Gram Schmidt Orthogonalization and Applications with Eigenmath

Dr. Wolfgang Lindner
dr.w.g.Lindner@gmail.com
Leichlingen, Germany
2021

Contents

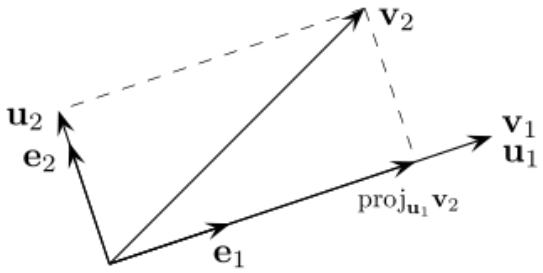
1 The GRAM–SCHMIDT orthogonalization process

Given an arbitrary *k-frame* (linear–independent set of vectors) $(v_1, ..., v_k)$ of the n-dimensional vector space V the GRAM SCHMIDT[1] orthogonalization process constructs a new k-frame $(u_1, ..., u_k)$, whose members are mutually orthogonal to each other and spans the same k-dimensional subspace of V.

To spend the following procedure a geometric insight, we remember at the geometric concept of the *orthogonal* **proj***ection* function $proj\colon V \to V$ defined by

$$\mathrm{proj}_{\mathbf{u}}(\mathbf{v}) = \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle} \mathbf{u}$$

where $\langle \mathbf{u}, \mathbf{v} \rangle$ denotes the *inner product* of the vectors $\mathbf{u}$ and $\mathbf{v}$.
The function *proj* projects the vector $\mathbf{v}$ orthogonally onto the line spanned by vector $\mathbf{u}$.[2]



u_1: the process starts with vector $u_1 := v_1$.
Figure 1: u_2: the process then constructs $u_2 := v_2 - proj_{u_1} v_2$.
e_1, e_2: finish with ortho*normal*ization of (u_1, u_2), see §2.2.

Therefore the GRAM–SCHMIDT process proceeds as follows:

$$\begin{aligned}
\mathbf{u}_1 &= \mathbf{v}_1, \\
\mathbf{u}_2 &= \mathbf{v}_2 - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_2) \\
\mathbf{u}_3 &= \mathbf{v}_3 - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_3) - \mathrm{proj}_{\mathbf{u}_2}(\mathbf{v}_3), \\
&\vdots \\
\mathbf{u}_k &= \mathbf{v}_k - \sum_{j=1}^{k-1} \mathrm{proj}_{\mathbf{u}_j}(\mathbf{v}_k)
\end{aligned}$$

Result: $(u_1, ..., u_k)$ is the required new *orthogonal* frame with same span.
Click here to see an 3D animation.
We now concretize this process as an EIGENMATH procedure.

[1] The author was student of Prof. Peter DOMBROWSKI, who was student of Erhard SCHMIDT.
[2] The picture is found at [13].

2 The Gram–Schmidt algorithm in Eigenmath

The following Eigenmath algorithm implements the Gram–Schmidt orthogonalization for Euclidean vector spaces, i.e for vector spaces equipped with an inner product $\langle \mathbf{u}, \mathbf{v} \rangle$. The example codes included in this vignette can be copied and pasted (CTRL-C and CTRL-V) into the Eigenmath Online interactive input box ☐RUN and then executed by pressing the RUN button. You can make your own variations, since this online box is editable.

2.1 Orthogonalization

The vectors of the frame $(v_1, ..., v_k)$ are input as rows of a matrix `v`, so that `v[j]` is the j–th vector v_j of the frame. This vector is replaced by an orthonormal vector `u[j]`, which is saved as the j–th row of the new matrix `u`.
We implement first two special low–dimensional versions and then the general version.

LEXICON	*Math*	Eigenmath
inner product in $\mathbb{R}^n$	$\langle \mathbf{a}, \mathbf{b} \rangle$	`dot(a,b)`
alias		`inner(a,b)`

2.1.1 2-dim Gram–Schmidt orthogonalization

```
# EIGENMATH
GramSchmidt2(v)= do(
    u=zero(2,2),          -- u = ((0,0),(0,0))
    u[1] = v[1],
    u[2] = v[2] - inner(u[1],v[2]) / inner(u[1],u[1])*u[1],
    u)

# TEST EXAMPLE
B = ((1,2),               -- v1 = (1,2)
     (3,4))               -- v2 = (3,4)
GramSchmidt2(B)           -- = u
```

Eigenmath output:

$$\begin{bmatrix} 1 & 2 \\ \frac{4}{5} & -\frac{2}{5} \end{bmatrix}$$

i.e. $u_1 = (1, 2)$ and $u_2 = (0.8, -0.4)$. We have $u_1 \perp u_2$, because $1 \cdot 0.8 + 2 \cdot (-0.4) = 0$.
▷*Mark & Copy the blue code lines, then paste it into this Online box and press RUN.*[3]

Exercise 1. Reformulate `GramSchmidt` using the function `proj(v,u) = dot(u,v)/dot(u,u)*u`.

[3]Do not forget to *click into the Online form* to give it the focus. You have the focus, if the Eigenmath Online frame change to blue. Please check, wether all input lines are pasted with the right NEW LINE ending! Otherwise correct the pasting online.

2.1.2 3-dim Gram–Schmidt orthogonalization

```
# EIGENMATH
GramSchmidt3(v)= do(
  u = zero(3,3),
  u[1] = v[1],
  u[2] = v[2] - inner(u[1],v[2])/inner(u[1],u[1])*u[1],
  u[3] = v[3] - inner(u[2],v[3])/inner(u[2],u[2])*u[2] -
                inner(u[1],v[3])/inner(u[1],u[1])*u[1],
  u)

B = ((-1,-2,0),(-1,0,-1),(2,-1,-2))
GramSchmidt3(B)
```

Eigenmath output:

$$\begin{bmatrix} -1 & -2 & 0 \\ -\frac{4}{5} & \frac{2}{5} & -1 \\ 2 & -1 & -2 \end{bmatrix}$$

2.1.3 k-dim Gram–Schmidt orthogonalization

```
# EIGENMATH
proj(v,u) = dot(u,v)/dot(u,u)*u   -- project v on u

GramSchmidt(B) = do(
  odim1 = dim(B,1),               -- (1)
  odim2 = dim(B,2),               -- (2)
  u  = zero(odim1,odim2),         -- (3)
  u[1]= B[1],                     -- (4)
  for( k,2,odim1,
       u[k] = B[k] - sum(j,1,k-1, proj(B[k],u[j]) ) ),     -- (5)
  u)                              -- (6)


B2 = ((1,2),(3,4))
u = GramSchmidt(B2)
u
dot(u, transpose(u))              -- (7)


B3 = ((-1,-2,0),(-1,0,-1),(2,-1,-2))
u = GramSchmidt(B3)
u
dot(u, transpose(u))
```

EIGENMATH output of second test input $B3$:

$$u = \begin{bmatrix} -1 & -2 & 0 \\ -\frac{4}{5} & \frac{2}{5} & -1 \\ 2 & -1 & -2 \end{bmatrix}$$

$$\begin{bmatrix} 5 & 0 & 0 \\ 0 & \frac{9}{5} & 0 \\ 0 & 0 & 9 \end{bmatrix}$$

Mark ▷ Copy the blue code lines. ▷ Then click here and Paste it into the input box □RUN.

Comment. The identifier `odim1` in (1) takes the number of vectors in the span B and `odim2` in (2) saves the dimension of the vectors in the vector space in question. In (3) we construct the suitable container matrix `u` as a zero matrix of dimension *odim1* × *odim2*, because in general this matrix is not quadratic. In (4) starts the procedure and (5) implements the u_k projection formula using a for loop construction. (6) gives back the filled solution matrix. Line (7) checks the orthogonality of the rows of `u`.

Exercise 2. (Gram–Schmidt via GAUSS-JORDAN.)
Write the vectors $v_1, ..., v_k$ as a matrix A and apply Gaussian elimination to row reduce (RREF) the augmented matrix $\left[A \star A^{\mathsf{T}} | A\right]$. Then the orthogonalized vectors u show up in place of A. *Hint*: use package `gjBox.txt` [5] and/or study the examples in [2, p. 20 ff].

2.2 Orthonormalization

To get an ortho*normal* frame **e** from the intermediate result **u** of mutually ortho*gonal* vectors we only have to normalize the members of the orthogonal frame **u**, i.e.

$$\mathbf{e}_k := \frac{\mathbf{u}_k}{\|\mathbf{u}_k\|}$$

LEXICON	*Math*	EIGENMATH
normalize $w \in \mathbb{R}^k$:	$\|\mathbf{w}\|$	`abs(w)`

```
# EIGENMATH  ONB = OrthoNormalBasis
normalize(u) = u / abs(u)

ONB(B) = do( onb=B,          -- input OrthoGonal frame B
             odim=dim(B,1),
             for(k,1,odim, onb[k] = B[k] / abs(B[k])),
             onb)

O = ((1,2),(4/5,-2/5))       -- orthogonal basis in rows as input
onb = ONB(O)
onb                          -- result OrthoNormal basis in rows
dot(onb,transpose(onb))      -- test for orthonormality
```

EIGENMATH output:

$$o_{nb} = \begin{bmatrix} \frac{1}{5^{1/2}} & \frac{2}{5^{1/2}} \\ \frac{2}{5^{1/2}} & -\frac{1}{5^{1/2}} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Mark ▷ Copy the blue code lines. ▷ Then click here and Paste it into the input box □RUN.

2.3 GRAM-SCHMIDT toolbox

The routines `GramSchmidt2(.)`, `GramSchmidt3(.)`, `GramSchmidt(.)` and `ONB(.)` are collected in a toolbox named `gsBox.txt`. This box is downloadable at [6].
We show its use by

Example 1. We check the example at WIKIPEDIA [13]▷Dansk language page.

```
run("Downloads/gsBox.txt")

v  = ((1,1,1,1),(1,1,0,0),(3,1,1,-1))
og = GramSchmidt(v)    -- make v orthogonal
og
onb=ONB(og)            -- make og orthonormal
onb                    -- read off as rows
```

EIGENMATH output:

$$o_g = \begin{bmatrix} 1 & 1 & 1 & 1 \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

$$o_{nb} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{bmatrix}$$

Mark ▷ Copy the blue code lines. ▷ Then click here and Paste it into the input box □RUN.

Exercise 3. Check all the concrete examples on the German, English, Spanish and Nederland language pages of [13] by use of the `gsBox.txt`.

3 Applications

3.1 Gram-Schmidt orthonormalization in Hilbert space $L_2[0,1]$

We run the second example of WikipediA [13]▷Nederland language page. In the 2D real vector space of the linear functions $f(t) = p + qt$ on the interval $[0,1]$, we have the inner product

$$\langle f_1, f_2 \rangle = \int_0^1 f_1(t) f_2(t)\, \mathrm{d}t$$

Task: orthonormalize the functions $f_1(x) = 1, f_2(x) = x$.
Solution. We implement the adapted inner product of space $L_2[0,1]$.

```
innerL2(f1,f2) = defint(f1*f2,x,0,1)  -- inner product as DEFinite INTegral

GramSchmidtL2(B) = do(
    O=(0,0),                           -- container with two zero functions
    O[1]=B[1],
    O[2]=B[2] - innerL2(O[1],B[2]) / innerL2(O[1],O[1])*O[1],
    O)                                 -- result as rows

B  = (1,x)
Bg = GramSchmidtL2(B)                 -- make B orthoGonal resp. innerL2(.)
Bg

normL2(f) = sqrt(defint(f*f,x,0,1))   -- norm in L2[0,1]

onbL2(B) = do(onb=B,
            odim=dim(B,1),
            for(k,1,odim, onb[k] = B[k]/normL2(B[k])),
            onb)

Bn=onbL2(Bg)                          -- make Bg orthoNormal resp. innerL2
Bn
```

Eigenmath output:

$$B_g = \begin{bmatrix} 1 \\ x - \frac{1}{2} \end{bmatrix}$$

$$B_n = \begin{bmatrix} 1 \\ -3^{1/2} + 2\,3^{1/2} x \end{bmatrix}$$

Mark ▷ Copy the blue code lines. ▷ Then click here and Paste it into the input box ☐RUN.

Exercise 4. Orthonormalize the set $\{1, x, x^2, x^3\}$.

3.2 GRAM-SCHMIDT orthogonalization in Clifford Algebra $c\ell(n)_{n=2,3}$

Task: orthogonalize the linear-independent vectors $a1 = (3,1)$ and $a2 = (2,2)$ using CLIFFORD algebra methods.

Solution: We implement the GRAM–SCHMIDT process for the CLIFFORD algebra $c\ell(2)$.

```
run("Downloads/EVA2.txt")              -- load package EVA2
cl(2)                                  -- invoke Clifford constructor
a1 = 3e1+1e2                           -- input data
a2 = 2e1+2e2                           -- second multivector

orth2(u,v) = do(
   isVector(u), isVector(v),           -- (1)
   test( outp(u,v) =0, print("not a frame",stop)),  -- (2)
   u0 = u,
   v0 = gp(outp(v,u0), inverse(u0)),   -- (3)
   ( (u0[2],u0[3]),                    -- (4)
     (v0[2],v0[3])) )

orth2(a1,a2)                           -- (5)

-- verify the result y:
y1 =    3e1+  1e2
y2 = -0.4e1+1.2e2
inp(y1,y2)                             -- (6)
inp(y1,y1)                             -- (7)
```

Result:

$$\begin{bmatrix} 3 & 1 \\ -0.4 & 1.2 \end{bmatrix}$$

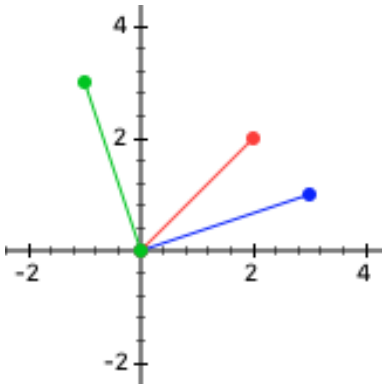


Figure 2: Basis $A = \left[\begin{smallmatrix} 3 & 1 \\ 2 & 2 \end{smallmatrix}\right]$ and its orthogonalization $Y = \left[\begin{smallmatrix} 3 & 1 \\ -0.4 & 1.2 \end{smallmatrix}\right]$.
The basis vectors (rows of A) are plotted as points in plane $\mathbb{R}^2$.
Blue: vector $a_1 = [3,1] \perp [-0.4, 1.2] = y_2$

Comment. We comment a little bit about the procedure `orth2`, which is the 2D pendent of `GramSchmidt`. In (1) we check, whether the input vectors u, v are indeed correct multivectors in `cl(2)`. Line (2) checks, whether the input vectors are linear independent by calculation of the area of the parallelogram with edges u and v – `outerp(u,v)` being the wegde product $u \wedge v$. Line (3) determines the new orthogonal vector v_0 by – picturally spoken – 'dividing the area by the old edge giving the orthogonal edge'. In (4) we return the new orthogonalized basis with its vectors as rows of the matrix. We pick only the 2^{th} and 3^{rd} coordinate because this is the vector part of the 4D CLIFFORD algebra number. In (6) we check the orthogonality using the inner product of the CLIFFORD algebra.

Exercise 5. Orthonormalize the linear independent set (y_1, y_2) of multivectors using the CLIFFORD algebra $c\ell(2)$ and the `EVA2` package function `normalize(u)`.

Exercise 6. Verify the result by invoking the GRAM–SCHMIDT process for $\mathbb{R}^2$.

```
run("Downloads/gsBox.txt")        -- load toolbox GramSchmidt
Y = GramSchmidt( ((3,1),(2,2)) )  -- input vectors as rows
Y                                 -- resulting ortho basis as rows
dot(Y, transpose(Y))              -- check orthogonality of new vectors
```

EIGENMATH output: `Y=((3,1),(-2/5,6/5))`

- Orthonormalize the result Y.

Exercise 7. Orthonormalize the linear independent set of vectors in §2.1.2 using the CLIFFORD algebra $c\ell(3)$ and the `EVA2` package functions `orthogonal(u,v,w)` and `normalize(u)`.

3.3 Excurs – The reciprocal (alias: dual) basis in $\mathbb{R}^2$ and $\mathbb{R}^3$

By a *pair* (B, R) *of bi-orthogonal bases* we mean a basis $B = (b_1, .., b_n)$ together with its correspondent *dual* basis $R = (r_1, .., r_n)$. In the 2D case such a pair of basis is characterized as follows: the first vector in the basis is orthogonal to the second vector in the dual basis and the second vector in the basis is orthogonal to the first vector in the dual basis, see Fig.2. Furthermore, the corresponding vectors in the two bases have an inner product equal to 1. Symbolically, "inner"-multiplying a dual vector r_i on a vector b_j in the original basis gives $r_i \cdot b_j = \delta_{ij} = \{^{1 \quad \text{if } i=j}_{0 \quad \text{if } i \neq j}\}$.
We may represent a basis by a matrix B whose rows are the basis vectors. In an orthonormal basis the basis vectors are mutually orthogonal, i.e. $B * B^t = I_n$ with I_n the n×n identity matrix. In contrast, for an bi-orthonormal basis pair (B, R) we have the defining relation $R^t * B = I_n$. Therefore it follows $R = (B^{-1})^t$.[4]

LEXICON	*orthogonal basis B*	dual basis pair (B,R)
orthogonal basis B:	$B * B^t = I_n$	
biorthogonal pair B, R of basis:		$R^t * B = I_n$

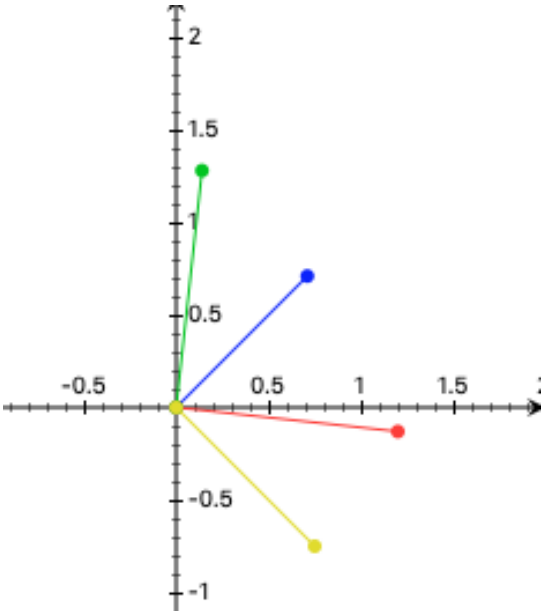


Figure 3: Basis $B = \begin{bmatrix} 0.71 & 0.71 \\ 1.20 & -0.13 \end{bmatrix}$ and its dual (reciprocal) basis $R = \begin{bmatrix} 0.14 & 1.28 \\ 0.75 & -0.75 \end{bmatrix}$.
Blue: first vector $b_1 = [0.71, 0.71] \perp [0.75, -0.75] = r_2$ second dual.
Red: second vector $b_2 = [1.2, -0.13] \perp [0.14, 1.28] = r_1$ first dual.

[4] see e.g. `https://en.wikipedia.org/wiki/Dual_basis`

3.3.1 The 2D reciprocal basis

Determine for the basis $B = \begin{bmatrix} 0.71 & 0.71 \\ 1.20 & -0.13 \end{bmatrix}$ its dual (reciprocal) basis R.[5]

Solution N^o**1**: We use the defining relation.

```
b1 = (0.71, 0.71)            -- first basis vector as row
b2 = (1.2 ,-0.13)            -- second basis vector as row

B = (b1,b2)                  -- basis vectors as rows
B                            -- show basis
                             -- Rt*B=I defining relation
R = transpose(inv(B))        -- <= therefore R= (1/B)^t
R

dot(transpose(R),B)          -- check relation Rt*B=I
```

Mark ▷ Copy the blue code lines. ▷ Then click here and Paste it into the input box □RUN.
EIGENMATH output:

$$R = \begin{bmatrix} 0.137668 & 1.27078 \\ 0.75188 & -0.75188 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Solution N^o**2**: We calculate inside the CLIFFORD algebra $c\ell(2)$ alias $\mathbb{G}^2$, see [3].

```
run("Downloads/EVA2.txt")
cl(2)

reci22(u,v) = do(                -- short version without input check
   uv = outp(u,v),
   ru = gp( v,outp(v,u) ) / magnitude(uv)^2,
   rv = gp( u,outp(u,v) ) / magnitude(uv)^2,
   ( (ru[2],ru[3]),              -- 1st dual basis vector
     (rv[2],rv[3])) )            -- 2nd dual basis vector as row

b1 = 0.71e1+0.71e2               -- 1st original basis vector
b1
b2 = 1.2e1-.13e2                 -- 2nd original basis vector
b2

"reciprocal basis via Clifford Algebra:"
reci22(b1,b2)                    -- read of dual basis as rows:
```

[5]To check the result we take the data from
https://www.wolframcloud.com/objects/demonstrations/APairOfBiorthogonalBasesInTheRealPlane-source.nb

```
r1 = 0.1376e1+1.2707e2        -- 1. dual basis vector
r2 = 0.7518e1-0.7518e2        -- 2. dual basis vector as row

inp(b1,r2)                    -- check orthogonality
inp(b2,r1)
```

Mark ▷ Copy the blue code lines. ▷ Then click here and Paste it into the input box □RUN.
EIGENMATH output:

reciprocal basis via Clifford Algebra:

$$\begin{bmatrix} 0.137668 & 1.27078 \\ 0.75188 & -0.75188 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

3.3.2 The 3D reciprocal basis

Task: Find the dual basis for the space $\mathbb{R}^3$ whose basis are given by:[6]
$\mathbf{a}_1 = (5, -2, 6), \mathbf{a}_2 = (-3, -1, -4), \mathbf{a}_3 = (9, -5, 7)$ or coded in matrix form as

$$A = (\mathbf{a}_1, \mathbf{a}_1, \mathbf{a}_1) = \begin{bmatrix} 5 & -2 & 6 \\ -3 & -1 & -4 \\ 9 & -5 & 7 \end{bmatrix}$$

Solution **N°1:** We use the following fact: For a given basis $A = (\vec{a}_1, \vec{a}_2, \vec{a}_3)$, we can find the *bi-orthogonal* (*dual* alias *reciprocal*) basis $B = (\vec{b}_1, \vec{b}_2, \vec{b}_3)$ by

$$\vec{b}_1 = \frac{\vec{a}_2 \times \vec{a}_3}{\vec{a}_1 \bullet (\vec{a}_2 \times \vec{a}_3)} \tag{3.1}$$

$$\vec{b}_2 = \frac{\vec{a}_3 \times \vec{a}_1}{\vec{a}_2 \bullet (\vec{a}_3 \times \vec{a}_1)} \tag{3.2}$$

$$\vec{b}_3 = \frac{\vec{a}_1 \times \vec{a}_2}{\vec{a}_3 \bullet (\vec{a}_1 \times \vec{a}_2)} \tag{3.3}$$

where the denominator $V := \vec{a}_1 \bullet (\vec{a}_2 \times \vec{a}_3) = \vec{a}_1 \bullet (\vec{a}_2 \times \vec{a}_3) = \vec{a}_3 \bullet (\vec{a}_1 \times \vec{a}_2)$ is the volume of the parallelepiped formed by the basis vectors $\vec{a}_1, \vec{a}_2$ and $\vec{a}_3$.[7]
This is one of those uncomfortable formulas in which you should be supported by a CAS like EIGENMATH. Let's go ahead.

[6] To check the result we take the data from `https://es.wikipedia.org/wiki/Base_dual`

[7] Function `Box(.)` is invariant to a cyclical exchange of its arguments, and which is equal to the determinant of the matrix which is formed from the basis vectors. – See e.g. [4].

```
##### RECIPROCAL via CROSS in R^3
a1 = ( 5,-2, 6)
a2 = (-3,-1,-4)
a3 = ( 9,-5, 7)

Box(A,B,C) = dot(A, cross(B,C))
Box(a1,a2,a3)

b1 = cross(a2,a3) / Box(a1,a2,a3)
b2 = cross(a3,a1) / Box(a1,a2,a3)
b3 = cross(a1,a2) / Box(a1,a2,a3)

b1                          -- 1st dual basis vector
B = (b1,b2,b3)              -- dual basis in rows of B
B
```

EIGENMATH output:

$$B = \begin{bmatrix} -\frac{9}{13} & -\frac{5}{13} & \frac{8}{13} \\ -\frac{16}{39} & -\frac{19}{39} & \frac{7}{39} \\ \frac{14}{39} & \frac{2}{39} & -\frac{11}{39} \end{bmatrix}$$

Mark ▷ Copy the blue code lines. ▷ Then click here and Paste it into the input box □RUN.

Solution **N°2:** (Using the CLIFFORD algebra $c\ell(3)$ alias $\mathbb{G}^3$, see [3])

```
##### RECIPROCAL via CLIFFORD Algebra cl(3)
run("Downloads/EVA2.txt")
cl(3)

reciprocal(u,v,w) = do(                 -- (1)
   u123 = outp(outp(u,v),w),            -- equivalent to Box()
   ru = gp(outp(v,w), inverse(u123)), -- ru is reciprocal to u
   rv = gp(outp(w,u), inverse(u123)), -- (2)
   rw = gp(outp(u,v), inverse(u123)),
   ( (ru[2],ru[3],ru[4]),  -- 3x3 matrix with dual basis
     (rv[2],rv[3],rv[4]),  --     saved in its rows
     (rw[2],rw[3],rw[4]))  )

a1 = 5e1-2e2+6e3      -- a1 expressed in vector basis (e1,e2,e3) of G^3
a2 =-3e1-1e2-4e3
a3 = 9e1-5e2+7e3

B = reciprocal(a1,a2,a3)   -- dual Basis as rows
B = transpose(B)           -- dual basis as columns
```

```
B

R=((-27/39,-15/39,  24/39),  -- Referenz matrix
   (-16/39,-19/39,  7/39),
   ( 14/39,  2/39,-11/39))
float(R)                     -- (3)
```

EIGENMATH output:

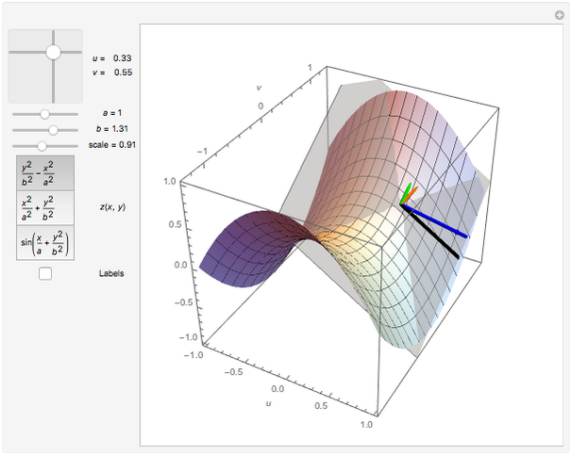
$$B = \begin{bmatrix} -0.692307 & -0.410256 & 0.358974 \\ -0.384615 & -0.487179 & 0.051282 \\ 0.615384 & 0.179487 & -0.282051 \end{bmatrix}$$

Mark ▷ Copy the blue code lines. ▷ Then click here and Paste it into the input box □RUN.

Comment. Function `reciprocal` determines the dual basis for a given basis A. We cite here a shortened version of the function in EIGENMATH package `EVA2.txt`[8]. (2) is the Geometric Algebra equivalent of the EIGENMATH term fo $b2$ in solution *1*. Observe, how the term `outerp(w,u)`, i.e. the wedge product $w \wedge u$ of w with u substitutes the traditional cross product in $\mathbb{R}^3$! In (3) we write the result of solution.1 using floating point values in order to verify the calculation.

Remark. The dual basis plays a role in Differential Geometry. A nice demo can be found at `https://demonstrations.wolfram.com/TheDualBasisForTheTangentSpaceOfA2DSurface/`

> ”This Demonstration illustrates the tangent plane basis and the reciprocal basis at points along a few sample surfaces. The tangent plane itself is shown and a control is provided to alter the position of the tangent plane along the surface. This lets you see the normality of the tangent plane basis vectors and their alternate indexed reciprocal vector. You can also compare the orientations of the tangent plane basis vectors with their reciprocals, and see how these pairs of vectors are not necessarily collinear. [P. Joot]



[8] See [1]

3.4 Eigenvalues and Eigenvectors

We give an raw implementation of the iterative so-called *QR method* to calculate the eigenvalues and eigenvectors of a symmetric quadratic real matrix A. We use our toolbox `gsBox.txt`.[9]

```
run("Downloads/gsBox.txt")      -- for use of GramSchmidt and ONB

A = ((1,3,4),(3,1,2),(4,2,1))   -- real quadratic symmetric matrix
A
A = float(A)                    -- make entries of A numerical decimal
V = unit(3)

for(k,1,100,                -- QR iteration to compute eigenvalues and EV's
    AT  = transpose(A),
    ATo = GramSchmidt(AT),  -- make AT  orthogonal
    ATn = ONB(ATo),         -- make ATo orthonormal
    U   = transpose(ATn),
    A   = dot(transpose(U),A,U),
    V   = dot(V,U) )

dot(U,transpose(U))              -- check U*Ut = unit(3)
A                                -- new A with eigenvalues on diagonal

"Eigenvalues:"
lambda = (A[1,1], A[2,2], A[3,3])
lambda

"Eigenvectors: "
V = transpose(V)
V                  --  read off Eigenvectors as rows
```

EIGENMATH output:

Eigenvalues:

$$\lambda = \begin{bmatrix} 7.07467 \\ -3.18788 \\ -0.886791 \end{bmatrix}$$

Eigenvectors: (only if A symmetric)

$$V = \begin{bmatrix} 0.634577 & 0.505785 & 0.584374 \\ -0.757161 & 0.255232 & 0.601302 \\ 0.154979 & -0.824038 & 0.544925 \end{bmatrix}$$

Mark ▷ Copy the blue code lines. ▷ Then click here and Paste it into the input box □RUN.

[9] We adapt here the solution of G. WEIGT [12] in order to show, that the functions `GramSchmidt(.)` and `ONB(.)` are involved. Therefore this version also works for arbitrary dimension of A.

Exercise 8. (Check for Eigenvector property)
Following the last EIGENMATH session, check wether the eigenvector property is fulfilled.

```
-- Is lambda1 Eigenvalue for Eigenvector V1?
A = ((1,3,4),(3,1,2),(4,2,1))
lambda[1]   -- first Eigenvalue
V[1]        -- first Eigenvector
dot(A,V[1]) - lambda[1]*V[1]   -- should be (0,0,0)
```

Exercise 9. (Reproduce an example calculation of WOLFRAMAlpha)
Do "gram schmidt `{{1,1,1},{2,1,0},{5,1,3}}`".
See `https://www.wolframalpha.com/input/?i=gram+schmidt+%7B%7B1%2C1%2C1%7D%2C%7B2%2C1%2C0%7D%2C%7B5%2C1%2C3%7D%7D&js=off`.

3.4.1 Eigenvalues of ellipse with equation $\frac{(x-2)^2}{9} + \frac{(y-1)^2}{4} = 1$

This ellipse has center $(x_o, y_o) = (2, 1)$, half major axis $a = 3$ and half minor axis $b = 2$. What are the eigenvalues of the ellipse? (We pretend we don't know it.)

```
# EIGENMATH: ellipse (x-2)^2/9+(y-1)^2/4=1
a=3
b=2
xo=2
yo=1
u = (xo,yo) + (a*cos(t),b*sin(t))
xrange = (-5,5)
yrange = (-5,5)
trange = (0,2pi)
draw(u,t)
```

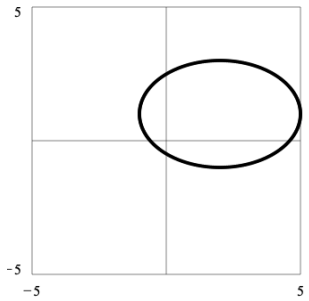


Figure 4: the ellipse $\frac{(x-2)^2}{9} + \frac{(y-1)^2}{4} = 1$.

We have:

$$\begin{aligned} \frac{(x-2)^2}{9} + \frac{(y-1)^2}{4} &= 1 \\ \begin{bmatrix} x-2 \\ y-1 \end{bmatrix}^T * \begin{bmatrix} \frac{1}{9} & 0 \\ 0 & \frac{1}{4} \end{bmatrix} * \begin{bmatrix} x-2 \\ y-1 \end{bmatrix} &= 1 \end{aligned}$$

The eigenvalues of the ellipse are p.d. the eigenvalues of the defining matrix $A = \begin{bmatrix} 1/9 & 0 \\ 0 & 1/4 \end{bmatrix}$. Therefore:

```
run("Downloads/gsBox.txt") -- load package for QR() function

A = ((1/9,0),(0,1/4))
A = float(A)               -- change A to decimal values, ONLY REALs
V = unit(2)                -- because A is of typ 2x2
QR()                       -- QR aufruf
"Eigenvalues:"             -- print Eigenvalues of transformed matrix A:
for(i,1,dim(A,1), print(A[i,i]))
```

EIGENMATH output: 0.11111 0.25
i.e. $\lambda_1 = \frac{1}{9}$ and $\lambda_2 = \frac{1}{4}$, because we know that the half axes values are $\frac{1}{\sqrt{\lambda_i}}$, λ_i being the i-th eigenvalue of A.

3.4.2 rotate ellipse with equation $\frac{(x-2)^2}{9} + \frac{(x-1)^2}{4} = 1$

We rotate the ellipse with major axis length $a = 3$ and minor axis length $b = 2$ with angle $\varphi = -30^o$.

```
# EIGENMATH (plot suggested by G. Weigt)
a=3
b=2
xo=2
yo=1

phi = -pi/6
R = ((cos(phi),-sin(phi)),(sin(phi),cos(phi)))    -- Rotation matrix

u = (a cos(t),b sin(t))   -- ellipse in polar coordinates at the origin
u = (xo,yo) + dot(R,u)    -- R*u rotates ellipse around the origin O=(0,0)
                          -- .. then we translate it to the center (xo,yo)
u                         -- the parametrization term of ellipse for draw

xrange = (-5,5)
yrange = (-5,5)
trange = (0,2pi)
draw(u,t)
```

$$u = \begin{bmatrix} \frac{3}{2} 3^{1/2} \cos(t) + \sin(t) + 2 \\ 3^{1/2} \sin(t) - \frac{3}{2} \cos(t) + 1 \end{bmatrix}$$

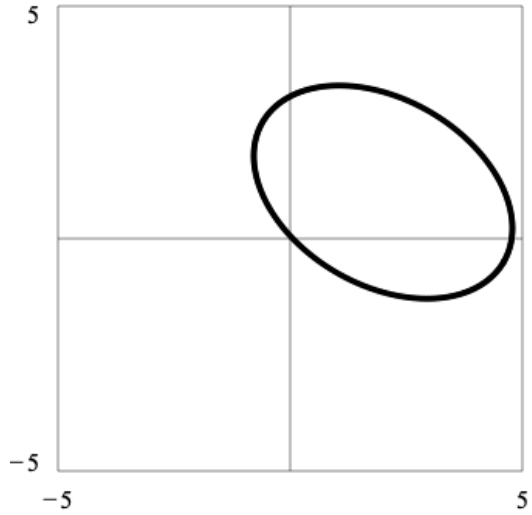


Figure 5: u: the polar coordinates term of rotated ellipse.
right: the rotated ellipse

Exercise 10. Do a qualitiy plot of the rotated ellipse using e.g. CalcPlot3D [8].

Exercise 11. What are the eigenvalues and eigenvectors of the rotated ellipse?

Hint: this leads to the discussion of the ellipse under the perspective of the so-called *principal axes transformation*. You find nice worked examples about this topic on the German wikipedia page:
`https://de.wikipedia.org/wiki/Hauptachsentransformation#Diagonalisierung_einer_symmetrischen_Matrix_(Hauptachsentheorem)`

3.5 GERSHGORIN circles for the location of Eigenvalues

The GERSHGORIN circle theorem marks a region in the complex plane that contains all the eigenvalues of a complex (or real) square matrix.

Theorem. (GERSHGORIN's Circle Theorem): Let $A = (a_{ij})$, $i, j = 1...n$ be a square matrix with entries from $\mathbb{C}$ (or $\mathbb{R}$).
The i-th GERSHGORIN circle is defined by its center and its radius as follows:
- The center point C_i is the i-th diagonal entry of the matrix A, i.e. $C_i = a_{ii}$
- The radius r_i is the sum of all entries in the i-th row except for the diagonal entry, i.e.

$$r_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \tag{3.4}$$

Then the eigenvalues of A lie in the union of the GERSHGORIN circular disks. □
We use formula (3.4) to implement the calculation of the GERSHGORIN circles in EIGENMATH for matrices over $\mathbb{R}$.

```
GerschgorinCircles(A) = do(
   check( dim(A,1)==dim(A,2) ),
   n = dim(A,1),
   for(i,1,n,
     R = sum(j,1,n, mag(A[i,j])) - mag(A[i,i]), -- radius
     C = (A[i,i],0),                            -- center
     print("Circle:", i,C,R)
   ))

A = ((2,0,-1),(-1,3,1),(0,2,5))
GerschgorinCircles(A)
```

EIGENMATH output for the last index:

Circle:

$i = 3$

$$C = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$$

$R = 2$

There are three circles, so we conclude that there are three eigenvalues of A near 2, 3 and 5 in a distance of 1, 2 and 2 resp.. We plot the situation with EIGENMATH.

3.5.1 Plot the GERSHGORIN circles with EIGENMATH

```
-- GERSCHGORIN circles
xrange = (0,8)
yrange = (-4,4)
```

```
trange = (0, 6*pi)                          -- (1)

f(t) = test(                                -- (2)
  t < 2 pi, (2,0) + 1*(cos(t),sin(t)),
  t < 4 pi, (3,0) + 2*(cos(t),sin(t)),
  t < 6 pi, (5,0) + 2*(cos(t),sin(t))
)       --  ^center ^radius of circle

draw(f,t)
```

Comment. In (1) we choose an interval of length $[0, 6\pi]$ for the parameter t which is three times as long as the parameter interval $[0, 2\pi]$ to draw one circle, so we are able to run through three circles one after the other given in polar coordinates. In (2) we define a piecewise defined function $f\colon [0, 6\pi] \to \mathbb{R}$ which glues the pieces together and draws the three circles in one plot:

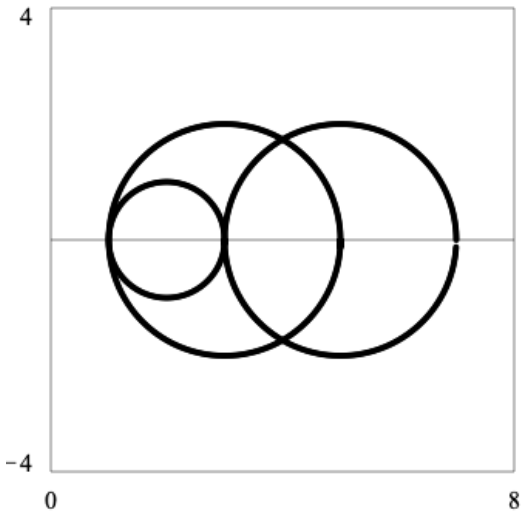


Figure 6: the three GERSCHGORIN circles to localize the tree eigenvalues.

3.5.2 Calculate the eigenvalues inside the GERSHGORIN circles

We now invoke our `QR()` routine to calculate the approximated (real parts of) eigenvalues.

```
run("Downloads/gsBox.txt")  -- load package for QR() function

A = ((2,0,-1),(-1,3,1),(0,2,5))
A = float(A)              -- change A to decimal values, ONLY REALs
V = unit(3)               -- because A is of typ 3x3
```

```
QR()                          -- invoke QR
"Eigenvalues:"                -- print out Eigenvalues of transformed matrix A
for(i,1,dim(A,1), print(A[i,i]))
```

EIGENMATH output:

Eigenvalues:

5.87513

2.07222

2.05265

Remark. Using other software like Maxima, askMathstudio, SageCell or WolframAlpha we find the approximated eigenvalues $\lambda_1 \approx 5.86$, $\lambda_2 \approx 2.06 + 0.72i$ and $\lambda_3 \approx 2.06 - 0.72i$.

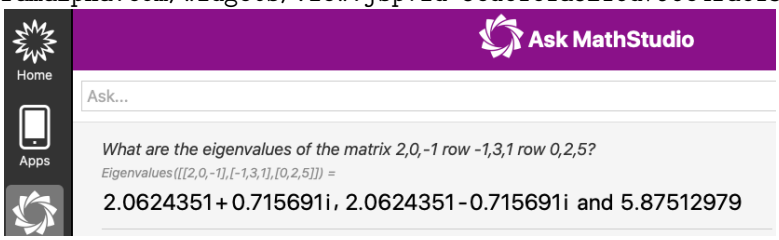
- askMathstudio: `http://mathstud.io/ask/` with the question
  Q: What are the eigenvalues of the matrix 2,0,-1 row -1,3,1 row 0,2,5?

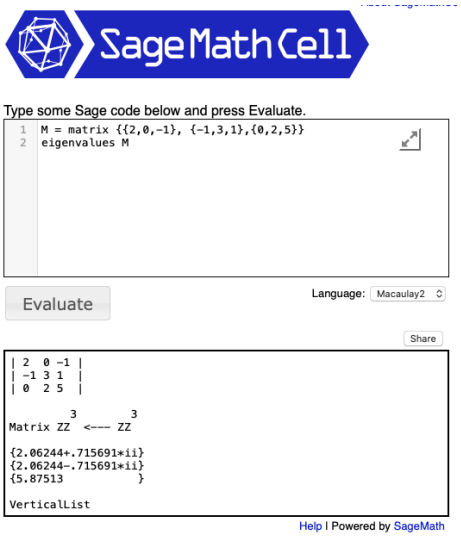
- SageCell: `https://sagecell.sagemath.org/?lang=macaulay2` with input

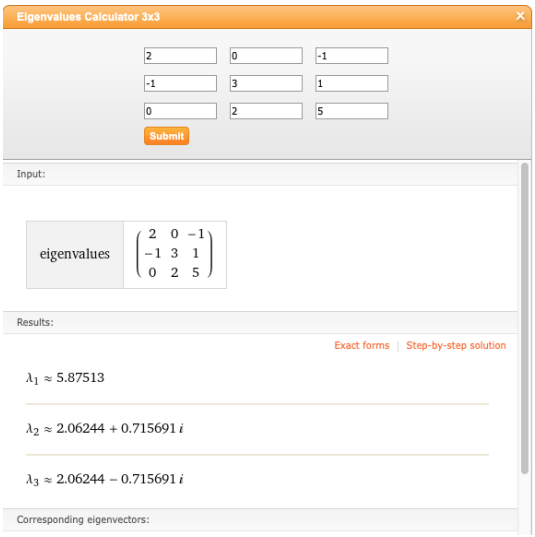
```
M = matrix {{2,0,-1}, {-1,3,1},{0,2,5}}
eigenvalues M
```

- WolframAlpha: fill in the form

`https://www.wolframalpha.com/widgets/view.jsp?id=5edc1c1a8216d796e4fac13256bab63d`







References

[1] Eyheramendy, B.(2012): *Clifford Algebra Calculator - The EVA package.*
url: http://beyhfr.free.fr/EVA2/crbst_38.html

[2] Lindner, W.(2020): *Regular Linear Systems. Gauss Algorithm. RREF.*
url: https://georgeweigt.github.io/Lindner/LAi2-Gauss-RREF.pdf

[3] Lindner, W.(2021): *Complex, Hyperbolic and Geometric Algebra Numbers.*
url: https://georgeweigt.github.io/Lindner/LAi6-complex-hyperbolics-quaternions-Geometric-Algebra.pdf

[4] Lindner, W.(2020): *Singular-Systems. Analytical-Geometry.*
url: https://georgeweigt.github.io/Lindner/LAi3-Singular-Systems-Analytical-Geometry.pdf

[5] Lindner, W.(2020): *Gauss Jordan package for* Eigenmath*: gjBox.txt*
url: https://georgeweigt.github.io/Downloads/gjBox.txt

[6] Lindner, W.(2021): *Gram Schmidt package for* Eigenmath*: gsBox.txt*
url: https://georgeweigt.github.io/Downloads/gsBox.txt

[7] Lounesto, P.(1987): *CLICAL, a calculator type computer program. Student Guide.*
url: https://users.aalto.fi/~ppuska/mirror/Lounesto/CLICAL.htm

[8] Seeburger, P. (2018): *CalcPlot3D.*
url: https://www.monroecc.edu/faculty/paulseeburger/calcnsf/CalcPlot3D/

[9] Steeb, W.-H., Lewien, D. (1992): *Algorithms and Computations with Reduce.*
Mannheim: Bibliographisches Institut.

[10] Weigt, G. (2021): Eigenmath *Homepage.*
url: https://georgeweigt.github.io

[11] Weigt, G. (2021): Eigenmath *online.*
url: https://georgeweigt.github.io/eigenmath-demo.html

[12] Weigt, G. (2021): Eigenmath *Eigenvalues demo.*
url: https://georgeweigt.github.io/eigenvalues.html

[13] WikipediA : Gram-Schmidt process.
url: https://en.wikipedia.org/wiki/Gram-Schmidt_process

[14] WikipediA : Principal Axes Theorem.
url: https://en.wikipedia.org/wiki/Principal_axis_theorem

[15] WikipediA : HauptAchsenTransformation
url: https://de.wikipedia.org/wiki/Hauptachsentransformation#Diagonalisierung_einer_symmetrischen_Matrix_(Hauptachsentheorem)