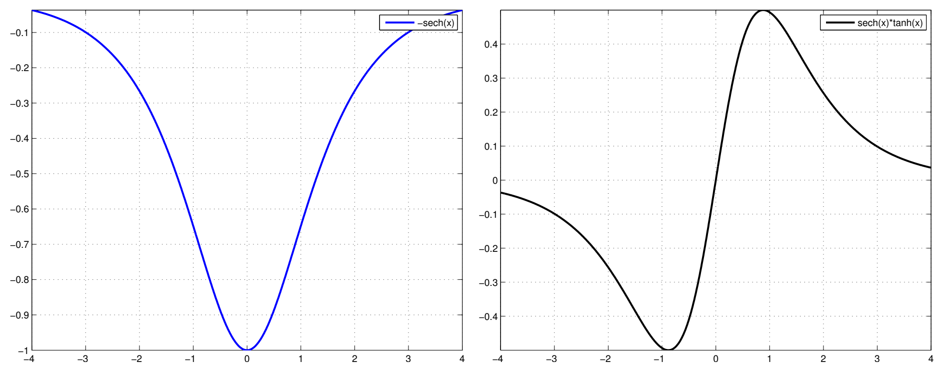

Figure 9: Function $f(x) = -\text{sech}(x)$ (left) and its derivative (right).

which is due to properties of the hyperbolic secant function):

```
Iteration    1:  x:  +0.1000000000000000
Iteration    2:  x:  −0.0016882781843722
Iteration    3:  x:  +0.0000000080201476
Iteration    4:  x:  −0.0000000000000000
```

Starting the iteration from $x = 0.2$ or $x = 0.5$, the method also converges to the local (and global) minimum. The iterates for starting guess of $x = 0.5$ are:

```
Iteration    1:  x:  +0.5000000000000000
Iteration    2:  x:  −0.3066343421104646
Iteration    3:  x:  +0.0546314000372350
Iteration    4:  x:  −0.0002727960502389
Iteration    5:  x:  +0.0000000000338348
Iteration    6:  x:  +0.0000000000000000
```

However, if the method is initialized with $x = 0.6$ (or any value larger than that), the method diverges. The first 10 iterates for a starting guess of $x = 0.6$ are:

```
Iteration    1:  x:  +0.6000000000000000
Iteration    2:  x:  −0.6691540935844730
Iteration    3:  x:  +1.1750390494712146
Iteration    4:  x:  +3.4429554757227767
Iteration    5:  x:  +4.4491237147096072
Iteration    6:  x:  +5.4499441189054147
Iteration    7:  x:  +6.4500548922755296
Iteration    8:  x:  +7.4500698791442161
Iteration    9:  x:  +8.4500719073107184
Iteration   10:  x:  +9.4500721817916382
```

This divergence is explained by the fact that the function is concave at $x = 0.6$.