F9Analytics, LLC

Cona Inverse Rotation

The 21st century N-Dimensional Non-Gaussian Algorithm for Solving System Problems of the form Ax = b

US Copyright Office

John J. Cona
January 31, 2018

Cona Inverse Rotation: The 21st century N-Dimensional Non-Gaussian Algorithm for Solving System Problems of the form Ax = b

- **Discovered By**: **John J. Cona, on 9/22/2017 9:48:07 AM**

The "Cona Inverse Rotation" algorithm is an update and extension of the original "Cona Inverse" previously registered with the **US Copyright Office Registration No. TXu 2-066-619**. This "Cona Inverse Rotation" algorithm should supersede the prior work in that the structure of the algorithm has been simplified and generalized to extend to n-dimensions.

The "Cona Inverse Rotation"; as it shall be known; is the 21st century algorithm for computing and solving n-dimensional system problems where a "Matrix Inverse" is required (i.e. $A^{-1)}$. It should be clarified that **this algorithm; the Cona Inverse Rotation, and its underlying methods are an original embodiment and not a derivation of any prior algorithms or methods in existence,** such as those known as Gaussian Methods (including the modified Gaussian LU Decomposition) that were all borrowed from the Chinese and date back to 200 BC.

It should be clarified that what makes this algorithm unique to physicists and mathematicians alike is not only its application of what one might consider a simple component-wise pairing of a traditional 2x2 Inverse across the n-dimensional planes (that is after the connection of such has been made), but also the mechanics or "first principles" which would allow one to construct or follow such a course of logic geometrically in the first place, which is exactly why it carries the name "Rotation".

In essence, what led to the discovery of "Cona Inverse Rotation" was simply thinking about rotations in the plane not in a single-sense as known today, but in a dual-sense as we will most definitely apply in the future. Without getting too technical, when one thinks of rotating a plane spanned by two coordinate axes, one typically rotates an axis in a single direction using a typical rotation convention similar to the Givens Rotation $G(i,j,\theta)$ where theta θ is the single direction radian that moves one axis in a counterclockwise direction against the other. In the Cona Inverse Rotation $C(i,j,Ci)$ the Ci that replaces theta θ represents two rotations albeit in a non-traditional dual-sense.

A few interesting properties to consider for those with intimate knowledge of matrix operations:

- When implementing the Cona Inverse Rotation there is no requirement to pre-compute unit vectors for a rotation, you can simply use full numeric form within the matrix. As one might imagine, this simplicity should not only improve its computational cost but also make it easier to teach to younger students – possibly even middle-school – no college required.

- By unifying two rotations into a single unified-rotation we reduce the computational path of the "Matrix Inverse" to that of a single triangularization – thus operating on the Lower and Upper("LU") matrix in unison across the entire n-dimensional space.

- For precision; should one let the Cona Inverse Rotation algorithm run indefinitely on an n-dimensional matrix, it will reach a mathematical limit or constant equal to the Identity Matrix.

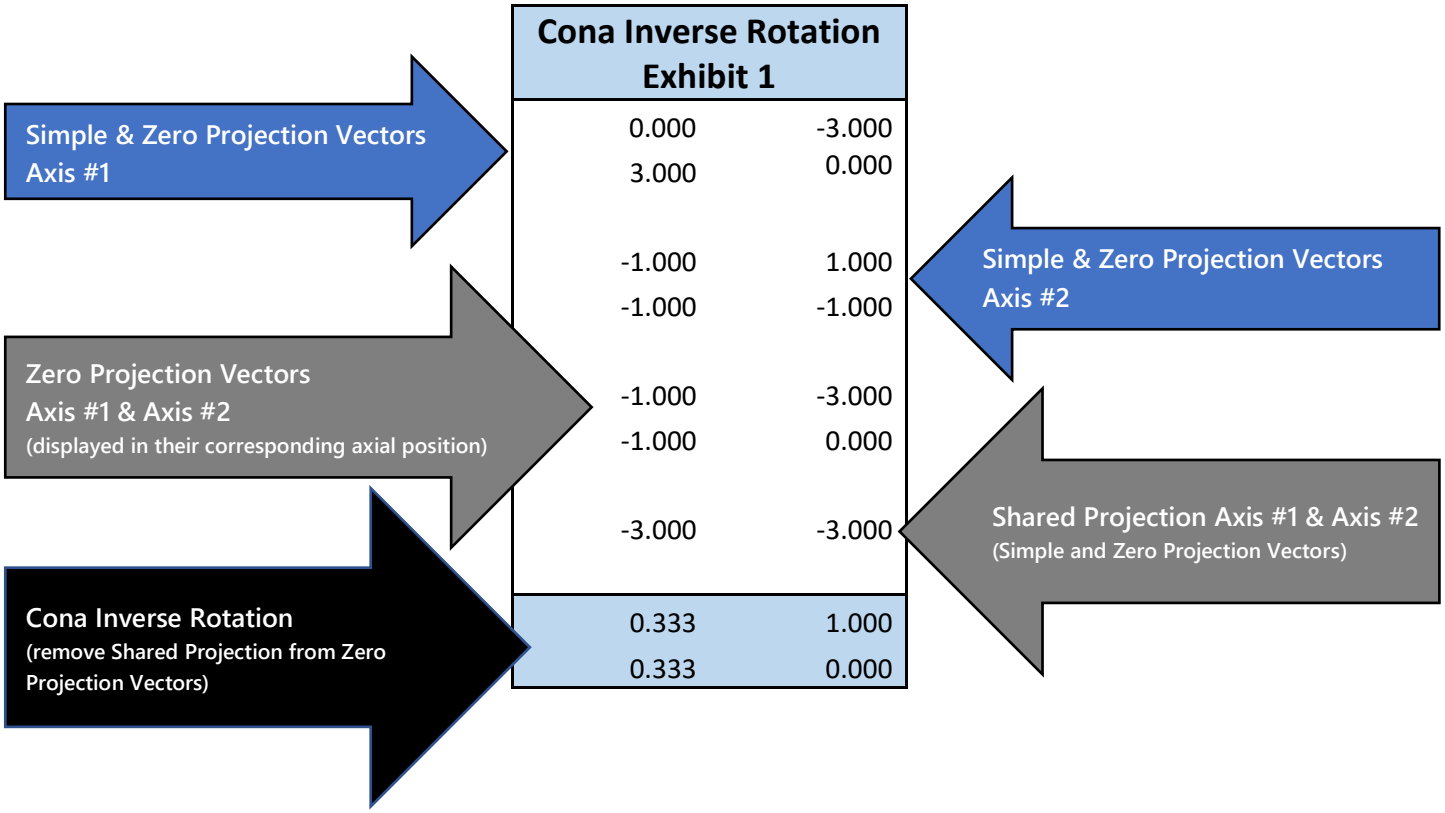
Cona Inverse Rotation: The 21st century N-Dimensional Non-Gaussian Algorithm for Solving System Problems of the form Ax = b

The "Cona Inverse Rotation" algorithm exploits what can be called the "Zero Projection Vectors" of every plane that interacts within the system A. The algorithm begins in a sequential manner by pairing the first-column first two components with their equivalent axial counterparts second-column first two components in an independent orthogonal mathematical rotation. Once the pairing is complete we thereby remove the "Shared Projection" between the axial counterpart Simple Vector and its corresponding paired Zero Projection Vector. To be clear the "Zero Projection Vector" when projected upon its own axial "Simple Vector" equals "Zero" – In essence it's the orthogonal equivalent of the axial "Simple Vector" rotated in the direction of its axial counterpart in the plane.

For example, in the following Matrix A the first "Cona Inverse Rotation" would be computed as follows:

A			
	0	1	-1
	3	-1	1
	1	1	-2

Beginning with the left-most axis of the Matrix, and working component-wise down the axial dimension, compute the "Cona Inverse Rotation" by selecting the Cosine and corresponding Sine components (see Exhibit 2) to form a "Simple Vector", then compute their corresponding "Zero Projection Vector" as shown in Exhibit 1. Align the "Zero Projection Vectors" with their corresponding axial "Simple Vector" (as displayed in Exhibit 1) and remove via division their "Shared Projection" column-wise, to arrive at the computed "Cona Inverse Rotation" (Exhibit 1, labeled with Black Arrow shaded in light-blue).



Cona Inverse Rotation: The 21st century N-Dimensional Non-Gaussian Algorithm for Solving System Problems of the form Ax = b

To extend the "Cona Inverse Rotation" algorithm to compute the Matrix Inverse of an n-dimensional matrix, use a rotational approach similar to the Givens Rotation with the exception that a known angle is not necessarily computed; in other words, **only** the pattern is followed (as shown in Exhibit 2). The numerical pattern followed in the "Cona Inverse Rotation" is a similar pattern known to mathematicians as a Jacobi Rotation or Givens Rotation with the exception that the "Cona Inverse Rotation" argument "Ci" (as noted in blue Exhibit 2) replaces the standard radian or degree theta θ. Most of the conceptual implementation and application of the "Cona Inverse Rotation" in n-dimensions will have Mathematicians, Physicists, and Educators alike very comfortable in that it still follows the same geometric rules albeit in an entirely new and modern computational approach.

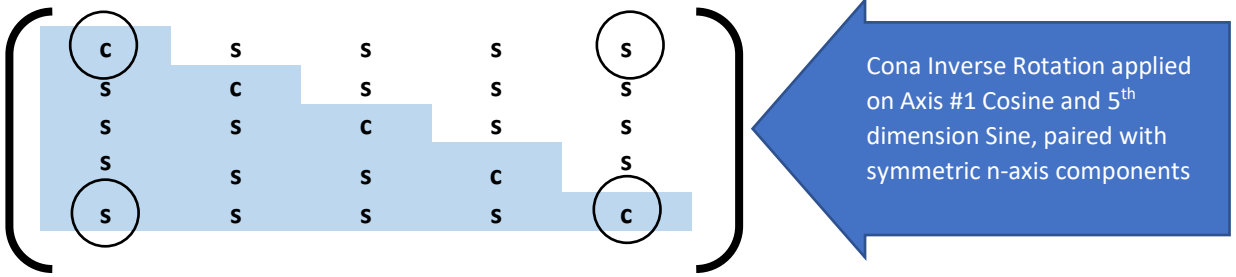
Exhibit 2

$$\mathbf{C}(i,j,\mathrm{Ci}) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}$$

To implement the "Cona Inverse Rotation" algorithm to compute the Matrix Inverse of an n-dimensional matrix, begin in the left-most axis of the matrix (Axis #1), always anchoring on the Cosine "c", thereby computing each individual "Cona Inverse Rotation" (as described in Exhibit 1) required to move Axis #1 into the "Identity" form where "c" is moved to "1" and "s" is moved to "0". As one moves down Axis #1 rotating it into "Identity" form, the other n-axis, via their paired relationship, will also be partially moved into the "Identity".

Each "Cona Inverse Rotation" is applied as an embedded 2 X 2 rotation matrix as to always anchor on the Cosine "c" and the Sine "s" component of the axis in question and its corresponding paired "c" and "s" axis components (as shown in Exhibit 3). Essentially, the rotational components are applied in a billiard pocket format, where as you move down each axis by component to obtain "Identity" form, you are always rotating on the plane interactions. As you obtain the "Identity" form of the left-most axis in the Matrix, move right to the next axis at the Diagonal and repeat until the entire matrix has been rotated into the full "Identity Matrix" – at this point the algorithm ends - you have computed the Inverse of the Matrix (that is the Matrix A you started with is now the Matrix Inverse A^{-1}, See Exhibit 4).

Exhibit 3



Cona Inverse Rotation: The 21st century N-Dimensional Non-Gaussian Algorithm for Solving System Problems of the form Ax = b

Example of the Cona Inverse Rotation applied to a simple 3-dimensional Matrix A to compute the Matrix Inverse and Solution to System Problem Ax = b.

Exhibit 4

Cona Inverse Rotation #1	
0.000	-3.000
3.000	0.000
-1.000	1.000
-1.000	-1.000
-1.000	-3.000
-1.000	0.000
-3.000	-3.000
0.333	1.000
0.333	0.000

Cona Inverse Rotation #2	
1.000	-1.000
1.000	1.000
-2.000	0.000
0.000	-2.000
-2.000	-1.000
0.000	1.000
-2.000	-2.000
1.000	0.500
0.000	-0.500

Cona Inverse Rotation #3	
1.000	0.500
-0.500	1.000
1.000	-1.000
1.000	1.000
1.000	0.500
1.000	1.000
0.500	0.500
2.000	1.000
2.000	2.000

A				b
	0	1	-1	-1
	3	-1	1	4
	1	1	-2	-3
	0.33	1.00	0.00	
	0.33	0.00	0.00	
	0.00	0.00	1.00	
	1	0	0	
	0	1	-1	
	1	1	-2	
	1.00	0.00	0.50	
	0.00	1.00	0.00	
	0.00	0.00	-0.50	
	1	0	0	
	0	1	-1	
	0	-0.5	1	
	1.00	0.00	0.00	
	0.00	2.00	1.00	
	0.00	2.00	2.00	
	1.00	0.00	0.00	
	0.00	1.00	0.00	
	0.00	0.00	1.00	

Cona Inverse Rotation #1

Cona Inverse Rotation #2

Cona Inverse Rotation #3

[1] The "Identity Matrix"

[1] The "Identity Matrix" following sequential "Cona Inverse Rotations"

Cona Inverse Rotations taken sequentially form the desired Matrix Inverse

Solution to System Problem, Ax = b				
			x	Check
0.333333	2.333333	1.333333	1	-1
0.333333	0.333333	0.333333	2	4
0	-1	-1	3	-3

- **Proof and verification** of the **Cona Inverse Rotation** algorithm claim is simple. To date, every textbook cites and describes algorithms primarily based on derivations from Johann Carl Friedrich Gauss – Known as Gaussian Methods (or similar modified forms known as LU Decomposition) that are borrowed methods from the Chinese dating back to 200 BCE– this new algorithmic approach named the "Cona Inverse Rotation" utilizes an entirely new and original logic that until now was undiscovered.
- **As Addendum** to the **Cona Inverse Rotation** is the base python code implementation on a simple 2x2 subspace plane that can be extended to n-dimensions as described herein.