Chapter 2
Eigenvalues, nonlinear equations, and optimization problems

Numerical Methods for Visual Computing
WS 18/19

Michael Moeller
Jonas Geiping
Visual Scene Analysis
Department of Computer Science
and Electrical Engineering
University of Siegen

Michael Moeller
Jonas Geiping

Eigenvalues

Michael Moeller
Jonas Geiping

Eigenvectors and eigenvalues

Eigenvector and eigenvalue

Let $A \in \mathbb{R}^{n \times n}$. We call $\lambda \in \mathbb{R}$ a (real) **eigenvalue** of A, if there exists a $0 \neq v \in \mathbb{R}^n$ such that

$$Av = \lambda v.$$

Any $v \neq 0$ that satisfies the above equation is called an **eigenvector** of A to the eigenvalue λ.

Why should one be interested in eigenvalues?

- We have seen, that eigenvalues are fundamental to the behavior of linear systems, e.g. the *condition* of a matrix A.
- Eigenvalues are often crucial for the choice of certain algorithm parameters.
- Eigenvalues and eigenvectors are important far beyond numerical algorithms. Disrespecting them can lead to quite catastrophic behavior!

`https://www.youtube.com/watch?v=xcjv_vQJO7U`

Catastrophic resonance

A short comment about what happened here: Many systems in physics behave approximately like an harmonic oscillator, which is characterized by

$$\frac{\partial^2 x}{\partial t^2}(t) = -\omega^2 \cdot x(t).$$

Assume the second derivative is somewhat like a multiplication with a matrix, then $-\omega^2$ would be the eigenvalue of this matrix.

Assuming $x(0) = x(2\pi) = 0$ one can easily show that

$$x(t) = \sin(\omega t)$$

solves the above equation.

Catastrophic resonance

Assume we apply an additional external force $F(t)$ that excites in the same frequency as the 'eigenvalue' of the second derivative ω, i.e.

$$\frac{\partial^2 x}{\partial t^2}(t) = -\omega^2 \cdot x(t) + F(t)$$

with $F(t) = \cos(\omega x)$,then the solution becomes

$$x(t) = A(t)\sin(\omega t)$$

with a **linearly increasing** amplitude $A(t) = \frac{t}{2\omega}$.

We are unrealistically ignoring any damping here, but the effect can be observed even in real life:
`https://www.youtube.com/watch?v=ePHn-w5Gia0`
`https://www.youtube.com/watch?time_continue=68&v=sH7XSX10QkM`

Michael Moeller
Jonas Geiping

Eigenvalues – a reminder

How can we compute pairs (λ, v) with

$$Av = \lambda v, \qquad v \neq 0?$$

The **determinant** is a mapping that assigns a number to each matrix $A \in \mathbb{R}^{n \times n}$. It has many interesting properties, one of them being that

$$\det(A) \neq 0 \;\Leftrightarrow\; A \text{ is invertible.}$$

This immediately yields that $0 \neq v$ is an eigenvector to A with eigenvalue λ, if

$$\det(A - \lambda I) = 0.$$

Since the determinant is given by

$$\det(A) = \sum_{\sigma \in P_n} \operatorname{sign}(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)}$$

(for P_n being the set of all permutations of $\{1, \ldots, n\}$), we can see that $p(\lambda) = \det(A - \lambda I)$ is a polynomial of degree n in λ.

Eigenvalues – a reminder

The quantity $p(\lambda) = \det(A - \lambda I)$ is called the **characteristic polynomial** of A. Roots of $p(\lambda)$ are eigenvalues of A.

In $\mathbb{R}$ not every polynomial of degree n has n roots, such that we cannot expect to find n real eigenvalues in general.

A matrix $A \in \mathbb{R}^{n \times n}$ is called **diagonalizable** if there exists an invertible matrix U as well as a diagonal matrix Σ such that

$$A = U \Sigma U^{-1}.$$

Relation between eigenvectors and diagonalization

If A is diagonalizable then the column $U_{:,i}$ is an eigenvector of A to the eigenvalue $(\Sigma)_{ii}$.

Diagonalizable - a reminder

We do not want to work over $\mathbb{C}$, and do not want to deal with the somewhat complicated issues of being diagonalizable or not. Hence, we turn to symmetric matrices $A \in \mathbb{R}^{n \times n}$.

Symmetric matrices $A \in \mathbb{R}^{n \times n}$ (i.e. those matrices for which $A = A^T$) can always be diagonalized with an orthogonal basis of eigenvectors, i.e.,

$$A = U \Sigma U^T$$

for a diagonal Σ and $UU^T = U^T U = I$.

From now on let us consider the (simpler) problem of determining eigenvalues of a symmetric matrix $A \in \mathbb{R}^{n \times n}$.

Determining the roots of the polynomial $p(\lambda) = \det(A - \lambda I)$ is not easy! Is there a way to work with the matrix A directly?

Determining eigenvalues

Let us first consider the problem of determining the eigenvalue of largest magnitude, i.e., finding the eigenvalue $\lambda_i = \Sigma_{ii}$ such that $|\lambda_i| = \max_j |\lambda_j|$.

Interestingly, it holds

$$A = U\Sigma U^T = \underbrace{UP}_{=:\tilde{U}} \underbrace{P^T\Sigma P}_{=:\tilde{\Sigma}} \underbrace{P^T U^T}_{=:\tilde{U}^T}$$

for any permutation matrix P, which allows us to reorder the values of Σ.Thus, without restriction of generality,

$$|\Sigma_{11}| \geq |\Sigma_{22}| \geq \ldots \geq |\Sigma_{nn}|.$$

The matrix U is invertible, and it's columns are eigenvectors of A. Let us abbreviate $u_i = U_{i,:}$ and represent an arbitrary v^0 in the basis formed by the u_i,

$$v^0 = \sum_{i=1}^{n} \alpha_i u_i.$$

Determining eigenvalues

For

$$v^0 = \sum_{i=1}^{n} \alpha_i u_i$$

we find

$$Av^0 = \sum_{i=1}^{n} \alpha_i A u_i = \sum_{i=1}^{n} \lambda_i \alpha_i u_i = \lambda_1 \sum_{i=1}^{n} \frac{\lambda_i}{\lambda_1} \alpha_i u_i.$$

Therefore

$$A^k v^0 = (\lambda_1)^k \sum_{i=1}^{n} \underbrace{\left(\frac{\lambda_i}{\lambda_1}\right)^k}_{=:\nu_i^k} \alpha_i u_i.$$

Interesting: If $|\lambda_i| < |\lambda_1|$ then

$$\lim_{k \to \infty} \nu_i^k = 0.$$

Determining eigenvalues

If we assume $|\lambda_1| > |\lambda_2| \geq \ldots$, then

$$A^k v^0 = (\lambda_1)^k \sum_{i=1}^{n} \left(\frac{\lambda_i}{\lambda_1}\right)^k \alpha_i u_i = \alpha_1 (\lambda_1)^k \left(u_1 + \sum_{i=2}^{n} \left(\frac{\lambda_i}{\lambda_1}\right)^k \frac{\alpha_i}{\alpha_1} u_i \right).$$

looks promising already. Only problem $\|A^k v^0\|$ goes to ∞ if $|\lambda_1| > 1$ and to 0 if $|\lambda_1| < 1$.

Idea: Normalize

$$\tilde{v}^k := A^k v^0, \quad v^k = \frac{\tilde{v}^k}{\langle v^0, \tilde{v}^k \rangle}.$$

A short computation shows

$$\langle v^0, \tilde{v}^k \rangle = \alpha_1 (\lambda_1)^k \left(1 + \sum_{i=2}^{n} \left(\frac{\lambda_i}{\lambda_1}\right)^k (\alpha_i)^2 \right)$$

Michael Moeller
Jonas Geiping

Determining eigenvalues

We find

$$v^k = \frac{\tilde{v}^k}{\langle v^0, \tilde{v}^k \rangle} = \frac{1}{1 + \sum_{i=2}^{n} \left(\frac{\lambda_i}{\lambda_1}\right)^k (\alpha_i)^2} \left(u_1 + \sum_{i=2}^{n} \left(\frac{\lambda_i}{\lambda_1}\right)^k \frac{\alpha_i}{\alpha_1} u_i \right)$$

Since $\left(\frac{\lambda_i}{\lambda_1}\right)^k \to 0$ for $k \to \infty$, we conclude $v^k \to u_1$.

Furthermore

$$\alpha_k = \frac{\langle v^{k+1}, v^0 \rangle}{\langle v^k, v^0 \rangle} \overset{k\to\infty}{\to} \lambda_1.$$

Power method

Power method. For $k = 1, \ldots, n_{\max}$ compute

- $\tilde{v}^k = Av^{k-1}$
- $v^k = \frac{\tilde{v}^k}{\langle v^0, \tilde{v}^k \rangle}$

Return $u_1^{approx} = \frac{v^{n_{\max}}}{\| v^{n_{\max}} \|_2}$ and $\lambda_1^{approx} = \frac{\langle \tilde{v}^{n_{\max}}, v^0 \rangle}{\langle v^{n_{\max}-1}, v^0 \rangle}$.

For the above method to work we need

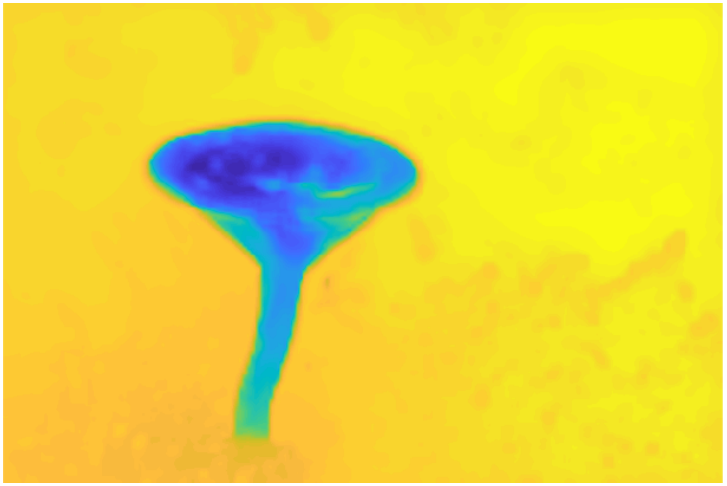
- A to have a basis of eigenvectors, e.g. A symmetric.
- $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$
- $\alpha_1 \neq 0$.

Quiz: How do we determine the smallest eigenvalue of a positive semi-definite matrix A?

How do we determine the eigenvalue of smallest magnitude of a symmetric invertible matrix A?

Application in image clustering

Image with segmentation



Second eigenvec.

General overview, see e.g. `https://www.cs.cmu.edu/~aarti/Class/10701/readings/Luxburg06_TR.pdf`.

Example: Spectral image clustering

Spectral clustering methods for images, e.g. normalized cuts[1]

- Build a graph with weights W that encode how similar pixel i and pixel j are
- Try to cluster the 'most similar' pixels by finding some kind of 'minimal cut' through the edges of the graph
- To avoid cutting a single edge, normalize your costs
- One can show that a relaxation of the resulting problem is given by the second largest eigenvector of the similarity matrix W (assuming that it's rows sum to one).

[1] http://repository.upenn.edu/cgi/viewcontent.cgi?article=1101&context=cis_papers

Example: Spectral image clustering

Original image

Part 1

Part 2

Modified image

All eigenvectors?

How can we compute more than the eigenvalue of largest magnitude?

Assume we know λ_1, u_1, $\|u_1\| = 1$, with $Au_1 = \lambda_1 u_1$ already. Simple idea

$$\tilde{v}^k = Av^{k-1} - \langle Av^{k-1}, u_1 \rangle u_1$$

and continue as usual.

Note that this step leads to $\tilde{v}^k \perp u_1$ for all k.

Repeating our previous computation (assuming $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$) one can indeed show convergence to λ_2 and a corresponding eigenvector.

All eigenvectors?

Natural extension for the third eigenvalue:

$$\tilde{v}^k = Av^{k-1} - \langle Av^{k-1}, u_1\rangle u_1 - \langle Av^{k-1}, u_2\rangle u_2.$$

Annoying: Very sequential, and eigenvectors need to be known beforehand to high accuracy.

Can we compute all eigenvectors of a (symmetric) matrix A 'on-the-fly', i.e. in a single algorithm?

Start with some $V^0 = (v_1, \ldots, v_n)$.

Compute $\tilde{V}^1 = AV^0$

Normalize the first column: $v_1^1 = \frac{1}{r_{11}}\tilde{v}_1^1$ for $r_{11} = \|\tilde{v}_1^1\|$.

Compute $v_2^1 = \frac{1}{r_{22}}(\tilde{v}_2^1 - r_{12}v_1^1)$ for $r_{12} = \langle \tilde{v}_2^1, v_1^1\rangle$ and $r_{22} = \|\tilde{v}_2^1 - r_{12}v_1^1\|$ to achieve that

$$\|v_2^1\| = 1, \qquad v_2^1 \perp v_1^1.$$

Michael Moeller
Jonas Geiping

All eigenvectors

We can continue the orthogonalization:

Assume we have $v_1^1, \ldots, v_l^1$ that all satisfy $\|v_i^1\| = 1$ and $\langle v_i^1, v_j^1 \rangle = 0$ for $i \neq j$.

Compute

$$v_{l+1}^1 = \frac{1}{r_{l+1\,l+1}} \left(\tilde{v}_{l+1}^1 - \sum_{i=1}^{l} r_{il+1} v_i^1 \right)$$

for $r_{il+1} = \langle v_i^1, \tilde{v}_{l+1}^1 \rangle$, and $r_{l+1\,l+1}$ such that $\|v_{l+1}^1\| = 1$.

Exciting: We end up with an orthogonal matrix V^1!

Even more exciting: Similar to the LR-decomposition, one can show that the entries r_{ij} from the above algorithm actually form an upper triangular matrix R^1 in such a way that

$$\tilde{V}^1 = V^1 R^1$$

Such a decomposition of a matrix $\tilde{V}$ into the product of an orthogonal matrix and an upper triangular matrix is called **QR-decomposition**. The above method to compute it is called **Gram-Schmidt process**.

All eigenvectors

We already have

$$\tilde{V}^1 = V^1 R^1,$$

but also know that $\tilde{V}^1$ was generated by multiplying our initial set of vectors V^0 with A. Thus,

$$AV^0 = V^1 R^1.$$

Iteratively, given V^k we compute the QR-decomposition of $AV^k = V^{k+1} R^{k+1}$ and expect that V^k converges to the matrix of eigenvectors of A. Therefore,

$$\Sigma^k := (V^k)^T A V^k$$

should converge to a diagonal matrix with the eigenvalues on the diagonal.

All eigenvectors!

We can rewrite the algorithm one more time to arrive at the most common form. Note that

$$\Sigma^k := (V^k)^T A V^k = (V^k)^T V^{k+1} R^{k+1}$$

is a QR-decomposition of Σ^k with $Q^{k+1} = (V^k)^T V^{k+1}$. If one requires the diagonal elements of R to be positive, one can show uniqueness of the QR-decomposition, and one can compute it with the Gram-Schmidt procedure as before.

We can then notice that

$$\begin{aligned}
\Sigma^{k+1} &= (V^{k+1})^T A V^{k+1} \\
&= (V^{k+1})^T A V^k (V^k)^T V^{k+1} \\
&= (V^{k+1})^T V^{k+1} R^{k+1} (V^k)^T V^{k+1} \\
&= R^{k+1} (V^k)^T V^{k+1} \\
&= R^{k+1} Q^{k+1}.
\end{aligned}$$

Therefore, the algorithm becomes: Compute $A^k = QR$, set $A^{k+1} = RQ$.

When and how does it work?

Convergence

Consider the QR-algorithm of setting $A^0 = A$ and for $A^k = QR$ being the QR-decomposition of A^k, set $A^{k+1} = RQ$.

If the eigenvalues λ_i of A satisfy

$$|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|,$$

then

- the diagonal of A^k converges to the eigenvalues of A,
- the lower left of A^k converges to zero,
- for symmetric matrices A the upper right also converges to zero. For general matrices, no claim about the convergence of the upper right can be made.

Things I did not tell you

For an 'industrial' implementation:

- The Gram-Schmidt method I presented is not stable.
- Either a modification or a different algorithm ('Householder') are used to compute the QR-decomposition.
- One does not apply the QR-algorithm to the matrix A itself, but rather to a matrix that is similar to A but allows more efficient decompositions.
- You may ignore these things in the exercise.

Michael Moeller
Jonas Geiping

Fun facts / motivation

- The QR-algorithm was invented and published in 1961 by the student John Francis.
- Francis never got a degree, left the field of numerical analysis, and became a system engineer. Did not keep in contact with any of his fellow students, or professors.
- Gene Goloub spend a lot of afford to find Francis. When he managed to do so in 2007(!) "*John was entirely unaware and quite amazed that there had been many references to and extensions of his early work and that his QR algorithm was considered one of the 10 most important algorithms of the twentieth century.*"[2]
- Francis has retired and was studying for a degree in mathematics in 2009.

[2] http://www.dm.unibo.it/~guerrini/html/an_09_10/QR_50_years_later.pdf

Nonlinear equations

Nonlinear equations

We understood how we can systematically solve the (nonlinear) equation

$$(A - \lambda I)v = 0$$

for λ and v simultaneously.

The above problem has a lot of structure and we can exploit many results from linear algebra.

What do we do for general nonlinear equations, i.e., finding a $u \in \mathbb{R}^n$ that satisfies

$$g(u) = 0$$

for some nonlinear function $g : \mathbb{R}^n \to \mathbb{R}^n$?

Newton method

Newton's method in 1d

Let us start with $g : \mathbb{R} \to \mathbb{R}$, and looking for a $u \in \mathbb{R}$ such that

$$g(u) = 0.$$

Similar to the iterative methods for linear equations the easiest approach is to set up a **fixed-point iteration**, i.e., define a function $G : \mathbb{R} \to \mathbb{R}$ is such a way that

$$G(u) = u \quad \Leftrightarrow \quad g(u) = 0.$$

One then iterates

$$u^{k+1} = G(u^k).$$

If the above scheme converges to some $\hat{u}$ (and G is continuous), one has $\hat{u} = G(\hat{u})$, and hence $\hat{u}$ also solves $g(\hat{u}) = 0$.

Michael Moeller
Jonas Geiping

Newton's method in 1d

The most famous way to show convergence of iteration of the form

$$u^{k+1} = G(u^k) \tag{FP}$$

is by **Banachs-fixed point theorem**.

In a nutshell Banachs-fixed point theorem says that (FP) is guaranteed to converge if $|G'(u)| \leq c < 1$ holds for all $u \in \mathbb{R}$.

One can also restrict G to a closed subset of $M \subset \mathbb{R}$ as long as G is a *self-map*, i.e., $G : M \to M$.

While the simplest choice is $G(u) = u - g(u)$, the choice $G(u) = u - g(u) \cdot h(u)$ works equally well as long as $h(u)$ is nonzero for all u.

A reasonable choice for h can be motivated geometrically using **Taylor's expansion**.

Taylor polynomial

Taylor's theorem

If a function $g : \mathbb{R} \to \mathbb{R}$ is s-times differentiable at a point v, then there exists a function $r_s : \mathbb{R} \to \mathbb{R}$ such that

$$g(u) = \underbrace{\left(\sum_{j=0}^{s} \frac{g^{(j)}(v)}{j!} (u-v)^j \right)}_{=:p_v^s(u) \text{ Taylor approximation}} + r_s(u)(u-v)^s,$$

and $\lim_{u \to v} r_s(u) = 0$.

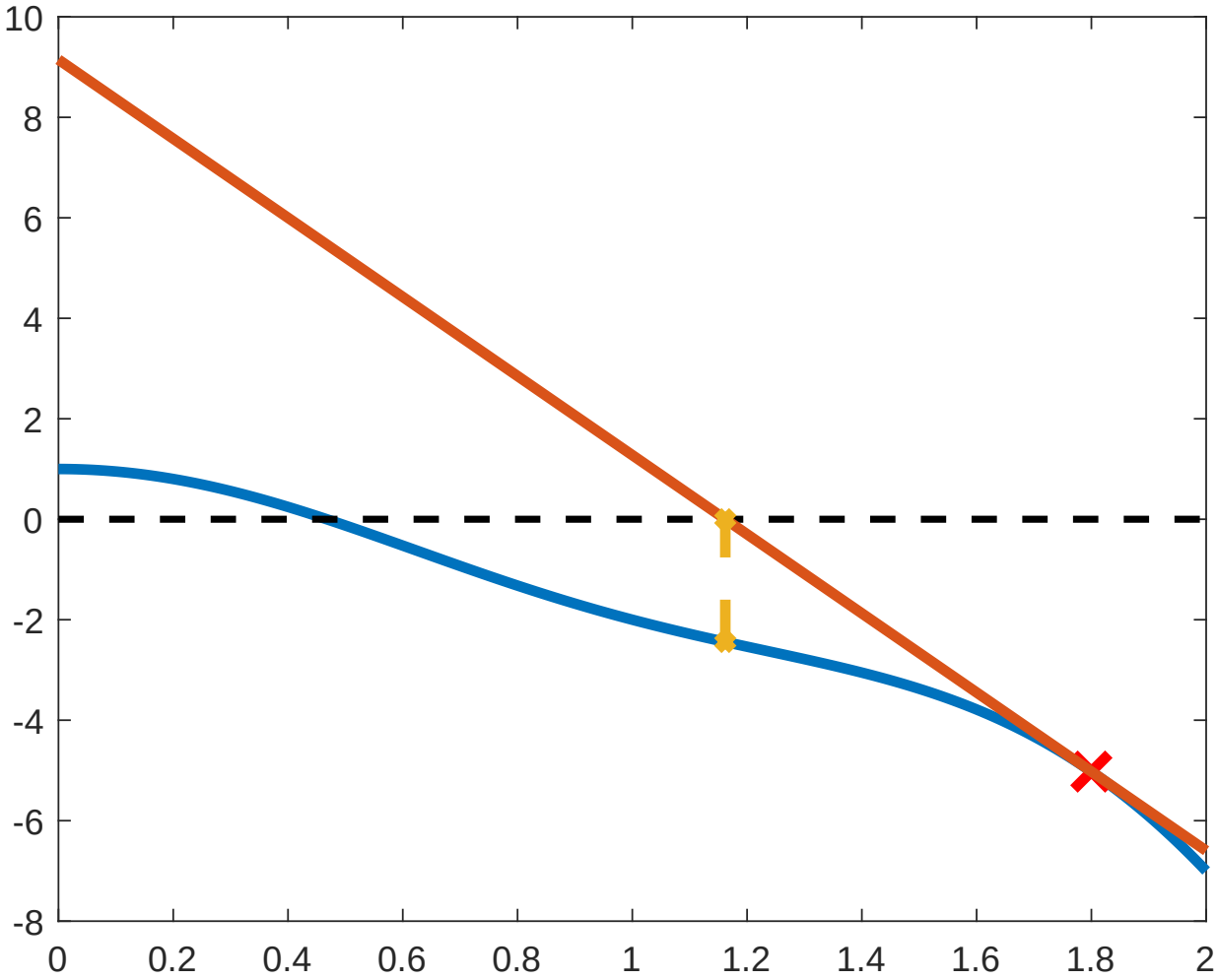
Idea for solving $g(u) = 0$: Iteratively solve

$$p_{u^k}^2(u) = g(u^k) + g'(u^k)(u - u^k) = 0,$$

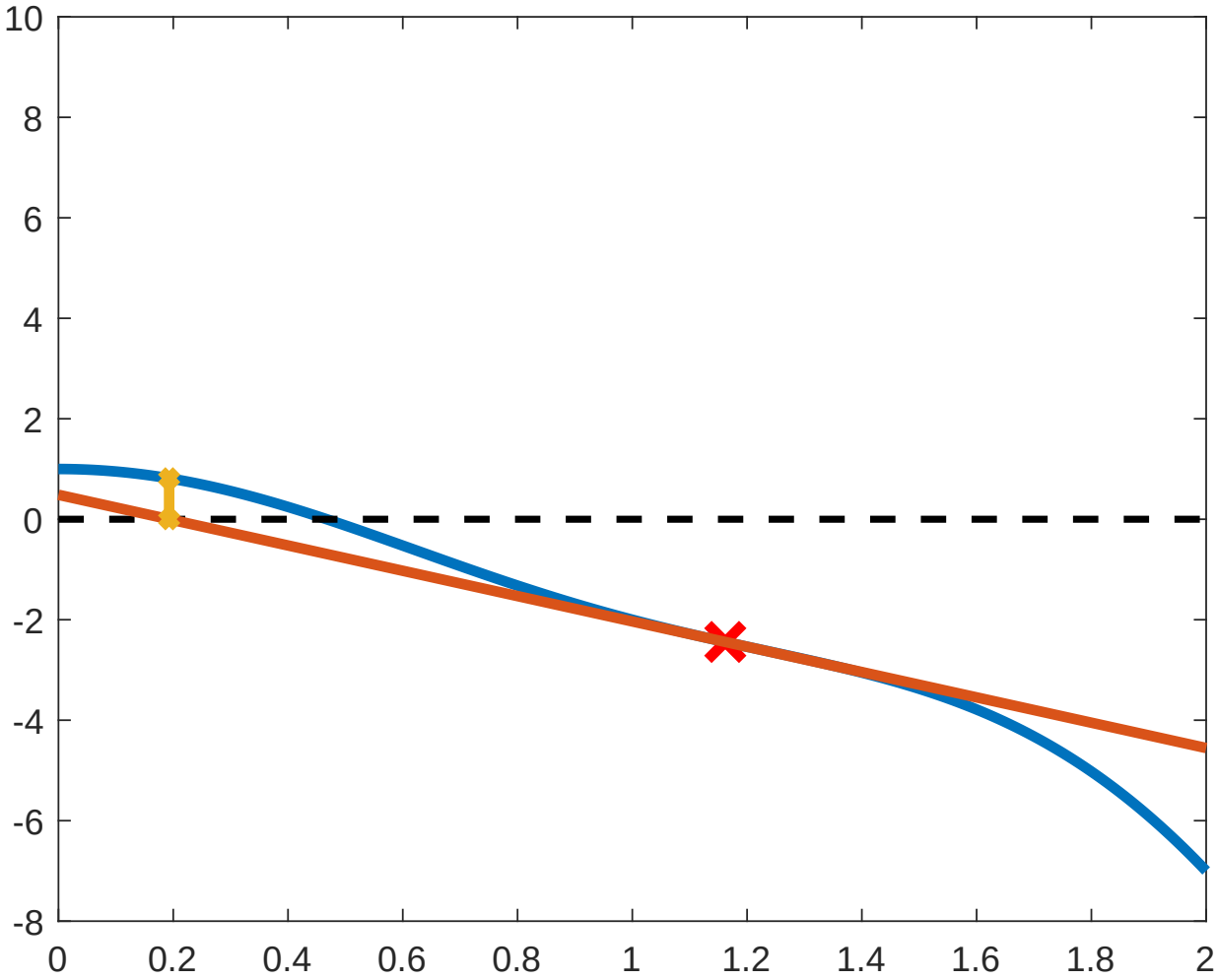
i.e., update

$$u = u^k - \frac{g(u^k)}{g'(u^k)}.$$

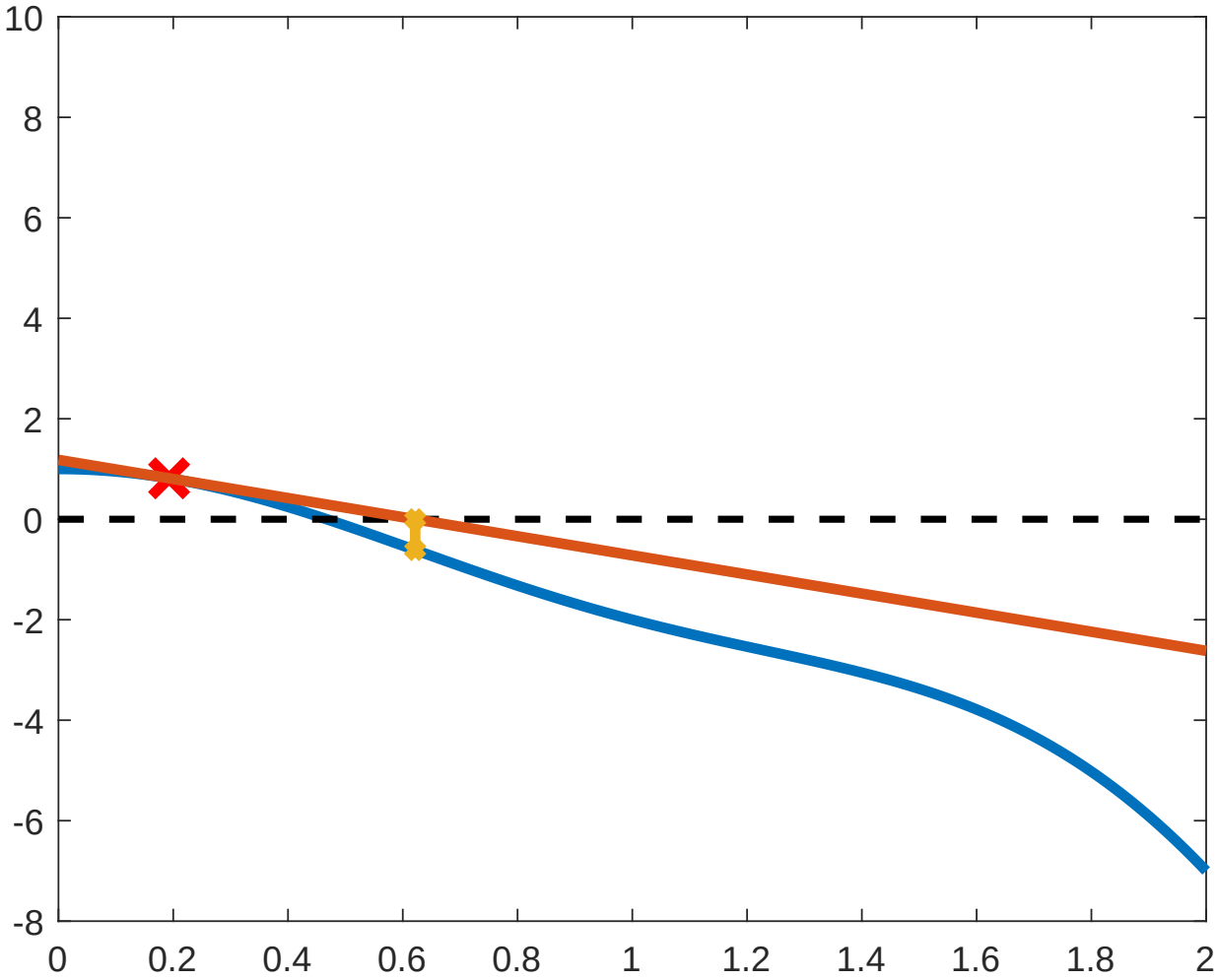
Newton's method



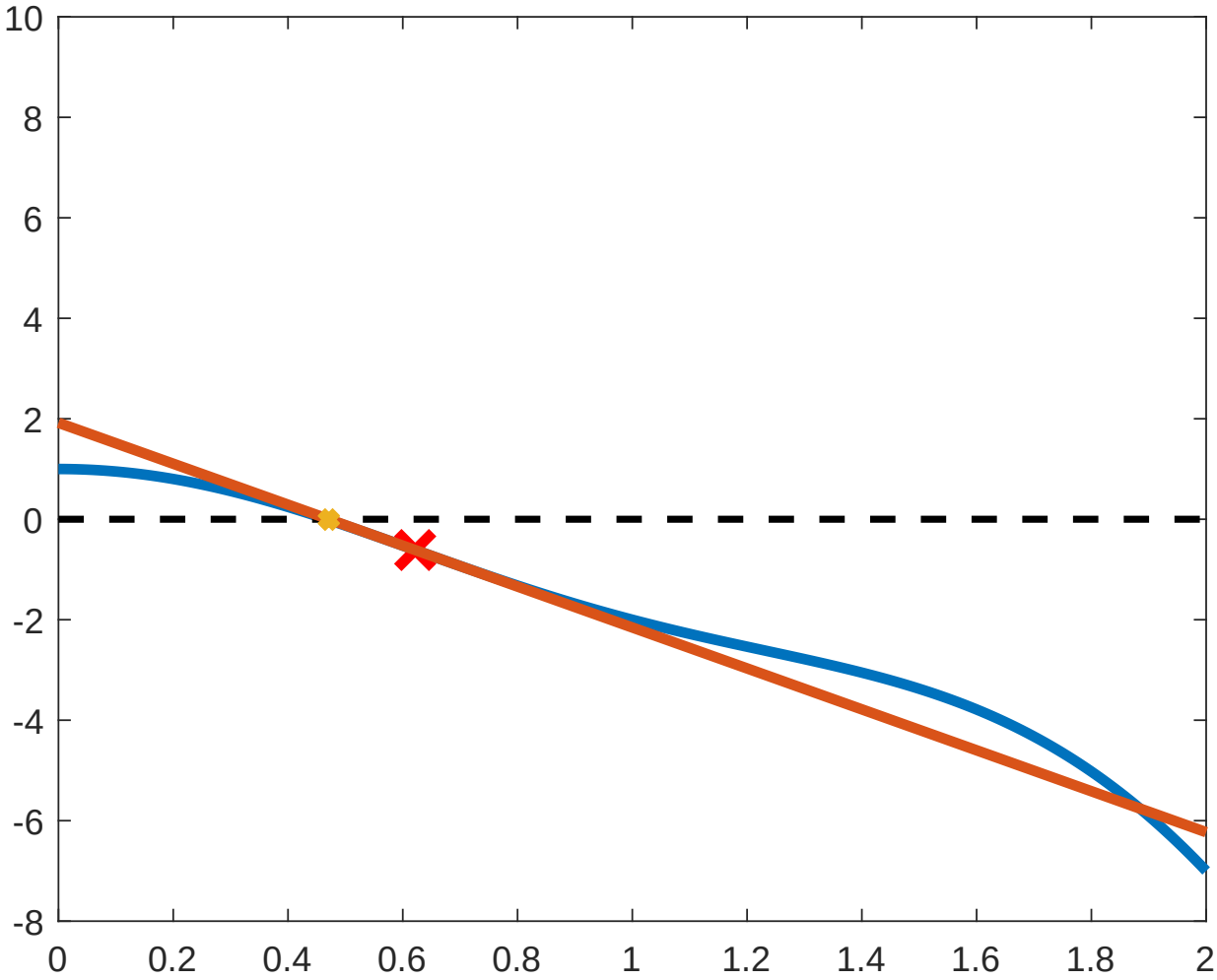
Newton's method



Newton's method



Newton's method



Newton's method

In terms of fixed-point iterations:

$$u^{k+1} = G(u^k) \quad \text{for} \quad G(u) = u - \frac{g(u)}{g'(u)}.$$

According to Banachs Fixed-Point theorem this works if $|G'(u)| \leq c < 1$ for all u.

If g is two times continuously differentiable we find

$$G'(u) = 1 - \frac{g'(u)\, g'(u) - g(u)\, g''(u)}{(g'(u))^2} = \frac{g(u)\, g''(u)}{(g'(u))^2}.$$

If $\hat{u}$ meets $g(\hat{u}) = 0$ and $g'(\hat{u}) \neq 0$, then we see that $G'(u) = 0$.

Thus, $|G'(u)| \leq c < 1$ holds in a small neighborhood of $\hat{u}$ from which one can indeed prove **local convergence**, i.e., if one starts 'close enough' to a solution, the method converges.

Newton's method – convergence

In some nice cases the method converges globally, e.g. Newton's method for computing the root $\sqrt{a}$ of a positive number $a > 0$ by solving

$$g(u) = u^2 - a = 0.$$

This is a proof you will do on this week's exercise sheet.

If g is two times continuously differentiable, and $g'(\hat{u}) \neq 0$ for $g(\hat{u}) = 0$, then Newton's method not only converges locally around $\hat{u}$, but it also yields a very fast, so-called **quadratic** convergence:

$$\|u^{k+1} - \hat{u}\| \leq c \cdot \|u^k - \hat{u}\|^2$$

for some constant c.

In general, we cannot expect more than local convergence. See Matlab code example for divergence.

Newton's method in multiple dimensions

Solving 1d problems only is a little boring, but the idea of Newton's method is straight forward to generalize to problems

$$g(u) = 0$$

for $g : \mathbb{R}^n \to \mathbb{R}^n$!

Do do so, we to recall/introduce some concepts of derivatives and Taylor expansions for functions $g : \mathbb{R}^n \to \mathbb{R}^n$.

Derivatives multiple dimensions

A vector valued function $g : \mathbb{R}^n \to \mathbb{R}^n$ gets a variable $u = (u_1, \ldots, u_n)^T$ as an input, and consists of n-many functions $g_i : \mathbb{R}^n \to \mathbb{R}$, i.e.

$$g(u) = \begin{pmatrix} g_1(u) \\ \cdots \\ \cdots \\ g_n(u) \end{pmatrix}.$$

The **partial derivative** $\frac{\partial h}{\partial u_i}$ of a function $h : \mathbb{R}^n \to \mathbb{R}$ with respect to u_i is the usual derivative of the function $\tilde{h} : \mathbb{R} \to \mathbb{R}$ which treats all variables u_j except u_i as constants.

Example: $h(u_1, u_2) = u_1^2 u_2 + u_2 - \cos(u_2 u_1)$ yields

$$\frac{\partial h}{\partial u_1}(u_1, u_2) = 2u_1 u_2 + u_2 \sin(u_2 u_1)$$
$$\frac{\partial h}{\partial u_2}(u_1, u_2) = u_1^2 + 1 + u_1 \sin(u_2 u_1)$$

Derivatives multiple dimensions

If for $g : \mathbb{R}^n \to \mathbb{R}^n$ all partial derivatives exist and are continuous, then the **Jacobi matrix**

$$Jg(u) = \begin{pmatrix} \frac{\partial g_1}{\partial u_1}(u) & \dots & \frac{\partial g_1}{\partial u_n}(u) \\ \dots & \dots & \dots \\ \dots & \dots & \dots \\ \frac{\partial g_n}{\partial u_1}(u) & \dots & \frac{\partial g_n}{\partial u_n}(u) \end{pmatrix}$$

is the natural generalization of the derivative to functions $g : \mathbb{R}^n \to \mathbb{R}^n$.

In most cases the Jacobian behaves similar to the usual derivative of scalar-valued functions. In particular, one can consider a first order **Taylor approximation** of a continuously differentiable function $g : \mathbb{R}^n \to \mathbb{R}^n$ via

$$g_v^{approx}(u) = g(v) + Jg(v)(u - v).$$

Newton's method in n dimensions

Following the idea's of the 1d Newton method, we iteratively compute new iterates via

$$g_{u^k}^{approx}(u^{k+1}) = 0,$$

which yields **Newton's method**

$$u^{u+1} = u^k - (Jg(u^k))^{-1}\, g(u^k). \qquad \text{(NM)}$$

Obviously, we need $Jg(u^k)$ to be invertible in each iteration.

The **convergence properties** are similar to the one-dimensional case, i.e., if $\hat{u}$ is a point where $g(\hat{u}) = 0$ and $Jg(\hat{u})$ is invertible, then Newton's method converges locally quadratic. Again, we cannot expect global convergence except in very particular cases.

Michael Moeller
Jonas Geiping

Optimization problems

Optimality conditions

A different problem, with yet very similar methodologies:
Minimize some energy $E : \mathbb{R}^n \to \mathbb{R}$, i.e., find an element $\hat{u}$ such that

$$E(\hat{u}) \leq E(u) \quad \forall u \in \mathbb{R}^n$$

We already learned about linear regression problems

$$\min_u \|Au - f\|^2$$

for some matrix A and a vector f, which led to the linear **Gaussian normal equation**

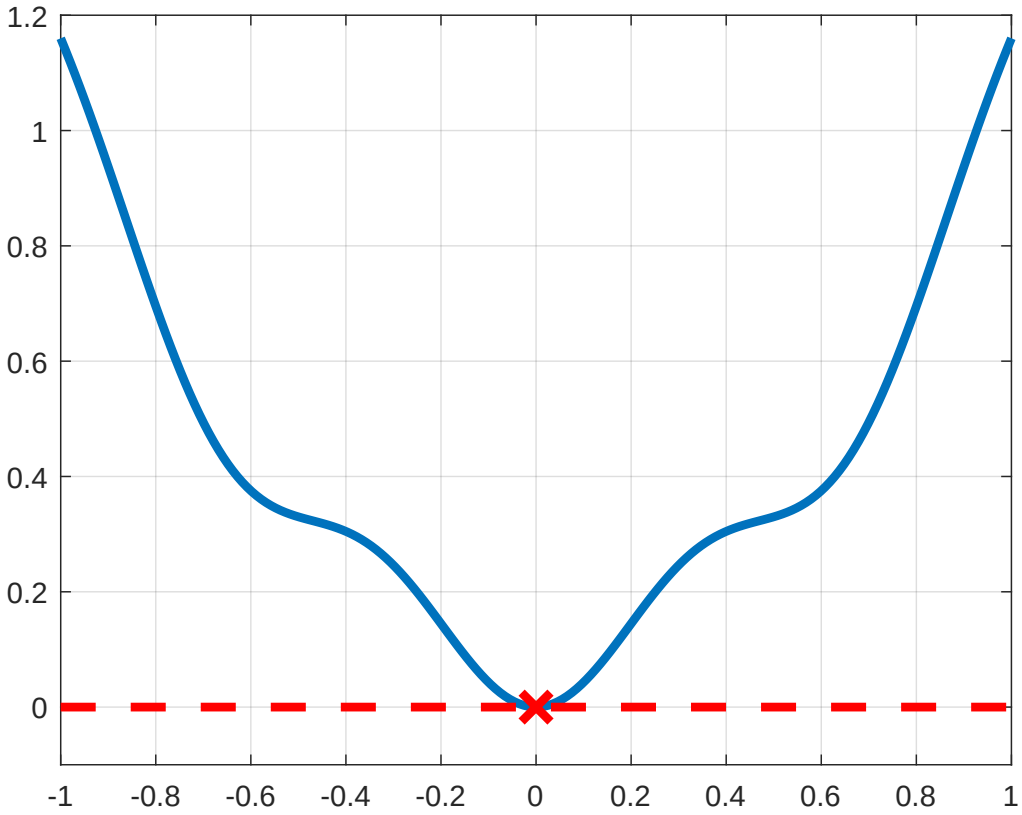
$$A^T Au = A^T f.$$

But what happens for E that are more general than $E(u) = \|Au - f\|^2$?

We need a way to determine what makes a $\hat{u}$ optimal!

Optimality conditions

Remember from school: For a differentiable $E : \mathbb{R} \to \mathbb{R}$ the necessary condition for $\hat{u}$ to be a minimizer of E is $E'(\hat{u}) = 0$.



Optimality conditions

The necessary condition in multiple dimensions, i.e., for a differentiable $E : \mathbb{R}^n \to \mathbb{R}$ are actually exactly the same, i.e.,

$$JE(\hat{u}) = 0, \tag{OC}$$

where JE is the Jacobian of E.

If we define $g = JE : \mathbb{R}^n \to \mathbb{R}^n$, then we are facing nothing but a nonlinear equation

$$g(u) = 0.$$

We can therefore apply the Newton method we studied in the previous chapter!

The condition (OC) is not sufficient in general, but we can also not hope to compute a global minimizer in general. An interesting class of functions for with (OC) is sufficient are *convex functions*.

Newton method

Let $E : \mathbb{R}^n \to \mathbb{R}$ be two times continuously differentiable. What does that mean?

The Jacobian JE is a function

$$JE : \mathbb{R}^n \to \mathbb{R}^n$$
$$JE(u) = \left(\frac{\partial E}{\partial u_1}(u), \ldots, \frac{\partial E}{\partial u_n}(u) \right).$$

If all partial derivatives of JE exist and are continuous, then the Jacobian of JE is the so-called **Hessian**

$$\nabla^2 E : \mathbb{R}^n \to \mathbb{R}^{n \times n}$$
$$\nabla^2 E(u) = \begin{pmatrix} \frac{\partial^2 E}{\partial u_1^2}(u) & \frac{\partial^2 E}{\partial u_1 \partial u_2}(u) & \ldots & \frac{\partial^2 E}{\partial u_1 \partial u_n}(u) \\ \frac{\partial^2 E}{\partial u_1 \partial u_2}(u) & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ \frac{\partial^2 E}{\partial u_1 \partial u_n}(u) & \frac{\partial^2 E}{\partial u_2 \partial u_n}(u) & \ldots & \frac{\partial^2 E}{\partial u_n^2}(u) \end{pmatrix}$$

and naturally represents the second derivative of E.

Michael Moeller
Jonas Geiping

Newton method

Remember that a necessary condition for minimizing a function $E : \mathbb{R}^n \to \mathbb{R}$ is

$$JE(u) = 0.$$

Technically, $JE(u) = \mathbb{R}^{1\times n}$. It is convenient to define the **gradient** ∇E via

$$\nabla E(u) = (JE(u))^T \in \mathbb{R}^{n\times 1}.$$

Applying Newton's method to $\nabla E(u) = 0$ yields

$$u^{k+1} = u^k - (\nabla^2 E(u^k))^{-1} \nabla E(u^k),$$

which is a very powerful tool as it can converge very quickly. There are, however, also some problems:

- Convergence can usually only be guaranteed locally
- Although we only have n unknowns, we have to invert an $n \times n$ matrix in each step.

Newton method

The fact that each Newton step is expensive brings us to a direction of approximations of the original Newton method, e.g., the **Quasi-Newton methods**. We will not detail this approach.

Let us consider the convergence issues instead: We can improve the convergence statements if we allow small steps in the 'Newton direction':

$$u^{k+1} = u^k - \tau^k (\nabla^2 E(u^k))^{-1} \nabla E(u^k),$$

for some $\tau^k > 0$.

For so called *strongly convex* functions (which we will also not discuss here), we can at least always find a τ^k such that the energy decreases monotonically, i.e.

$$E(u^{k+1}) \leq E(u^k).$$

Michael Moeller
Jonas Geiping

Gradient descent

Gradient descent

Let us go back to the idea of fixed point equations. If we want to solve

$$g(u) = 0,$$

we construct a function G such that

$$G(u) = u \quad \Leftrightarrow g(u) = 0,$$

and iterate

$$u^{k+1} = G(u^k).$$

One of the first suggestions in the class was to consider

$$G(u) = u - g(u)$$

or, more generally,

$$G(u) = u - \tau g(u)$$

for some scalar $\tau > 0$.

Michael Moeller
Jonas Geiping

Gradient descent

Applying this simple idea to the necessary condition $\nabla E(u) = 0$ arising from minimization problems, we obtain

$$u^{k+1} = u^k - \tau \nabla E(u^k).$$

This is a very famous update called **gradient descent**.

It **converges globally under much less restrictive assumption**; mainly it is enough for ∇E to meet

$$\|\nabla E(u) - \nabla E(v)\| \leq L\|u - v\|, \qquad \forall u, v \qquad \text{(Lip)}$$

for a constant $L \geq 0$, and choose $\tau < \frac{1}{L}$.

If ∇E meets (Lip), we call ∇E **Lipschitz-continuous**.

Gradient descent with line search

If ∇E is continuous but not necessary Lipschitz continuous one can adapt the step size $\tau = \tau^k$ at each iteration.

A common trick to obtain a globally convergent method is to

- compute the Newton direction

$$z^k = -(\nabla^2 E(u^k))^{-1} \nabla E(u^k)$$

- check if z^k meets a certain stability criterion
- if yes, look for a good step size τ^k to update

$$u^{k+1} = u^k + \tau^k z^k$$

- if no, look for a good step size τ^k for a gradient descent update

$$u^{k+1} = u^k - \tau^k \nabla E(u^k)$$

This is called **globalized Newton method**.

Globalized Newton method

Pick $u^0 \in \mathbb{R}^n$, $\beta \in]0,1[$, $\gamma \in]0,1[$, $\alpha_1, \alpha_2 > 0$, and $p > 0$. Iterate

- If $\nabla E(u^k) = 0$, stop.
- Determine z^k as the solution of

$$\nabla^2 E(u^k) z^k = -\nabla E(u^k).$$

  If this is possible and the resulting z^k meets

$$-\langle \nabla E(u^k), z^k \rangle \geq \min(\alpha_1, \alpha_2 \|z^k\|^p) \|z^k\|^2,$$

  set $v^k = z^k$. Otherwise set $v^k = -\nabla E(u^k)$.
- Determine τ^k to be the largest number in $\{1, \beta, \beta^2, \ldots\}$ such that

$$E(u^k - \tau^k v^k) \leq E(u^k) + \tau^k \gamma \langle \nabla E(u^k), v^k \rangle$$

- Update

$$u^{k+1} = u^k + \tau^k v^k.$$