Nonlinear Equations

1 Introduction

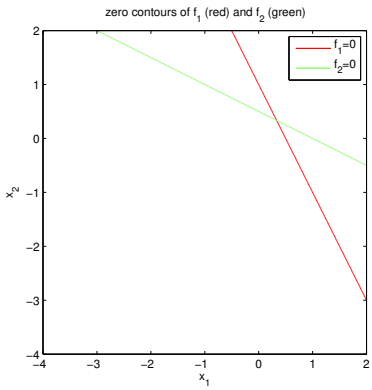
In applications we usually need to find several unknown values $x_1, \ldots, x_n$. We have n equations for $x_1, \ldots, x_n$

$$\begin{aligned} f_1(x_1, \ldots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, \ldots, x_n) &= 0 \end{aligned}$$

and we want to find the solutions.

In many cases the problem can be (approximatively) described by **linear equations**. In this case we have n linear equations for n unknowns. We will get a unique solution if the matrix is nonsingular.
Example with $n = 2$: Find x_1, x_2 such that

$$\begin{aligned} 2x_1 + x_2 - 1 &= 0 \\ x_1 + 2x_2 - 1 &= 0 \end{aligned}$$

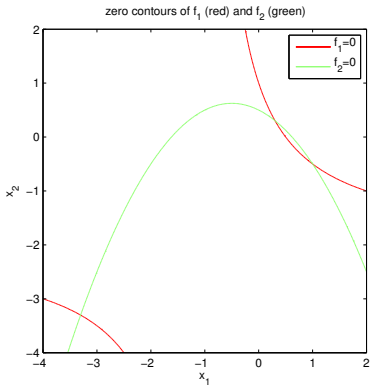


Here we have one solution $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}$ which is the intersection of the red and the green line.

In other cases the problem is nonlinear, and we obtain n **nonlinear equations**.

Example with $n = 2$: Find x_1, x_2 such that

$$\begin{aligned} 2x_1 + x_2 + x_1 x_2 - 1 &= 0 \\ x_1 + 2x_2 + x_1^2 - 1 &= 0 \end{aligned}$$



Here we have three solutions $\begin{bmatrix} .3028 \\ .3028 \end{bmatrix}, \begin{bmatrix} 1 \\ -.5 \end{bmatrix}, \begin{bmatrix} -3.3028 \\ -3.3028 \end{bmatrix}$. Here is how I found the first solution in **Matlab**:

```
f = @(x) [ 2*x(1)+x(2)+x(1)*x(2)-1 ; x(1)+2*x(2)+x(1)^2-1 ]  % Define function f
xs = fsolve(f,[0;0])                                        % Find solution near [0;0]
```

2 One nonlinear equation

2.1 Bisection Method

Assume that the function f is continuous. If we have two function values $f(a), f(b)$ with opposite signs then the intermediate value theorem guarantees that there must be a point $x_* \in (a,b)$ with $f(x_*) = 0$. This motivates the bisection method:

Bisection method Find $x_* \in [a_0, b_0]$ such that $f(x_*) = 0$.
Given:

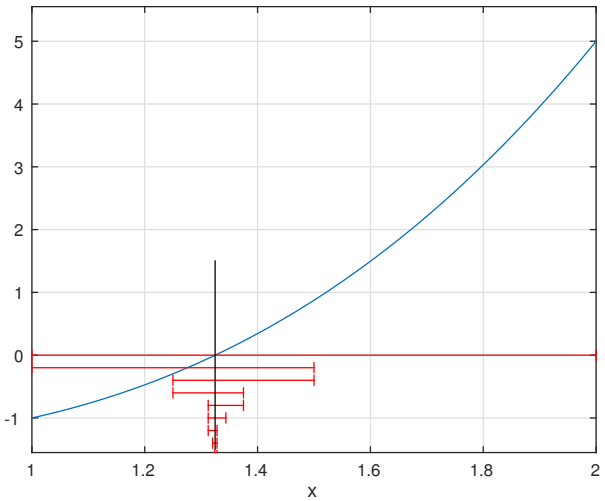
- subroutine to evaluate $f(x)$
- initial guesses a_0, b_0 where $f(a_0)$ and $f(b_0)$ have different signs

this algorithm computes a sequence of intervals $[a_k, b_k]$ such that $x_* \in [a_k, b_k]$

Algorithm: For $k = 0, 1, 2, \ldots$:
$c_k := (a_k + b_k)/2$
evaluate $f(c_k)$
If $f(c_k) = 0$: stop
If $f(c_k), f(a_k)$ have different sign: $[a_{k+1}, b_{k+1}] := [a_k, c_k]$
If $f(c_k), f(a_k)$ have same sign: $[a_{k+1}, b_{k+1}] := [c_k, b_k]$

Example: For $f(x) = x^3 - x - 1$ and $a_0 = 1$, $b_0 = 2$ we obtain the following intervals: $f(1) = -1 < 0$, $f(2) = 5 > 0$

k	$[a_k, b_k]$	$b_k - a_k$	
0	[1.0 , 2.0]	1	$f(1.5) = .875 > 0$
1	[1.0 , 1.5]	0.5	$f(1.25) = -.29688 < 0$
2	[1.25 , 1.5]	0.25	$f(1.375) = .22461 > 0$
3	[1.25 , 1.375]	0.125	$f(1.3125) = -.051514 < 0$
4	[1.3125 , 1.375]	$6.3 \cdot 10^{-2}$	$f(1.34375) = .082611 > 0$
5	[1.3125 , 1.34375]	$3.1 \cdot 10^{-2}$	$f(1.328125) = .014576 > 0$
6	[1.3125 , 1.328125]	$1.6 \cdot 10^{-2}$	
⋮	⋮	⋮	
50	[1.3247179572447454, 1.3247179572447463]	$8.9 \cdot 10^{-16}$	
51	[1.3247179572447458, 1.3247179572447463]	$4.4 \cdot 10^{-16}$	
52	[1.3247179572447458, 1.3247179572447461]	$2.2 \cdot 10^{-16}$	



In practice we assume that the function is given as a subroutine which we can call (in Matlab we can specify a function in an m-file). Then the only substantial cost for each step is this call to this external function:

The cost of one bisection step is one function evaluation.

In theory we can let this continue forever, unless we hit for some k the solution x_* exactly. Then we get an infinite sequence a_k with $a_{k+1} \geq a_k$ and an infinite sequence b_k with $b_{k+1} \leq b_k$.

Theorem 2.1. *Assume that the function f is continuous on $[a_0, b_0]$. If $f(a_0)$ and $f(b_0)$ have different sign, then the bisection method converges:*

$$\lim_{k\to\infty} a_k = \lim_{k\to\infty} b_k = x_* \qquad \text{with } f(x_*) = 0.$$

Proof. As $b_k - a_k = 2^{-k}(b_0 - a_0)$ goes to zero as $k \to \infty$ we must have that both a_k and b_k converge to the same point x_* as $k \to \infty$. Since $f(a_k) \leq 0$ and $f(b_k) \geq 0$ we must have $f(x_*) = 0$ by the continuity of f. □

Note that the midpoint c_k satisfies $|c_k - x_*| \leq (b_k - a_k)/2$, therefore we have **decreasing error bounds E_k satisfying**

$$\boxed{|c_k - x_*| \leq E_k, \qquad E_{k+1} = \frac{1}{2}E_k}$$

For each step we have $E_{k+1} = \frac{1}{2}E_k$, so the error is reduced by at least a factor $\frac{1}{2}$.

Essentially, we obtain one new binary digit of the result with each bisection step, or about 3 new decimal digits with 10 bisection steps.

If for some method we get approximations x_k satisfying $|x_k - x_*| \leq E_k$ where the error bounds satisfy

$$E_k \leq CE_k$$

with a fixed factor $C < 1$ we say that the method is **convergent of order 1** (since we have $E_{k+1} \leq C \cdot E_k^1$ with exponent 1).

Bisection method:

- assumption: $f(x)$ is continuuous, we have a_0, b_0 where $f(a_0)$, $f(b_0)$ have different sign
- cost per step: 1 function evaluation
- improvement of error bound for each step: $E_{k+1} = \frac{1}{2}E_k$
- "bracketing": we can guarantee $a_k \leq x_* \leq b_k$ for each step

2.2 Secant Method

Assume that we have two function values $f(a)$ and $f(b)$. Based on this information we want to find a good guess c for the solution x_*: We can approximate $f(x)$ by the linear interpolation

$$p(x) = f(b) + f[a,b](x - b)$$

where $f[a,b] = \frac{f(b)-f(a)}{b-a}$. Then we find c such that $p(c) = 0$: Solving $f(b) + f[a,b](c - b) = 0$ for c gives

$$c = b - f(b)/f[a,b].$$

If we have two initial guesses x_0, x_1 we can use this to find an improved guess x_2. Using x_1, x_2 we find x_3, etc.

Algorithm: Secant Method Find $x_* \in [a_0, b_0]$ such that $f(x_*) = 0$.
Given:

- subroutine to evaluate $f(x)$
- two initial guesses x_0, x_1

Then this algorithm computes a sequence x_k which may or may not converge to a root x_*.

Algorithm: For $k = 1, 2, 3, \ldots$:
$x_{k+1} := x_k - f(x_k)/f[x_{k-1}, x_k]$

Example: Consider the function $f(x) = x^3 - x - 1$ and the initial guesses $x_0 = 1$, $x_1 = 2$. We can measure the "number of correct digits" as $N_k := -\log_{10}|x_k - x_*|$. We want to see how N_k grows with each step.

k	x_k	N_k	N_k/N_{k-1}
2	1.16666666666666666666666666666666666666666666666666666666666667	0.8	
3	1.25311203319502074688796680497925311203319502074688796680	1.1	1.4292
4	1.32720644584165640039570185142407424920450200352602540210	1.9	1.6624
5	1.32385009638764090385381405944349497906016891171355395298	3.1	1.6084
6	1.32470793653208797064174666448878197322217786129602539372	5.0	1.6329
7	1.32471796535381767575939715803702106931272485403036767120	8.1	1.6185
8	1.32471795724467030158200181494839983320960900093813494824	13.1	1.6216
9	1.32471795724474602596033662546920425813781535049399611421	21.2	1.6190
10	1.32471795724474602596090885447809738111461705368557107541	34.4	1.6191
11	1.32471795724474602596090885447809734073440405690173336451	55.7	1.6185

Errors of the secant method

During the algorithm we have $a = x_{k-1}$ and $b = x_k$. We then compute $c = x_{k+1}$ using the secant. We want to show that the new error $|c - x_*|$ is small:

From the interpolation error we know that

$$f(x_*) - p(x_*) = R(x_*), \qquad R(x_*) = \frac{1}{2} f''(t) \cdot (x_* - a)(x_* - b)$$

(where t is somewhere between a, b, x_*). If $|f''(t)| \le C_2$ we have $|R(x_*)| \le \frac{C_2}{2} |x_* - a| \cdot |x_* - b|$.

Note that $f(x_*) = 0 = p(c)$. Hence

$$\underbrace{p(c) - p(x_*)}_{f[a,b] \cdot (c - x_*)} = R(x_*)$$

since $p(x)$ is a linear function with slope $f[a,b]$. Therefore

$$c - x_* = \frac{R(x_*)}{f[a,b]}$$

We have $f[a,b] = f'(s)$ with $s \in [a,b]$. If $|f'(s)| \ge C_1 > 0$ we therefore have with $D := \frac{C_2}{2C_1}$

$$\boxed{|c - x_*| \le D\,|a - x_*| \cdot |b - x_*|} \tag{1}$$

Since $a = x_{k-1}$, $b = x_k$, $c = x_{k+1}$ we obtain

$$|x_{k+1} - x_*| \le D\,|x_{k-1} - x_*| \cdot |x_k - x_*|$$

Let $e_k := D\,|x_k - x_*|$. Multiplying by D gives

$$e_{k+1} \le e_{k-1} e_k.$$

Now assume that

$$e_0 \le q, \quad e_1 \le q \qquad \text{with } q < 1$$

Then we obtain

$$e_0 \le q^1, \quad e_1 \le q^2, \quad e_2 \le q^3, \quad e_3 \le q^5, \quad \dots \quad e_k \le q^{F_k}$$

with the **Fibonacci number** F_k (defined by $F_0 = 1$, $F_1 = 1$, $F_{k+1} = F_k + F_{k-1}$). Since $q < 1$ and $F_k \to \infty$ for $k \to \infty$ we obtain convergence $e_k = D \cdot |x_k - x_*| \to 0$ if our assumptions

$$\left|f''(t)\right| \le C_2, \qquad \left|f'(t)\right| \ge C_1 > 0 \tag{2}$$

are satisfied. The order of convergence corresponds to the ratio F_k/F_{k-1} which converges to the golden ratio $\frac{\sqrt{5}+1}{2}$.

Theorem 2.2. *Assume that $f(x_*) = 0$ and*

- *$f'(x)$ and $f''(x)$ exist and are continuous near x_**
- *$f'(x_*) \neq 0$.*

Then there exists $\delta > 0$, $C > 0$ such that for $|x_0 - x_*| \leq \delta$, $|x_1 - x_*| \leq \delta$ we have

- *$\lim_{k\to\infty} x_k = x_*$ (convergence)*
- *$|x_k - x_*| \leq E_k$ and $E_{k+1} \leq CE_k^\alpha$ with $\alpha = \frac{\sqrt{5}+1}{2}$ (convergence with order $\alpha > 1$)*

Proof. Pick $\varepsilon > 0$ such that on the interval $B_\varepsilon = [x_* - \varepsilon, x_* + \varepsilon]$ we have that $|f'(x)| > 0$ and f'' is continuous:

$$\text{For } x \in B_\varepsilon: \qquad \left|f'(x)\right| \geq C_1 > 0, \quad \left|f''(x)\right| \leq C_2 \tag{3}$$

with some constants C_1, C_2. Let $D = \frac{C_2}{2C_1}$. Pick $q < 1$ such that $\delta := q/D \leq \varepsilon$.

Now assume $|x_{k-1} - x_*| \leq \delta$, $|x_k - x_*| \leq \delta$. Since $\delta \leq \varepsilon$ we have $x_{k-1}, x_k, x_* \in B_\varepsilon$. We now have

$$|x_{k+1} - x_*| = \frac{|f''(t)|}{2\,|f'(s)|}\,|x_k - x_*| \cdot |x_{k-1} - x_*|$$

where the intermediate points s, t are located between x_0, x_1, x_*. Hence we have $s, t \in B_\varepsilon$ and (3) gives

$$|x_{k+1} - x_*| \leq D\,|x_k - x_*| \cdot |x_{k-1} - x_*| \leq \underbrace{D\delta}_{q<1} \cdot \delta < \delta$$

so that we also have $|x_{k+1} - x_*| \leq \delta$.

Therefore we obtain by induction that $|x_k - x_*| \leq \delta$ for $k = 0, 1, 2, \ldots$, and that

$$|x_{k+1} - x_*| \leq D\,|x_k - x_*| \cdot |x_{k-1} - x_*|$$

As we saw above, this implies that $e_k := D\,|x_k - x_*|$ satisfies $e_k \leq q^{F_k}$ where F_k are the Fibonacci numbers. Since $q < 1$ and $F_k \to \infty$ we obtain convergence $\lim_{k\to\infty} x_k = x_*$.

It remains to prove convergence of order $\alpha = \frac{\sqrt{5}+1}{2}$: We have shown $e_k \leq \tilde{E}_k := q^{F_k}$. Since the Fibonacci numbers satisfy $F_{k+1} - \alpha F_k = (1-\alpha)^{k+1} \geq -1$ we have

$$\begin{aligned} & F_{k+1} \geq \alpha F_k - 1 \\ \Rightarrow \quad & q^{F_{k+1}} \leq q^{\alpha F_k} \cdot q^{-1} \\ \Rightarrow \quad & \tilde{E}_{k+1} \leq \tilde{E}_k^\alpha \cdot q^{-1} \end{aligned}$$

□

2.3 Newton Method

For the secant method we used the interpolating polynomial with the nodes a, b. Now assume that $a = b$, and that we know $f(a)$ and $f'(a)$. We can approximate $f(x)$ by the linear interpolation

$$p(x) = f(a) + f[a,a](x - a)$$

where $f[a,a] = f'(a)$. Then we find c such that $p(c) = 0$: Solving $f(a) + f[a,a](c-a) = 0$ for c gives

$$c = b - f(a)/f[a,a].$$

If we have an initial guesses x_0 we can use this to find an improved guess x_1, etc.:

Algorithm: Newton Method

- Initial guess x_0
- For $k = 1, 2, 3, \ldots$:
  $x_{k+1} := x_k - f(x_k)/f'(x_k)$

Example: Consider the function $f(x) = x^3 - x - 1$ and the initial guess $x_0 = 1$. We can measure the "number of correct digits" as $N_k := -\log_{10}|x_k - x_*|$:

k	x_k	N_k	N_k/N_{k-1}
1	1.50000000000000000000000000000000000000000000000000000000000000000000000000000000	0.8	
2	1.34782608695652173913043478260869565217391304347826086957	1.6	2.1636
3	1.32520039895090687451516382845111004395847955376602268110	3.3	2.0269
4	1.32471817399905373435873101429361486184335021364673358323	6.7	2.0093
5	1.32471795724478980823964747415808243636452539861027911560	13.4	2.0046
6	1.32471795724474602596090885626441930491868291749622751396	26.7	2.0023
7	1.32471795724474602596090885447809734073440405690173336751	53.5	2.0011

Errors of the Newton method

For the errors we obtain from (1) with $a = b = x_k$, $c = x_{k+1}$ that

$$|x_{k+1} - x_*| \le D\,|x_k - x_*|^2$$

if the assumptions (2) hold. Multiplying this by D gives with $e_k := D\,|x_k - x_*|$ that

$$e_{k+1} \le e_k^2$$

If $e_0 \le q < 1$ we therefore obtain $e_1 \le q^2$, $e_2 \le q^4$, $e_3 \le q^8, \ldots$

$$e_k \le q^{(2^k)}$$

This means that the error converges to zero as $k \to \infty$, and we obtain the following theorem:

Theorem 2.3. *Assume that $f(x_*) = 0$ and*

- *$f'(x)$ and $f''(x)$ exist and are continuous near x_**
- *$f'(x_*) \neq 0$.*

Then there exists $\delta > 0$, $C > 0$ such that for $|x_0 - x_*| \le \delta$ we have

- $\lim_{k\to\infty} x_k = x_*$ *(convergence)*
- $|x_{k+1} - x_*| \le C\,|x_k - x_*|^2$ *(convergence of order 2)*

Proof. Exactly like the proof of Theorem 2.2. □

2.4 Local Convergence

Note that for both the secant and the Newton method we can only guarantee convergence if the initial guesses are sufficiently close to the solution x_*. This type of result is called "local convergence": If the initial guess is to far away we may not have convergence at all. If however the initial guess satisfies $|x_0 - x_*| \le \delta$ then the accuracy increases very quickly (doubling the number of correct digits with each step for the Newton method).

This is in contrast to the bisection method which is guaranteed to converge, no matter how large the initial interval $[a_0, b_0]$ is (but we need $f(a_0)f(b_0) < 0$). This type of result is called "global convergence".

Example 1: Consider the Newton method for the function $f(x) = \tan^{-1}(x)$ which has the zero $x_* = 0$.

For $|x_0| \le 1.39$ the Newton method converges to 0.

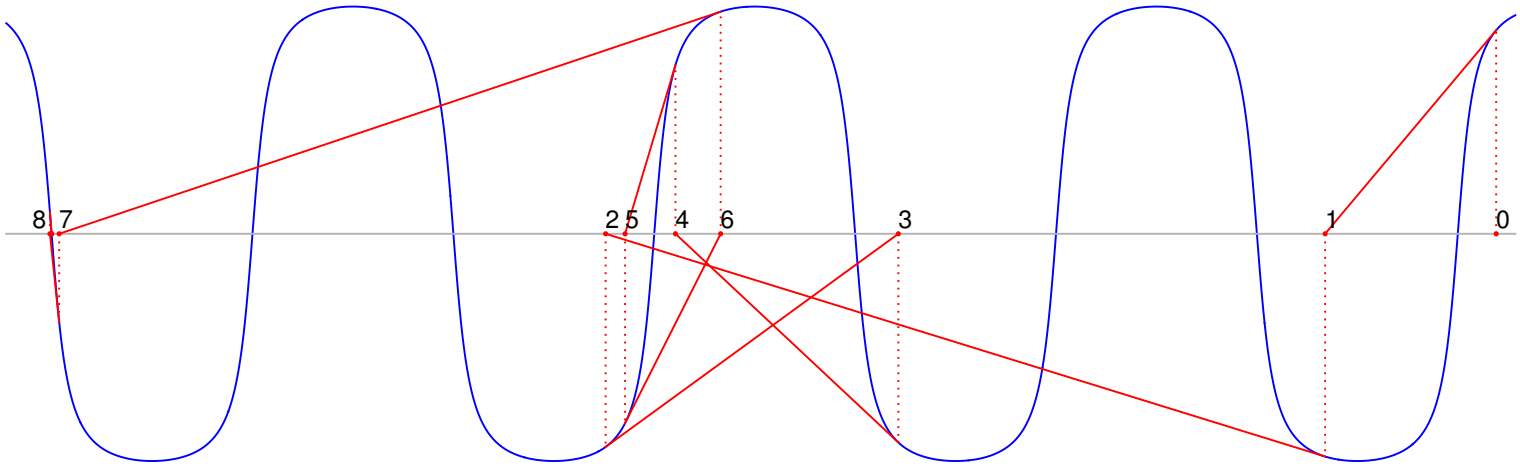
For $|x_0| \ge 1.4$ the Newton method diverges, with $|x_k| \to \infty$ as $k \to \infty$.

For $x_0 = 1.39174520027073\ldots$ the Newton method gives $x_1 = -x_0$, $x_2 = x_0$, $x_3 = -x_0, \ldots$

Example 2: Consider the Newton method for $f(x) = \tan^{-1}(5\sin x)$ which has zeros $k\pi$, $k \in \mathbb{Z}$.

For $x_0 = 0.6$ we obtain

k	x_k
1	-2.07458463997807
2	-13.3266896191295
3	-8.74903429240199
4	-12.2345381742364
5	-13.0229622940453
6	-11.5280491387632
7	-21.8729923205714
8	-22.0175237627052
9	-21.9908377144459
10	-21.9911485756392
11	-21.991148575128552669238503680 7



This illustrates a typical behavior for the Newton and secant methods: First the values jump around wildly. Then we hit "by chance" one of the δ-intervals around a zero. Now the superlinear convergence sets in, and we get full machine accuracy in about 4 iterations.

2.5 Hybrid methods and the Matlab command `fzero`

The bisection method has several advantages:

- it is guaranteed to converge, even if the function is not differentiable (only continuity is needed)
- it gives at each step an interval containing the zero x_* ("bracketing")

However, the bisection method converges very slowly (one new binary digit of accuracy with each step).

The Newton and secant method converge much faster ("superlinear convergence": the number of correct digits gets multiplied by $\alpha > 1$ with each step), however this only works if

- $f'(x)$ and $f''(x)$ exist and are continuous
- $f'(x_*) \neq 0$
- we are already close to the zero (δ in Theorem 2.2, Theorem 2.3).

If not all of these conditions are satisfied the Newton or secant method may be worse than the bisection method, or may not converge at all.

Therefore we would like to use a **hybrid method** which combines the secant method and the bisection method such that it combines the best properties of both methods:

- it is guaranteed to converge, even if the function is not differentiable (only continuity is needed)
- it gives at each step an interval containing the zero x_* ("bracketing")
- if f' and f'' are continuous and $f'(x_*) \neq 0$ we will have superlinear convergence

The **Matlab command `fzero`** uses a "hybrid method" which combines a secant-like method with the bisection method (it uses interpolation with a quadratic polynomial instead of a linear polynomial, "Brent algorithm").

- **`xs = fzero(f,[a,b])`** : give interval $[a,b]$ such that $f(a), f(b)$ have different sign. This will return a solution `xs` in this interval.
- **`xs = fzero(f,x0)`** : give initial guess x_0. This may return a solution xs far away from x_0, or may not converge at all.

Example: Find x such that $x^{10} - .01 = 0$.

```
>> f = @(x) x^10-.01
>> xs = fzero(f,[0,1])
xs =
         0.630957344480193
```

If the function is given in an m-file `f.m` we have to use `fzero(@f,[a,b])` .

We can specify options with the `optimset` command (use `doc fzero` in Matlab for details):

```
>> opt = optimset('Display','iter');   % show information about each iteration
>> xs = fzero(f,[0,1],opt)

 Func-count    x          f(x)             Procedure
    2              0         -0.01        initial
    3           0.01         -0.01        interpolation
    4          0.505   -0.00892127        bisection
    5          0.505   -0.00892127        bisection
    6       0.543642   -0.00774509        interpolation
    7       0.648071     0.0030685        bisection
    8       0.618438   -0.00181609        interpolation
    9       0.629455  -0.000235499        interpolation
   10        0.63098   3.59311e-06        interpolation
   11       0.630957  -3.86467e-08        interpolation
   12       0.630957  -6.24737e-12        interpolation
   13       0.630957             0        interpolation

Zero found in the interval [0, 1]
xs =
         0.630957344480193
```

We see that the algorithm uses three times a bisection step because the interpolation method does not make much progress.

3 Review of Taylor theorem

Tangent line approximation for a function $g(x)$

For a function $g(x)$ the first order Taylor polynomial $p(x)$ is the **tangent line** at the point $x^{(0)}$:

$$p(x) = g(x^{(0)}) + g'(x^{(0)})(x - x^{(0)}).$$

Taylor's theorem gives for the error $R(x) := g(x) - p(x)$

$$|R(x)| \leq c_2 \left| x - x^{(0)} \right|^2 \tag{4}$$

Here we need that g'' exists and is continuous on some interval B. If $|g''(t)| \leq M$ for $t \in B$ and $x^{(0)}, x \in B$ then (4) holds with $c_2 = \frac{1}{2}M$.

Tangent plane approximation for a function $g(x_1, x_2)$

For a function $g(x_1, x_2)$ the first order Taylor polynomial $p(x_1, x_2)$ is the **tangent plane** at the point $(x_1^{(0)}, x_2^{(0)})$:

$$p(x_1, x_2) = g(x_1^{(0)}, x_2^{(0)}) + \frac{\partial g}{\partial x_1}(x_1^{(0)}, x_2^{(0)}) \cdot (x_1 - x_1^{(0)}) + \frac{\partial g}{\partial x_2}(x_1^{(0)}, x_2^{(0)}) \cdot (x_2 - x_2^{(0)}).$$

We then have for the error $R(x_1, x_2) := g(x_1, x_2) - p(x_1, x_2)$

$$|R(x_1, x_2)| \leq c_2 \left\| x - x^{(0)} \right\|_\infty^2$$

Here we need that all first order partial derivatives $\frac{\partial g}{\partial x_1}, \frac{\partial g}{\partial x_2}$ and all second order partial derivatives $\frac{\partial^2 g}{\partial x_1^2}, \frac{\partial^2 g}{\partial x_1 \partial x_2}, \frac{\partial^2 g}{\partial x_2^2}$ exist and are continuous in some convex region B. If $\left| \frac{\partial^2 g}{\partial x_i \partial x_j}(t) \right| \leq M_{ij}$ for $t \in B$ and $x^{(0)}, x \in B$ then (4) holds with $c_2 = \frac{1}{2}(M_{11} + M_{12} + M_{21} + M_{22})$. (Note that $M_{12} = M_{21}$)

Linear approximation of a function $g(x_1, \ldots, x_n)$

For a function $g(x_1, \ldots, x_n)$ of n variables the first order approximation $p(x)$ at a point $x^{(0)}$ is given by

$$\begin{aligned} p(x) &= g(x^{(0)}) + \frac{\partial g}{\partial x_1}(x^{(0)}) \cdot (x_1 - x_1^{(0)}) + \cdots + \frac{\partial g}{\partial x_n}(x^{(0)}) \cdot (x_n - x_n^{(0)}) \\ &= g(x^{(0)}) + \left[\frac{\partial g}{\partial x_1}(x^{(0)}), \ldots, \frac{\partial g}{\partial x_n}(x^{(0)}) \right] (x - x^{(0)}) \end{aligned}$$

and we have for the error $R(x) := g(x) - p(x)$

$$|R(x_1, x_2)| \leq c_2 \left\| x - x^{(0)} \right\|_\infty^2$$

Here we need that all first order partial derivatives $\frac{\partial g}{\partial x_i}$ for $i = 1, \ldots, n$ and all second order partial derivatives $\frac{\partial^2 g}{\partial x_i \partial x_j}$ for $i, j = 1, \ldots, n$ exist and are continuous in some convex region B. If $\left| \frac{\partial^2 g}{\partial x_i \partial x_j}(t) \right| \leq M_{ij}$ for $t \in B$ and $x^{(0)}, x \in B$ then (4) holds with $c_2 = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n M_{ij}$. (Note that $M_{ij} = M_{ji}$)

4 Nonlinear system

We have n nonlinear equations $f_1(x_1,\ldots,x_n)=0,\ldots,f_n(x_1,\ldots,x_n)=0$. We define the vector-valued function $f(x)$ as

$$f(x)=\begin{bmatrix} f_1(x_1,\ldots,x_n) \\ \vdots \\ f_n(x_1,\ldots,x_n) \end{bmatrix}$$

The Jacobian $f'(x)$ (often denoted by $Df(x)$) is the $n\times n$ matrix of first partial derivatives

$$f'(x)=\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

Then Taylor's theorem for functions $g(x_1,\ldots,x_n)$ gives that

$$f(x)=\underbrace{f(x^{(0)})+f'(x^{(0)})(x-x^{(0)})}_{p(x)}+R(x)$$

We assume that the second order partial derivatives $\dfrac{\partial^2 f_i}{\partial x_j \partial x_k}(x)$ exist and are continous. Then the remainder term $R(x)=f(x)-p(x)$ satisfies

$$\|R(x)\|_\infty \le c_2 \left\|x-x^{(0)}\right\|_\infty^2$$

if we have bounds $\left|\frac{\partial^2 f_i}{\partial x_j \partial x_k}(x)\right| \le M_{ijk}$ for $i,j,k=1,\ldots,n$ and

$$c_2 := \frac{1}{2}\max_{i=1\ldots n}\left(\sum_{j=1}^{n}\sum_{j=1}^{n} M_{ijk}\right). \tag{5}$$

We start with an inital guess $x^{(0)}$ and compute $b:=f(x^{(0)})$ and $A:=f'(x^{(0)})$. Then we can approximate the function $f(x)$ by the Taylor approximation $p(x)=b+A(x-x^{(0)})$. Instead of $f(x)=\vec{0}$ we solve solve $p(x)=\vec{0}$ as follows: Let $d=x-x^{(0)}$, solve the linear system $Ad=-b$, then let $x^{(1)}:=x^{(0)}+d$.

Algorithm: Newton Method

- Initial guess $x^{(0)}$
- For $k=0,1,2,\ldots$:
  $b:=f(x^{(k)})$
  $A:=f'(x^{(k)})$
  solve $Ad=-b$ for d (use Gaussian elimination with pivoting)
  $x^{(k+1)}:=x^{(k)}+d$

Example

We want to solve the nonlinear system

$$\begin{aligned} 2x_1+x_1x_2&=2 \\ 2x_2-x_1x_2^2&=2 \end{aligned}$$

We therefore define $f(x)=\begin{bmatrix} 2x_1+x_1x_2-2 \\ 2x_2-x_1x_2^2-2 \end{bmatrix}$ and obtain the Jacobian matrix $f'(x)=\begin{bmatrix} 2+x_2, & x_1 \\ -x_2^2, & 2-2x_1x_2 \end{bmatrix}$.

We use the initial guess $x^{(0)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.

Step 1: $b = f(x^{(0)}) = \begin{bmatrix} -2 \\ -2 \end{bmatrix}, A = f'(x^{(0)}) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$. Solving $Ad = -b$ gives $d = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $x^{(1)} = x^{(0)} + d = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

Step 2: $b = f(x^{(1)}) = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, A = f'(x^{(1)}) = \begin{bmatrix} 3 & 1 \\ -1 & 0 \end{bmatrix}$. Solving $Ad = -b$ gives $d = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$ and $x^{(2)} = x^{(1)} + d = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$.

Step 3: $b = f(x^{(2)}) = \begin{bmatrix} -2 \\ 4 \end{bmatrix}, A = f'(x^{(2)}) = \begin{bmatrix} 5 & 0 \\ -9 & 2 \end{bmatrix}$. Solving $Ad = -b$ gives $d = \begin{bmatrix} .4 \\ -.2 \end{bmatrix}$ and $x^{(3)} = x^{(2)} + d = \begin{bmatrix} .4 \\ 2.8 \end{bmatrix}$.

We can perform the Newton method in Matlab:

```
f = @(x)  [ 2*x(1)+x(1)*x(2)-2; 2*x(2)-x(1)*x(2)^2-2 ]; % define the function f(x)
fp = @(x) [ 2+x(2), x(1) ; -x(2)^2, 2-2*x(1)*x(2) ];    % define the Jacobian f'(x)
x = [0;0];                                               % initial guess
for i=1:7
  b = f(x);
  A = fp(x);
  d = -A\b;
  x = x+d                                                % print out x
end
```

Running this gives the output

```
x =
     1
     1
x =
     0
     3
x =
                  0.4
                  2.8
x =
         0.483870967741935
          1.99354838709677
x =
          0.50009892401114
          1.99939860092483
x =
         0.499999985726356
          1.99999999518732
x =
                  0.5
                    2
```

All the following iterates are the same in machine precision. Here the exact solution is $x^* = \begin{bmatrix} \frac{1}{2} \\ 2 \end{bmatrix}$. We obtain for the errors

k	3	4	5	6	7
$\left\|x^{(k)} - x^*\right\|_\infty$	0.8	$1.6 \cdot 10^{-2}$	$6.0 \cdot 10^{-4}$	$1.4 \cdot 10^{-8}$	$2.2 \cdot 10^{-16}$

It seems that we have $\left\|x^{(k+1)} - x^*\right\|_\infty \leq C \left\|x^{(k)} - x^*\right\|_\infty^2$, i.e., convergence of order 2.

Errors of the Newton method

For $p(x) = f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)})$ Taylor's theorem gives for $x = x^*$

$$f(x^*) - p(x^*) = R(x^*)$$

Since $f(x^*) = \vec{0} = p(x^{(k+1)})$ we get

$$p(x^{(k+1)}) - p(x^*) = R(x^*)$$

From $p(x) = b + A(x - x^{(0)})$ we get $p(x^{(k+1)}) - p(x^*) = A\left(x^{(k+1)} - x^*\right)$ so that

$$\begin{aligned} x^{(k+1)} - x^* &= A^{-1}R(x^*) \\ \left\|x^{(k+1)} - x^*\right\|_\infty &\le \left\|A^{-1}\right\|_\infty \|R(x^*)\|_\infty \\ \left\|x^{(k+1)} - x^*\right\| &\le \left\|A^{-1}\right\|_\infty c_2 \left\|x^{(k)} - x^*\right\|_\infty^2 \end{aligned}$$

with c_2 from (5). If $\left\|A^{-1}\right\|_\infty \le c_1$ we obtain with $D := c_1 c_2$

$$\left\|x^{(k+1)} - x^*\right\|_\infty \le D \left\|x^{(k)} - x^*\right\|_\infty^2$$

Therefore we obtain the following theorem:

Theorem 4.1. *Assume that $f(x^*) = 0$ and*

- $\dfrac{\partial f_i}{\partial x_j}$ *and* $\dfrac{\partial^2 f_i}{\partial x_j \partial x_k}$ *exist and are continuous near x_* for $i, j, k = 1, \ldots, n$*
- *the matrix $f'(x^*)$ is nonsingular.*

Then there exists $\delta > 0$, $C > 0$ such that for $\left\|x^{(0)} - x_*\right\| \le \delta$ we have

- $\lim\limits_{k \to \infty} x^{(k)} = x^*$ *(convergence)*
- $\left\|x^{(k+1)} - x^*\right\| \le C \left\|x^{(k)} - x^*\right\|^2$ *(convergence of order 2)*

Proof. Since $f'(x^*)$ is nonsingular and $f'(x)$ is continuous, we can find $\varepsilon > 0$ such that on $B_\varepsilon := \{x \mid \|x - x^*\| \le \varepsilon\}$ we have

$$\left\|f'(x)^{-1}\right\| \le c_1.$$

We can then determine M_{ijk} such that $\left|\dfrac{\partial^2 f_i(x)}{\partial x_j \partial x_k}\right| \le M_{ijk}$ on B_ε. Define c_2 by (5) and $D := c_1 c_2$. Then we have for $x^{(k)} \in B_\varepsilon$ that $\left\|x^{(k+1)} - x^*\right\|_\infty \le D \left\|x^{(k)} - x^*\right\|_\infty^2$. Now we proceed exactly as in the proof of Theorem 2.3. □

5 Newton method with damping

The Newton method converges very quickly to a solution x^* if the intial error $x^{(0)} - x^*$ is sufficiently small. If the initial error $x^{(0)} - x^*$ is too large the Newton method can take us farther away from the solution, see the example with $n = 1$ and $f(x) = \tan^{-1} x$ in section 2.4.

In this example the Newton step d goes into the correct direction of decreasing absolute value$|f(x)|$, but the Newton step is too large and gives $x_{k+1} = x_k + d$ with $|f(x_{k+1})| > |f(x_k)|$. In such a case we should not accept the value $x_{k+1} = x_k + d$ and should rather try a smaller step $x_{k+1} = x_k + \alpha d$ with some $\alpha < 1$.

We obtain the following algorithm: "**Newton method with damping**"

$b := f(x^{(k)}), A := f'(x^{(k)})$
solve linear system $Ad = -b$
$\alpha := 1$
While $\left\|f(x^{(k)} + \alpha d)\right\| \geq \left\|f(x^{(k)}\right\|$:
    $\alpha := \alpha/2$
$x^{(k+1)} := x^{(k)} + \alpha d$

For $n = 1$ it is clear that the Newton direction $d = -f(x^{(k)})/f'(x^{(k)})$ points in the direction of decreasing $|f(x)|$, hence the while loop will terminate at some point.

We want to show that this works for any n and any choice of norm: Recall that the Newton direction d satisfies $f'(x^{(k)})d = -f(x^{(k)})$.

Claim: For $\alpha > 0$ sufficiently small we have $\left\|f(x^{(k)} + \alpha d)\right\| < \left\|f(x^{(k)}\right\|$

Proof: We use Taylor's theorem and get for $\alpha < 1$

$$f(x^{(k)} + \alpha d) = f(x^{(k)}) + \alpha \underbrace{f'(x^{(k)})d}_{-f(x^{(k)})} + R, \qquad \|R\| \leq c\|d\|^2 \tag{6}$$

$$\left\|f(x^{(k)} + \alpha d)\right\| = (1-\alpha)\left\|f(x^{(k)})\right\| + \|R\| \leq (1-\alpha)\left\|f(x^{(k)})\right\| + \alpha^2 c\|d\|^2 \tag{7}$$

The Newton method with damping should converge to a solution x^* even if $x^{(0)}$ is not extremely close to x^*, but it may still fail: **The following cases are possible:**

1. $x^{(k)}$ **converges to a solution** x^*: In this case we will have $\alpha_k = 1$ if $\left\|x^{(k)} - x^*\right\|$ is sufficiently small. Hence we will have $\alpha_k = 1$ for all steps $k \geq K$ and therefore convergence of order 2 (assuming $f'(x^*)$ is nonsingular).
   Proof: From (7) we get for $\alpha = 1$ with $A = f'(x^{(k)})^{-1}$ and $\left\|A^{-1}\right\| \leq c_1$

$$\left\|f(x^{(k)} + d)\right\| \leq c\|d\|^2 \leq c\left\|A^{-1}f(x^{(k)})\right\|^2 \leq \underbrace{cc_1^2\left\|f(x^{(k)})\right\|}_{<1} \cdot \left\|f(x^{(k)})\right\|$$

   since $\|f(x)\| < 1/(cc_1^2)$ if $\|x - x^*\|$ is sufficiently small.
2. $x^{(k)}$ **converges to local minimum** $\tilde{x}$ **of** $\|f(x)\|$ **with** $\|f(\tilde{x})\| > 0$**.** In this case $\alpha_k \to 0$. Therefore we should stop the algorithm if α_k gets too small. *Example:* $f(x) = x^2 + 1, x_0 = 1$.
3. $x^{(k)}$ **does not converge:** $x^{(k)}$ may be unbounded with decreasing values $\left\|f(x^{(k)}\right\| > 0$. *Example:* $f(x) = 1/x, x_0 = 1$.

6 Nonlinear least squares problem

We have N functions $f_1(x_1, \ldots, x_n), \ldots, f_N(x_1, \ldots, x_n)$ for n unknowns with $N > n$. We define the vector-valued function $f(x)$ as

$$f(x) = \begin{bmatrix} f_1(x_1, \ldots, x_n) \\ \vdots \\ f_N(x_1, \ldots, x_n) \end{bmatrix}$$

We cannot expect to find $x \in \mathbb{R}^n$ such that $f(x) = \vec{0}$ since we have more equations than unknowns. But we can try to find $x \in \mathbb{R}^n$ such that the vector $f(x)$ becomes "as small as possible":

$$\text{Find } x \in \mathbb{R}^n \text{ such that } \|f(x)\|_2 \text{ is minimal}$$

The Jacobian $f'(x)$ (often denoted by $Df(x)$) is the $N \times n$ matrix (more rows than columns) of first partial derivatives

$$f'(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_N}{\partial x_1} & \cdots & \frac{\partial f_N}{\partial x_n} \end{bmatrix}$$

We start with an inital guess $x^{(0)}$. Then we approximate the function $f(x)$ by the Taylor approximation $p(x) = b + A(x - x^{(0)})$ with $b := f(x^{(0)})$ and $A := f'(x^{(0)})$. Instead of $\|f(x)\| = \min$ we solve $\|p(x)\| = \min$ as follows: Let $d = x - x^{(0)}$, solve the linear least squares problem $\|Ad + b\| = \min$, then let $x^{(1)} := x^{(0)} + d$.

Algorithm: Gauss-Newton Method

- Initial guess $x^{(0)}$
- For $k = 0, 1, 2, \ldots$:
  $b := f(x^{(k)})$
  $A := f'(x^{(k)})$
  find d such that $\|Ad + b\|$ is minimal (use normal equations or QR decomposition)
  $x^{(k+1)} := x^{(k)} + d$

Convergence of the Gauss-Newton method: We assume that $F(x) := \|f(x)\|_2^2 = f_1(x)^2 + \cdots + f_N(x)^2$ has a local minimum at $x^* \in \mathbb{R}^n$. Therefore $\dfrac{\partial F}{\partial x_j}(x^*) = 0$ for $i = 1, \ldots, n$, i.e., with $A_* := f'(x^*)$ we have the normal equations

$$A_*^\top f(x^*) = \vec{0} \tag{8}$$

If our current approximation is $x^{(k)}$ we consider the Taylor approximation $p(x) = b + A(x - x^{(k)})$ with $b = f(x^{(k)})$ and $A = f'(x^{(k)})$. Then we determine $x^{(k+1)}$ such that $\left\|p(x^{(k+1)})\right\|_2$ is minimal, hence we have the normal equations

$$A^\top p(x^{(k+1)}) = \vec{0} \tag{9}$$

For the Taylor approximation we know that

$$f(x^*) - p(x^*) = r(x^*), \qquad \|r(x^*)\| \le C_2 \left\|x^* - x^{(k)}\right\|^2 \tag{10}$$

where C_2 depends on the size of the second order partial derivatives $\dfrac{\partial^2 f_i}{\partial x_j \partial x_k}$. We also have

$$\|A_* - A\| = \left\|f'(x^*) - f'(x^{(k)})\right\| \le C_2 \left\|x^* - x^{(k)}\right\|$$

From (10) we obtain

$$A^\top f(x^*) - A^\top p(x^*) = A^\top r(x^*)$$

Now (8), (9) give for the first term

$$\begin{aligned} A^\top f(x^*) = & \underbrace{A_*^\top f(x^*)}_{0} + (A - A_*)^\top f(x^*) \\ & = \overbrace{A^\top p(x^{(k+1)})} + (A - A_*)^\top f(x^*) \end{aligned}$$

yielding

$$\underbrace{A^\top \Big(p(x^{(k+1)}) - p(x^*)\Big)}_{A\left(x^{(k+1)} - x^*\right)} = (A_* - A)^\top f(x^*) + A^\top r(x^*)$$

and with the matrix $M := A^\top A$

$$x^{(k+1)} - x^* = M^{-1}(A_* - A)^\top f(x^*) + M^{-1} A^\top r(x^*)$$

$$\left\|x^{(k+1)} - x^*\right\| \le \left\|M^{-1}\right\| \left(C_2 \left\|x^{(k)} - x^*\right\| \|f(x^*)\| + \left\|A^\top\right\| C_2 \left\|x^{(k)} - x^*\right\|^2\right)$$

$$\left\|x^{(k+1)} - x^*\right\| \le D \left(c \|f(x^*)\| \cdot \left\|x^{(k)} - x^*\right\| + \left\|x^{(k)} - x^*\right\|^2\right)$$

where D is a bound for $C_2 \left\|M^{-1}\right\| \left\|A^{\top}\right\|$. If the residual $\|f(x^*)\|$ is zero (usually not satisfied) we get quadratic convergence. If $\varepsilon := c\,\|f(x^*)\|$ is small the error $\left\|x^{(k)} - x^*\right\|$ will at first decrease as with quadratic convergence, until $\left\|x^{(k)} - x^*\right\| \approx \varepsilon$. From then on we will **only have convergence of order 1** (if the residual $\|f(x^*)\|$ is sufficiently small). If the residual $\|f(x^*)\|$ is too large the **Gauss-Newton method may not be locally convergent**.