O: Kritik, Verbesserungsvorschläge?
A: Kurvenintegrale, ∇-Felder
N: QR-Algo, Householder, Givens (wdh) (wdh)

Globalübung 10 Matte 3-WS21

13.01.21
Lambert Theisen

A KURVENINTEGRALE

Kurvenintegral (1ster Art):
→ für zB Flächeninhalt unter f entlang γ
→ Masse berechnen für gegebene Dichte (→ siehe HA)

Sei $\gamma: [a,b] \to \mathbb{R}^n$ eine C^1-Kurve mit Weg $\Gamma \subset \Omega$.
Dann ist das Kurvenintegral für ein Skalares Feld $f: \Omega \to \mathbb{R}$ entlang Γ

$$\int_\Gamma f\,ds = \int_a^b f(\gamma(t)) \, \|\gamma'(t)\|_2 \, dt$$

Beispiel CHA: Sei $\Gamma = \gamma([0,2\pi])$, $\gamma: [0,2\pi] \to \mathbb{R}^2$, $\gamma(t) = \begin{bmatrix}\cos(t)\\ \sin(t)\end{bmatrix}$ $\Rightarrow \gamma'(t) \begin{bmatrix}-\sin(t)\\ \cos(t)\end{bmatrix}$

mit Dichte $\rho: \Gamma \to \mathbb{R}$ $\rho(x,y) = x^2 + y^2$ ($\mathbb{R}^2$). Berechne Masse von Γ.

$\rho(\gamma_1(t), \gamma_2(t))$

$$M = \int_0^{2\pi} \rho(\gamma(t)) \cdot \|\gamma'(t)\| \, dt = \int_0^{2\pi} [\cos^2 + \sin^2(t)] \cdot \sqrt{[-\sin(t)]^2 + [\cos(t)]^2}$$

$$= \int_0^{2\pi} 1 \cdot 1 \, dt = 2\pi$$

3D komplexen Integralen.

Arbeitsintegral (Kurvenintegral 2ter Art) → Nun vektorwertige Fkt.

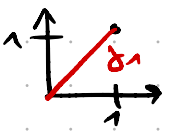
Sei $F_i: \Omega \to \mathbb{R}$ für $i \in \{1, \dots, n\}$ skalare Felder zum Vektorfeld $F: \Omega \to \mathbb{R}^n$, $F = (F_1, \dots, F_n)$.
Das Arbeitsintegral von F entlang Γ lautet:

$$\int_\Gamma F \cdot dx := \int_a^b \langle F(\gamma(t)), \gamma'(t) \rangle \, dt$$

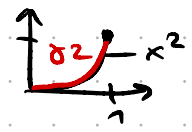
→ "Arbeit $\hat{=}$ Kraft (F) mal Weg (Γ)"

Beispiel

Ex: Given $f: \mathbb{R}^2 \to \mathbb{R}^2$, $f(x,y) = \begin{pmatrix} xy \\ x-y \end{pmatrix}$



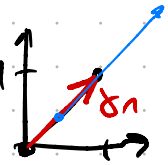
→ Calc $\int_{\gamma_1} f \cdot dx$, $\int_{\gamma_2} f \cdot dx$ for



(2)

(I)

1) Parametrisierung



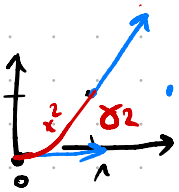
$\Rightarrow \gamma_1(t) = \begin{pmatrix} t \\ t \end{pmatrix}$, $t \in [0,1]$

2) Tangentialvektor: $\gamma_1'(t) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ $\quad [t \cdot t \;\; t-t]$

3) Integral: $\int_{\gamma_1} f \cdot dx = \int_0^1 \langle \underbrace{f(\gamma(t))}_{=}, \gamma_1' \rangle = \int_0^1 \left\langle \begin{pmatrix} t^2 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\rangle$

$= \int_0^1 t^2 dt = \frac{1}{3}$

(II) 1) Para:



$\Rightarrow \gamma_2(t) = \begin{pmatrix} t \\ t^2 \end{pmatrix}$, $t \in [0,1]$

2) Tang'vektor: $\gamma_2'(t) = \begin{pmatrix} 1 \\ 2t \end{pmatrix}$

3) Integral: $\int_{\gamma_2} f \cdot dx = \int_0^1 \langle f(\gamma_2(t)), \gamma_2'(t) \rangle \, dt$ $\quad \langle x,y \rangle \; x \cdot y$

$= \int_0^1 \left\langle \begin{pmatrix} t \cdot t^2 \\ t - t^2 \end{pmatrix}, \begin{pmatrix} 1 \\ 2t \end{pmatrix} \right\rangle dt$

$= \int_0^1 t^3 + 2t^2 - 2t^3 dt$ $\quad -\frac{1}{2}t^4$ $\quad -\frac{1}{2}$

$\left[\frac{1}{4}t^4 + \frac{2}{3}t^3\right]_0^1 = \frac{1}{4} + \frac{2}{3}$

$= \frac{3}{12} + \frac{8}{12} - \frac{6}{12}$

$= 5/12$

→ **Hat f ein Potential?** : Nein, weil I ≠ II obwohl $\gamma_1(a) = \gamma_2(a)$, $\gamma_2(b) = \gamma_2(b)$

GRADIENTENFELDER

Sei $\Omega \subset \mathbb{R}^n$. Ein Vektorfeld $F: \Omega \to \mathbb{R}^n$ heißt Gradientenfeld (konservatives Vektorfeld) falls es eine stetig diff'bare Fkt. $\varphi: \Omega \to \mathbb{R}$ gibt mit $\boxed{F = \nabla \varphi}$, φ heißt Potential

Einfachheit von Arbeitsintegralen bei Gradientenfeldern

(F ist Gradientenfeld mit φ)

$$\rightarrow \boxed{\int_\Gamma F \cdot dx = \ldots = \varphi(\gamma(b)) - \varphi(\gamma(a))}$$ (hängt nur von Randpunkten ab!)

→ Falls Γ geschlossene Kurve ($\gamma(a) = \gamma(b)$): $\boxed{\int_\Gamma F \cdot dx = 0}$

zsm. hängend

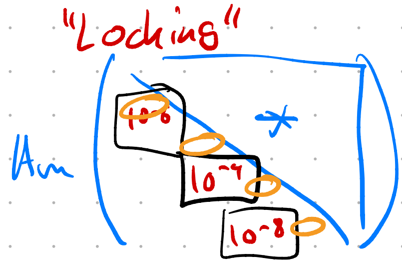
Satz 1 zu Existenz eines Potentials: Wenn Ω zsm.hängend und $F: \Omega \to \mathbb{R}^n$ Vektorfeld. Falls $\int_\Gamma F \cdot dx = 0$ $\forall$ geschlossenen Wege $\Rightarrow F$ hat Potential.

Alle geschlossenen Wege auszuprobieren wird schwierig.... ↦ Lösung nächste VL :-)

⑶

[N] QR-Algorithm for Eigenvalue Problems

$$[A] = [Q^T] \,\Big[\overset{R}{\underset{0}{\diagdown}} \Big], \qquad Q^T \equiv Q^{-1}$$



- Given $A \in \mathbb{R}^{n \times n}$
- Set $A_0 = A$
- for $m = 0, 1, 2, \dots$ do
  - Compute $A_m = Q_m R_m$ with $Q_m^T = Q_m^{-1}$ ortho. & R_m upper-Δ (QR-Decomp.)
  - Assign $A_{m+1} = R_m Q_m = Q_m^T A Q_m$ (similarity trafo ⟺ same Eigen)

→ $A_m = Q_{m-1}^T A_{m-1} Q_{m-1} = Q_{m-1}^T Q_{m-2}^T A_{m-2} Q_{m-2} Q_{m-1}$

$= \dots = (Q^{(m)})^T A \, Q^{(m)}$ with $Q^{(m)} := Q_0 Q_1 \cdot \ldots \cdot Q_{m-1}$ is ortho.

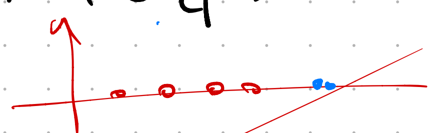
Buch-Templates for the Solution of Eigenv. Probs (Heiß Cover)

→ **Eigenvalue-Estimates**: stehen auf Diagonalen von A_m (upper-Δ $\begin{bmatrix} * & * & * \\ 0 & * & * \\ & & * \end{bmatrix}$)

→ **Eigenvector-Estimates**: stehen in Spalten von $Q^{(m)}$

~~→ **Konvergenz**: (siehe Lecture): $A_m = \Lambda + \mathcal{O}(p^m)$~~

~~mit $p := \max_{i=1 \dots n} \left| \frac{\lambda_{i+1}}{\lambda_i} \right|$~~



Pü4

~~↳ Wenn zwei EVal nahe beieinander ⇒ nicht so gut~~ (⟹ QR mit Shift)

~~↳ Wenn komplex konjugierte EVals ⇒ A_m konvergiert zu oberer Block Δ-Matrix~~

~~zB $A_m \to \begin{bmatrix} [* & *] & * & * \\ [* & *] & * & * \\ 0 & 0 & [*] & * \\ 0 & 0 & 0 & [*] \end{bmatrix}$~~ $\tilde{A}$ hat dann EVals λ_j & λ_{j+1}

Berechnung der QR-Zerlegungen:

→ Gram-Schmidt: "Wenn A vollen Rang, mache Spalten von A zu OGB ⇒ Q"

→ Householder: "Bringe A auf obere Dreiecksform ⇒ R, $Q \triangleq \prod$ Trafos"

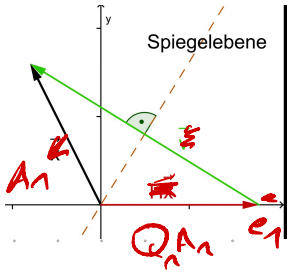
→ Givens-Rotationen: "Entferne alle Einträge von A unterhalb der Diag"

Recall: Householder Spiegelung für QR Zerlegung

→ Matrix $A = [A_1 \cdots A_n]$ gegeben

→ Erste Householder Matrix $Q_1 = I - 2\frac{v_1 v_1^T}{v_1^T v_1}$ mit $\begin{cases} \cdot v_1 = A_1 + \alpha_1 e_1 \\ \cdot \alpha_1 = \mathrm{sgn}(A_{11})\|A_1\|_2 \end{cases}$ $\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$

→ $Q_1 A = \begin{bmatrix} * & * & \cdots & * \\ 0 & * & \cdots & * \\ \vdots & \vdots & & \vdots \\ 0 & * & \cdots & * \end{bmatrix}$ (erste Spalte wird auf e_1 projiziert.)



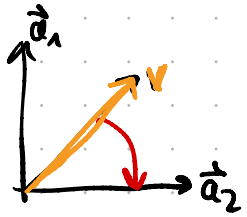
→ $Q_2 Q_1 A = \left[\begin{array}{c|ccc} * & * & * & * \\ \hline 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \end{array}\right]$ ↝ Nutze $A_1' := (Q_1 A)[2{:}end, 2{:}end]$ für Konstruktion von $Q_2 = \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & Q_2' \end{array}\right]$ usw....

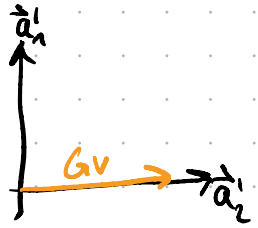
$R = \tilde{Q} A \qquad A = QR$

→ $R = \underbrace{Q_{n-1} \cdot \ldots \cdot Q_1}_{\tilde{Q}} A = \begin{bmatrix} * & * & \cdots & * \\ & * & \cdots & * \\ & & \ddots & \vdots \\ 0 & & & * \end{bmatrix}$ obere Δ-Matrix

→ **Was ist Q^2:** $R = \tilde{Q} A \iff \tilde{Q}^{-1} R = A \iff \tilde{Q}^T R = A \iff \boxed{Q = \tilde{Q}^T}$

Recall: Givens Rotationen : Drehung in Ebene





Basis koeff von $\vec{a}_1' = 0$

$G(i,j,\theta) = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & \cdots & s & & \\ & & \vdots & & \vdots & & \\ & & -s & \cdots & c & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix}$ (i), (j) mit $\begin{cases} c = \cos\theta = \frac{a_{jj}}{\rho} \\ s = \sin\theta = a_{ij}/\rho \\ \rho = \sqrt{a_{jj}^2 + a_{ij}^2} \quad ?! \end{cases}$

$G(n,1) \cdot A = \begin{bmatrix} * & \cdots & * \\ \vdots & & \vdots \\ * & \cdots & * \\ 0 & & \end{bmatrix}$

→ Kann für QR benutzt werden

→ Insbesondere QR für Hessenbergs Matrizen in $\mathcal{O}(n-1)$ Givens-Rotationen [mit konst oder variablen Kosten!]

Eintrag (i,j) wird zu Null

Demo:
- QR Algorithmus
- Konvergenz
- Householder Hessenberg Trafo
- Givens QR für tridiagonale Matrizen

QR Algorithm Native

We use Julia's standard `qr()` function and implement:

1. Given $A \in \mathbb{R}^{n \times n}$
2. Initialize $Q^{(m+1)} = I$
3. For $k = 1, \ldots, m$:
   1. Calculate QR-Decomposition: $A_k = Q_k R_k$
   2. Update: $A_{k+1} = R_k Q_k$
4. Return diagonal entries of A_m and $Q^{(m)} = Q_m \cdots Q_0 Q_1$

```
qra_general (generic function with 1 method)
```

```
function qra_general(A, m)
    @assert size(A)[1] == size(A)[2] && length(size(A))==2
    n = size(A)[1]
    Qm = I(n)
    for k=1:m
        Q, R = qr(A)
        A = R * Q
        Qm = Qm * Q
    end
    return diag(A), Qm
end
```

QR Algorithm for symmetric Hessenberg matrices

Symmertric hessenberg matrices are tridiagonal (only diag plus uppe and lower sub-diagonal). For the QR decomposition, we only have to make the lower sub diagonal entries to zero to obtain the upper right triangular matrix. This can be done by using Givens roations:

Givens Rotations [1]

Given a matrix A, we can make to entry A_{ij} to zero with $A_{new} = G(A, i, j)$ where

$$G(A, i, j) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix},$$

with

- $r = \sqrt{A_{jj}^2 + A_{ij}^2}$
- $s = -A_{ij}/r$
- $c = A_{jj}/r$

[1]: https://en.wikipedia.org/wiki/Givens_rotation

```
givens_rotation_matrix (generic function with 1 method)
```

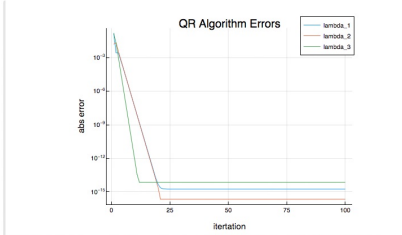
```
function givens_rotation_matrix(A, i, j)
    n = size(A)[1]
    r = sqrt(A[j,j]^2 + A[i,j]^2)
    s = -A[i,j]/r
    c = A[j,j]/r
    G = sparse(1.0I, n, n)
    G[i,i] = c
    G[j,j] = c
    G[i,j] = s
    G[j,i] = -s
    return G
end
```

Check Convergence

```
begin
    N = 100
    tape = Array[]
    for k=1:N
        push!(tape, sort(qra_general(A, k)[1], rev=false)) # #dont do this, very inefficient#
    end
end
```

```
errors =
▸ Array{Float64,1}[Float64[0.137636, 0.00263023, 0.00266808, 0.000628078, 0.00012773, 2
```

```
errors = [
    abs.(map(x -> x[1], tape) .- eigen(A).values[1]),
    abs.(map(x -> x[2], tape) .- eigen(A).values[2]),
    abs.(map(x -> x[3], tape) .- eigen(A).values[3]),
]
```




```
plot([errors[1],errors[2],errors[3]], yaxis=:log, label=["lambda_1" "lambda_2" "lambda_3"], title="QR Algorithm Errors", xlabel="iteration", ylabel="abs error")
```

Improve QR Algorithm with Upper Hessenberg Matrix Preconditioning

- Idea: QR decomposition needs $\mathcal{O}(n^3)$, QR for Hessenberg matrices is easier done in $\mathcal{O}(n^2)$. Linear complexity is even possible if A is symmetric. In this case, the QR decomposition only needs $\mathcal{O}(n)$ Givens rotations with constant effort.
- Therefore transform the matrix A to upper Hessenberg form with similarity transforms in $\mathcal{O}(n^3)$ (also cubic, but only needs to be done once) and use this matrix for the QR algorithm.

Upper Hessenberg Shape

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} & \cdots & h_{1n} \\ h_{21} & h_{22} & h_{23} & \cdots & h_{2n} \\ 0 & h_{32} & h_{33} & \cdots & h_{3n} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & h_{nn\ 1} & h_{nn} \end{pmatrix}$$

Algorithm [1]:

1. Given: $A \in \mathbb{R}^{n \times n}$
2. For $k = 1 \ldots n \quad 2$ do:
   1. $[v, \beta] \leftarrow \text{house}(A(k+1:n, k))$
   2. $A(k+1:n, k:n) \leftarrow (I - \beta v v^T) A(k+1:n, k:n)$
   3. $A(1:n, k+1:n) \leftarrow A(1:n, k+1:n)(I - \beta v v^T)$

with Householder reflection vector v and weight $\beta = 2/(v^T v)$.

[1]: https://www.tu-chemnitz.de/mathematik/numa/lehre/nla-2015/Folien/nla-kapitel6.pdf

```
upperhessenberg (generic function with 1 method)
```

```
function upperhessenberg(A)
    @assert size(A)[1] == size(A)[2] && length(size(A))==2
    n = size(A)[1]
    for k = 1:n-2
        v, β = householdervec(A[k+1:n,k])
        A[k+1:n, k:n] = (I(n-k) - β * v * v') * A[k+1:n, k:n]
        A[1:n, k+1:n] = A[1:n, k+1:n] * (I(n-k) - β * v * v')
    end
    return A
end
```

Speed Check

We still loose against Julia's native `qr`-method. 😐

- Homework: Improve this.

```
QRA General:
  0.166218 seconds (30.90 k allocations: 51.228 MiB, 4.84% gc time)
QRA Own:
  0.417899 seconds (99.22 k allocations: 752.204 MiB, 13.56% gc time)
```

```
with_terminal() do
    N = 100
    M = 50
    C = Array(Symmetric(rand(N,N)))

    # @time qr(C)
    # @time qr_symm_hess(C)
    # @time upperhessenberg(C)

    println("QRA General:")
    @time Am1, Qm1 = qra_general(C, M)
    println("QRA Own:")
    @time Am2, Qm2 = qra_symm(C, M)
end
```

0.14 ms