

- Testate, Evaluation
- Kurvenintegrale, V-Felder
- QR-Algo, Householder, Givens

Globalübung 10 Mathe 3 - WS20

13.01.21
Lambert Theisen

(1)

- Testate (PÜ3) am Do, 14.01.21 (Moodle!) (bis 14:00 heute!)
- o Evaluierung der Übung

I) Lambert

<https://www.campus.rwth-aachen.de/evasys/online.php?pswd=VXKPV>



II) Ulilka:

<https://www.campus.rwth-aachen.de/evasys/online.php?pswd=AC3RS>



A KURVENINTEGrale

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

Sei $\gamma: [a, b] \rightarrow \mathbb{R}^n$ eine C^1 -kurve mit Weg $\Gamma \subset \Omega$.

Dann ist das Kurvenintegral für ein Skalarfeld $f: \Omega \rightarrow \mathbb{R}$ entlang Γ

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

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

CHA

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

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

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

3D
komplexe
Integrale.

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

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

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

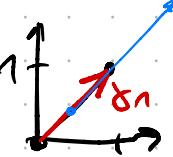
→ "Arbeit = Kraft (F) mal Weg (Γ)"

Beispiel

(2)

(I)

1) Parametrisierung



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

2) Tangential vector: $\gamma_1'(t) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

$$\begin{bmatrix} t \cdot t \\ t - t \end{bmatrix}$$

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

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

(II)

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

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

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

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

$$= \int_0^1 \cancel{t^3} - 2t^2 dt = \left[\frac{1}{4}t^4 - \frac{2}{3}t^3 \right]_0^1 = \frac{1}{4} - \frac{2}{3} = \underline{\underline{-\frac{5}{12}}}$$

Bsp für
einf. Wegzsm
aber kein
Pkt.

GRADIENTENFELDER

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

Einfachheit von Arbeitsintegralen bei Gradientenfeldern (F ist Gradientenfeld mit φ)

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

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

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

Alle geschlossenen Wege auszuprobiieren wird schwierig... DAHER \exists

Einfach zum hängende Mengen: $\Omega \subset \mathbb{R}^n$ heißt einfach weg zsm. hängend, falls Ω weg zsm. hängend & sich jeder geschlossene Weg in stetiger Art und Weise auf einen Pkt. in Ω zum Zielen lässt. (3)

Satz 2 zu

Existenz eines Potentials: Sei Ω einfach weg zsm. hängend ($\{\textcircled{0}\}, \{\textcircled{0}\}, \{\textcircled{0}\}, \{\textcircled{0}\}$)

und $\mathbf{F}: \Omega \rightarrow \mathbb{R}^n$ ein diff'bares Vektorfeld. Falls $\nabla \mathbf{F}$ symmetrisch ($\operatorname{rot} \mathbf{F} = 0, \nabla \times \mathbf{F} = 0$) dann hat \mathbf{F} ein Potential.

Bsp 1 (VL): $\mathbf{F}(x) = \begin{pmatrix} 2x_1 x_2^2 \\ 2x_1^2 x_2 + 1 \end{pmatrix}$ auf \mathbb{R}^2

$$\Rightarrow \nabla \mathbf{F} = \begin{bmatrix} 2x_2^2 & 4x_1 x_2 \\ 4x_1 x_2 & 2x_1^2 \end{bmatrix} \text{ symmetrisch!}$$

$$\rightarrow \frac{\partial \varphi(x)}{\partial x_1} = F_1 = 2x_1 x_2^2 \Rightarrow \varphi(x_1, x_2) = x_1^2 x_2^2 + g(x_2)$$

$$\rightarrow \frac{\partial \varphi(x)}{\partial x_2} = 2x_1^2 x_2 + g'(x_2) \stackrel{!}{=} F_2 = 2x_1^2 x_2 + 1 \Rightarrow g(x_2) = x_2$$

$$\Rightarrow \varphi = x_1^2 x_2^2 + x_2$$

Beispiel: a) $D_f = \mathbb{R} \times \mathbb{R}^+ \times \mathbb{R}$

weil $\ln(y) \rightarrow$ einfache Weg zsm.

D_f ist einfach zsm. hängend ✓

$$b) \operatorname{rot} \mathbf{f} = \nabla \times \mathbf{f} = \begin{pmatrix} \partial_y f_3 - \partial_z f_2 \\ \partial_z f_1 - \partial_x f_3 \\ \partial_x f_2 - \partial_y f_1 \end{pmatrix} = \begin{pmatrix} 0 \\ -3x^2 \sin(z) - (-3x^2 \sin(z)) \\ 0 \end{pmatrix}$$

$\nabla \mathbf{f}$ Diagonalen weglassen

$= 0 \Rightarrow \mathbf{f}$ hat Potential! nach Satz 2

Bestimmung Potential φ :

$$\begin{aligned} \partial_x \varphi &\stackrel{!}{=} f_1 = 3x^2 \cos(z) \Rightarrow \varphi(x, y, z) = x^3 \cos(z) + k_1(y, z) \\ \partial_y \varphi &\stackrel{!}{=} f_2 = \ln(y) \Rightarrow \varphi(x, y, z) = y \ln(y) - y + k_2(x, z) \\ \partial_z \varphi &\stackrel{!}{=} f_3 = -x^3 \sin(z) \Rightarrow \varphi(x, y, z) = x^3 \cos(z) + k_3(x, y) \end{aligned}$$

$$\Rightarrow \varphi(x, y, z) = y \ln(y) - y + x^3 \cos(z) + k, k \in \mathbb{R}$$

c) Arbeitsintegral direkt: (skipped, siehe oben): $\int_{\gamma} \mathbf{f} \cdot d\mathbf{x} = \dots = 2$ ↗

$$\gamma(t) = \begin{pmatrix} t \\ e^t \\ 0 \end{pmatrix}, t \in [0, 1] \quad (\text{Parametrisierung})$$

d) Arbeitsintegral mit Potential: \rightsquigarrow Einsetzen {Anfangs} & {Endpunkt} von γ : $\{(0, 1, 0)^T\} \quad \{(1, e, 0)^T\}$

$$\Rightarrow \int_{\gamma} \mathbf{f} \cdot d\mathbf{x} = \varphi\left(\begin{pmatrix} 1 \\ e \\ 0 \end{pmatrix}\right) - \varphi\left(\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\right) = \underbrace{\left[e \ln(e) - e + 1^3 \cos(0) \right]}_{e} - \underbrace{\left[1 \ln(1) - 1 + 0^3 \cos(0) \right]}_{0} = 2$$

Bsp 2 (VL): $\mathbf{F}(x) = \begin{bmatrix} -x_2/(x_1^2 + x_2^2) \\ x_1/(x_1^2 + x_2^2) \end{bmatrix}$ auf $\Omega = \mathbb{R}^2 \setminus \{\textcircled{0}\}$

hätte $\nabla \mathbf{F}$ symmetrisch (und würde $\varphi(x) = -\arctan\left(\frac{x_2}{x_1}\right)$ haben)

Allerdings gilt für geschl. Weg $\Gamma = \gamma([0, 2\pi])$ mit $\gamma(t) = \begin{bmatrix} \cos(t) \\ \sin(t) \end{bmatrix} : \int_{\Gamma} \mathbf{F} = \dots = 2\pi \neq 0$

→ Einfach zum hängend ist entscheidend!

Aufgabe 45. (Arbeitsintegral / Potential)
Gegeben sei das Vektorfeld

$$\mathbf{f} : D_f \rightarrow \mathbb{R}^3, \quad \mathbf{f}(x, y, z) = \begin{pmatrix} 3x^2 \cos z \\ \ln y \\ -x^3 \sin z \end{pmatrix}.$$

a) Was ist der Definitionsbereich D_f von f ?

b) Weisen Sie nach, dass f ein Potential besitzt, und berechnen Sie dieses.

c) Berechnen Sie das Arbeitsintegral

$$\int_{\Gamma} \mathbf{f} \cdot d\mathbf{x}$$

direkt. Die Kurve γ ist dabei die Verbindung der Punkte $(0, 1, 0)^T$ und $(1, e, 0)^T$ entlang des Graphen der Funktion $g(x) = e^x$ in der xy -Ebene.

d) Berechnen Sie das Arbeitsintegral

$$\int_{\Gamma} \mathbf{f} \cdot d\mathbf{x}$$

mit Hilfe des in b) bestimmten Potentials.

Folgerung:
Kann nur eine
Konstante
sein



N

QR-Algorithm for Eigenvalue Problems

④

- Given $A \in \mathbb{R}^{n,n}$
- Set $A_0 = A$
- for $m=0,1,2,\dots$ do

$$[A] = [Q] \begin{bmatrix} R \\ \vdots \\ Q^m \end{bmatrix}$$

- Compute $A_m = Q_m R_m$ with $Q_m^T = Q_m^{-1}$ ortho. & R_m upper-Δ (QR-Decomp.)
- Assign $A_{m+1} = R_m Q_m = Q_m^T A Q_m$ (similar trafo \Leftrightarrow same Eigen)

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

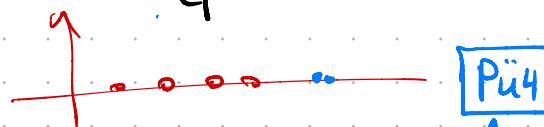
$$= \dots = (Q^{(m)})^T A Q^{(m)} \text{ with } Q^{(m)} := Q_0 Q_1 \cdots Q_{m-1} \text{ is ortho.}$$

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

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

\rightarrow Konvergenz: (siehe Lecture) : $A_m = \Lambda + O(\rho^m)$

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



\hookrightarrow Wenn zwei Elval nahe beieinander \Rightarrow nicht so gut (und QR mit Shift)

\hookrightarrow Wenn komplexe konjugierte Elvals \Rightarrow A_m konvergiert zu oberer Block A-Matrix

z.B. $A_m \rightarrow \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ 0 & 0 & [*] & * \\ 0 & 0 & 0 & [*] \end{bmatrix}$ $\tilde{\Lambda}$ hat dann Elvals $\lambda_1 \& \lambda_{2n}$

\rightarrow Implementierung: \rightarrow Problem: Jede QR-Decomp. braucht $O(n^3)$

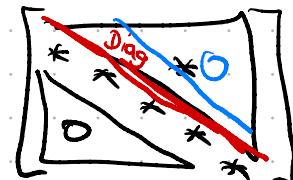
\rightarrow Beobachtung: QR-Decomp für Hessenberg-Matrizen braucht nur $O(n^2)$ und $\tilde{H} = RQ$ bleibt obere Hessenberg (\rightarrow siehe HA).

\Rightarrow Vorbereitung der Matrix auf obere Hessenberg Gestalt (per Ähnlichkeitstraf)

\Rightarrow Hessenberg braucht zwar auch $O(n^3)$, dann aber pro Iterationsschritt nur noch $O(n^2)$ Operationen.

\Rightarrow Für symmetrische Matrizen braucht man nur $O(n)$ Givens-Rotationen mit konstantem Aufwand.

$\hookrightarrow A_{ij} = 0$



5

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 $\begin{cases} \cdot v_1 = A_1 + \alpha_1 e_1 \\ \cdot \alpha_1 = \text{sgn}(A_{11}) \|A_1\|_2 \end{cases}$

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

→ $R = \underbrace{Q_{n-1} \cdots Q_1}_Q A = \begin{bmatrix} * & * & \\ 0 & * & \\ \vdots & \ddots & * \\ 0 & & \end{bmatrix}$ obere A -Matrix

→ $R = \tilde{Q} A \Rightarrow \tilde{Q}^{-1} R = A \Leftrightarrow \tilde{Q}^T R = A \Leftrightarrow \boxed{Q = \tilde{Q}^T}$

→ Kann auch für Hessenberg-Trafo genutzt werden indem man nur die entsprechende Untermatrix betrachtet

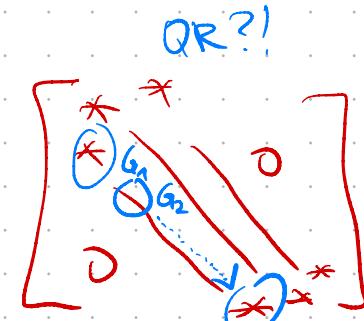
$$A = \begin{bmatrix} * & \cdots & * \\ * & \cdots & * \\ \vdots & \cdots & * \\ * & \cdots & * \end{bmatrix} \text{ und } P_1 A P_1^{-1} = \begin{bmatrix} * & & \\ * & * & \\ \vdots & \ddots & * \\ 0 & 0 & \ddots \\ 0 & 0 & 0 & * & \cdots & * \end{bmatrix} \text{ und usw...}$$

$\left[\begin{array}{c|c} I_n & 0 \\ \hline 0 & P_1 \end{array} \right]$

↑ wie oben

Recall: Givens Rotationen : Drehung

$$G(i,j,\theta) = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & c & \cdots & -s & 0 \\ 0 & -s & \cdots & c & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \text{ mit } \begin{cases} c = \cos \theta = \frac{a_{ii}}{\rho} \\ s = \sin \theta = \frac{a_{ij}}{\rho} \\ \rho = \sqrt{a_{ii}^2 + a_{ij}^2} \end{cases}$$



$$G(n,1) \cdot A = \begin{bmatrix} * & * \\ \vdots & \vdots \\ 0 & * \end{bmatrix} \rightarrow \text{Kann für QR benutzt werden}$$

→ Insbesondere QR für Hessenberg Matrizen in $O(n-1)$ Givens-Rotationen mit konst oder variablen Eintrag (i,i) wird zu Null

→ HA

QR?

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 $k = 1, \dots, m$:
 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 $Q^{(m)} = Q_m \cdots Q_1 Q_1$

`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

Symmetric hessenberg matrices are tridiagonal! (only diag plus upper 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 rotations:

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

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

with

$$r = \sqrt{A_{ii}^2 + A_{jj}^2}$$

$$s = -A_{ij}/r$$

$$c = A_{jj}/r$$

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

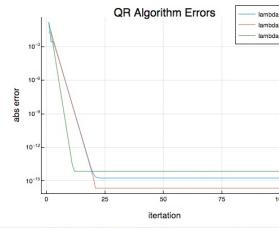
`givens_rotation_matrix` (generic function with 1 method)

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

Check Convergence

```
begin
    N = 100
    tape = Array[]
    for k=1:N
        push!(tape, sort(qra_general(A, k)[1], rev=false)) # don't do this, very inefficient!
    end
end

errors =
> Array{Float64,1}[float64(0.137638, 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[1].errors[1], errors[1].errors[1].errors[2], errors[1].errors[1].errors[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

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

Algorithm [1]:

- Given $A \in \mathbb{R}^{n \times n}$
 - For $k = 1 \dots n-2$ do:
 - $[v, \beta] \leftarrow \text{house}(A(k+1:n, k))$
 - $A(k+1:n, k:n) \leftarrow (I - \beta v v^T) A(k+1:n, k:n)$
 - $A(1:n, k+1:n) \leftarrow A(1:n, k+1:n) (I - \beta v v^T)$
- with Householder reflection vector v and weight $\beta = 2/(v^T v)$.
- [1]: <https://www.tu-chemnitz.de/mathematik/nma/lehre/la-2015/Folien/bla-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-1
        v, B = householdervec(A(k+1:n, k))
        A[k+1:n, k:n] = (I(n-k) - B * v * v^T) * A[k+1:n, k:n]
        A[1:n, k+1:n] = A[1:n, k+1:n] * (I(n-k) - B * v * v^T)
    end
    return A
end
```

Speed Check

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

- Homework: Improve this.

QRA General:
0.160218 seconds (30.99 k allocations: 51.228 MiB, 4.84% gc time)
QRA Own:
0.417899 seconds (99.22 k allocations: 762.264 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 A1, Q1 = qra_general(C, M)
    println("QRA Own:")
    @time A2, Q2 = qra_symm(C, M)
    end
```