TODO | 0 : Alles ok?
A Kurven
N QR-Alg, Householder, Givens

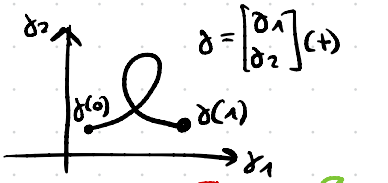
Globalübung 10
Mathe 3 – WS 24

Lambert Theisen

18.12.24 (1)

A Kurven:

→ stetige Abbildung $\gamma : [a,b] \to \mathbb{R}^n$.

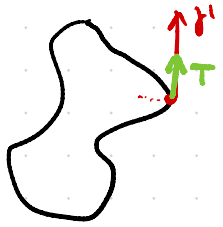


- **Weg**: $\hat{=}$ Bild $\Gamma = \gamma([a,b])$ — γ = Parametrisierung von Γ
- **Einfachheit**: Falls γ injektiv ("doppelpunktfrei") (schneidet sich nicht)

- **Geschlossenheit**: Falls $\gamma(a) = \gamma(b)$

- **$C^\wedge$-Kurve**: Falls γ stetig diff'bar

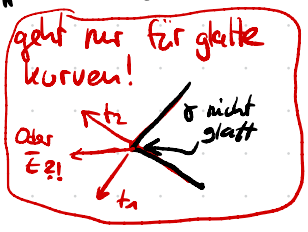
- **Tangentialvektor im Pkt. $\gamma(t)$**: $\gamma'(t) = (\gamma_1'(t), \dots, \gamma_n'(t))^T$



- **Glattheit**: Falls γ eine $C^\wedge$-Kurve & $\|\gamma'(t)\| \neq 0 \ \forall t \in [a,b]$
- **Tangentialeinheitsvektor für glatte Kurven**: $T : [a,b] \to \mathbb{R}^n$: $T(t) := \frac{\gamma'(t)}{\|\gamma'(t)\|} \Rightarrow \|T(t)\| = 1$ ✓

Vektor! Skalar!

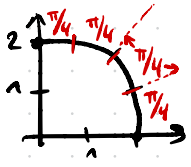
- **Umparametrisierung** $\tilde{\gamma} := \gamma \circ \phi : [c,d] \to [a,b] \to \mathbb{R}^n, t \mapsto \gamma(\phi(t))$
  → Falls ϕ Diffeomorphismus ("gute Trafo") & $\gamma \in C^\wedge \Rightarrow \tilde{\gamma} \in C^\wedge$ — Transformation

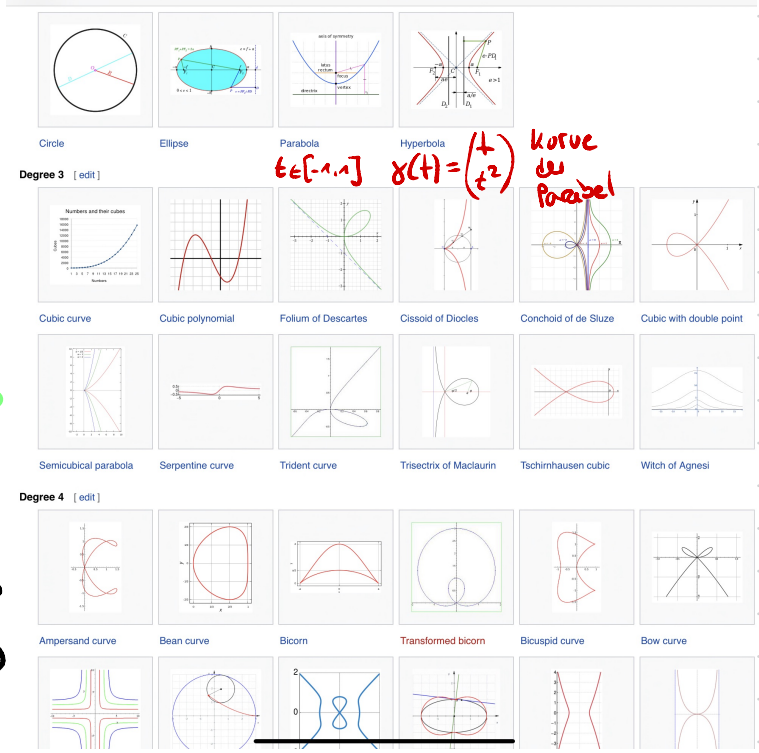


- **Stückweise Eigenschaften**:

→ zb Polygonzug stückweise glatt

- **Bogenlänge für $C^\wedge$-Kurven**:
  → $L(\gamma) = \int_a^b \|\gamma'(t)\| dt$
  "Quasi Maßband an Kurve anlegen"





- **Invarianz Bogenlänge unter Umparametrisierung**:

Sei $\phi : [c,d] \to [a,b]$ Diffeomorphismus und $\tilde{\gamma} := \gamma \circ \phi$, dann

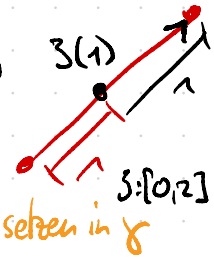
$$L(\tilde{\gamma}) = \int_c^d \|\tilde{\gamma}'(s)\| ds = \int_c^d \|\gamma'(\phi(s))\| \, |\phi'(s)| ds = \int_a^b \|\gamma'\| dt = L(\gamma)$$

($\phi^{-1}(t)$)

$\|\gamma'\| = 1$ wäre schön.... 🤔

weil sub $t = \phi(s)$, $\phi'(s) ds = dt$

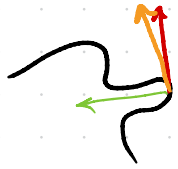
Natürliche Parametrisierung: $\gamma: [0, L] \to \mathbb{R}^n$ ist nat. Param. eines Weges Γ falls $\|\gamma'(t)\|_2 = 1$. [Einfaches Ablesen der Bogenlänge]



→ Konstruktion: Benutze Diffeomorphismus $\psi: [a,b] \to [0,L]$

mit $t \mapsto \psi(t) := \int_a^t \|\gamma'(s)\| \, ds$ nach t umformen und einsetzen in γ

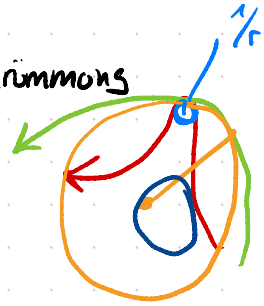
Normaleneinheitsvektor: Sei $T(t) := \frac{\gamma'(t)}{\|\gamma'(t)\|}$ Tang'enh'vektor, dann



$$N(t) := \frac{T'(t)}{\|T'(t)\|}$$ Norm'enh'vektor.

Krümmung: Sei γ eine natürliche Param., dann ist die Krümmung von γ in γ:

$$\kappa(\gamma) := \|T'(\gamma)\|$$

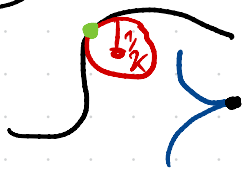


Kreiskrümmung: $\gamma: [0, 2\pi r] \to \mathbb{R}$, $s \mapsto \gamma(s) = \begin{bmatrix} r\cos(\frac{s}{r}) \\ r\sin(\frac{s}{r}) \end{bmatrix}$

$$\gamma'(s) = \begin{bmatrix} -\sin(\frac{s}{r}) \\ \cos(\frac{s}{r}) \end{bmatrix} \Rightarrow T(t) = \frac{\gamma'(t)}{\|\gamma'(t)\|_2} = \gamma'(t) \text{ weil } \|\gamma'(t)\|_2 = \sqrt{\sin^2+\cos^2} = 1$$

$$\Rightarrow T'(\gamma) = \begin{bmatrix} -\frac{1}{r}\cos(\frac{s}{r}) \\ -\frac{1}{r}\sin(\frac{s}{r}) \end{bmatrix} \Rightarrow \kappa(\gamma) = \sqrt{\frac{1}{r^2}(\underbrace{\sin^2+\cos^2}_{=1})} = \frac{1}{r}$$

$2r \downarrow \Leftarrow$  $\Rightarrow \kappa \uparrow$



Beispiel: a) **Check:** Bdg. muss erst parametrisiert werden!

$$x^{2/3} + y^{2/3} = a^{2/3} \text{ mit } \gamma: [0, 2\pi] \to \mathbb{R}^2$$

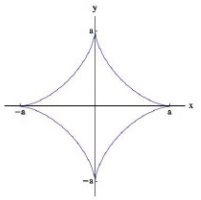
$$[a\cos^3(t)]^{2/3} + [a\sin^3(t)]^{2/3} = a^{2/3}$$

$$\Leftrightarrow a^{2/3}\cos^2(t) + a^{2/3}\sin^2(t) = a^{2/3} \quad \checkmark$$

b) **Bogenlänge** $L(\gamma) = \int_0^{2\pi} \|\gamma'(t)\|_2 \, dt$

Daher $\gamma'(t) = \begin{bmatrix} \partial/\partial t(a\cos^3(t)) \\ \partial/\partial t(a\sin^3(t)) \end{bmatrix} = \begin{bmatrix} -3a\cos^2(t)\sin(t) \\ 3a\sin^2(t)\cos(t) \end{bmatrix}$

Aufgabe 68. (Bogenlänge einer Kurve)



The so-called astroid (Sternkurve) is described by the Cartesian equation

$$x^{2/3} + y^{2/3} = a^{2/3}$$

for some $a > 0$.

$\gamma(t) = \begin{bmatrix} a\cos^3(t) \\ a\sin^3(t) \end{bmatrix}$

a) Verify that the astroid can be expressed in parametric form $\gamma: [0, 2\pi] \to \mathbb{R}^2$ for

$$x(t) = a\cos^3 t \quad ; \quad y(t) = a\sin^3 t$$

b) Find the length of the astroid.

c) Find the area of the astroid using the change of variables $F: (x, y) \mapsto (r, \varphi)$ defined as

$$x = r\cos^3\varphi \quad , \quad y = r\sin^3\varphi$$

Begin by verifying that

$$J(r,\varphi) = \left\| \begin{matrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \varphi} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \varphi} \end{matrix} \right\| = \frac{8}{3} r\,(1 - \cos 4\varphi)$$

and notice that we can describe the astroid in terms of the coordinates (r, φ) as $0 \le r \le a$ and $0 \le \varphi \le 2\pi$.

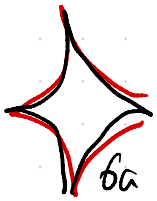
$$\Rightarrow \|\gamma'(t)\|_2 \overset{a>0}{=} 3a\sqrt{(\cos^2\sin)^2+(\sin^2\cos)^2} = 3a\sqrt{\underbrace{(\cos^2+\sin^2)}_{=1}(\sin^2\cos^2)}$$

Finden $\cos^2+\sin^2$

$$\Rightarrow L(\gamma) = \int_0^{2\pi} \|\gamma'(t)\|dt = 3\int_0^{2\pi} \sqrt{\sin^2(t)\cos^2(t)}\,dt$$

$$= 3a\int_0^{2\pi} \left(\underbrace{\sin(t)\cos(t)}_{=\frac{1}{2}\sin(2t)}\right)^2)^{1/2} dt = 3a\int_0^{2\pi} \left|\tfrac{1}{2}\sin(2t)\right| dt \quad \text{(Formelzettel)}$$

$$= 12a\int_0^{\pi/2} \tfrac{1}{2}\sin(2t)\,dt = 12a\cdot\tfrac{1}{2} = 6a$$



c) **Fläche** (nicht so wichtig) siehe Tips

$$A = \int_0^a \int_0^{2\pi} 1\cdot J(r,\varphi)$$

$$= \int_0^a \int_0^{2\pi} \tfrac{3}{8}r(1-\cos(4\varphi))\,d\varphi\,dr$$

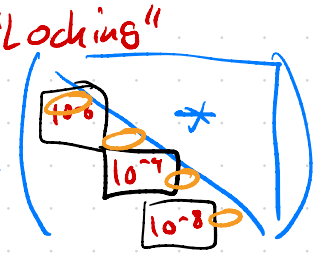
$$= \ldots \overset{!}{=} \frac{3}{8}\pi a^2$$

- Symbolic tools (Sympy, Julia, Matlab, Wolfram Alpha) manchmal hilfreich

N QR-Algorithm for Eigenvalue Problems

$Ax = \lambda x$

"Locking"



$$[A] = [Q^T] [R] \quad (Q^T = Q^{-1})$$

- 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 $\Leftrightarrow$ same Eigen)

Buch: Templates for the solution of Eigenv. Probs (Hoffbauer Cover)

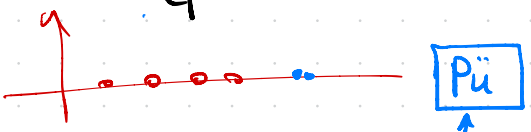
$$\rightarrow A_m = Q_{m-1}^T A_{m-1} Q_{m-1} = Q_{m-1}^T Q_{m-2}^T A_{m-2} Q_{m-2} Q_{m-1}$$
$$= \dots = (Q^{(m)})^T A \, Q^{(m)} \quad \text{with } Q^{(m)} := Q_0 Q_1 \cdot \dots \cdot Q_{m-1} \text{ is ortho.}$$

→ **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): $A_m = \Lambda + \mathcal{O}(\rho^m)$

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



↳ Wenn zwei Eval nahe beieinander ⇒ nicht so gut (⇒ QR mit Shift)

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

zB $A_m \rightarrow \begin{bmatrix} \hat{A} & & * & * \\ & & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix}$ $\hat{A}$ hat dann Evals λ_j & λ_{j+1}

Berechnung der QR-Zerlegungen:

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

→ Householder: "Bringe A auf obere Dreiecksform ⇒ R, Q = ∏ Trafos"

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

→

→ **Implementierung**: → Problem: Jede QR-Decomp. braucht $\mathcal{O}(n^3)$

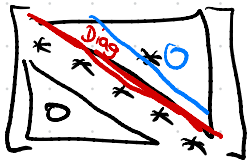
→ **Beobachtung**: QR-Decomp für Hessenberg-Matrizen braucht nur $\mathcal{O}(n^2)$ und $A^{k+1} = RQ$ bleibt obere Hessenberg (→ siehe HA).

⇒ Vorbereitung der Matrix auf obere Hessenberg Gestalt (per Ähnlichkeitstrafo)

⇒ Hessenberg braucht zwar auch $\mathcal{O}(n^3)$ (einmal) dann aber pro Iterationsschritt nur noch $\mathcal{O}(n^2)$ Operationen.

⇒ Für symmetrische (Hessenberg) Matrizen braucht man nur $\mathcal{O}(n)$ Givens-Rotationen mit konstantem Aufwand.

($A_{ij} = 0$)

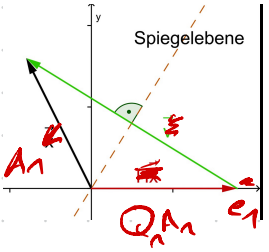


Recall: Householder Spiegelung für QR Zerlegung

→ Matrix $A = \begin{bmatrix} | & & | \\ A_1 & \cdots & A_n \\ | & & | \end{bmatrix}$ gegeben

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

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



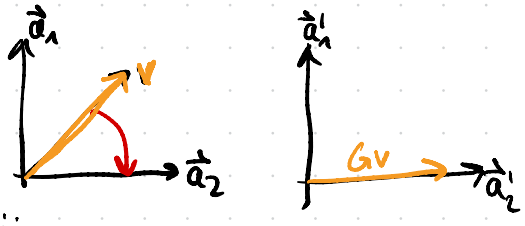
→ $Q_2 Q_1 A = \begin{bmatrix} * & * & * \\ 0 & * & * \\ 0 & 0 & * \end{bmatrix}$ no Notiz $A_1' := (Q_1 A)[2{:}end, 2{:}end]$ für Konstruktion von $Q_2 = \begin{bmatrix} 1 & 0 \\ 0 & Q_2' \end{bmatrix}$ usw....

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

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

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

Recall: Givens Rotationen : Drehung in Ebene:



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

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

→ kann für QR benutzt werden

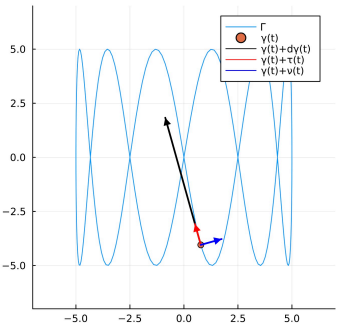
$G(n,1) \cdot A = \begin{bmatrix} * & \cdots & * \\ \vdots & & \\ * & & \\ 0 & & \end{bmatrix}$

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

Eintrag (i,j) wird zu Null

Demo: Kurven

8.953539062730911



- 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 $Q^{(n+1)} = I$
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 $A_{\text{new}} = G(A, i, j)$ 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)
```

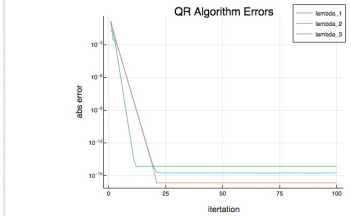
```
function givens_rotation_matrix(A, i, j)
    n = size(A)[1]
    r = sqrt(A[j,j]^2 + A[i,j]^2)
    s = -A[i,j]/r
    c = A[j,j]/r
    G = sparse(1.0I, n, n)
    G[i,i] = c
    G[j,j] = c
    G[i,j] = s
    G[j,i] = -s
    return G
end
```

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