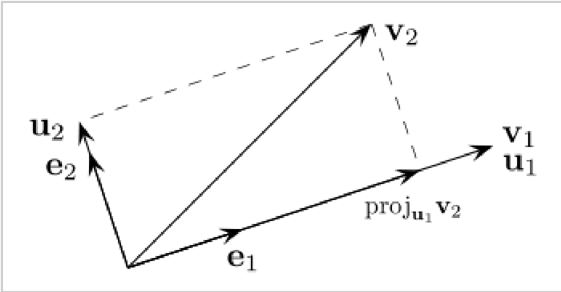
Gram–Schmidt process

From Wikipedia, the free encyclopedia



The first two steps of the Gram–Schmidt process

In mathematics, particularly linear algebra and numerical analysis, the **Gram–Schmidt process** is a method for orthonormalising a set of vectors in an inner product space, most commonly the Euclidean space $\mathbf{R}^n$. The Gram–Schmidt process takes a finite, linearly independent set $S = \{v_1, ..., v_k\}$ for $k \leq n$ and generates an orthogonal set $S' = \{u_1, ..., u_k\}$ that spans the same k-dimensional subspace of $\mathbf{R}^n$ as S.

The method is named after Jørgen Pedersen Gram and Erhard Schmidt but it appeared earlier in the work of Laplace and Cauchy. In the theory of Lie group decompositions it is generalized by the Iwasawa decomposition.[1]

The application of the Gram–Schmidt process to the column vectors of a full column rank matrix yields the QR decomposition (it is decomposed into an orthogonal and a triangular matrix).

Contents [hide]

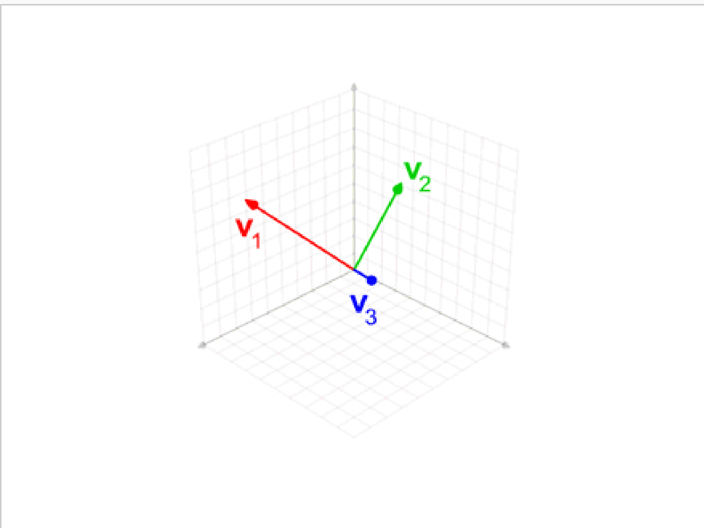
The Gram–Schmidt process [edit]

We define the projection operator by

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

where $\langle \mathbf{v}, \mathbf{u} \rangle$ denotes the inner product of the vectors **v** and **u**. This operator projects the vector **v** orthogonally onto the line spanned by vector **u**. If **u**=0, we define $\mathrm{proj}_0(\mathbf{v}) := 0$. i.e., the projection map proj_0 is the zero map, sending every vector to the zero vector.

The Gram–Schmidt process then works as follows:



The Gram-Schmidt process being executed on three linearly independent, non-orthogonal vectors of a basis for $\mathbf{R}^3$. Click on image for details.

$$\begin{aligned}
\mathbf{u}_1 &= \mathbf{v}_1, & \mathbf{e}_1 &= \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|} \\
\mathbf{u}_2 &= \mathbf{v}_2 - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_2), & \mathbf{e}_2 &= \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|} \\
\mathbf{u}_3 &= \mathbf{v}_3 - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_3) - \mathrm{proj}_{\mathbf{u}_2}(\mathbf{v}_3), & \mathbf{e}_3 &= \frac{\mathbf{u}_3}{\|\mathbf{u}_3\|} \\
\mathbf{u}_4 &= \mathbf{v}_4 - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_4) - \mathrm{proj}_{\mathbf{u}_2}(\mathbf{v}_4) - \mathrm{proj}_{\mathbf{u}_3}(\mathbf{v}_4), & \mathbf{e}_4 &= \frac{\mathbf{u}_4}{\|\mathbf{u}_4\|} \\
&\;\;\vdots & &\;\;\vdots \\
\mathbf{u}_k &= \mathbf{v}_k - \sum_{j=1}^{k-1} \mathrm{proj}_{\mathbf{u}_j}(\mathbf{v}_k), & \mathbf{e}_k &= \frac{\mathbf{u}_k}{\|\mathbf{u}_k\|}.
\end{aligned}$$

The sequence $\mathbf{u}_1, \ldots, \mathbf{u}_k$ is the required system of orthogonal vectors, and the normalized vectors $\mathbf{e}_1, \ldots, \mathbf{e}_k$ form an ortho*normal* set. The calculation of the sequence $\mathbf{u}_1, \ldots, \mathbf{u}_k$ is known as *Gram–Schmidt orthogonalization*, while the calculation of the sequence $\mathbf{e}_1, \ldots, \mathbf{e}_k$ is known as *Gram–Schmidt orthonormalization* as the vectors are normalized.

To check that these formulas yield an orthogonal sequence, first compute ‹ $\mathbf{u}_1,\mathbf{u}_2$ › by substituting the above formula for $\mathbf{u}_2$: we get zero. Then use this to compute ‹ $\mathbf{u}_1,\mathbf{u}_3$ › again by substituting the formula for $\mathbf{u}_3$: we get zero. The general proof proceeds by mathematical induction.

Geometrically, this method proceeds as follows: to compute $\mathbf{u}_i$, it projects $\mathbf{v}_i$ orthogonally onto the subspace U generated by $\mathbf{u}_1, \ldots, \mathbf{u}_{i-1}$, which is the same as the subspace generated by $\mathbf{v}_1, \ldots, \mathbf{v}_{i-1}$. The vector $\mathbf{u}_i$ is then defined to be the difference between $\mathbf{v}_i$ and this projection, guaranteed to be orthogonal to all of the vectors in the subspace U.

The Gram–Schmidt process also applies to a linearly independent countably infinite sequence $\{\mathbf{v}_i\}_i$. The result is an orthogonal (or orthonormal) sequence $\{\mathbf{u}_i\}_i$ such that for natural number n: the algebraic span of $\mathbf{v}_1, \ldots, \mathbf{v}_n$ is the same as that of $\mathbf{u}_1, \ldots, \mathbf{u}_n$.

If the Gram–Schmidt process is applied to a linearly dependent sequence, it outputs the **0** vector on the ith step, assuming that $\mathbf{v}_i$ is a linear combination of $\mathbf{v}_1, \ldots, \mathbf{v}_{i-1}$. If an orthonormal basis is to be produced, then the algorithm should test for zero vectors in the output and discard them because no multiple of a zero vector can have a length of 1. The number of vectors output by the algorithm will then be the dimension of the space spanned by the original inputs.

A variant of the Gram–Schmidt process using transfinite recursion applied to a (possibly uncountably) infinite sequence of vectors $(v_\alpha)_{\alpha<\lambda}$ yields a set of orthonormal vectors $(u_\alpha)_{\alpha<\kappa}$ with $\kappa \leq \lambda$ such that for any $\alpha \leq \lambda$, the completion of the span of $\{u_\beta : \beta < \min(\alpha, \kappa)\}$ is the same as that of $\{v_\beta : \beta < \alpha\}$. In particular, when applied to a (algebraic) basis of a Hilbert space (or, more generally, a basis of any dense subspace), it yields a (functional-analytic) orthonormal basis. Note that in the general case often the strict inequality $\kappa < \lambda$ holds, even if the starting set was linearly independent, and the span of $(u_\alpha)_{\alpha<\kappa}$ need not be a subspace of the span of $(v_\alpha)_{\alpha<\lambda}$ (rather, it's a subspace of its completion).

Example [edit]

Consider the following set of vectors in $\mathbf{R}^2$ (with the conventional inner product)

$$S = \left\{ \mathbf{v}_1 = \begin{pmatrix} 3 \\ 1 \end{pmatrix}, \mathbf{v}_2 = \begin{pmatrix} 2 \\ 2 \end{pmatrix} \right\}.$$

Now, perform Gram–Schmidt, to obtain an orthogonal set of vectors:

$$\mathbf{u}_1 = \mathbf{v}_1 = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$$

$$\mathbf{u}_2 = \mathbf{v}_2 - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_2) = \begin{pmatrix} 2 \\ 2 \end{pmatrix} - \mathrm{proj}_{\left(\begin{smallmatrix} 3 \\ 1 \end{smallmatrix}\right)}\left(\begin{pmatrix} 2 \\ 2 \end{pmatrix}\right) = \begin{pmatrix} 2 \\ 2 \end{pmatrix} - (4/5)\begin{pmatrix} 3 \\ 1 \end{pmatrix} = \begin{pmatrix} -2/5 \\ 6/5 \end{pmatrix}.$$

We check that the vectors $\mathbf{u}_1$ and $\mathbf{u}_2$ are indeed orthogonal:

$$\langle \mathbf{u}_1, \mathbf{u}_2 \rangle = \left\langle \begin{pmatrix} 3 \\ 1 \end{pmatrix}, \begin{pmatrix} -2/5 \\ 6/5 \end{pmatrix} \right\rangle = -\frac{6}{5} + \frac{6}{5} = 0,$$

noting that if the dot product of two vectors is *0* then they are orthogonal.

For non-zero vectors, we can then normalize the vectors by dividing out their sizes as shown above:

$$\mathbf{e}_1 = \frac{1}{\sqrt{10}}\begin{pmatrix}3\\1\end{pmatrix}$$
$$\mathbf{e}_2 = \frac{1}{\sqrt{\frac{40}{25}}}\begin{pmatrix}-2/5\\6/5\end{pmatrix} = \frac{1}{\sqrt{10}}\begin{pmatrix}-1\\3\end{pmatrix}.$$

Numerical stability [edit]

When this process is implemented on a computer, the vectors $\mathbf{u}_k$ are often not quite orthogonal, due to rounding errors. For the Gram–Schmidt process as described above (sometimes referred to as "classical Gram–Schmidt") this loss of orthogonality is particularly bad; therefore, it is said that the (classical) Gram–Schmidt process is numerically unstable.

The Gram–Schmidt process can be stabilized by a small modification; this version is sometimes referred to as **modified Gram-Schmidt** or MGS. This approach gives the same result as the original formula in exact arithmetic and introduces smaller errors in finite-precision arithmetic. Instead of computing the vector $\mathbf{u}_k$ as

$$\mathbf{u}_k = \mathbf{v}_k - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_k) - \mathrm{proj}_{\mathbf{u}_2}(\mathbf{v}_k) - \cdots - \mathrm{proj}_{\mathbf{u}_{k-1}}(\mathbf{v}_k),$$

it is computed as

$$\begin{aligned}
\mathbf{u}_k^{(1)} &= \mathbf{v}_k - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_k),\\
\mathbf{u}_k^{(2)} &= \mathbf{u}_k^{(1)} - \mathrm{proj}_{\mathbf{u}_2}(\mathbf{u}_k^{(1)}),\\
&\;\;\vdots\\
\mathbf{u}_k^{(k-2)} &= \mathbf{u}_k^{(k-3)} - \mathrm{proj}_{\mathbf{u}_{k-2}}(\mathbf{u}_k^{(k-3)}),\\
\mathbf{u}_k^{(k-1)} &= \mathbf{u}_k^{(k-2)} - \mathrm{proj}_{\mathbf{u}_{k-1}}(\mathbf{u}_k^{(k-2)}).
\end{aligned}$$

Each step finds a vector $\mathbf{u}_k^{(i)}$ orthogonal to $\mathbf{u}_k^{(i-1)}$. Thus $\mathbf{u}_k^{(i)}$ is also orthogonalized against any errors introduced in computation of $\mathbf{u}_k^{(i-1)}$.

This method is used in the previous animation, when the intermediate v'_3 vector is used when orthogonalizing the blue vector v_3.

Algorithm [edit]

The following algorithm implements the stabilized Gram–Schmidt orthonormalization. The vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ are replaced by orthonormal vectors which span the same subspace.

for *i* **from** 1 **to** *k* **do**

$\mathbf{v}_i \leftarrow \frac{\mathbf{v}_i}{\|\mathbf{v}_i\|}$ (*normalize*)

for *j* **from** i+1 **to** k **do**

$\mathbf{v}_j \leftarrow \mathbf{v}_j - \mathrm{proj}_{\mathbf{v}_i}(\mathbf{v}_j)$ (*remove component in direction* $\mathbf{v}_i$)

next j

next i

The cost of this algorithm is asymptotically $2nk^2$ floating point operations, where n is the dimensionality of the vectors (Golub & Van Loan 1996, §5.2.8).

Determinant formula [edit]

The result of the Gram–Schmidt process may be expressed in a non-recursive formula using determinants.

$$\mathbf{e}_j = \frac{1}{\sqrt{D_{j-1}D_j}}\begin{vmatrix}
\langle\mathbf{v}_1,\mathbf{v}_1\rangle & \langle\mathbf{v}_2,\mathbf{v}_1\rangle & \dots & \langle\mathbf{v}_j,\mathbf{v}_1\rangle\\
\langle\mathbf{v}_1,\mathbf{v}_2\rangle & \langle\mathbf{v}_2,\mathbf{v}_2\rangle & \dots & \langle\mathbf{v}_j,\mathbf{v}_2\rangle\\
\vdots & \vdots & \ddots & \vdots\\
\langle\mathbf{v}_1,\mathbf{v}_{j-1}\rangle & \langle\mathbf{v}_2,\mathbf{v}_{j-1}\rangle & \dots & \langle\mathbf{v}_j,\mathbf{v}_{j-1}\rangle\\
\mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_j
\end{vmatrix}$$

$$\mathbf{u}_j = \frac{1}{D_{j-1}} \begin{vmatrix} \langle \mathbf{v}_1, \mathbf{v}_1 \rangle & \langle \mathbf{v}_2, \mathbf{v}_1 \rangle & \dots & \langle \mathbf{v}_j, \mathbf{v}_1 \rangle \\ \langle \mathbf{v}_1, \mathbf{v}_2 \rangle & \langle \mathbf{v}_2, \mathbf{v}_2 \rangle & \dots & \langle \mathbf{v}_j, \mathbf{v}_2 \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \mathbf{v}_1, \mathbf{v}_{j-1} \rangle & \langle \mathbf{v}_2, \mathbf{v}_{j-1} \rangle & \dots & \langle \mathbf{v}_j, \mathbf{v}_{j-1} \rangle \\ \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_j \end{vmatrix}$$

where D_0=1 and, for $j \geq 1$, D_j is the Gram determinant

$$D_j = \begin{vmatrix} \langle \mathbf{v}_1, \mathbf{v}_1 \rangle & \langle \mathbf{v}_2, \mathbf{v}_1 \rangle & \dots & \langle \mathbf{v}_j, \mathbf{v}_1 \rangle \\ \langle \mathbf{v}_1, \mathbf{v}_2 \rangle & \langle \mathbf{v}_2, \mathbf{v}_2 \rangle & \dots & \langle \mathbf{v}_j, \mathbf{v}_2 \rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \mathbf{v}_1, \mathbf{v}_j \rangle & \langle \mathbf{v}_2, \mathbf{v}_j \rangle & \dots & \langle \mathbf{v}_j, \mathbf{v}_j \rangle \end{vmatrix}.$$

Note that the expression for $\mathbf{u}_k$ is a "formal" determinant, i.e. the matrix contains both scalars and vectors; the meaning of this expression is defined to be the result of a cofactor expansion along the row of vectors.

The determinant formula for the Gram-Schmidt is computationally slower (exponentially slower) than the recursive algorithms described above; it is mainly of theoretical interest.

Alternatives [edit]

Other orthogonalization algorithms use Householder transformations or Givens rotations. The algorithms using Householder transformations are more stable than the stabilized Gram–Schmidt process. On the other hand, the Gram–Schmidt process produces the jth orthogonalized vector after the jth iteration, while orthogonalization using Householder reflections produces all the vectors only at the end. This makes only the Gram–Schmidt process applicable for iterative methods like the Arnoldi iteration.

Yet another alternative is motivated by the use of Cholesky decomposition for inverting the matrix of the normal equations in linear least squares. Let $\mathbf{V}$ be a full column rank matrix, which columns need to be orthogonalized. The matrix $\mathbf{V}^*\mathbf{V}$ is Hermitian and positive definite, so it can be written as $\mathbf{V}^*\mathbf{V} = \mathbf{L}\mathbf{L}^*,$ using the Cholesky decomposition. The lower triangular matrix $\mathbf{L}$ with strictly positive diagonal entries is invertible. Then columns of the matrix $\mathbf{U} = \mathbf{V}(\mathbf{L}^{-1})^*$ are orthonormal and span the same subspace as the columns of the original matrix $\mathbf{V}$. The explicit use of the product $\mathbf{V}^*\mathbf{V}$ makes the algorithm unstable, especially if the product's condition number is large. Nevertheless, this algorithm is used in practice and implemented in some software packages because of its high efficiency and simplicity.

In quantum mechanics there are several orthogonalization schemes with characteristics better suited for applications than the Gram–Schmidt one. The most important among them are the symmetric and the canonical orthonormalization (see Solivérez & Gagliano).[*clarification needed*]

References [edit]

1. ^ Cheney, Ward; Kincaid, David (2009). *Linear Algebra: Theory and Applications*. Sudbury, Ma: Jones and Bartlett. pp. 544, 558. ISBN 978-0-7637-5020-6.

- Bau III, David; Trefethen, Lloyd N. (1997), *Numerical linear algebra*, Philadelphia: Society for Industrial and Applied Mathematics, ISBN 978-0-89871-361-9.
- Golub, Gene H.; Van Loan, Charles F. (1996), *Matrix Computations* (3rd ed.), Johns Hopkins, ISBN 978-0-8018-5414-9.
- Greub, Werner (1975), *Linear Algebra* (4th ed.), Springer.
- Soliverez, C. E.; Gagliano, E. (1985), *Orthonormalization on the plane: a geometric approach*, Mex. J. Phys. **31** (Nº 4), pp. 743l758.

External links [edit]

Mathematics portal

- Hazewinkel, Michiel, ed. (2001), "Orthogonalization", *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Harvey Mudd College Math Tutorial on the Gram-Schmidt algorithm
- Earliest known uses of some of the words of mathematics: G The entry "Gram-Schmidt orthogonalization" has some information and references on the origins of the method.
- Demos: Gram Schmidt process in plane and Gram Schmidt process in space
- Gram-Schmidt orthogonalization applet
- NAG Gram–Schmidt orthogonalization of n vectors of order m routine

- Proof: Raymond Puzio, Keenan Kidwell. "proof of Gram-Schmidt orthogonalization algorithm" (version 8). PlanetMath.org.

v · t · e	Linear algebra [hide]
Basic concepts	Scalar · Vector · Vector space · Vector projection · Linear span · Linear map · Linear projection · Linear independence · Linear combination · Basis · Column space · Row space · Dual space · Orthogonality · Kernel · Eigenvalues and eigenvectors · Least squares regressions · Outer product · Inner product space · Dot product · Transpose · **Gram–Schmidt process** · Linear equations
Matrices	Block · Decomposition · Invertible · Minor · Multiplication · Rank · Transformation · Cramer's rule · Gaussian elimination
Numerical	Floating point · Matrix Laboratory · Numerical stability · Basic Linear Algebra Subprograms (BLAS) · Sparse matrix · Comparison of linear algebra libraries · Comparison of numerical analysis software

Categories: Linear algebra | Functional analysis