The Gram Schmidt Orthogonalisation process with Eigenmath

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

Contents

1 The Gram–Schmidt orthogonalization process **2**

2 The Gram–Schmidt algorithm in Eigenmath **3**

2.1 Orthogonalization . . . 3

2.2 Orthonormalization . . . 4

3 Examples **6**

3.1 Example . . . 7

1 The Gram–Schmidt orthogonalization process

Given an arbitrary k-frame (linear–independent set) $(v_1, ..., v_k)$ of the n-dimensional vector space V the Gram Schmidt 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 procedure a geometric insight, we remember first at the geometric concept of the *orthogonal* **proj***ection* function $proj: V \to V$ 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 u and v.
The function *proj* projects the vector v orthogonally onto the line spanned by vector u.
(The picture is found at `https://en.wikipedia.org/wiki/Gram-Schmidt_process`)

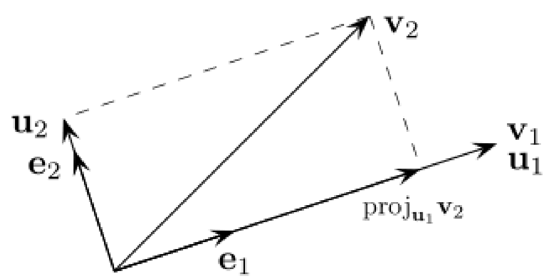

Figure 1: e_1, e_2: this 2D subspace (plane) is also spanned by (v_1, v_2).
u_1: the process starts with vector $u_1 := v_1$.
u_2: the process then goes to $u_2 := v_2 - proj_{u_1} v_2$.

The Gram–Schmidt process then 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.
Click here to see an 3D animation.
We now concretize the process as an Eigenmath procedure.

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 code included in this section can be copied and pasted (CTRL-C and CTRL-V) into the Eigenmath Online Demo–form at .. and then executed by pressing the RUN button. You can make your own variations, since this online form is editable.

2.1 Orthogonalization

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

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

2.1.1 2-dim Gram–Schmidt orthogonalization

```
# EIGENMATH
GramSchmidt2(B)= do(
    O=zero(2,2),
    O[1]=B[1],
    O[2]=B[2] - inner(O[1],B[2])/inner(O[1],O[1])*O[1],
    O)

B=((1,2),(3,4))
GramSchmidt2(B)
```

▷*Mark & Copy the blue code lines, then past it into the form and press RUN.*[1]

2.1.2 3-dim Gram–Schmidt orthogonalization

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

[1] Do not forget to click into the Online form to give it the focus.

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

2.1.3 k-dim Gram–Schmidt orthogonalization

```
# EIGENMATH
-- general GramSchmidt wL 29.3.21

GramSchmidt(B) = do(
  Odim = dim(B,1),
  O    = zero(Odim,Odim),
  O[1]= B[1],
  for( k,2,Odim,
       O[k] = B[k] -
               sum( j,1,k-1,
               dot(O[j],B[k])/dot(O[j],O[j]) * O[j]) ),
  O)

B2 = ((1,2),(3,4))
GramSchmidt(B2)

B3 = ((-1,-2,0),(-1,0,-1),(2,-1,-2))
GramSchmidt(B3)
```

via gj
If the rows $v1, vk$ are written as a matrix A A, then applying Gaussian elimination to the augmented matrix
$\left[AA^{\mathsf{T}}|A\right]$ will produce the orthogonalized vectors in place of A A. However the matrix A A T AA^{T} must be brought to row echelon form, using only the row operation of adding a scalar multiple of one row to another.[2] For example, taking $\mathbf{v}_1 = \begin{bmatrix} 3 & 1 \end{bmatrix}, \mathbf{v}_2 = \begin{bmatrix} 2 & 2 \end{bmatrix}$ as above, we have

2.2 Orthonormalization

```
# EIGENMATH
normalize(u) = test(abs(u)=0,u,u/abs(u))

-- round to n decimals after point
rnd(u,n) = do( test(number(n),Fixnum=n,Fixnum=4),
               float(floor(u*10^Fixnum+0.5)/10^Fixnum))

rnd(123.456789, 3)

-- OrthoNormalBasis/Bein
ONB(B)=do(onb=B,
```

```
          Odim=dim(B,1),
          for(k,1,Odim, onb[k] = B[k]/abs(B[k])), onb)

onb=ONB(B3)
onb

dot(onb[1],onb[1])
```

– GramSchmidt – B basis as matrix rows

3 Examples

3.0.1 Example: Ellipse with equation $\frac{(x-2)^2}{9} + \frac{(x-1)^2}{4} = 1$

has center $(x_o, y_o) = (2, 1)$ and half major axis $a = 3$ and half minor axis $b = 2$.

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

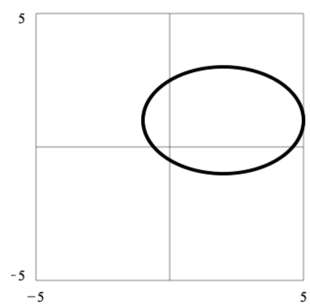

Figure 2: the ellipse $\frac{(x-2)^2}{9} + \frac{(x-1)^2}{4} = 1$.

3.0.2 Example: Ellipse with equation $\frac{(x-2)^2}{9} + \frac{(x-1)^2}{4} = 1$

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

```
# EIGENMATH (solution by G. Weigt, 31.3.21)
a=3
b=2
xo=2
yo=1

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

```
u = (a cos(t),b sin(t))
u = (xo,yo) + dot(R,u)
u

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

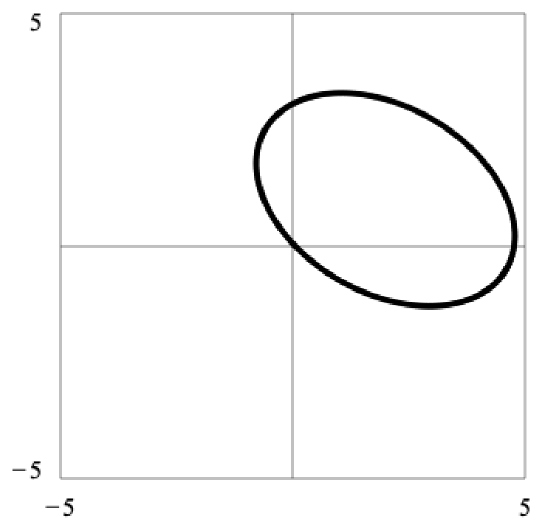

Figure 3: Blue: the ellipse $\frac{(x-2)^2}{9} + \frac{(x-1)^2}{4} = 1$.
Red: the rotated ellipse with eqaution ..

3.1 Example

References

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

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

[3] SEEBURGER, P. (2018): *CalcPlot3D.*
url: `https://c3d.libretexts.org/CalcPlot3D/index.html`

[4] STEEB, W.-H., LEWIEN, D. (1992): *Algorithms and Computations with Reduce.* Mannheim: Bibliographisches Institut.

[5] WEIGT, G. (2021): EIGENMATH *Homepage.*
url: `https://georgeweigt.github.io`

[6] WEIGT, G. (2021): EIGENMATH *online.*
url: `https://georgeweigt.github.io/eigenmath-demo.html`

[7] WEIGT, G. (2021): EIGENMATH *QR demo.*
url: `https://georgeweigt.github.io/quaternions.html`

[8] WIKIPEDIA|Principal Axes Theorem.
url: `https://en.wikipedia.org/wiki/Principal_axis_theorem`

[9] WIKIPEDIA|HauptAchsenTransformation
url: `https://de.wikipedia.org/wiki/Hauptachsentransformation#Diagonalisierung_einer_symmetrischen_Matrix_(Hauptachsentheorem)`