Towards Verified Linear Algebra Programs Through Equivalence

POPL 2025

CoqPL 2025

Yihan (Yuki) Yang, Mohit Tekriwal, John Sarracino, Matthew Sottile, Ignacio Laguna

Jan 25, 2025

LLNL-PRES-872110

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

SIGPLAN

HARVEY MUDD COLLEGE

Lawrence Livermore National Laboratory

Motivation

- Scientific computing is rapidly evolving. Some evolving trends
  - Data compression FP formats to overcome cost of data movement in FP arithmetic
  - Reduced & mixed precision or other exotic precision formats in next generation hardware & accelerators
  - Parallelization on GPUs, FPGAs and shift towards heterogeneous architectures

- Need for next generation of numerical algorithms that takes into account all these aspects in order to meet the demands of applications on the evolving hardware

- For the case when these next gen algorithms are variants of existing algorithms
  - How do we verify the correctness of their implementation for safe deployment?

Program equivalence

- A solution: Prove the equivalence between the new variant and its classical counterpart
- Classical counterpart extensively studied in numerical analysis literature and serve as a ground truth for correctness of new variants
- Recent work: Equivalence between dense matrix-vector product & sparse matrix-vector product with varying sparse matrix representations (**LGTM framework @PLDI’24 : Gladshtein et al.**)
- In this work, we prove equivalence between variants of a fundamental orthogonalization algorithm, Gram-Schmidt: **Classical Gram-Schmidt** and **Modified Gram-Schmidt (more stable numerically)**

Gram-Schmidt (GS) algorithm

- Gram-Schmidt is an **orthogonalization** algorithm: computes an orthogonal set of vectors

- Applications of orthogonalization algorithm
  - Machine learning
    - feature engineering through techniques like Principal component analysis
    - Optimization and regularization
    - Data compression, image processing and collaborative filtering through techniques like Singular Value Decomposition
  - Cryptography: Lattice reduction techniques to optimize the basis of a lattice for better efficiency and security
  - Signal processing: Noise reduction and multi-channel communication
  - …

Research contribution

We extended the Mathcomp library in Coq to include basic linear algebra theorems

- Basis Theorem, dot product properties, definition of projections...

The linear algebra library is used to prove the properties of GS

- Finished the proof for CGS using this library

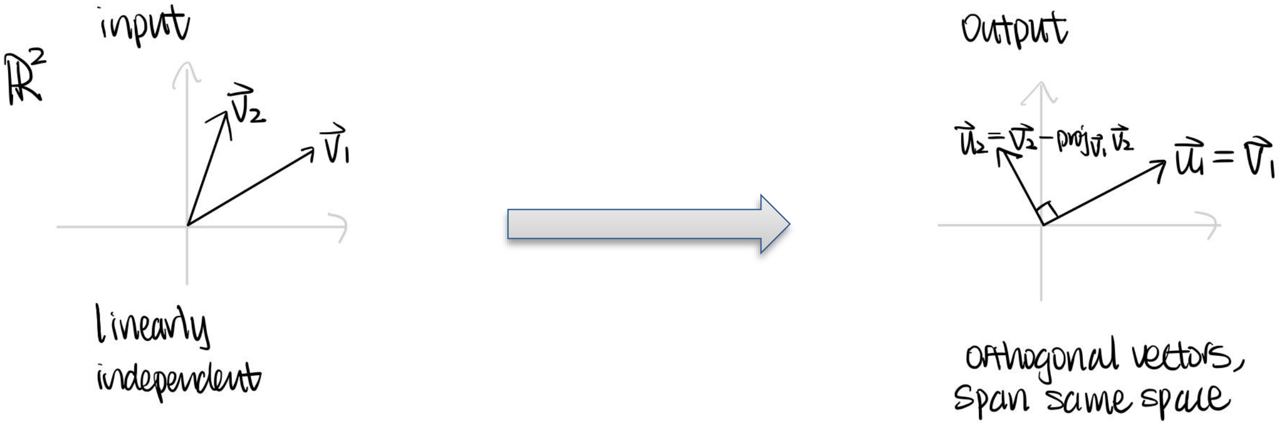
The library can be extended to floating point reasoning and error analysis.

Mathcomp: Assia Mahboubi, & Enrico Tassi. (2022). Mathematical Components (1.0.2) [Computer software]. Zenodo. https://doi.org/10.5281/zenodo.7118596

What is Gram-Schmidt Algorithm?

Input: a finite, linearly independent set of vectors $S = \{v_1, \ldots, v_k\}$ in $\mathbb{R}^n$ for $k \leq n$

Output: an orthogonal set $S' = \{u_1, \ldots, u_k\}$ that spans the same k-dimensional subspace of $\mathbb{R}^n$ as S.

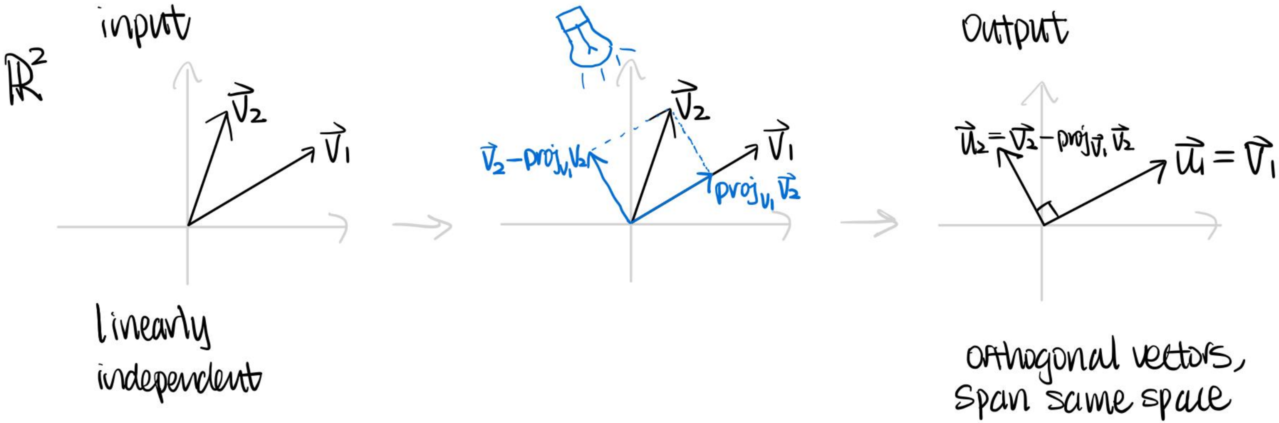

input
$\mathbb{R}^2$
$\vec{v}_2$
$\vec{v}_1$
linearly independent
Output
$\vec{u}_2 = \vec{v}_2 - \mathrm{proj}_{\vec{v}_1}\vec{v}_2$
$\vec{u}_1 = \vec{v}_1$
orthogonal vectors, span same space

What is Gram-Schmidt Algorithm?

Input: a finite, linearly independent set of vectors $S = \{v_1, \dots, v_k\}$ in $\mathbb{R}^n$ for $k \leq n$

Output: an orthogonal set $S' = \{u_1, \dots, u_k\}$ that spans the same k-dimensional subspace of $\mathbb{R}^n$ as S.

$$\mathrm{proj}_{\mathbf{b}}\, \mathbf{a} = \frac{\mathbf{a} \cdot \mathbf{b}}{\mathbf{b} \cdot \mathbf{b}} \mathbf{b}\ .$$


input
$\mathbb{R}^2$
$\vec{v}_2$
$\vec{v}_1$
linearly independent
$\vec{v}_2 - \mathrm{proj}_{v_1} \vec{v}_2$
$\vec{v}_2$
$\vec{v}_1$
$\mathrm{proj}_{v_1} \vec{v}_2$
Output
$\vec{u}_2 = \vec{v}_2 - \mathrm{proj}_{\vec{v}_1} \vec{v}_2$
$\vec{u}_1 = \vec{v}_1$
orthogonal vectors, span same space

GS has Different Versions, CGS and MGS

CGS

```
for $i = 1 : n$ do
    $u_i = v_i$
    for $k = 1 : i - 1$ do
        $u_i = u_i - \left( \frac{u_k^T v_i}{u_k^T u_k} \right) u_k$
    end for
end for
```

- output vectors not perfectly orthogonal in floating points
- large rounding error
- Numerically **unstable**

GS has Different Versions, CGS and MGS

CGS

```
for $i = 1 : n$ do
    $u_i = v_i$
    for $k = 1 : i - 1$ do
        $u_i = u_i - \left(\frac{u_k^T v_i}{u_k^T u_k}\right) u_k$
    end for
end for
```

- output vectors not perfectly orthogonal in floating points
- large rounding error
- Numerically **unstable**

MGS

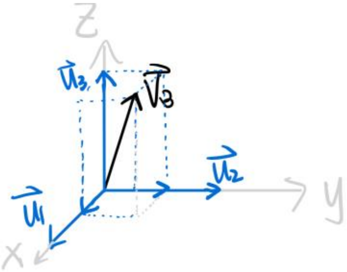
```
for $i = 1 : n$ do
    $u_i = v_i$
end for
for $i = 1 : n$ do
    for $k = i + 1 : n$ do
        $u_k = u_k - \left(\frac{u_i^T u_k}{u_i^T u_i}\right) u_i$
    end for
end for
```

- **exact equivalence** to CGS in real numbers
- small rounding error
- Numerically **stable**

GS has Different Versions, CGS and MGS

CGS

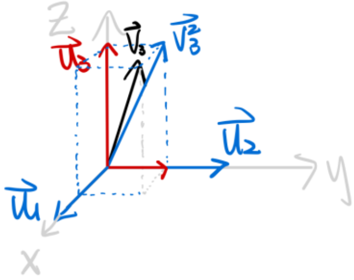
```
for $i = 1 : n$ do
    $u_i = v_i$
    for $k = 1 : i - 1$ do
        $u_i = u_i - \left(\frac{u_k^T v_i}{u_k^T u_k}\right) u_k$
    end for
end for
```



$$\vec{U_3} = \vec{V_3} - \mathrm{proj}_{\vec{U_1}} \vec{V_3} - \mathrm{proj}_{\vec{U_2}} \vec{V_3}$$

MGS

```
for $i = 1 : n$ do
    $u_i = v_i$
end for
for $i = 1 : n$ do
    for $k = i + 1 : n$ do
        $u_k = u_k - \left(\frac{u_i^T u_k}{u_i^T u_i}\right) u_i$
    end for
end for
```



$$\vec{V_3^2} = \vec{V_3} - \mathrm{proj}_{\vec{U_1}} \vec{V_3}$$

$$\vec{U_3} = \vec{V_3^2} - \mathrm{proj}_{\vec{U_2}} \vec{V_3^2}$$

Loss of Orthogonality in CGS

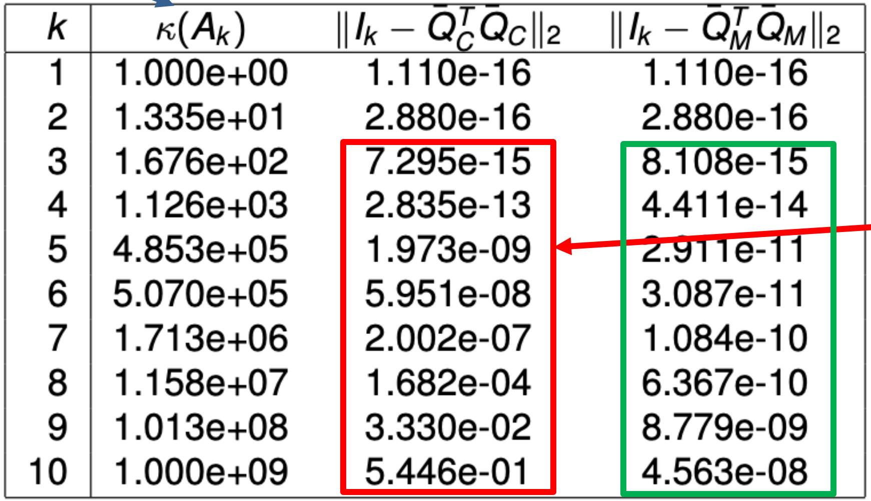
- CGS loses orthogonality after a couple iteration (in floating points)

κ: condition number: sensitivity of the output relative to errors in the input

Q_C: orthogonal basis using CGS

Q_M: orthogonal basis using MGS

k: dimension



k	$\kappa(A_k)$	$\|I_k - \bar{Q}_C^T\bar{Q}_C\|_2$	$\|I_k - \bar{Q}_M^T\bar{Q}_M\|_2$
1	1.000e+00	1.110e-16	1.110e-16
2	1.335e+01	2.880e-16	2.880e-16
3	1.676e+02	7.295e-15	8.108e-15
4	1.126e+03	2.835e-13	4.411e-14
5	4.853e+05	1.973e-09	2.911e-11
6	5.070e+05	5.951e-08	3.087e-11
7	1.713e+06	2.002e-07	1.084e-10
8	1.158e+07	1.682e-04	6.367e-10
9	1.013e+08	3.330e-02	8.779e-09
10	1.000e+09	5.446e-01	4.563e-08

The error of CGS quickly diverges away from 0

https://www.cis.upenn.edu/~cis6100/Gram-Schmidt-Bjorck.pdf

Goal

- Prove CGS and MGS with the goal of proving stability and convergence
- Proving equivalence (both in reals and in floats) between different variants of linear algebra algorithms.
- Prove actual programs → need computable definition

However...

- Isabelle/HOL
  - Computable definition
  - Target only CGS, did not include numerical stability
  - Only proved high level mathematical properties
- LEAN
  - Use a non-computable definition of CGS
  - No MGS, no numerical stability

Our work provides formalization keeping in mind the numerical properties such as stability, numerical convergence, and floats in the long run.

GS Specification

Input: a finite, linearly independent set of vectors $S = \{v_1, \dots, v_k\}$ in $\mathbb{R}^n$ for $k \leq n$

Output: an orthogonal set $S' = \{u_1, \dots, u_k\}$ that spans the same k-dimensional subspace of $\mathbb{R}^n$ as S.

GS Specification

Input: a finite, linearly independent set of vectors $S = \{v_1, \ldots, v_k\}$ in $\mathbb{R}^n$ for $k \leq n$

Output: an orthogonal set $S' = \{u_1, \ldots, u_k\}$ that spans the same k-dimensional subspace of $\mathbb{R}^n$ as S.

projections, dot product..

What is Mathcomp Missing?

**Vector* and *Matrix* are libraries in Mathcomp

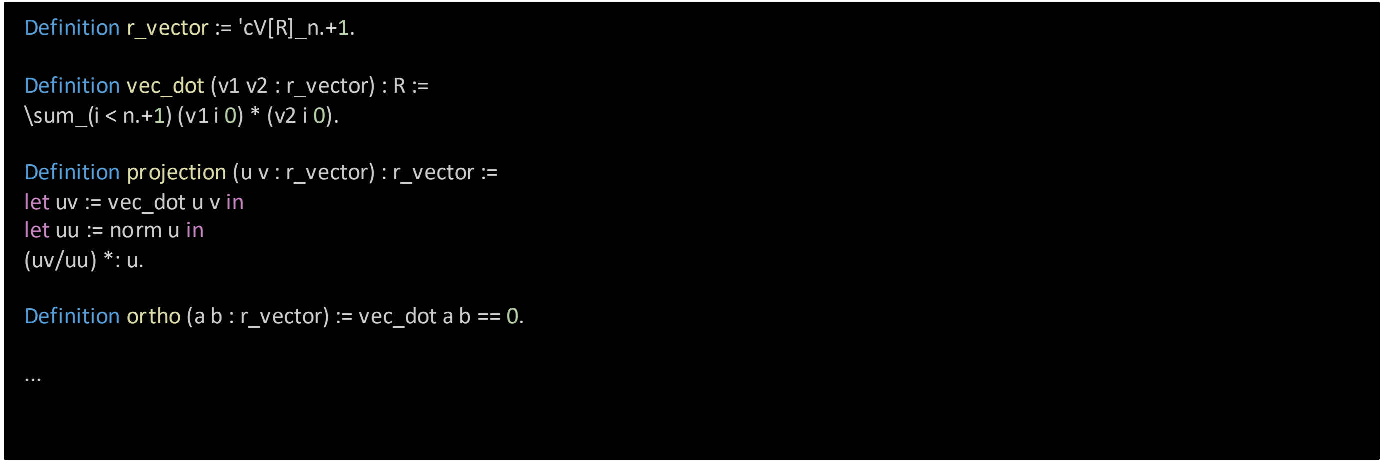
Mathcomp have a lot of low-level definitions

Definitions	Location in Mathcomp
vector	Defined in *Vector*, implicit in *Matrix*
linear independence	Defined in *Vector* as "free"
vector space/span	Defined in *Vector*
projection	Defined in *Vector*, but very limited and lack algebraic properties

Missing high level linear algebra theorems

Definitions	Location in Mathcomp
dot product	no explicit definition or properties
orthogonality	Not defined
Orthogonal Decomposition Theorem	Not defined
Basis Theorem	Not defined

We used Mathcomp to created a library that can help us prove the GS theorem.

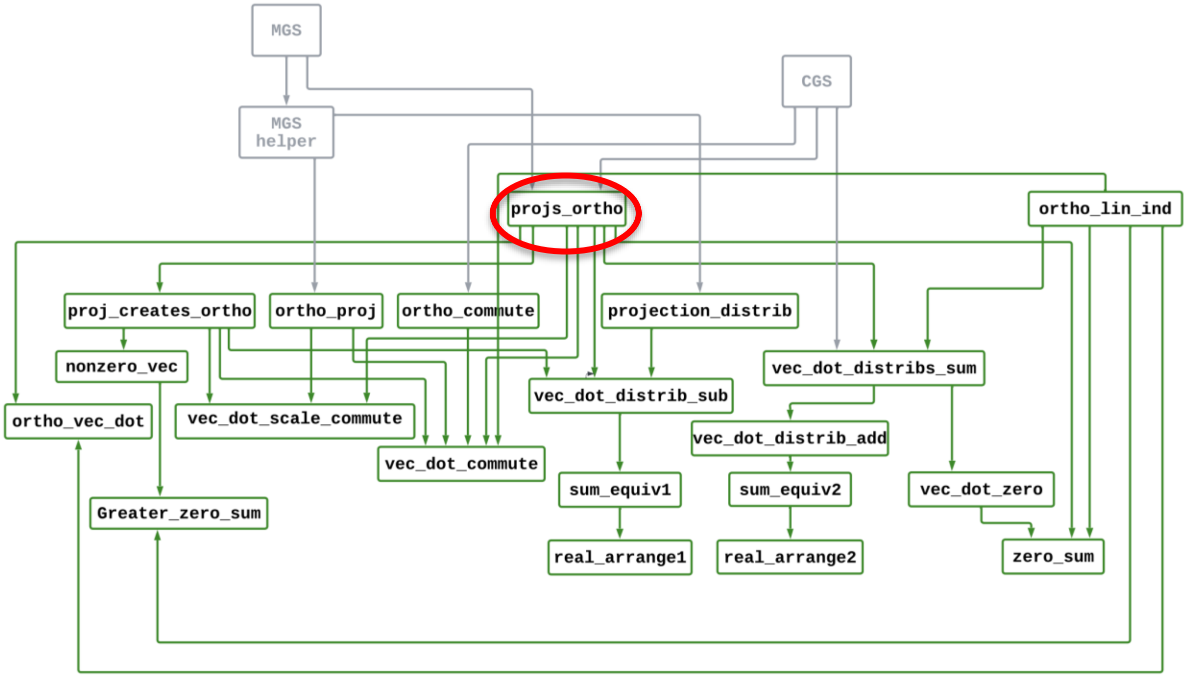


```
Definition r_vector := 'cV[R]_n.+1.

Definition vec_dot (v1 v2 : r_vector) : R :=
\sum_(i < n.+1) (v1 i 0) * (v2 i 0).

Definition projection (u v : r_vector) : r_vector :=
let uv := vec_dot u v in
let uu := norm u in
(uv/uu) *: u.

Definition ortho (a b : r_vector) := vec_dot a b == 0.

...
```

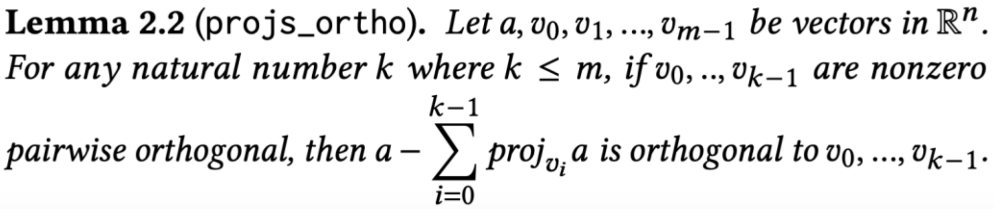
Key Takeaway:
- Add definitions and properties towards building a comprehensive linear algebra library

Dependency Graph 1

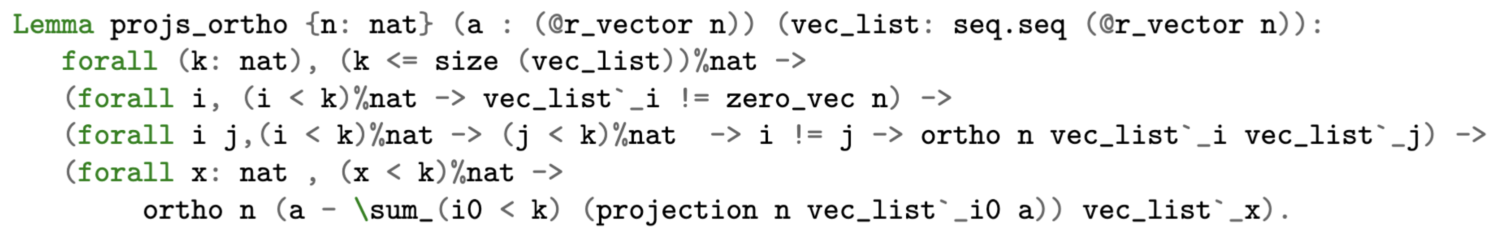
We mechanized fundamental linear algebra theorems, including the Basis Theorem, the Orthogonal Decomposition Theorem (projs_ortho), and basic properties of vectors.



Key Results – Theorem 1



Lemma 2.2 (projs_ortho). *Let $a, v_0, v_1, ..., v_{m-1}$ be vectors in $\mathbb{R}^n$. For any natural number k where $k \le m$, if $v_0, .., v_{k-1}$ are nonzero pairwise orthogonal, then $a - \sum_{i=0}^{k-1} proj_{v_i} a$ is orthogonal to $v_0, ..., v_{k-1}$.*



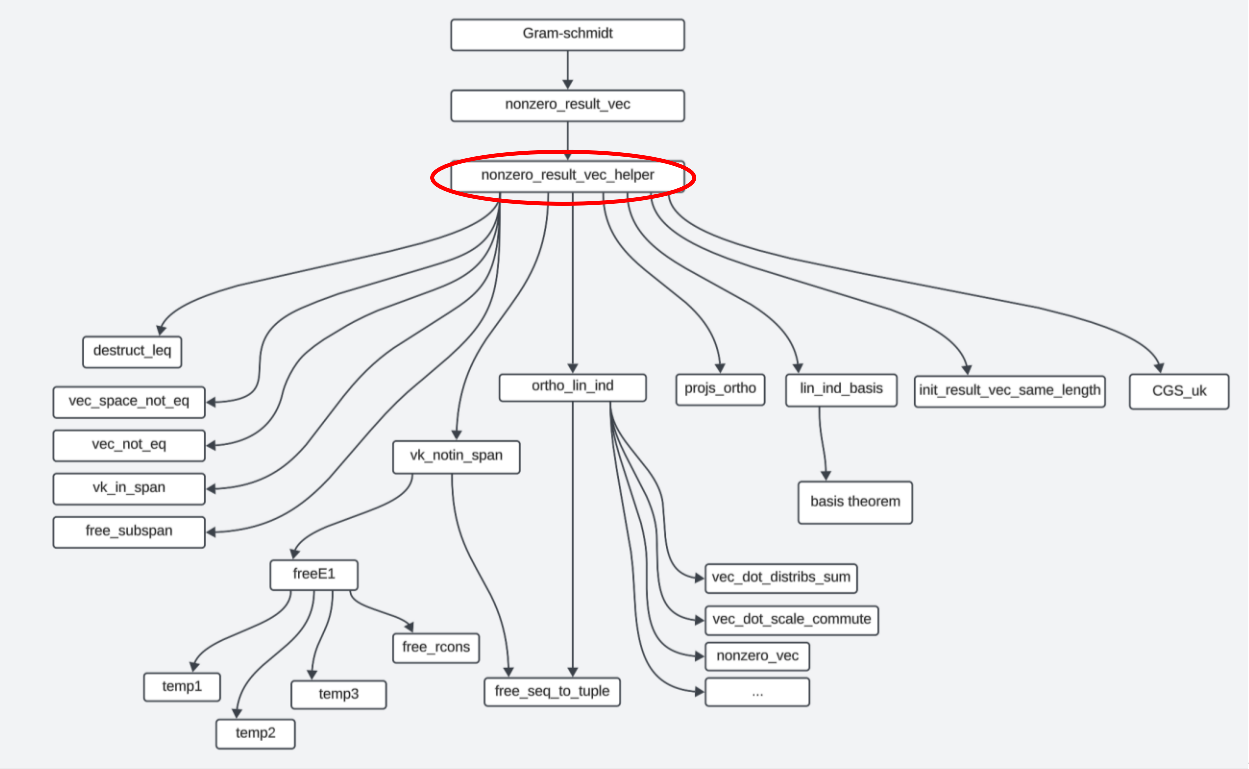
```
Lemma projs_ortho {n: nat} (a : (@r_vector n)) (vec_list: seq.seq (@r_vector n)):
   forall (k: nat), (k <= size (vec_list))%nat ->
   (forall i, (i < k)%nat -> vec_list`_i != zero_vec n) ->
   (forall i j,(i < k)%nat -> (j < k)%nat  -> i != j -> ortho n vec_list`_i vec_list`_j) ->
   (forall x: nat , (x < k)%nat ->
        ortho n (a - \sum_(i0 < k) (projection n vec_list`_i0 a)) vec_list`_x).
```

Key Takeaway:

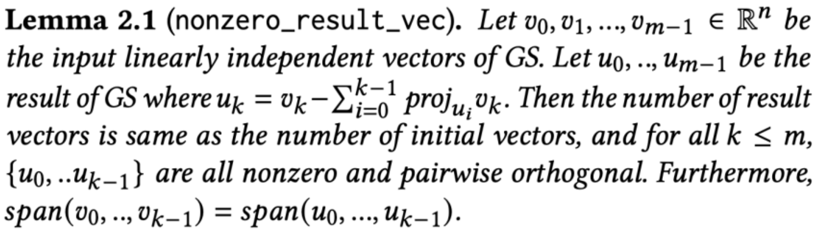
- Tweaked the specification a little to match its use case.
- When $k = m$, this is the Orthogonal Decomposition Theorem.
- The proof requires both Mathcomp and our library.

Dependency Graph 2

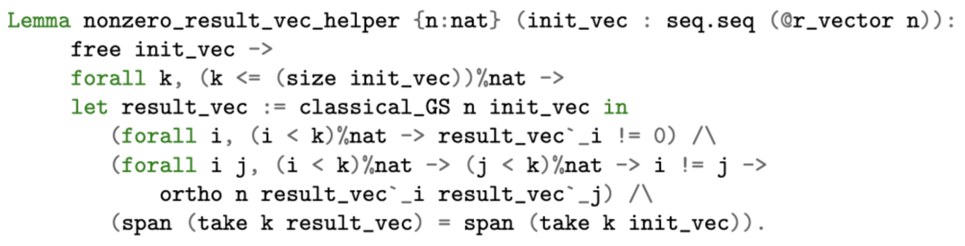
We used the linear algebra library to prove the correctness of CGS.



Key Results – Theorem 2



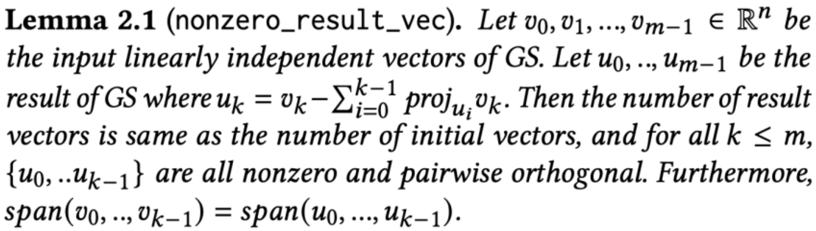
Lemma 2.1 (nonzero_result_vec). *Let $v_0, v_1, ..., v_{m-1} \in \mathbb{R}^n$ be the input linearly independent vectors of GS. Let $u_0, .., u_{m-1}$ be the result of GS where $u_k = v_k - \sum_{i=0}^{k-1} proj_{u_i} v_k$. Then the number of result vectors is same as the number of initial vectors, and for all $k \leq m$, $\{u_0, ..u_{k-1}\}$ are all nonzero and pairwise orthogonal. Furthermore, $span(v_0, .., v_{k-1}) = span(u_0, ..., u_{k-1})$.*



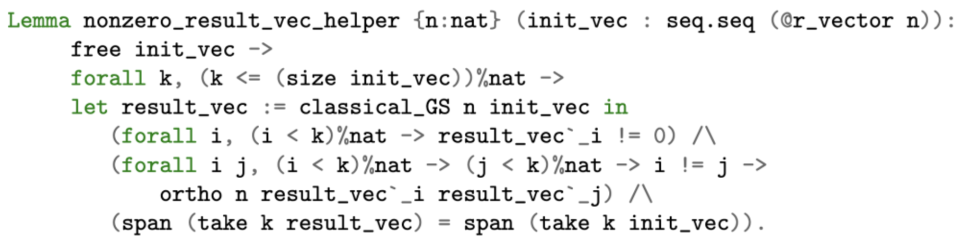
```
Lemma nonzero_result_vec_helper {n:nat} (init_vec : seq.seq (@r_vector n)):
      free init_vec ->
      forall k, (k <= (size init_vec))%nat ->
      let result_vec := classical_GS n init_vec in
         (forall i, (i < k)%nat -> result_vec`_i != 0) /\
         (forall i j, (i < k)%nat -> (j < k)%nat -> i != j ->
             ortho n result_vec`_i result_vec`_j) /\
         (span (take k result_vec) = span (take k init_vec)).
```

- Main helper lemma to prove CGS
- Independently devised the specification
- Stronger than what is in the textbook

Key Results – Theorem 2



Lemma 2.1 (nonzero_result_vec). *Let $v_0, v_1, ..., v_{m-1} \in \mathbb{R}^n$ be the input linearly independent vectors of GS. Let $u_0, .., u_{m-1}$ be the result of GS where $u_k = v_k - \sum_{i=0}^{k-1} proj_{u_i} v_k$. Then the number of result vectors is same as the number of initial vectors, and for all $k \leq m$, $\{u_0, ..u_{k-1}\}$ are all nonzero and pairwise orthogonal. Furthermore, $span(v_0, .., v_{k-1}) = span(u_0, ..., u_{k-1})$.*

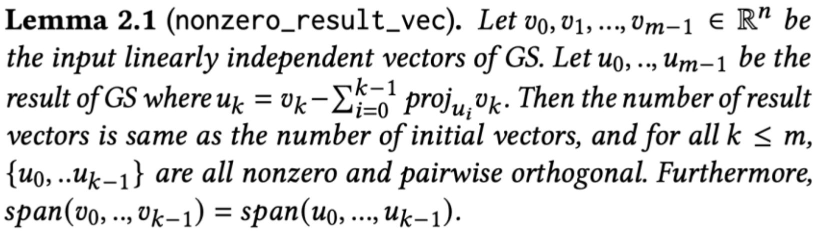


```
Lemma nonzero_result_vec_helper {n:nat} (init_vec : seq.seq (@r_vector n)):
     free init_vec ->
     forall k, (k <= (size init_vec))%nat ->
     let result_vec := classical_GS n init_vec in
        (forall i, (i < k)%nat -> result_vec`_i != 0) /\
        (forall i j, (i < k)%nat -> (j < k)%nat -> i != j ->
            ortho n result_vec`_i result_vec`_j) /\
        (span (take k result_vec) = span (take k init_vec)).
```

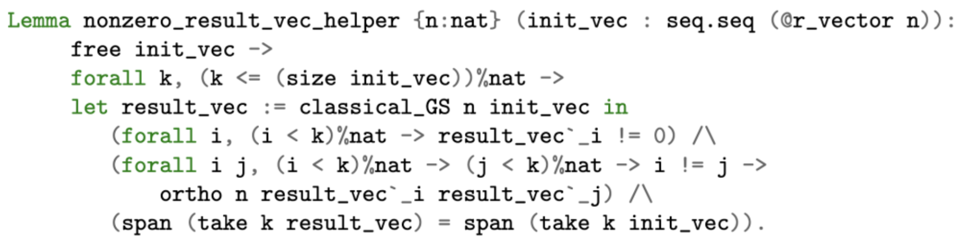
- Main helper lemma to prove CGS
- Independently devised the specification
- Stronger than what is in the textbook

IH: For a certain $k \leq m$, $\{u_0, \ldots, u_{k-1}\}$ are all nonzero and pairwise orthogonal, and that $span(v_0, .., v_{k-1}) = span\ (u_0, .., u_{k-1})$

Key Results – Theorem 2

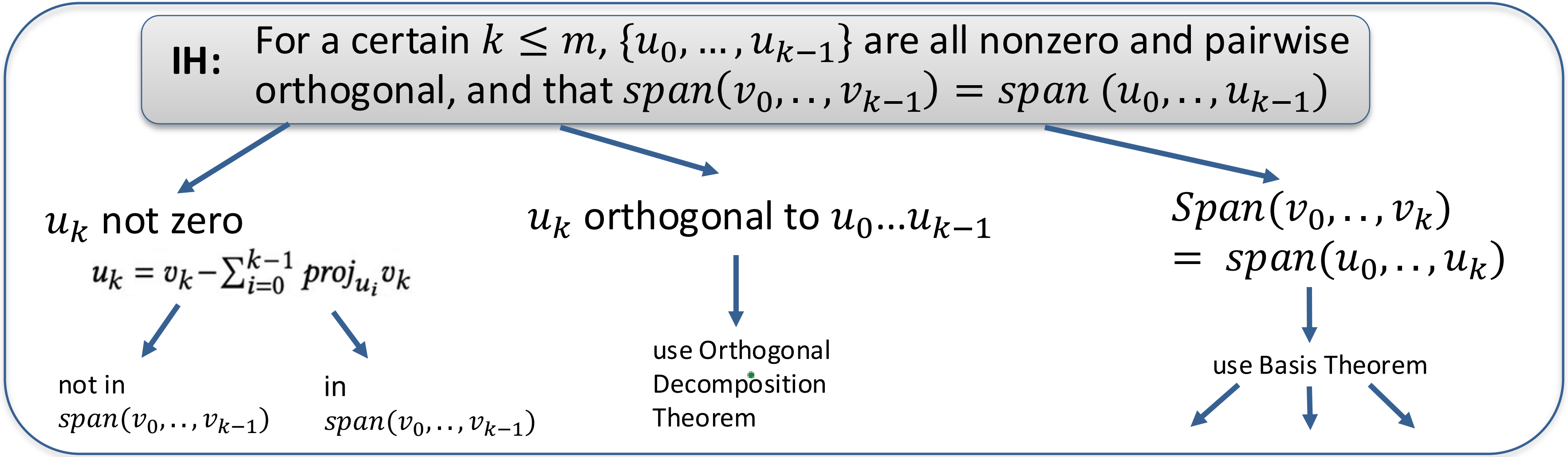


Lemma 2.1 (nonzero_result_vec). *Let $v_0, v_1, ..., v_{m-1} \in \mathbb{R}^n$ be the input linearly independent vectors of GS. Let $u_0, .., u_{m-1}$ be the result of GS where $u_k = v_k - \sum_{i=0}^{k-1} proj_{u_i} v_k$. Then the number of result vectors is same as the number of initial vectors, and for all $k \leq m$, $\{u_0, ..u_{k-1}\}$ are all nonzero and pairwise orthogonal. Furthermore, $span(v_0, .., v_{k-1}) = span(u_0, ..., u_{k-1})$.*



```
Lemma nonzero_result_vec_helper {n:nat} (init_vec : seq.seq (@r_vector n)):
      free init_vec ->
      forall k, (k <= (size init_vec))%nat ->
      let result_vec := classical_GS n init_vec in
         (forall i, (i < k)%nat -> result_vec`_i != 0) /\
         (forall i j, (i < k)%nat -> (j < k)%nat -> i != j ->
            ortho n result_vec`_i result_vec`_j) /\
         (span (take k result_vec) = span (take k init_vec)).
```

- Main helper lemma to prove CGS
- Independently devised the specification
- Stronger than what is in the textbook



IH: For a certain $k \leq m$, $\{u_0, \ldots, u_{k-1}\}$ are all nonzero and pairwise orthogonal, and that $span(v_0, .., v_{k-1}) = span\ (u_0, .., u_{k-1})$

u_k not zero

$u_k = v_k - \sum_{i=0}^{k-1} proj_{u_i} v_k$

not in $span(v_0, .., v_{k-1})$

in $span(v_0, .., v_{k-1})$

u_k orthogonal to $u_0 \ldots u_{k-1}$

use Orthogonal Decomposition Theorem

$Span(v_0, .., v_k) = span(u_0, .., u_k)$

use Basis Theorem

Equivalence Proof Sketch (what is next?)

Correctness of CGS

- Done

$$u_i = v_i - \sum_{k=1}^{i-1} proj_{u_k} v_i$$

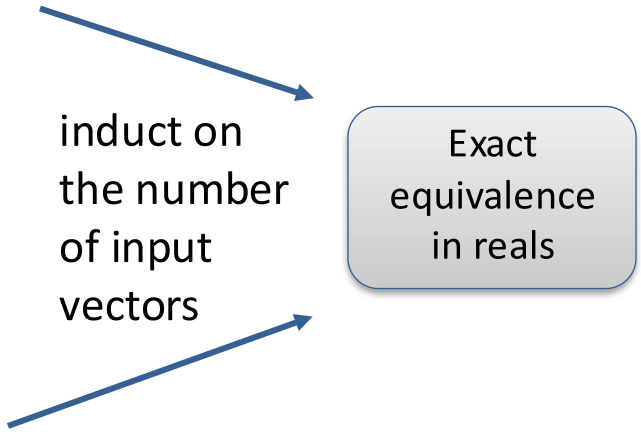
Correctness of MGS

$$v_i^i = v_i - \sum_{k=1}^{i-1} proj_{v_k^k} v_i$$

- Reuse CGS helper lemmas
- An Additional lemma proved on pen and paper:

For all $0 \leq k < m$, if $v_i^i \perp v_j^j$ for i, $j \ = \ 0, .., k-1$, then $v_k^h = v_k - \sum_{i=o}^{h-1} proj_{v_i^i} v_k$ for $0 \leq h \leq k$.

- Reuse Orthogonal Decomposition Theorem



induct on the number of input vectors

Exact equivalence in reals

Future Work (Mathcomp)

- Current work could be implemented into Mathcomp
- Suggestion: Could add our work to the Vector library as we have similar definition on vectors.
- Future applications and use case of our library:
  - Ordinary Differential Equation
    - People can use our library to mechanize ODE solving techniques that involves linear algebra
  - QR Decomposition
    - To solve linear least squares problem
  - …Theorems that involves linear algebra

Future Work

Reasoning about floating-point programs

- Adapts the equality constrains of our lemmas to symbolic bounds.

Reasoning about low-level HPC programs

- Extends a separation logic with support for floating-point.

Modular equivalence framework for floating-point programs

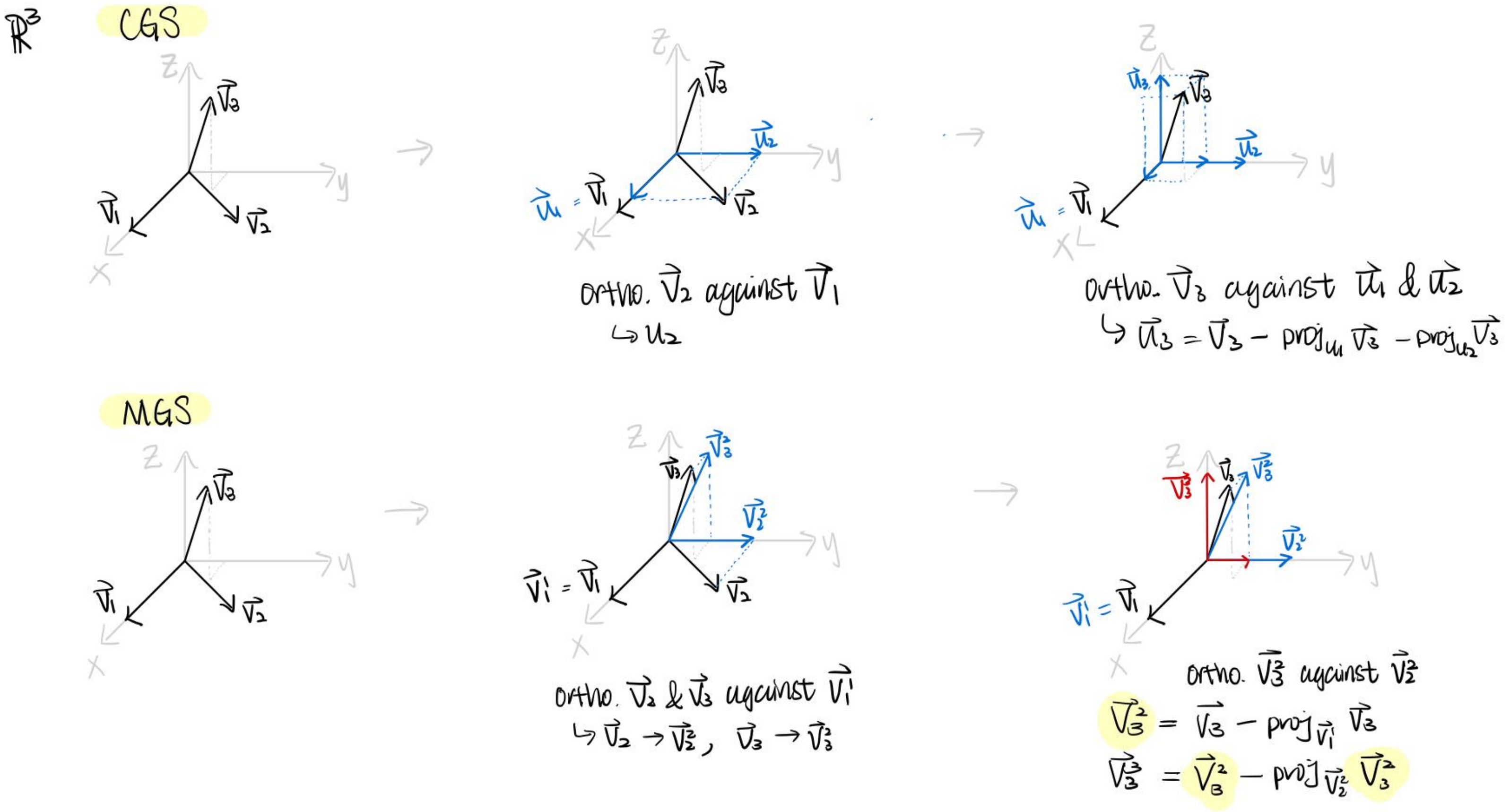
- Significantly automate the currently very tedious and error-prone method of manually constructing FP error bounds.

Conclusion and Personal Takeaway

- We extended the Mathcomp library to include basic linear algebra theorems
  - Basis Theorem, dot product properties, definition of projections…
- The linear algebra library is used to prove the properties of GS
  - Finished the proof for CGS using this library
- The library can be extended to floating point reasoning and error analysis.

- Mechanize math is ~~easy~~ hard
- Formalizing a software is challenging because there are missing pieces → have to develop a library
- Lack of automation in theorem proving → need for better automation
- Grateful to have Flocq and Mathcomp to reason about floats

GS has different versions, CGS and MGS



ortho. $\vec{V}_2$ against $\vec{V}_1$

↳ u_2

ortho. $\vec{V}_3$ against $\vec{u}_1$ & $\vec{u}_2$

↳ $\vec{u}_3 = \vec{V}_3 - \text{proj}_{u_1}\vec{V}_3 - \text{proj}_{u_2}\vec{V}_3$

ortho. $\vec{V}_2$ & $\vec{V}_3$ against $\vec{V}_1^1$

↳ $\vec{V}_2 \to \vec{V}_2^2$, $\vec{V}_3 \to \vec{V}_3^2$

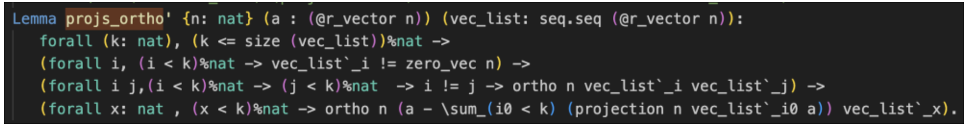
ortho. $\vec{V}_3^2$ against $\vec{V}_2^2$

$$\vec{V}_3^2 = \vec{V}_3 - \text{proj}_{\vec{V}_1^1}\vec{V}_3$$

$$\vec{V}_3^3 = \vec{V}_3^2 - \text{proj}_{\vec{V}_2^2}\vec{V}_3^2$$

Key results – Theorem 1

Lemma 2.2 (projs_ortho). *Let $a, v_0, v_1, \dots, v_{m-1}$ be vectors in $\mathbb{R}^n$. For any natural number k where $k \le m$, if $v_0, .., v_{k-1}$ are nonzero pairwise orthogonal, then $a - \sum_{i=0}^{k-1} proj_{v_i} a$ is orthogonal to $v_0, \dots, v_{k-1}$.*



```
Lemma projs_ortho' {n: nat} (a : (@r_vector n)) (vec_list: seq.seq (@r_vector n)):
  forall (k: nat), (k <= size (vec_list))%nat ->
  (forall i, (i < k)%nat -> vec_list`_i != zero_vec n) ->
  (forall i j,(i < k)%nat -> (j < k)%nat  -> i != j -> ortho n vec_list`_i vec_list`_j) ->
  (forall x: nat , (x < k)%nat -> ortho n (a - \sum_(i0 < k) (projection n vec_list`_i0 a)) vec_list`_x).
```

$\left(a - \sum_{i=0}^{k-1} proj_{v_i} a\right) \cdot v_x = a \cdot v_x - \left(\sum_{i=0}^{k-1} proj_{v_i} a\right) \cdot v_x$	distribute dot product	Our work
$= a \cdot v_x - \left(\sum_{i=0, i \neq x}^{k-1} proj_{v_i} a + proj_{v_x} a\right) \cdot v_x$	pull out v_x term	mathcomp
$= a \cdot v_x - \left(\sum_{i=0, i \neq x}^{k-1} proj_{v_i} a\right) \cdot v_x - proj_{v_x} a \cdot v_x$	distribute v_x	Our work
$= a \cdot v_x - \left(\sum_{i=0, i \neq x}^{k-1} \frac{v_i \cdot a}{v_i \cdot v_i} v_i\right) \cdot v_x - \frac{v_x \cdot a}{v_x \cdot v_x} v_x \cdot v_x$	expand $proj$	Our work
$= a \cdot v_x - \left(\sum_{i=0, i \neq x}^{k-1} \frac{v_i \cdot a}{v_i \cdot v_i} (v_i \cdot v_x)\right) - \frac{v_x \cdot a}{v_x \cdot v_x} (v_x \cdot v_x)$	use vec_dot_scale_commute	Our work
$= a \cdot v_x - \left(\sum_{i=0, i \neq x}^{k-1} \frac{v_i \cdot a}{v_i \cdot v_i} \times 0\right) - \frac{v_x \cdot a}{v_x \cdot v_x} (v_x \cdot v_x)$	$v_i \cdot v_x = 0$	mathcomp
$= a \cdot v_x - v_x \cdot a$	$v_x \neq 0$ and nonzero_vec.	Our work
$= 0.$		

MGS

MGS_prop

Let $v_0, ..., v_{m-1}$ be the input vectors of MGS. Let $v_0^0, v_1^1, ..., v_{m-1}^{m-1}$ be the result of MGS.

For all $0 \leq k < m$, if $v_i^i \perp v_j^j$ for $i, j = 0, ...k-1$, then $v_k^k = v_k - \sum_{i=0}^{k-1} proj_{v_i^i} v_k$.

proof.

Fix k. We are going to prove a stronger result:

$$v_k^h = v_k - \sum_{i=0}^{h-1} proj_{v_i^i} v_k.$$

for $0 \leq h \leq k$. This way, it is easy to see that when $h = k$, we get want we want: $v_k^k = v_k - \sum_{i=0}^{k-1} proj_{v_i^i} v_k$.

We are going to prove this by inducting on h.

base case: $h = 0$. $v_k^0 = v_k - \sum_{i=0}^{-1} proj_{v_i^i} v_k = v_k$. By definition of MGS, this holds.

Inductive hypothesis: Assume $v_k^h = v_k - \sum_{i=0}^{h-1} proj_{v_i^i} v_k$ for some $0 < h < k$.

Inductive step: We want to show $v_k^{h+1} = v_k - \sum_{i=0}^{h} proj_{v_i^i} v_k$ assuming $h + 1 \leq k$ (so we are still in the valid range).

Math: MGS_helper

let $v_0, ..v_{m-1}$ be the input vector. let $v_0^0, ..., v_{m-1}^{m-1}$ be the result of MGS.

Then, $v_k^k = v_k - \sum_{i=0}^{k-1} proj_{v_i^i} v_k$ for $k = 0, .., m-1$ and $v_k^k \perp v_0^0, .., v_{k-1}^{k-1}$.

proof. Induction on k.

Base case: $k = 0$, then $v_0^0 = v_0 - 0 = v_0$ is true by definition.

Inductive hypothesis: assume $v_k^k = v_k - \sum_{i=0}^{k-1} proj_{v_i^i} v_k$ and $v_k^k \perp v_0^0, .., v_{k-1}^{k-1}$ for some $k < m$.

Inductive step: We want to show $v_{k+1}^{k+1} = v_{k+1} - \sum_{i=0}^{k} proj_{v_i^i} v_{k+1}$ and $v_{k+1}^{k+1} \perp v_0^0, .., v_k^k$ assuming $k + 1 < m$.

Passing the IH $v_k^k \perp v_0^0, .., v_{k-1}^{k-1}$ into MGS_prop we can get $v_{k+1}^{k+1} = v_{k+1} - \sum_{i=0}^{k} proj_{v_i^i} v_{k+1}$.

Using this result, together with some proof about $v_0^0, ..v_k^k$ being nonzero, we can use projs_ortho to show v_{k+1}^{k+1} is orthogonal to $v_0^0, ..v_k^k$, which completes the proof.