Department of Physics and Mathematics

Engineering Physics

Selected Research Topics

‘Non-Euclidean Economic Data Spaces: Differential Geometry and Curvature as an Analytical Framework’

Author: Stoopen Secades, Enrique

Advisor: Dr. Sebastián Nájera Valencia
Committee Member 1: Dr. Miguel Ángel García Aspeitia
Committee Member 2: Dr. Nelson Omar Muriel Torrero

XIX · XI · MMXXV

Dedicated to all the people who, just like me, suffer from this terrible disease called "curiosity."

Abstract

This work presents the first iteration of the **Riemannian Henry Embedding Method (RHEM1)**, a geometric approach to data analysis that combines tools from *differential geometry*, *mathematical physics*, and *machine learning*. The model aims to characterize the intrinsic structure of high-dimensional datasets through the estimation of local metrics, principal curvatures, and global properties of the underlying manifold.

As a case study, the methodology was applied to a global financial dataset composed of eight representative variables from international markets between 2018 and 2024. The results show that variations in the intrinsic curvature of the data space can be linked to major economic events —such as the cryptocurrency market crash (2018), the COVID-19 pandemic (2020), the global inflationary surge (2021), and the Russian invasion of Ukraine (2022)— revealing structural transitions in the coherence of the global economic system.

RHEM1 demonstrates that geometry can serve as a bridge between the exact and social sciences, offering a new way to interpret complex phenomena through the shape and structure of data.

Keywords: intrinsic curvature, data geometry, manifold learning, ISOMAP, local PCA, non-Euclidean economics, financial analysis.

Artificial Intelligence Usage Acknowledgment

This research was developed with the assistance of OpenAI's **ChatGPT**, used primarily as a tool for improving clarity, structure, and exploring ideas during the writing process (and, occasionally, as a therapist and crisis-management counselor). All final decisions, content, and interpretations reflect my own thinking, research, academic responsibility, and perception of the world.

Acknowledgments

I have always tried to challenge my own limits—both in what I can do and in what I long to understand. And so far, I can say that I am the luckiest man for having grown up in a home that pushed me to better myself in every possible way.

This thesis is not just that. It's not just words, graphs, and bits of code glued onto paper. What you see here is the fruit of my effort and the path I have walked throughout this wonderful journey that Engineering Physics has been. A journey that would have been impossible to complete without the encouragement of many people.

Thank you to my mom, Viviana Secades, for giving me immense love throughout my life, and for always supporting every decision I have made. For being my mom and my example of compassion, understanding, and empathy. Thank you, ma, for teaching me and shaping me into a man of integrity, and above all, for cheering me on every step of the way. Thank you for believing in me, for respecting and celebrating my authenticity and identity—not only as a professional, but as a person.

Thank you also to my dad, Enrique Stoopen Margain, for teaching me what it means to be a professional, for demanding excellence from a place of love, and for always leading by example: discipline, hard work, honesty, integrity, and humanity. Thank you for giving me the world, pá, and for always making it clear that I can count on you. I am deeply proud to carry your name, and I only hope my future lives up to it.

Thank you to my brother Vicente, for being an example of authenticity and creativity in my life. Thank you for offering a new perspective on the world and for never taking anything for granted. In a way, that is exactly what I am trying to do with this work.

And of course, it is impossible not to thank my brother Lorenzo—my best friend and the fuel of my life. Thank you, Wiwi, for being there for me. Despite the distance, I always feel you close. You inspire me to be a better person, a better brother, and a better friend. I am incredibly proud to be able to say you are my brother.

Thank you to my grandmother Lourdes for her warmth, affection, and encouragement in difficult moments.

Infinite thanks to Universidad Iberoamericana for giving me my education. It is an honor and a privilege to call it my *alma mater*. Thank you, FISMAT and all its members, for allowing me to be myself throughout my degree, for respecting my identity and my particular style, and for supporting me as a student at all times.

Thank you as well to my professors: Dr. Miguel Ángel García Aspeitia, Dr. Carla

Valencia Negrete, Dr. Gerardo Martínez, Dr. Nelson, and Ing. Salvador Arteaga. I deeply appreciate your work and passion for teaching. Thank you for infecting me with this terrible disease called curiosity. I will always be in your debt for your kindness, patience, and warmth.

To Dr. Sebastián Nájera Valencia, my advisor: thank you for believing in me even when I didn't. Thank you for sharing both your humanity and your brilliance with me in every class and every meeting. I will never be able to thank you enough for everything you have done for me, both academically and personally. You are one of my greatest examples.

To my friends in Bergen —Kiki, Anouk, Melly, Janèle, Annia, Chinguiz, Rüya, Luciano, Phillipa, Lucrecia, Sofía— thank you for being part of the best year of my life.

Thank you as well to my friends from the Ibero: Regina, Nico, Alonso, Brandon, Andrés and, above all, Mark —my best friend in school— for sharing this journey with me, for supporting me, and for helping me without hesitation. I wouldn't be here without you, my friend. Thank you.

And finally, I want to thank myself.

Thank you, Henry, for failing Calculus 1 with the intention of learning. Thank you for the sleepless nights and countless sacrifices. Thank you for holding on to your identity and for being yourself at all times. Thank you for having the courage to choose a life of discomfort just to feed your hunger for knowledge. Thank you for asking those "dumb" questions in class. And above all, thank you for always seeking humor and beauty in every aspect of life.

Table of Contents

1. Chapter 1: Our planet, Earth, is perfectly flat. (?)

1.1. Introduction

Our planet, Earth, is perfectly flat. To a few people, still not fully initiated into the modern world, this statement might seem perfectly reasonable. Others —like I hope you, dear reader— might be thinking: "How can a physics engineering student proudly claim such nonsense in his final undergraduate thesis?"

If that's your reaction, I'm afraid to tell you that the phrase is correct... partially.

For practical purposes, the world we live in is perfectly flat. For us —and our tiny human scale— the laws of physics follow Newtonian mechanics, and geometry is the geometry we've always known. In our mundane reality, the sum of the interior angles of a triangle is 180 degrees —exactly half the sum of any quadrilateral we can draw on a sheet of paper. For us, the shortest distance between any two points is a perfectly straight line, and parallel lines never intersect, no matter how long we draw them.

For a civil engineer about to raise a building, the Earth is perfectly flat. For an urban surveyor, the fact that he lives on a "sphere" is completely negligible. And for a farmer tracing furrows in his field, the Earth is as flat as his wish for rain.

Fun fact: approximately 4 % of the world's population believes that the Earth is flat, ignoring modern (and not-so-modern) scientific studies and handing over their intellect to contemporary conspiracies whose sole purpose is to misinform about the most basic notions of the world we inhabit.

The other 96 %, who know the Earth is not flat... do not know just how not-flat it really is.

Just like the engineer, the surveyor, and the farmer, modern economists and financial analysts also live in a perfectly flat world, unaware of how non-flat it can truly be...

This happens because, in geometric terms, our planet is a differentiable manifold. Which means that —believe it or not— in the first words I wrote, I wasn't wrong.

That phrase:

"Our planet, Earth, is perfectly flat." ... is not incorrect. It is simply incomplete.

To be technically correct, we must go back to one of the fundamental principles of differential geometry. And also —to establish the starting point of this work— I want you to take the following with you:

Our planet, Earth, is perfectly flat... locally.

Even though the existence of people who believe our planet is flat is not necessarily a reason for mockery (although... maybe a little), this intuition is simply natural at our small scale.

To our eyes, the horizon is a straight line —a comforting illusion, much like the idea that we understand our place in this vast universe. And this everyday notion has been more than enough to carry humanity through great advancements: from Egyptian and Babylonian mathematics used in astronomy and land measurement to medieval navigation maps known as portolans.

In fact, if you stood at the seashore, you would see that roughly 5 to 6 km away, a medium-sized ship embarking on a journey begins to disappear behind that distance.

It is only at larger scales that disregarding the natural curvature of our environment becomes illogical. It is at these scales —both in distance and velocity— that Newton's flat physics and Galileo's "flat" reality begin to feel the effects of the very reality they aim to describe...

And it is precisely at this point that we realize mathematical models are nothing more than attempts —as imperfect as the humans who design them— to naïvely describe their reality.

It is instinctive, because we live in a reality that presents itself as perfectly flat, but stepping back just a little reveals the curvature of our deception.

But what happens when we realize the model doesn't fit reality as we would like? Well, there are two options: either we change reality (which is a bit difficult, if you ask me), or we change and adjust our model —that is, we change the scale at which we are performing our analysis.

An example of this is the renowned physicist Albert Einstein, who, with his genius and curiosity, realized that Newton's model of the universe was imperfect at cosmic scales. So he increased the scale, shifted the perspective, and discovered that the universe we inhabit curves to the rhythm of mass and energy.

Analogously, to have a model capable of describing socioeconomic phenomena, we also need to increase the scale. We must analyze how flat the space in which this information lives actually is.

For this, we will need a branch of mathematics known as differential geometry. It is here that we will broaden our perception.

Differential geometry is nothing more than a tool which, far from trying to describe reality at the human scale, questions it. It is the study of curved and twisted spaces

through calculus. With it, we can study and analyze how a body moves in a space that is not necessarily flat.

By using differential geometry in our analysis process, what we are really saying is that we no longer assume we are working in a flat data space, as is usually done; and consequently, that there exists some type of intrinsic curvature that (perhaps) governs the dynamics of these socioeconomic phenomena.

Because just as the Earth is not flat —no matter how much it seems to be— economic data spaces are not necessarily flat either.

Throughout this work, we will explore key concepts that will help us give a non-traditional interpretation to an economic dataset. From redefining everyday ideas such as distance, straight line, and curvature; to technical concepts such as what a datum actually is, what a dataset is, what an algorithm is, locality, machine learning model, geodesic, and differentiable manifold.

With these concepts, we will analyze the geometric structures underlying our data spaces. The goal is to change the scale and transform our way of seeing and interpreting the data, appealing to their intrinsic morphology, and therefore, to their real nature and dynamics.

With this, another tool comes into play —very trendy, yes, but also powerful— called machine learning (ML).

You've likely heard of ML at some point in your life. But to level the field, I find it prudent to define it clearly, leaving no room for subjectivity:

Machine Learning is a branch of artificial intelligence concerned with the development of algorithms and statistical models that allow a system to learn patterns from data and improve its performance on a specific task without being explicitly programmed for it.

Thus, by respecting the natural geometry of the data, we can apply ML models that learn directly from the patterns implicit in their original structure —without flattening them, without forcing them— and thereby predict economic phenomena more faithfully and deeply.

Since most machine learning models assume that data live in a flat space —i.e., without curvature— it becomes relevant to apply a method that does not share this assumption and observe how it behaves. The method chosen, for its mathematical elegance and refined brilliance, is ISOMAP. This tool allows us to perform dimensionality reduction while preserving geodesic distances in the data space, with the goal of capturing its underlying structure. From this representation, we will seek to estimate the local metric of the embedding and subsequently attempt to compute the intrinsic curvature of the space where the economic data reside.

1.2. General and Specific Objectives

The general objective of this work is to reclaim my love for physics and my passion for machine learning —to apply knowledge from different courses that inspired me to think beyond the obvious, beyond the superficial.

More concretely, the academic objective of this work is the exploration and analysis of the data space where economic phenomena "live." By exploring various nonlinear dimensionality reduction techniques such as ISOMAP, as well as mathematical tools from differential geometry and vector and tensor analysis, we aim to highlight the presence of curvature in these spaces. If this can be demonstrated, the efficiency of traditional (Euclidean) models used to study these phenomena can be questioned, and potential "econorelativistic" effects can be discussed.

1.2.1. Specific Objectives

- Critical analysis of the limitations of Euclidean paradigms in traditional economic models.
- Effective reconstruction of the metric of the emerging data space after dimensionality reduction.
- Discussion of economic implications of a possible intrinsic curvature in the data space.
- Questioning the linear interpretation of economic behavior by proposing a geometrically rich perspective more closely tied to the complexity of the data space.

1.3. Hypothesis

There exists an intrinsic geometric structure —a nontrivial morphology— in the space where data associated with economic phenomena reside. This space, or data space, possesses particular characteristics that vary depending on the dataset and could be described using tools from differential geometry.

If this structure is indeed curved (that is, if it exhibits nonzero curvature), then it may be evidenced through nonlinear dimensionality reduction techniques such as ISOMAP, followed by local metric reconstruction. The presence of curvature in these spaces would imply that traditional economic models, which assume a flat Euclidean framework, leave out fundamental properties of the underlying data reality.

Consequently, this plausible curvature could open new avenues for economic modeling that more faithfully reflect the real structure of the data, with potential applications in crisis prediction, the identification of emerging dynamics, and complex economic phenomenology.

The purpose of this work is to show that the space in which the data under analysis live has curvature $\neq 0$. To this end, the following methodology is proposed.

1. Nonlinear dimensionality reduction preserving geodesic distances between data points.
2. Reconstruction (estimation) of an induced metric on the representative manifold.
3. Curvature computation via the scalar curvature derived from the estimated metric.
4. Characterization of the data space through Christoffel symbols, the Riemann tensor, etc.
5. Geometric analysis and interpretation.

2. Chapter 2: The Compass of Curved Worlds

2.1. Euclidean Geometry

It would be disrespectful to begin the theoretical framework of this research without mentioning the "father of geometry," Euclid (325–265 B.C.). At that time, people already had knowledge and a solid understanding of many concepts attributed to Euclid.

This is because Euclid was the one who compiled and organized that knowledge, a knowledge that until then came from humanity's ability to perceive shapes and structures, to compare sizes and distances. It was with the evolution of the human intellect that the study of geometry became formalized.

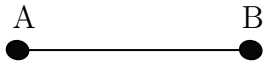
It was the ancient Greeks who decided to begin a theoretical and mathematical analysis of geometry, guided by illustrious figures such as Pythagoras, Archimedes, Hippocrates, Thales of Miletus, and, of course, Euclid. This resulted in a new way of seeing the world, compared to the more "practical" and ancient techniques used by civilizations such as the Egyptians or the Babylonians. [1]

Without any intention of boring you, dear reader, I will now describe what is truly important in this brief geometric introduction: Euclid's five axioms. Not only will we explain them briefly, but we will also reflect on their importance and the infinite possibilities that arise when we stop following the "geometric rules," the axioms of Euclid.

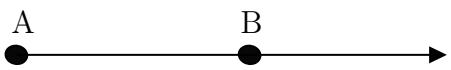
"More concretely, in the mathematical context, axioms are fundamental propositions that do not need to be proven because they are considered self-evident." [2]

In his book *Elements*, Euclid postulates five axioms to generalize the construction of geometric knowledge up to that point (and for many centuries after his death). These axioms are as follows: [3]

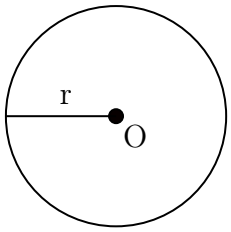
1. Se puede trazar una línea recta entre cualesquiera dos puntos.



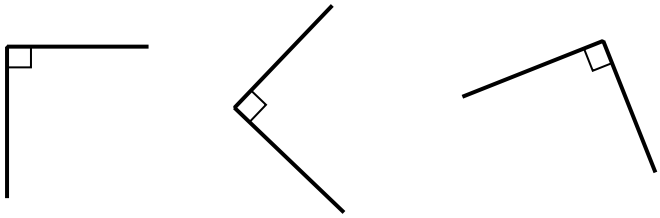
2. Se puede producir una línea recta finita continua en una línea recta.



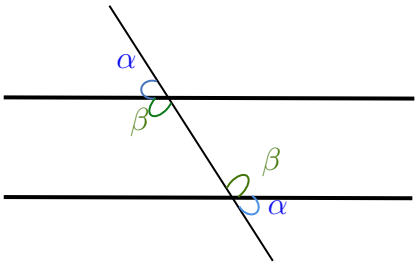
3. Se puede describir un círculo con cualquier centro y cualquier radio.



4. Todos los ángulos rectos son iguales entre sí.



5. Si una línea corta a otras dos formando ángulos internos menores de 180°, las líneas se encontrarán al prolongarse.



We can think of these axioms as bricks. Bricks that laid the foundation for the knowledge and development of what we now know as Euclidean geometry. This is the type of geometry we learned in elementary and middle school: that the angle sum of a triangle is always 180°, or that the area of a circle is πr^2.

These axioms defined our entire analogue understanding of the morphology of our surroundings for several centuries.

As an additional note and reflection, I would like to say that I find it magical and impressive what human beings can achieve by representing their environment on a piece of parchment, stone, or paper —and what can be deduced through observation and questioning.

2.2. Foundations of Differential Geometry

2.2.0.1. Dimensionality I'm not sure how technical my language needs to be to properly define dimensionality.

That said, I think my definition is elegant and a solid recap of what I've learned in classes and books.

For me, the dimensionality of a space is the number of independent coordinates needed to describe it locally.

- $\mathbb{R}^1$: One dimension. One coordinate is enough to describe the entire space.
- $\mathbb{R}^2$: Two dimensions, like x and y in a Cartesian plane. Think of it as a sheet of paper where we can talk about length and height.
- $\mathbb{R}^3$: Three dimensions, like the space we live in. With three coordinates x, y, and z (by convention), we describe length, height, and depth.
- $\mathbb{R}^n$: Now there's no example I can really give you; it's impossible for us to visualize more than three spatial dimensions. But the point stands: we would need n coordinates to describe this space. Length, height, depth... and whatever comes next.

Assuming we have a manifold $\mathcal{M}$ of dimension n at some arbitrary point x, we say that n is the dimensionality of the manifold. We can also call it an n-manifold or denote it as $\mathcal{M}^n$. [4]

2.2.0.2. Locality You'll also see me use the word "locality" many times. And this is one of the main differences from classical geometry.

Regardless of the manifold's complex morphology, we can take any point in it and define a small neighborhood where we recover Euclidean space $\mathbb{R}^n$.

We say that M is *locally Euclidean of dimension* n if, for every point $p \in M$, there exists an open set $U \subset M$ containing p and a homeomorphism[1]

$$\varphi : U \longrightarrow V \subset \mathbb{R}^n,$$

where V is an open set[2] of $\mathbb{R}^n$.

With this concept, we can now properly define an n-manifold.

[1] A *homeomorphism* is a bijective, continuous function with continuous inverse between topological spaces.

[2] An *open set* is a subset of a topological space in which every point has a neighborhood fully contained in it.

2.2.1. Metric Structures

Let's now say that we have points in this space that we don't fully understand. This space has a strange shape. The first thing we must do to perform mathematics in it is to have a notion of "distance." In our mundane reality, we measure distances with a ruler. I also want to recover the ability to compute angles between curves. This is a metric structure: the "ruler and compass" that allows us to geometrically characterize the manifold.

2.2.1.1. Riemannian Metric g_{ij} Thinking of a metric as an analogy to a ruler and compass, we can say that there are infinitely many... an infinite set of measuring instruments, depending on what we want to measure and how.

Just like rulers with different markings or different types of compasses, we have many metrics in differential geometry. Here we focus on the Riemannian metric, which lies at the heart of differential geometry. For this, we start with a pillar of Euclidean geometry: the dot product.

A commutative and linear operation from a vector field to the real plane.

$$\vec{v} \cdot \vec{w} = \sum_{i=1}^{n} v^i w^i.$$

We also have the norm of a vector:

$$|v| = \langle v, v \rangle^{1/2}.$$

And the angle between two nonzero vectors:

$$\cos\theta = \frac{\langle v, w \rangle}{|v|\,|w|}.$$

Now, given a manifold, we can equip it with a metric —that is, with instruments that help us characterize it geometrically.

Up to this point, we only have the ability to "go" from one point to another; we still can't measure lengths or angles.

To do so, we define a tangent space at each point p of the manifold $\mathcal{M}$. We denote this space by $T_p\mathcal{M}$, and g_p will be the inner product on it, so we use [5]:

$$\langle v, w \rangle_g = g_p(v, w)$$

2.2.1.2. Distance and Geodesics The ability to measure distances or lengths is arguably the most important aspect of a Riemannian metric. But distances of what? Lengths of what?

Naturally, distances between points —yes— points in the manifold. But more fundamentally, what we are doing is measuring the length of curves: curves going from one point to another, parameterized and continuous on an interval.

$$\gamma : I \to M, \quad I \subseteq \mathbb{R}.$$

If γ is defined on $[a, b] \subset \mathcal{M}$, its length is:

$$L_g(\gamma) = \int_a^b |\gamma'(t)|_g \, dt$$

A geodesic is then the curve that minimizes the distance between two points of the manifold, measured with the metric g. Another way to see it is as the curve whose tangent vector remains parallel to itself.

I won't get into tedious details here, but I'll write down the geodesic equation anyway, just in case you, reader, are curious.

2.2.1.3. Levi-Civita Connection Defining a connection is the next natural step in our geometric journey.

With the metric (g), we know how to measure. With a connection, we know how directions change between different points of the manifold.

As with metrics, there are infinitely many possible connections —you could even define your own if you ever needed to.

One way I like to think of a connection is imagining ourselves on a giant torus (and no, I don't mean an angry horned animal, but something resembling a giant donut). If we stand on this giant donut, we could place a tangent vector at each point on the surface. I find it useful to imagine this tangential vector as an arrow sitting locally on the surface.

If we walk along a curve on this not-flat surface, what we want is a way to define "not rotating" that arrow.

On a flat surface this is easy; on a curved one we need a new notion of derivative that respects the morphology of the surface. This derivative is called the Levi-Civita connection.

In its most fundamental form, the connection is a covariant derivative that lets us differentiate vector fields with respect to other vector fields. Think of it like standing in a city surrounded by tall buildings. You can feel the wind's direction: maybe hitting your face, maybe your back. Now imagine turning down a different street —how does the wind change? That is what the covariant derivative tells us.

$$\nabla_X Y, \quad X, Y \text{ vector fields on } \mathcal{M}.$$

Of the infinite possible connections, the Levi-Civita connection is the only one satisfying two key conditions:

- **Torsion-free:**
$$\nabla_X Y - \nabla_Y X = [X, Y]$$
- **Metric compatibility:**
$$X[g(Y, Z)] = g(\nabla_X Y, Z) + g(Y, \nabla_X Z)$$

By the existence and uniqueness theorem, every Riemannian manifold admits a unique connection with these two properties. Its coefficients are the Christoffel symbols, which encode how directions change.

The formula is:

$$\Gamma^{\lambda}{}_{\mu\nu} = \tfrac{1}{2}\, g^{\lambda\rho} \left(\partial_\mu g_{\nu\rho} + \partial_\nu g_{\mu\rho} - \partial_\rho g_{\mu\nu}\right)$$

2.2.2. Curvature and Tensor Formalism

2.2.2.1. Riemann Tensor I could probably spend the next 20 pages of this document talking about the Riemann tensor —the soul of differential geometry. Obviously we're not doing that, because it would require defining far too many things, and that is not the purpose of this thesis.

But it is indispensable to have at least a notion of what this mathematical object is.

The Riemann tensor measures how a vector changes when it is parallel transported along a closed loop on a manifold.

In Euclidean space, the vector always returns unchanged to its initial position. On a curved surface, it does not.

In simpler terms: the Riemann tensor measures the noncommutativity of the covariant derivative.

It is a direct measure of the manifold's intrinsic curvature.

$$R^{\rho}{}_{\sigma\mu\nu} = \partial_\mu \Gamma^{\rho}{}_{\nu\sigma} - \partial_\nu \Gamma^{\rho}{}_{\mu\sigma} + \Gamma^{\rho}{}_{\mu\lambda}\Gamma^{\lambda}{}_{\nu\sigma} - \Gamma^{\rho}{}_{\nu\lambda}\Gamma^{\lambda}{}_{\mu\sigma}$$

The tensor satisfies the following properties:

- **Antisymmetry:**
$$R_{\rho\sigma\mu\nu} = -R_{\rho\sigma\nu\mu}$$
- **Pair symmetry:**
$$R_{\rho\sigma\mu\nu} = R_{\mu\nu\rho\sigma}$$
- **First Bianchi identity:**
$$R_{\rho\sigma\mu\nu} + R_{\rho\mu\nu\sigma} + R_{\rho\nu\sigma\mu} = 0$$

2.2.2.2. Ricci and Scalar Curvature R From this magnificent object —which contains much (if not all) of the curvature information of the manifold— we derive the Ricci tensor and the scalar curvature R.

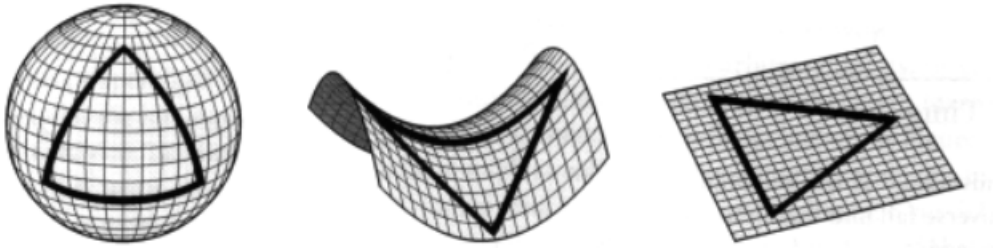
These are simpler ways to store the curvature information encoded in the Riemann tensor. You can think of it in terms of "compressing information."

The Riemann tensor has four indices; it carries a lot of curvature information that may be difficult to interpret. A contraction gives the Ricci tensor $R_{\mu\nu}$, which represents the average curvature around a point.

Another contraction, now with the inverse metric, gives the scalar curvature:

$$R = g^{\mu\nu} R_{\mu\nu}$$

- If $R > 0$: the space curves like a **sphere**.
- If $R = 0$: the space is **flat**.
- If $R < 0$: the space curves like a **saddle**.



2.2.3. Intrinsic vs. Extrinsic Curvature and Examples

Etymologically, it's not hard to understand the difference between intrinsic and extrinsic.

Intrinsic curvature does not depend on the surrounding space in which the manifold is embedded. Extrinsic curvature is how the object appears when embedded in a higher-dimensional space.

I know this sounds confusing, so let me use a classic analogy...

Imagine an ant unknowingly walking on a cylinder much larger than itself. For the ant, that surface feels perfectly flat —its curvature is imperceptible. But you, who can see the cylinder from outside, know perfectly well that it is not flat and that its curvature is nonzero.

You are measuring the cylinder's **extrinsic** curvature, while the ant is measuring **intrinsic** curvature.

2.3. Machine Learning and Data Analysis

Machine Learning is a type of artificial intelligence. And at the time of writing this, it is extremely trendy and increasingly accessible.

ML aims to make a computer program "learn" without explicit instructions. It has several branches and emblematic models. I will mention three that I find particularly important:

- **Supervised learning:** based on labeled data. The model trains on a subset of data and therefore knows the correct output. This allows it to adjust parameters, perfect the model, and predict future outcomes.
- **Reinforcement learning:** reinforcement learning is like training a dog. The model receives a reward or punishment for good or bad behavior. Of course, mathematically —I don't want to see anyone hitting their computer to "train models."
- **Unsupervised learning:** the model receives unlabeled data. It has no target output. Its purpose is to find patterns and structures in the data that may not be apparent to humans. Naturally, this is the type of model we will design for our geometric analysis to see whether the morphology of the data space reveals interesting insights about the chosen phenomenon.

2.3.1. What Is the Data Space?

This is a good moment to define something that will be fundamental from here onward: the data space.

Mexicans live in a designated region we call Mexico; Australians live in Australia; Norwegians in Norway.

Likewise, colors "live" in a color space governed by chromatic rules, as described in color theory.

So it doesn't sound crazy to say that the data of a socioeconomic phenomenon must "live" somewhere.

It is appropriate to name and define that place.

We will call this place the *data space (DS)*.

Now an important question arises: what differentiates the data space from more popular concepts like a dataset, a database, or a data structure?

A dataset is well defined by ISO (2022), ISO/IEC 22989:2022. [6]

A dataset is a collection of data with a defined structure —for example, a collection of Instagram posts containing the hashtags #rugby or #TaylorSwift.

A database is a static, stored collection of data accessible through a DBMS.

A data structure is an internal organization of data that allows easier manipulation and interpretation.

So what is a data space?

I don't know whether it's because I've been thinking about this for so long, or because the definition reflects my personality... but I like to think of a data space as something much more fundamental and "natural."

I define the data space as follows:

Data space: We will call data space the intrinsic "place" where data associated with a given phenomenon reside.

Such a space is geometrically characterizable; it possesses its own structure and metric that determine the relationships between data. The data space does not emerge from the data —rather, the data emerge within a space whose geometry and structure already exist.

2.3.2. Dimensionality Reduction Techniques

In ML there is a concept that is treated almost like a taboo —a problem that can significantly affect model performance.

This is known as the *curse of dimensionality*. It occurs when we have data describing high-dimensional phenomena that cannot be faithfully represented in lower dimensions.

The term was coined by Bellman in *Dynamic Programming* [7]. It is easy to understand that this "curse" is not exclusive to programming; it has haunted physicists and astronomers for centuries, and now its analogue appears in information sciences.

But this does not mean that high-dimensional data spaces leave us trapped. There are many dimensionality reduction techniques that help mitigate this "curse" using different algorithms and models. I will briefly mention three:

- **Principal Component Analysis (PCA):** A method dating back to before World War II, whose main function is computing covariance and correlation matrices. Eigenvalues are then computed and interpreted as principal axes to reduce dimensionality while preserving as much variance as possible. [8]
- **t-SNE:** A high-dimensional data visualization technique. It constructs neighborhood probabilities based on point densities. The low-dimensional representation yields clusters that reflect the local geometry —though not the global morphology. [9]
- **ISOMAP:** My favorite, and honestly a great inspiration for this thesis. From *Isometric Mapping*, this method builds a matrix of geodesic distances using shortest-path algorithms. A multidimensional scaling is then performed to preserve the geometry of the data. The issue: it is computationally expensive and very sensitive to parameter choices. [10]

2.3.3. Geometry and Learning on Manifolds

Manifold learning arises from the need to deal with the curse of dimensionality. These are the techniques mentioned earlier (and more) that aim to uncover underlying structures in a lowerdimensional data space.

My goal is to create, design, or at least attempt to build a model that, inspired by others, belongs to this family of machine learning approaches.

Ideally, something that allows us to characterize both the local and global geometry of an n-dimensional data space without explicitly reducing it to a d-dimensional one.

3. Chapter 3: Proposed Methodology

The model is called RHEM (*Riemannian Henry Embedding Method*). This is because everyone calls me Henry and I have always thought it would be incredible to have my name next to Riemann's in something... (not that I consider myself worthy of that, I just thought it was now or never).

Throughout the experiments, two models ended up being built, very similar to each other: RHEM v.1 and RHEM v.2, both with the same purpose but different applicability. More on that below. In the first stages, and in broad terms, the steps are the same for both models.

3.1. RHEM1

In this first iteration of the model, we will use a series of steps which, for those who know or are familiar with ISOMAP, will look quite similar to the initial stages of that method.

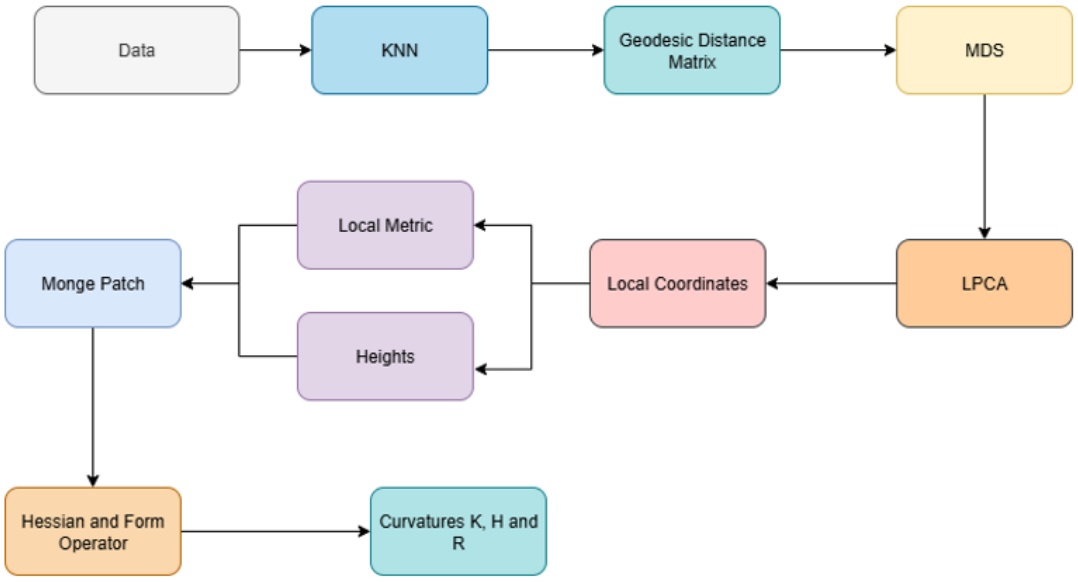


Figura 1: RHEM1 pipeline

3.1.1. KNN

KNN is a classification algorithm. Basically, we are saying that if something looks like a dog, and everything around it also looks like a dog, then they are dogs.

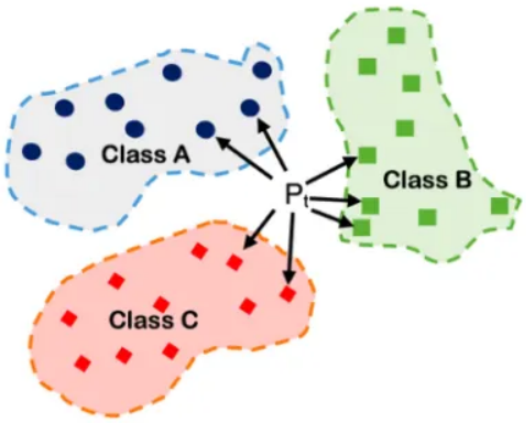
KNN, or K-nearest neighbors, is the step where, for each point in our data space, we assign a number K of nearby neighbors.

There are many ways and criteria to determine what "nearby" means in this context. For the purposes of this work, since we want to recover locality and Euclidean geometry near a given point, we will use the Euclidean distance:

$$\text{dist}(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$$

With this, we aim to generate a sort of connection between the data: a traversable distance.

We compute this distance between a reference point and the other points, and then take the K points whose distance is smallest.



Naturally, K will be a sensitive value, something that can affect the results. If K is too small, we risk losing connectivity between data points. On the other hand, if K is too large, we risk losing the sense of locality we want to recover at this initial stage, something related to what is known as "small worlds."

The result will be a graph that allows us to recover locality and, later on, numerically compute geodesic distances between any pair of points.

3.1.2. Geodesics

We previously mentioned geodesics, so I will not go into mathematical detail about what they represent, other than reminding you that we can interpret them as the

shortest path between any two points.

We also saw a way to compute them mathematically. The problem is that, in this discrete space, those formulas are not particularly practical.

However, we can rely on existing algorithms that compute these shortest or optimal paths.

I am sure you have already used these algorithms at some point. Every time you open Google Maps or Waze, a set of possible routes is computed and the shortest one is selected. And although there are several algorithms that do this, here we will experiment with two: Floyd–Warshall (FW) and Dijkstra's algorithm, and compare their computational efficiency later in a JupyterLab notebook.

The result will be a symmetric matrix D representing the true connectivity, the true distance between points in our space —no longer the Euclidean distance. With this, we can begin to describe the global geometry of the space.

3.1.3. MDS: Multidimensional Scaling

So far, we have the following: a matrix we call D that contains the shortest distance between any two points in our data space. We obtained it using a shortest-path algorithm and a graph G of points connected to their K nearest neighbors.

We have distances. Shortest distances, yes, but still distances in an N-dimensional space. What we want now is to reduce the dimensionality of this space. [11]

To do that, we will use MDS (Multidimensional Scaling), as in ISOMAP.

The goal is to determine coordinates or positions from these distances. A nice analogy is that I give you three pieces of string: one of length 5, one of 4, and one of 3 units. You already know those pieces form a right triangle; you just need to determine its position.

That is exactly what we will do here: assign positions that satisfy the distances in this matrix —which already represent shortest distances— so that we can reduce the dimensionality of the space while preserving those distances as much as possible.

$$\text{MDS}: \ D \longmapsto X \in \mathbb{R}^{n \times d} \ \Big| \ \|x_i - x_j\| \approx D_{ij}$$

where:

- D_{ij} is the original distance between points i and j,
- x_i is the new coordinate of point i in low dimension.

With this, we can reduce the dimension while preserving (as far as possible) the original geometry.

3.1.4. LPCA: Local Principal Component Analysis

We can now move on to the local geometry. To do this, we will perform LPCA (Local Principal Component Analysis), a principal component analysis carried out locally for each point. We are, in essence, approximating this geometry from the information itself, from the data.

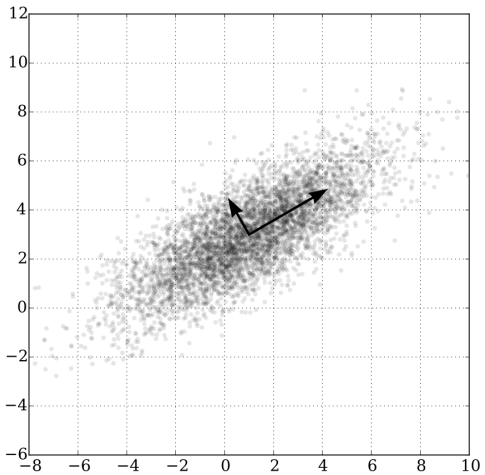
Given an arbitrary point $x_0 \in DS$, we consider the same K nearest neighbors that we used in KNN to perform a principal component analysis and estimate the tangent plane and its normal vector.

We can think of the normal vector as a reference with respect to which we can measure extrinsic curvature.

The tangent space at the point will be denoted $T_{x_0}DS$, and on it we will define a new coordinate system. In this space we recover locality and can perform differential calculus.

One could think of $T_{x_0}DS$ as the local flat space where Euclidean geometry is recovered.

These new coordinates will be given by the LPCA results. The directions with high data variance —those that maximize variance— will be the coordinates of the tangent plane, while the direction with the smallest variance will be the normal direction.



3.1.5. Local coordinates

In this space where we recover locality, we can build a local patch with its own coordinates. We can think of this space as the flat neighborhood of x_0 where we project its K nearest neighbors.

We center the local data:

$$Y_i = x_i - x_0, \quad \text{for } i = 1, \ldots, K.$$

Let $T_{x_0}\mathcal{M}$ be the tangent plane estimated via LPCA, spanned by the principal eigenvectors:

$$T_{x_0}\mathcal{M} = \text{span}(v_1, v_2, \ldots, v_d),$$

where d is the intrinsic dimension of the manifold.

The projection of Y_i onto the tangent plane is:

$$(u_{i1}, u_{i2}, \ldots, u_{id}) = (\langle Y_i, v_1 \rangle, \langle Y_i, v_2 \rangle, \ldots, \langle Y_i, v_d \rangle).$$

In this way, we obtain local coordinates

$$(u_{i1}, \ldots, u_{id}) \in \mathbb{R}^d, \quad \text{for } i = 1, \ldots, K,$$

which allow us to work with a local parametrization of the manifold around x_0.

3.1.6. Local metric

If you recall, something essential for our geometric characterization is the metric: the "ruler and compass" that tells us how to measure distances and compute angles.

We denote this metric by g, and it is also known as the *first fundamental form*. What we will do is compute g for this patch in the local coordinates.

We go from a cloud of points in this local patch to an analysis of how those points are distributed in each direction. The result is a matrix M containing this information. For a two-dimensional tangent plane, it is given by:

$$M = \begin{pmatrix} M_{uu} & M_{uv} \\ M_{uv} & M_{vv} \end{pmatrix}$$

- M_{uu} measures how spread out the points are in the u direction.
- M_{vv} measures how spread out the points are in the v direction.
- M_{uv} measures how u and v vary together, indicating whether the cloud is tilted or rotated.

The entries of the matrix are:

$$M = \begin{pmatrix} \frac{\sum_i w_i\, u_i^2}{\sum_i w_i} & \frac{\sum_i w_i\, u_i v_i}{\sum_i w_i} \\ \frac{\sum_i w_i\, u_i v_i}{\sum_i w_i} & \frac{\sum_i w_i\, v_i^2}{\sum_i w_i} \end{pmatrix}$$

The weights w_i are used to give more importance to neighbors closer to x_0.

The trace of the matrix, ideally diagonal, is:

$$\text{trace}(M) = M_{uu} + M_{vv}$$

and with this we can normalize to remove the scale and analyze only the shape of the patch, that is, its anisotropy.

The ratio

$$r \;=\; \frac{\lambda_{\text{máx}}}{\lambda_{\text{mín}}}$$

acts as a quick indicator of **anisotropy**. When $r \approx 1$, the local shape is essentially isotropic (almost circular), while large values of r indicate that the cloud is **stretched** in some principal direction.

Finally,

$$g = \frac{M}{\textbf{trace}(M)}.$$

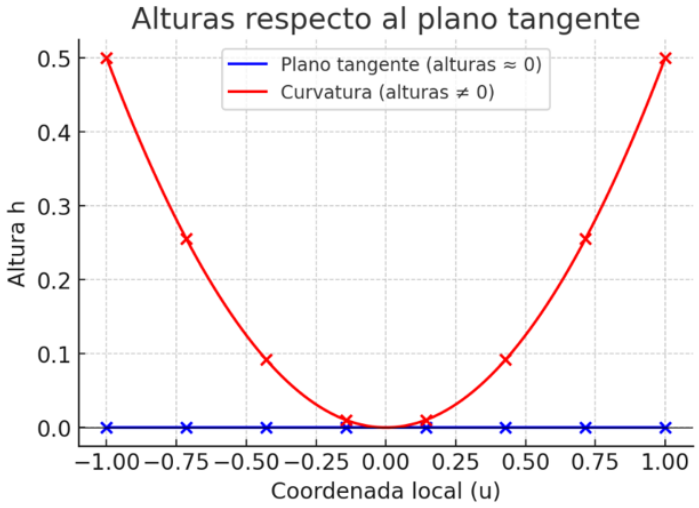
3.1.7. Heights

On the other hand, in this local patch with its new coordinate system, if we are in 2D we can think of these coordinates as (u, v). We also have a normal direction of lowest variance among the neighbors of a point x_0 and its K nearest neighbors x_i.

We will compute the height of each neighbor with respect to the tangent plane generated by x_0. We can think of x_0 as an "anchor" that keeps this tangent plane fixed.

In this way, we can see how far above or below the tangent plane each neighbor lies. If the generated patch were truly flat, all heights would be exactly zero.

Otherwise, we can say that there is curvature in that patch.



Thus, we obtain information of the form (u, v, h) for each point in the patch, which gives us information about the extrinsic curvature of the space.

3.1.8. Monge patch

We interpret the height of each nearby neighbor as $h(u, v)$; we are seeing how points move in the u and v directions and how their height h changes.

We can now perform a parabolic fit:

$$h(u, v) \approx a\,u^2 + b\,uv + c\,v^2 + \alpha\,u + \beta\,v + \gamma.$$

The linear terms do not tell us anything about curvature; they only encode tilt or inclination.

The real information we care about lies in the values of a, b, and c.

The coefficients a and c are the curvatures in the principal directions, while b tells us about the twisting that may exist.

These coefficients are what we will use to build the Hessian matrix H.

3.1.9. Hessian and shape operator

The Hessian matrix, which we denote by H, has as entries the second derivatives of $h(u, v)$. These derivatives tell us how the slope changes. If this change is large, it means there is curvature.

The relevant second derivatives are:

$$\frac{\partial^2 h}{\partial u^2} = 2a, \qquad \frac{\partial^2 h}{\partial u\,\partial v} = b, \qquad \frac{\partial^2 h}{\partial v^2} = 2c.$$

Therefore, the **Hessian** of the Monge parametrization $h(u, v)$, evaluated at $(u, v) = (0, 0)$ corresponding to the point x_0, is:

$$H = \begin{pmatrix} 2a & b \\ b & 2c \end{pmatrix}.$$

The eigenvalues are the principal curvatures κ_1, κ_2, and the eigenvectors of H are the principal directions along which the space bends more or less.

3.1.10. Curvature

With this, we can now estimate the classical curvatures that describe the local geometry of the space. We will compute the Gaussian curvature, the mean curvature, and the scalar curvature, all from the principal curvatures κ_1 and κ_2 obtained from the Hessian.

$$K = \kappa_1 \, \kappa_2 \qquad \text{(Gaussian curvature)}$$

$$H = \frac{\kappa_1 + \kappa_2}{2} \qquad \text{(mean curvature)}$$

$$R = 2K \qquad \text{(scalar curvature)}$$

- **Principal curvatures** κ_1, κ_2: they measure how much the surface bends in the two directions where curvature is maximum and minimum. Together, they fully describe the local curvature.
- **Gaussian curvature** $K = \kappa_1 \kappa_2$: it indicates the type of curvature at the point.
  - $K > 0$: the surface is convex (bowl-shaped).
  - $K < 0$: the surface is hyperbolic (saddle-shaped).
  - $K = 0$: the surface is locally flat or cylindrical.
- **Mean curvature** $H = \frac{\kappa_1 + \kappa_2}{2}$: it represents the average curvature at the point.
- **Scalar curvature** $R = 2K$: a rescaled version of the Gaussian curvature, used in Riemannian geometry and general relativity to describe how space curves around the point.

3.2. RHEM2

At the time of writing this, the methodology is being generalized to be compatible with various types of curvature characterization. The updated version of RHEM will be presented in Volume II of this text.

4. Chapter 4: Architecture of a Curved Space

In order to validate the general methodology proposed, it is necessary to analyze a familiar and well-understood space. I considered many possibilities and, to make my life easier, I decided to analyze spheres. Spheres whose curvature I know, whose data I can easily simulate, and that can help me validate the accuracy of the proposed model.

The expected results are numerical values of principal curvatures, mean curvature, and scalar curvature. With this, it is enough to start a geometric characterization of a data space that we will eventually not know beforehand and that might, just might, give us a new perspective on the data that live in it.

To do this, we will use an example that is easy to simulate, work with, and compute: a sphere in 3D, $S^2 \subset \mathbb{R}^3$.

4.1. Motivation for the Test Case

Before applying the methodology to real economic data, it is necessary to validate that the proposed pipeline **??** is capable of detecting curvature in a data space where it is known. This test aims to demonstrate the feasibility of the method in a controlled setting.

We will use Python for all of the code, and JupyterLab as a controlled environment where we can run the different experiments needed.

Throughout the steps, we will perform tests that I call *Sanity Checks*, just to make sure everything is on track.

4.2. Construction of the Synthetic Dataset

4.2.1. Choice of Geometry

A set of 800 synthetic data points was generated, distributed over a surface of known curvature. In this case, a unit sphere of radius $R = 2$ was used, whose scalar curvature is $R = \frac{1}{2}$ at every point.

4.2.2. Data Generation

The following equations were used to obtain points uniformly distributed over the spherical surface:

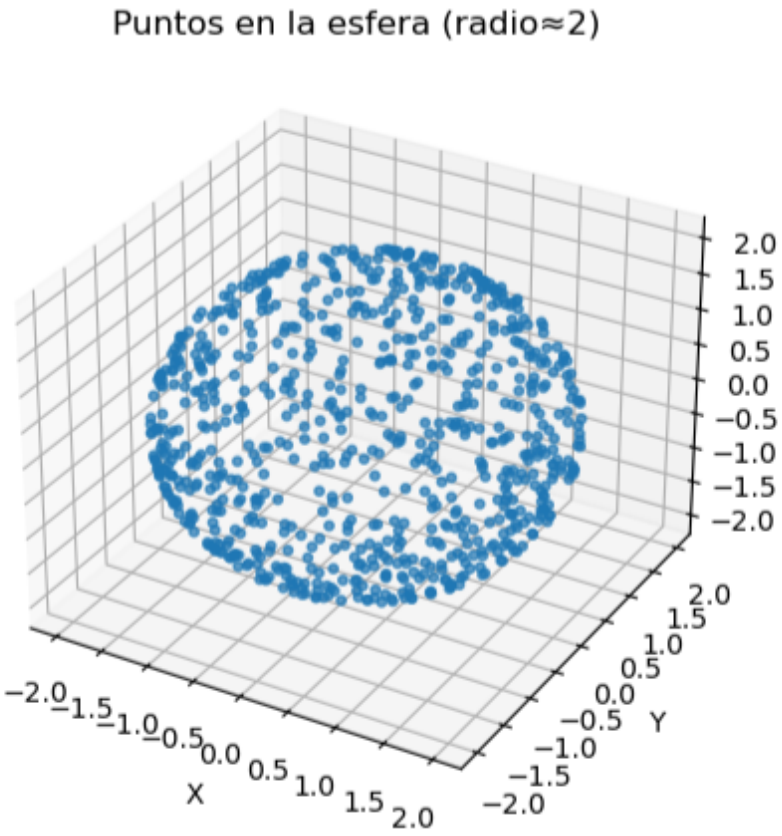
$$\begin{cases} x = \sin\theta\cos\phi \\ y = \sin\theta\sin\phi \\ z = \cos\theta \end{cases} \qquad \text{where} \quad \theta \in [0, \pi],\ \phi \in [0, 2\pi]$$

We generated $N = 800$ points, adding a small Gaussian noise to simulate imperfections.

Sanity Check:

```
First 5 observations of X:
[[ 0.29   -0.23   -1.96]
 [ 0.54    1.84    0.53]
 [ 0.80   -0.98   -1.54]
 [-0.59    1.79    0.63]
 [ 0.97   -0.38    1.70]]
```

We confirm that 800 points were generated and that the mean radius is 2. We have our data space!

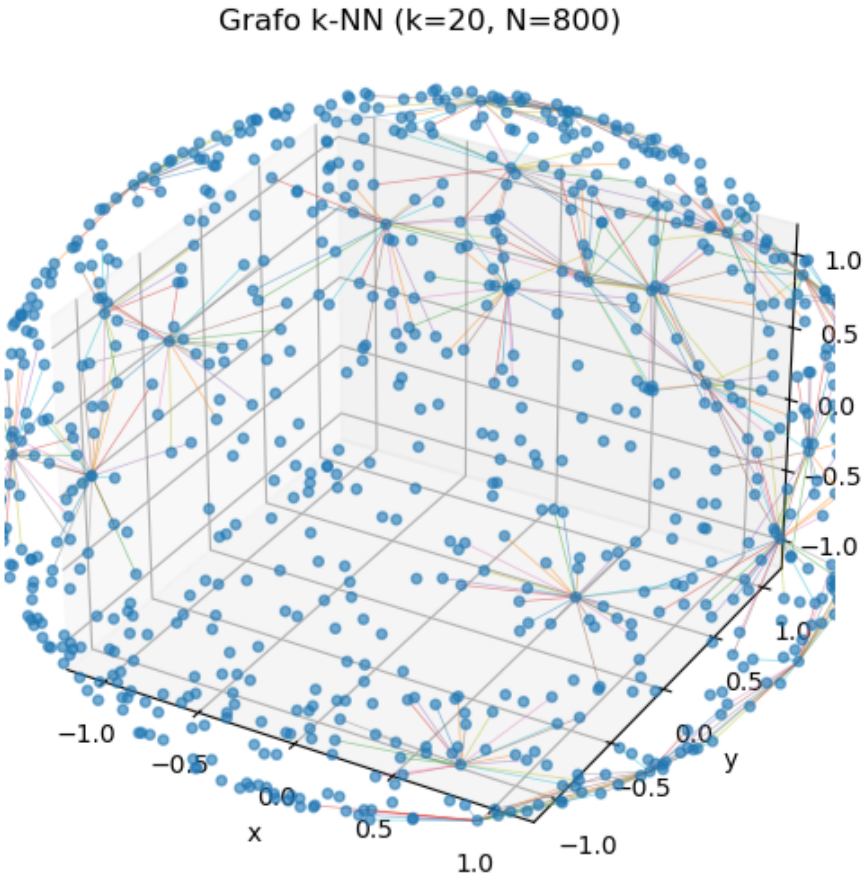


4.2.3. KNN

We chose to work with $k = 20$ nearest neighbors to generate the graph G.

The resulting graph contains $|V| = 800$ nodes and $|E| = 8766$ edges.

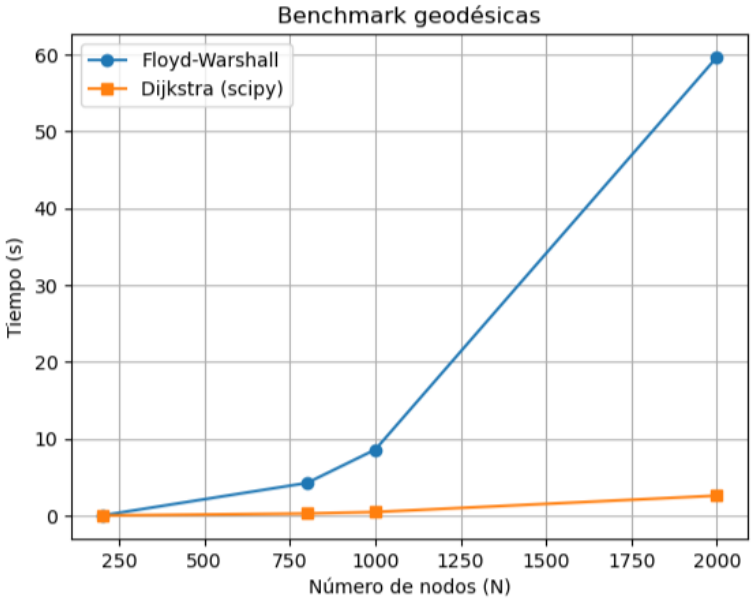
This confirms the construction of the desired graph. We will be performing an analysis at a good scale: not too small, and not so large as to generate *small worlds*.



The graph density is important, since later it guarantees a good numerical approximation of the geodesic distances. This will be fundamental for preserving the geometry of the space as much as possible when performing dimensionality reduction with MDS.

4.2.4. Geodesics

With the graph G generated, we can run an experiment in JupyterLab to see which *shortest path* method we can use. The main idea here is to use something that does not require too much computational power. The algorithms compared are Floyd–Warshall and Dijkstra.



Dijkstra's algorithm was used to obtain the minimum distance between every pair of nodes. The implementation used corresponds to an optimized version based on priority queues, with a total computation time of approximately 0.34 seconds.

Since neighborhood graphs can have weakly connected components, a disconnection-handling strategy was implemented by assigning a large value (`big_value`) to pairs of nodes with no valid path. Afterwards, we performed structural checks on D:

Tabla 1: Sanity check of the geodesic distance matrix D.

Dimension of D	800×800
Algorithm used	Optimized Dijkstra
Computation time	~ 0.34 s
Symmetry $D = D^{\mathsf{T}}$	True
Zero diagonal $D_{ii} = 0$	True
Disconnection handling	`big_value` strategy

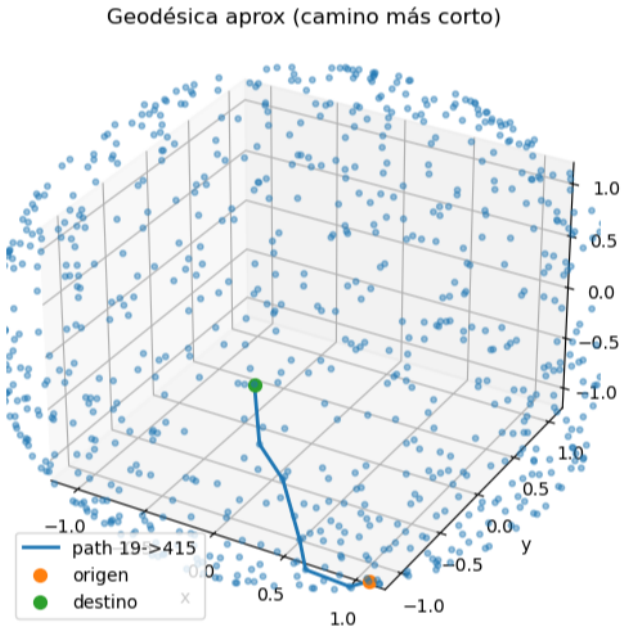
In particular, this allows us to apply methods such as Isomap, curvature analysis, and parallel transport on a geometrically consistent basis. This is why D is of utmost importance.

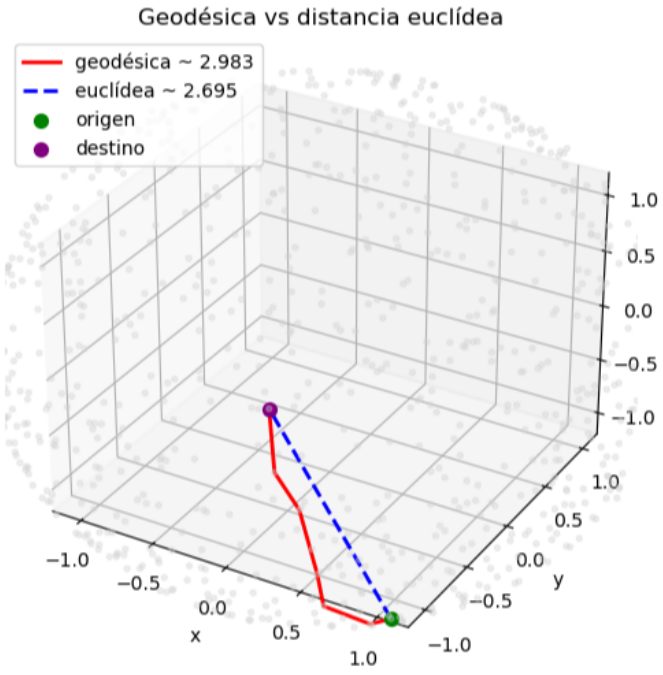
This matrix now correctly reflects the true connectivity between points in this space, which in this case we already know.

We can now look at an example between two points. I chose 19 and 419 for reasons I do not plan to discuss here. Anyone interested in my pseudo-obsession with numerology can contact me and we can gladly discuss it over a beer.

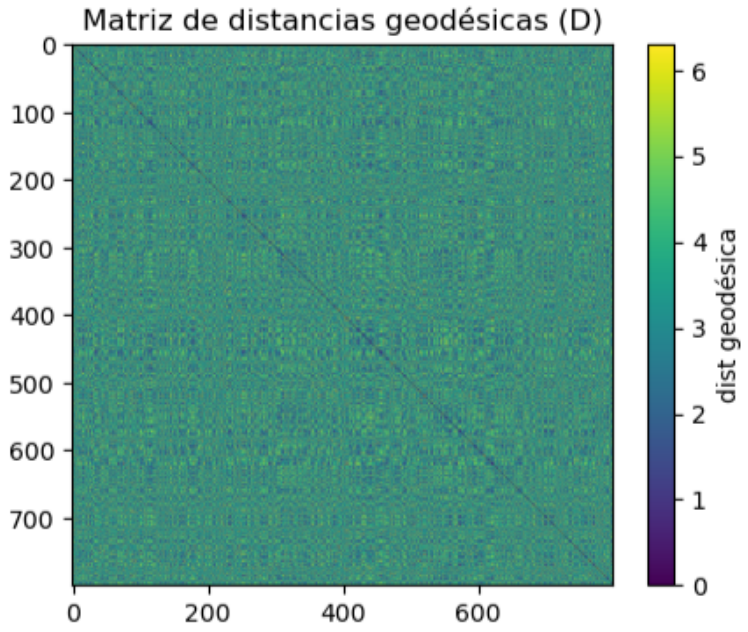
Anyway, the result is the following, and I think the difference between Euclidean

distance and the geodesic distance we computed is going to be very clear.





We carry out this computation for all points. We can visualize it via a heatmap that lets us inspect the structure of the matrix.

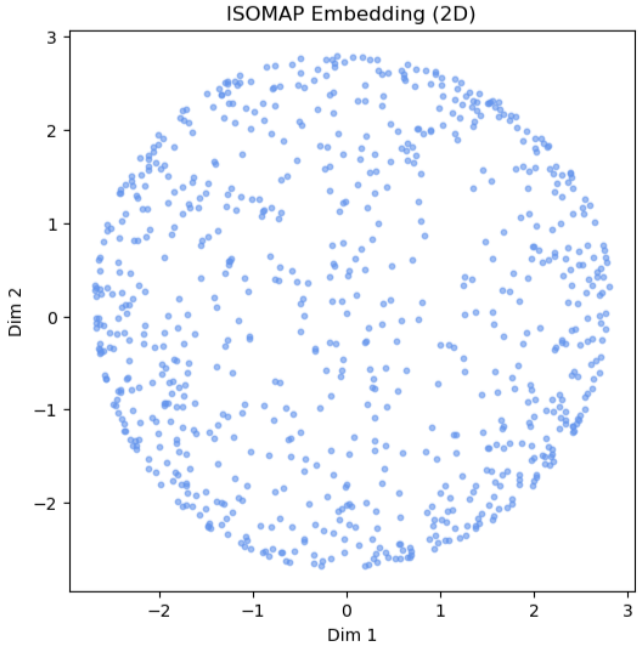


The main diagonal appears in dark color, indicating zero distances ($D_{ii} = 0$), as expected, since the distance from a point to itself is naturally zero. I also hope it is clear that the matrix is symmetric, since the distance from point A to point B must always be the same as from point B to point A.

4.2.5. MDS

With the distance matrix we now have, we can perform multidimensional scaling to reduce the dimensionality of the data while preserving these distances, which will now be represented in Euclidean fashion on a two-dimensional plane.

Two embeddings were computed: one in $d = 2$ and one in $d = 3$. In both cases, the result shows an approximately spherical structure, consistent with the geometry of the original dataset. The circular shape in the 2D projection is a direct result of preserving geodesic distances over a curved surface. This behavior is characteristic of ISOMAP, where dimensionality reduction maintains the innate nonlinear manifold structure of the space chosen to test the method.

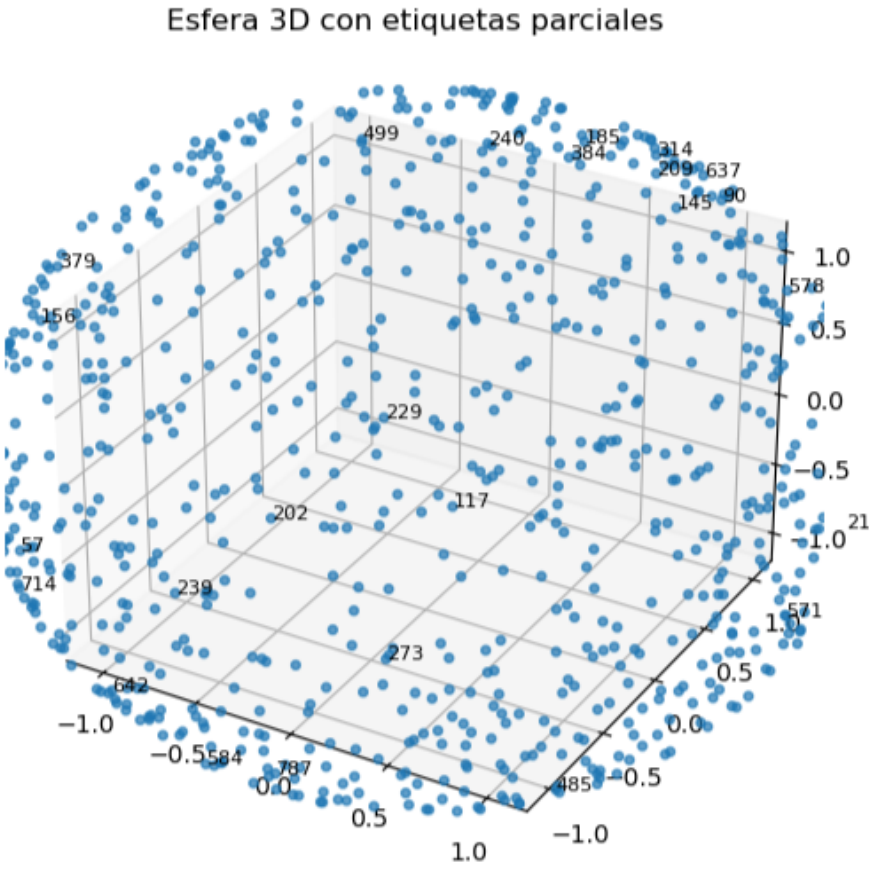


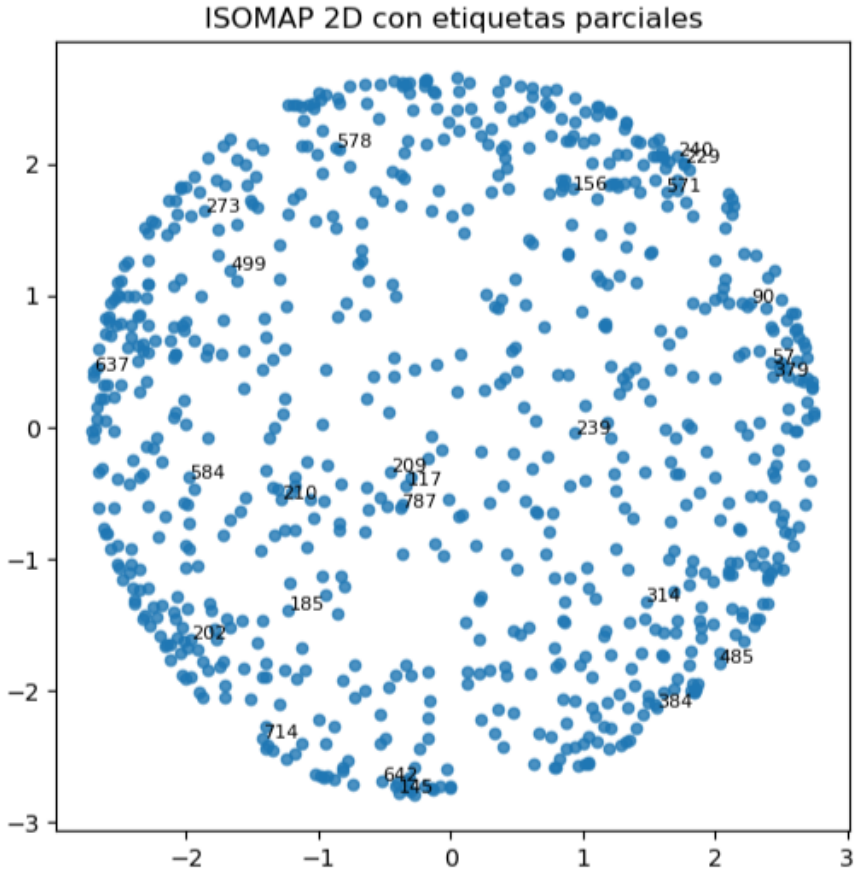
To validate that the embedding preserves the intrinsic geometry, we compared distances between randomly selected point pairs before and after dimensionality reduction. For example, for the pair of observations $(i, j) = (19, 419)$ we obtained:

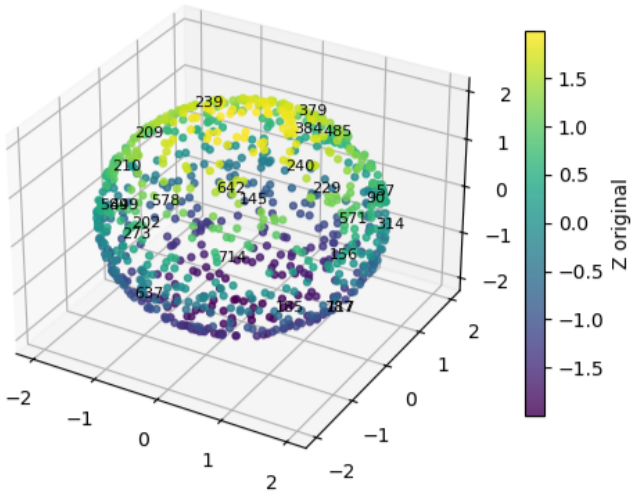
$$d_{\text{geod}}(19, 419) = 3{,}6593, \qquad d_{\text{embed}}(19, 419) = 3{,}3772.$$

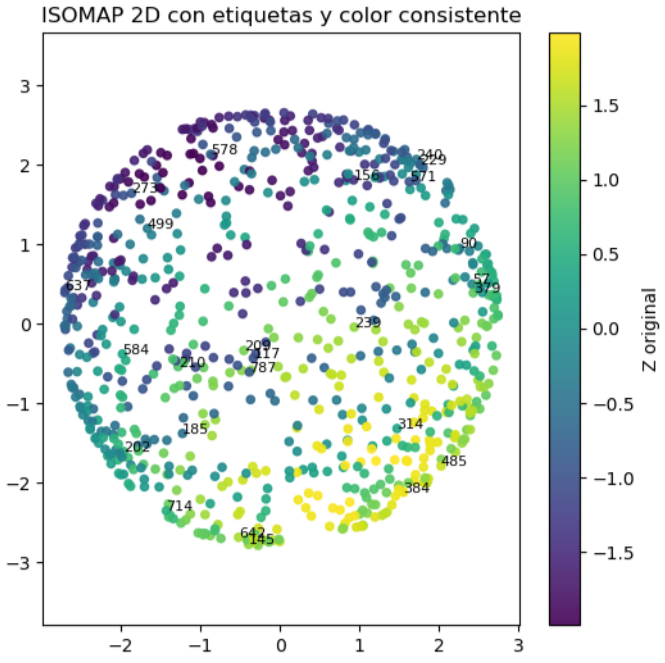
These quantities are comparable, which shows that the embedding approximately preserves the original metric relationships —especially considering that we reduced an entire dimension.

To better visualize what has happened, 25 points in the space were randomly labeled:


Esfera 3D con etiquetas parciales
499
240
185
384
314
209
637
145
90
379
578
156
229
202
117
57
714
239
273
642
584
787
485
571
21
1.0
0.5
0.0
−0.5
−1.0


ISOMAP 2D con etiquetas parciales
578
240
229
156
571
273
499
90
637
57
379
239
584
209
117
210
787
185
314
202
485
384
714
642
145
2
1
0
−1
−2
−3
−2
−1
0
1
2
3





And with this, ISOMAP is concluded, and we see that it turns out to be an essential method in situations where the structure of the data space is intrinsically nonlinear and requires a description in terms of differential geometry.

4.2.6. LPCA

We now perform principal component analysis at the local level, that is, on the nearest neighbors chosen from the beginning for each point.

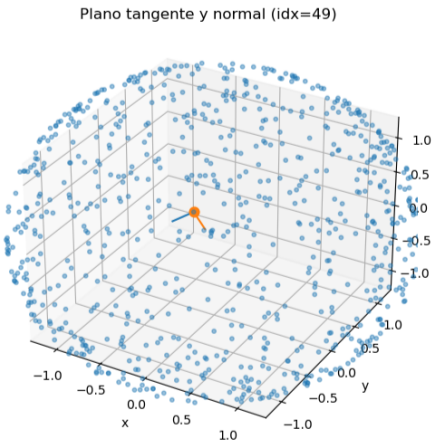
The first step is to center the reference point. For visualization, I chose point 49. Around this point, we take the cloud of the 20 nearest neighbors (as in KNN)

and perform a geometric decomposition to obtain the directions along which the neighbors spread out the most. These give us the directions of greatest variance, with which we construct the tangent plane at point 49. We also obtain the direction of least variance, which gives us the normal vector.

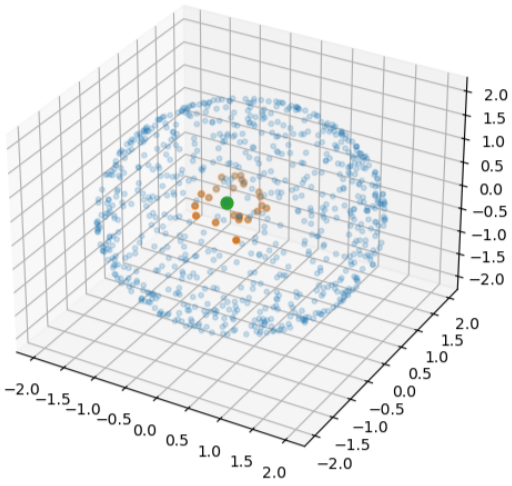
The local dispersion values at that point were:

$$\sigma_1 = 1{,}56, \qquad \sigma_2 = 1{,}50,$$

The similarity between the two values indicates that the surface is locally almost isotropic, with no dominant direction of curvature, which is exactly what we expect when working on a sphere.



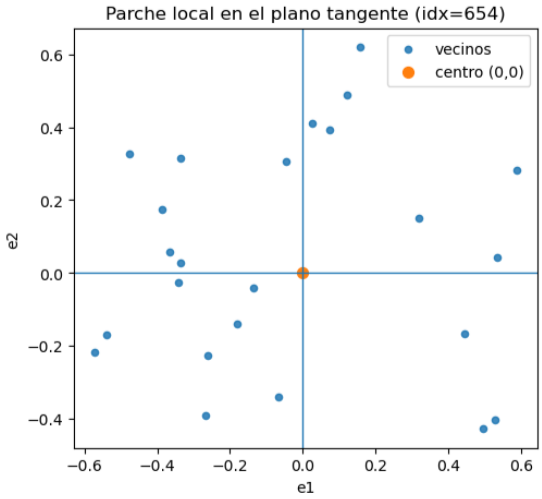
And the nearest neighbors used for the analysis:



4.2.7. Local Coordinates

To estimate the local coordinates, we project that cloud of points —the nearest neighbors— onto the tangent plane we previously computed. Again, we make sure that the patch is centered at the origin, anchored there via the reference point.

As a *Sanity Check*, we inspect and verify that it is indeed at the origin $(0, 0)$; otherwise, we would have a conceptual error.



$$\text{Local variances of the patch (2D)} \mapsto (\sigma_1^2, \sigma_2^2) = (0{,}09947079,\ 0{,}08923594)$$

$$\text{Anisotropy ratio } \mapsto \rho = \frac{\sigma_1^2}{\sigma_2^2} = 1{,}1146942706647378$$

This is exactly what happens on a real sphere. Every point on the sphere "looks" the same.

4.2.8. Local Metric

When estimating the metric, we do so in two steps.

First, we project the nearest neighbors onto the tangent plane and express them in the basis $\{e_1, e_2\}$, which is the basis of the tangent plane. Then, we take the nearest neighbors and express them in terms of the generated coordinates.

Afterwards, we compute a covariance matrix S.

What we are doing is using covariance as an empirical expression of the metric tensor g.

We then normalize by the trace so that:

$$g = (d/\text{trace}(S)) * S$$

And the results obtained were:

$$g \approx \begin{pmatrix} 1{,}03 & -0{,}52 \\ -0{,}52 & 0{,}62 \end{pmatrix}, \qquad \lambda_{\text{máx}} \approx 1{,}03,\ \lambda_{\text{mín}} \approx 0{,}96, \qquad \frac{\lambda_{\text{máx}}}{\lambda_{\text{mín}}} \approx 1{,}07 \approx 1.$$

- The local patch is **almost isotropic**: there is no clearly dominant direction.
- Geodesic distances are well preserved in the embedding.
- The estimated tangent plane is **consistent** with the local geometry.
- The surface is **smooth** in the neighborhood considered.
- The curvature is similar in all directions, as in a **sphere-like surface**.

4.3. Final Results of the Test

Coefficients of the quadratic fit

$$(a, b, c, \alpha, \beta, \gamma) \approx (-0{,}512631,\ -1{,}639\times10^{-4},\ -0{,}5083,\ 2{,}387\times10^{-3},\ 5{,}257\times10^{-2},\ 2{,}208\times10^{-4})$$

$$k_1 \approx -0{,}512637, \qquad k_2 \approx -0{,}508300$$

$$K = k_1 k_2 \approx 0{,}260754, \qquad H = \frac{k_1 + k_2}{2} \approx -0{,}510469, \qquad R = 2K \approx 0{,}521144$$

Quantity	**Theoretical value (sphere $R_s = 2$)**	**Obtained value**	**% Error**
k_1	0,5	0,5126	2,52 %
k_2	0,5	0,5083	1,66 %
K	0,25	0,2608	4,30 %
H	0,5	0,5105	2,10 %
$R = 2K$	0,5	0,5211	4,22 %

The observed accuracy (errors $< 5\,\%$) confirms that the local geometric estimation (recovery of the tangent plane, local coordinates, and quadratic fit) is consistent with the theoretical curvature of a smooth sphere of radius 2.

5. Chapter 5: When the Space Reveals Its Curvature

So now, what is all this actually for?

Several months of research have gone by —and at least a couple of years— since my curiosity and ambition started gravitating toward differential geometry and *machine learning*. Up to this point, what we have built is a mathematical tool capable of estimating, with reasonably good precision, the geometric properties of a smooth, well-behaved manifold.

In essence, that methodology *is* the research. But now we get to the most exciting part: figuring out how this tool can help us gain intuition about real phenomena, those that, in one way or another, influence our daily lives.

And I have to admit it: this part genuinely excites me. I am not an economist or a finance expert, and I do not have a perfectly clear idea of which dataset will be the most revealing or which phenomenon will be "the best" to analyze. It could be *stocks, bonds, market fluctuations, or economic crises*. At some point I also considered studying *quality-of-life indicators* and seeing whether the curvature of those data could offer any insight into the human development index under certain parameters.

In short, I feel like a kid with his first microscope. The model still needs to be refined, generalized, and expanded in future volumes and research projects —which I will undoubtedly pursue as my academic life moves forward— but for now I already have something. Something I can experiment with, modify, and feed with real data.

That is why I think this chapter deserves a title like this:

Revealing the curvature of the data space

Because, although imperfect, this tool already allows us to see the hidden geometry behind real-world phenomena.

5.1. Application to Real Economic Data

For this first application to real data, I think it is a good idea to use a set of variables that can serve as a kind of X-ray of the economic state of the world. The selected data were taken over the period from January 1, 2018 to December 31, 2024, with values recorded on a daily basis.

The analysis was therefore carried out on the following eight variables:

- **BTC-USD:** Bitcoin price in US dollars (represents digital assets and speculative volatility).
- **ETH-USD:** Ethereum price (second-largest cryptocurrency, complements the crypto market).
- **GC=F:** gold (classic safe-haven asset, often inversely correlated with risk).
- **CL=F:** WTI crude oil (proxy for energy and productive activity).
- **DX-Y.NYB:** US dollar index (measures the strength of the dollar against a basket of currencies).
- **ÎXIC:** Nasdaq index (US technology market).
- **ĜSPC:** S&P 500 (broad US equity market).
- **T̂NX:** 10-year US Treasury yield (represents risk-free rate and monetary policy stance).

The aim, at first, is to be able to describe the economic state of the world on a given day. That alone makes this a great data space to characterize, simply because we have good temporal density, which will allow us to analyze curvature and map it to important global events (at least, that is the idea).

The mission now is to determine, using RHEM1, the geometric characterization of this chosen data space.

5.1.1. Data Acquisition and Processing

The data were obtained directly from Yahoo Finance, which very kindly makes all the data they collect available to me without asking for anything in return. Through *yfinance*, the download was carried out simultaneously for all the variables just described. The data were obtained with the following command:

```
data = yf.download(tickers, start="2018-01-01", end="2025-01-01")["Adj Close"]
```

Now, you may call me uncultured, but I am just now finding out that markets close on weekends and holidays... and it would be a real shame if that lack of information messed up the results. That is why it is important to carry out at least a minimal data cleaning. And not only because markets close, but because some of them close and others do not!

Bitcoin trades 24/7, while the S&P 500 is closed on weekends. To handle this, we applied a forward fill and a backward fill:

- `ffill()` ("forward fill"): copies the last valid value forward in time.

- `bfill()` ("backward fill"): fills gaps at the beginning with the next known value.

This preserves temporal continuity without distorting long-term trends. At this point, all the series have the same number of days and there are no empty cells left.

Now... we have another small problem: the magnitudes of these variables. Their price levels are simply not very compatible with each other.

For example, Bitcoin might be at 50,000 US dollars, while oil trades around 75 dollars. To deal with this, we transform prices into *log returns* for each series, which measure relative variation:

$$r_t = \log\left(\frac{P_t}{P_{t-1}}\right)$$

In other words, this homogenizes the data so that we can compare asset performance at a given moment in time.

There is another aspect we must consider: asset volatility. While Bitcoin can have large swings in a single day, gold or the US dollar typically do not. For this reason, it is necessary to standardize; otherwise, the most volatile assets would dominate any analysis we perform later on.

$$Z_{t,i} = \frac{r_{t,i} - \mu_i}{\sigma_i}$$

where:

- μ_i: mean return of asset i.
- σ_i: its standard deviation.

The result is that rows represent days and columns represent the variables of interest, now standardized and normalized, yielding a data space $X \in \mathbb{R}^{2556\times 8}$ that we can analyze with RHEM1.

This matters, and it makes for a good dataset, because if we analyze consecutive days, we can talk about a local economic structure; whereas if we look at dates that are far apart, we can see macroeconomic effects, and the curvature of this data space may tell us something about how the market changes over time.

5.1.2. Embedding via ISOMAP

Now we can finally feed the beast. I remember joking with my friend Andrés —to whom I am very grateful for our conversations and his encouragement— and telling him that RHEM is actually a chained beast in my basement, and that I was going to feed it data one by one, stoking the beast's desperation. This, more than anything, reflects my mental stability after the Electrodynamics course. In any case, this is clearly impossible and makes no technical or ethical sense whatsoever.

But the point of that brief anecdote is that we now have the data ready and, at last, we can perform the analysis we have been looking forward to.

The first step is, naturally, the first part of the algorithm, which corresponds to a classic ISOMAP.

We start with the KNN graph, for which we selected $k = 40$ nearest neighbors for each point in the space. That is, each day x_i is assigned its 40 most similar days, using Euclidean distance as the similarity criterion.

This represents a grouping of similar conditions between points; in other words, we assign each day its 40 most similar days and build an undirected graph that serves as a map between these neighborhoods.

We then compute geodesics, using the connection graph obtained via KNN to define the minimum path length between any two points in the space.

If the geodesic distances were identical to the Euclidean distances between points, then we could say that the data space is flat.

From this, we obtain a matrix containing the computed distances, and with MDS, the next step, we assign coordinates in a lower-dimensional space.

It is important to emphasize that the embedding does not have to be in 2D or 3D; it can be in any lower dimension, with the goal of obtaining a reduced-dimensional representation that preserves, as much as possible, the geometry of the space.

It is at this point that we make a significant sacrifice: we are sacrificing information in order to recover interpretability.

5.1.3. Reconstruction of the Local Metric and Curvatures

Proceeding with the estimation of the local metric, the first step in the method is the computation of principal components at a local level (LPCA). For this we again rely on the k nearest neighbors, the same ones used in KNN, to identify the principal directions of variation in the data.

We do so via the covariance matrix, whose eigenvectors become the basis of the tangent plane: a space where we recover Euclidean behavior, or "flatness." This is the highly valuable space in differential geometry where we can perform differential calculus.

Once the tangent plane is well defined, we project the nearest neighbors onto this plane and compute their heights.

We project the neighbors onto the tangent plane and onto the normal directions:

$$U = (X_{\text{neighbors}} - x_i)E_i \quad \text{(coordinates in the tangent plane)}$$

$$H = (X_{\text{neighbors}} - x_i)F_i \quad \text{(heights along the normal directions)}$$

Thus we obtain two sets of coordinates:

$$U \in \mathbb{R}^{m\times 2} : \text{ describes the shape of the neighborhood "seen from above".}$$

$H \in \mathbb{R}^{m\times 6}$: tells us how much the manifold "rises or sinks" along each normal direction.

Intuition:

One way to think about it is to imagine a sheet curved in space. If we define a tangent plane at a point, the heights $h(u, v)$ tell us how far the sheet pulls away from that plane in each direction.

Then, for each column of H, that is, for each normal component, we fit a quadratic surface. The coefficients of that fit describe the local curvature in that normal direction.

From each Hessian we obtained its eigenvalues —the principal curvatures—:

$$k_1, k_2 = \text{eig}(H^{(\alpha)})$$

We then computed the geometric invariants:

Quantity	Formula	Meaning
K (Gaussian)	$K = \det(H^{(\alpha)})$	Local intrinsic curvature (depends only on the manifold, not on the *embedding*).
H (mean)	$H = \frac{1}{2}\,\mathrm{tr}(H^{(\alpha)})$	Extrinsic measure of how much the surface bends.
R (scalar)	$R = 2K$	2D version of the Ricci scalar, useful for comparing *patches*.

Tabla 2: Local geometric invariants obtained from each Hessian.

Afterwards, since there are 6 normal directions, we aggregate their contributions:

$$K_{\text{intr}} = \sum_{\alpha=1}^{6} \det(H^{(\alpha)}), \qquad |\mathbf{H}| = \|(H^{(1)}, \ldots, H^{(6)})\|$$

What they represent:

- K_{intr}: measures how the data space curves "from within".
  - Positive → sphere-like (coherent regimes, synchronized behavior of variables).
  - Negative → saddle-like (diverging directions, breakdown of linearity).
- $|\mathbf{H}|$: measures the extrinsic bending, that is, how far the data space departs from the plane that contains it.

6. Results and Conclusions

6.0.1. Results obtained

On data import:

Shape of X	(2556, 8) $\Rightarrow$ $T = 2556$ days, $d = 8$ variables
Dates	2018-01-02 $\rightarrow$ 2024-12-31
Variables	['BTC-USD', 'CL=F', 'DX-Y.NYB', 'ETH-USD', 'GC=F', 'ĜSPC', 'ÎXIC', 'T̂NX']

First 5 points:

```
[[ 2.57486250e+00 -2.40831544e-03 -1.89678693e-02  2.94087996e+00
  -3.46864694e-02 -2.94868373e-02 -3.19291668e-02 -8.56456819e-03]
 [ 3.85590760e-01  7.35670640e-01  9.62147319e-01  1.84075011e+00
   2.08256791e-01  5.86789484e-01 -2.68188336e-01 -3.22033586e-01]
 [ 7.03908160e-01  2.17225275e-01 -1.00008306e+00  3.96825918e-01
   2.75162708e-01  8.59339621e-01 -3.87784712e-02 -1.77047802e-02]
 [ 3.08949525e-02 -3.35293732e-01  2.97782110e-01  3.58586002e-01
   5.24492617e-02  6.47662226e-01  3.22938989e-01  3.22030511e-01]
 [ 3.15389649e-01 -2.40831544e-03 -1.89678693e-02  9.29768556e-01
  -3.46864694e-02 -2.94868373e-02 -3.19291668e-02 -8.56456819e-03]]
```

Tabla 3: Sanity check of the input matrix $X \in \mathbb{R}^{2556\times 8}$.

On KNN:

Description	**Value**
Graph with	2556 nodes and 77447 edges
Nodes	2556
Edges	77447

Tabla 4: Summary of the k-nearest neighbors graph: connectivity structure of the *data space*.

Parameter	Value
knn_idx	(2556, 40)
knn_dist	(2556, 40)
sorted distances	True
median distance to the k-th neighbor	1.371923295586892

Tabla 5: Sanity check of the k-nearest neighbors graph: dimensions and distance metrics.

Check	Result
Geodesic distance matrix	(2556, 2556)
Symmetric?	True
Zero diagonal?	True
Distance between node A and B ($D[15, 2000]$)	2.358752910491206

Tabla 6: Sanity check of the geodesic distance matrix.

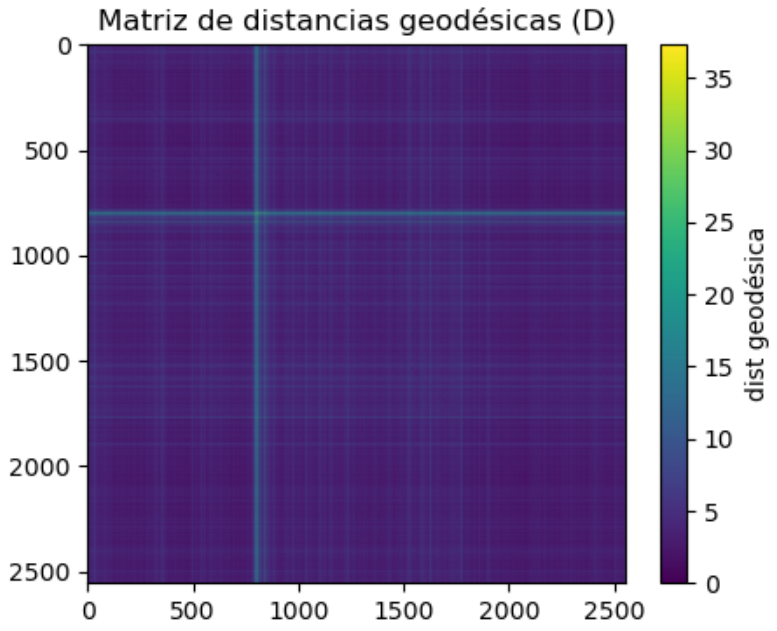


Figura 2: Heatmap of the geodesic distance matrix.

On MDS:

Check	Result
Mean of each column (Y_5 centered)	[-6.10e-17, 2.51e-16, -2.24e-16, -9.57e-16, -3.22e-16]
Original geodesic distance (0,1)	3.564937681747706
Embedded distance (0,1)	2.460981570402122

Tabla 7: Sanity check of ISOMAP embedding: centering and partial preservation of distances.

On local geometry:

Variable	Shape
E	(8, 2)
F	(8, 6)
U	(40, 2)
H_s	(40, 6)

Tabla 8: Sanity check of the local patch: dimensions of projections and heights in the tangent plane.

- E: orthonormal basis of the tangent plane, obtained via local PCA (dimension $(D, 2)$).
- F: normal vectors orthogonal to the tangent plane (dimension (D, q)).
- U: coordinates of neighbors projected onto the tangent plane (dimension $(m, 2)$).
- H_s: heights of neighbors along the normal directions (dimension (m, q)).

Check	Result
g (estimated local metric)	$\begin{bmatrix} 1{,}13811827 & 0{,}19897809 \\ 0{,}19897809 & 0{,}86188173 \end{bmatrix}$
trace(g)	2.0
eigvals(g)	[1.24221672, 0.75778328]
Largest/smallest ratio	1.639277022432253

Tabla 9: Sanity check of the estimated local metric on the patch.

(k_1, k_2)	$\|H\|$	K_{intr}
$(-0{,}472,\ 0{,}966)$	1.9455	2.7009
$(-1{,}962,\ -0{,}894)$		
$(-1{,}374,\ -0{,}909)$		

Tabla 10: Sanity check of the quadratic fit per normal direction: curvatures (first 3 pairs).

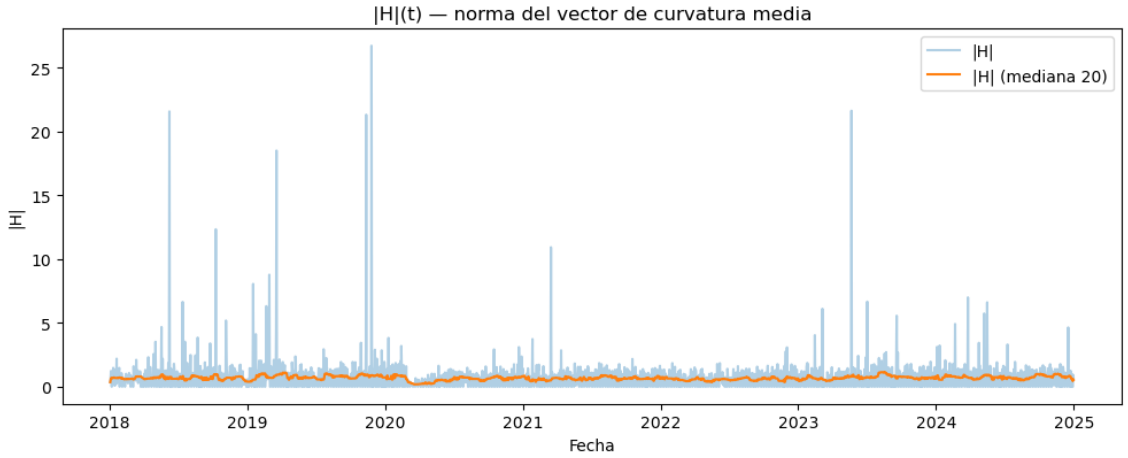


Figura 3: Temporal evolution of the norm of the mean curvature vector $|H|(t)$.

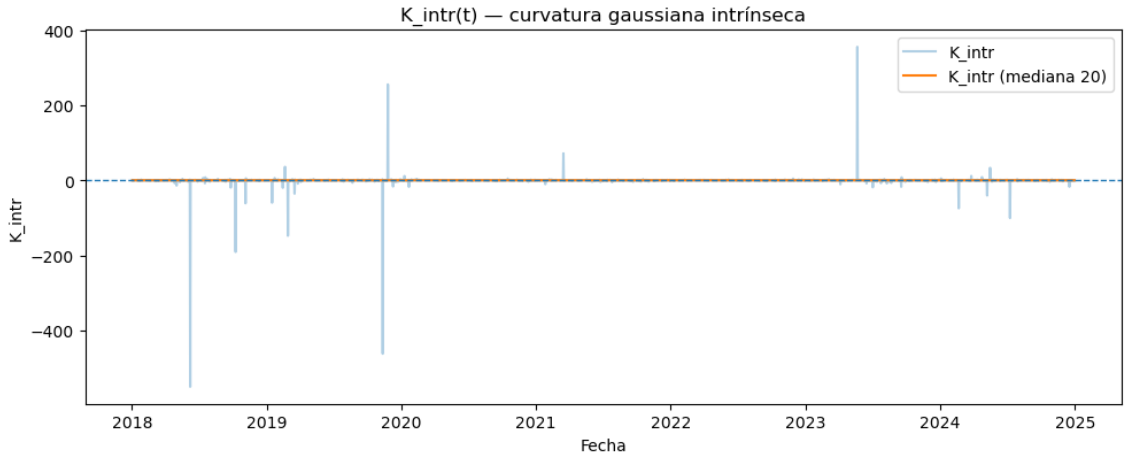


Figura 4: Temporal evolution of the intrinsic Gaussian curvature $K_{\text{intr}}(t)$.

6.1. Discussion of the results

6.1.1. Discussion of the geometric results

We can see that after data preprocessing we obtain an 8-dimensional space, given by the assets we chose to analyze. After normalization and standardization, we end up with a space of 2556 points in $\mathbb{R}^8$. This is the *data space* on which the global analysis via ISOMAP is carried out, followed by local geometric characterization, thus completing this first iteration of RHEM1.

The graph generated via the 40 nearest neighbors seems to be a good initial approximation, although it would be wise to run experiments with different values for k, ideally between 30 and 60 neighbors, to see how the estimates of the local patches behave.

We verify that KNN is implemented correctly through the shape of the graph and its high connectivity after handling possible disconnections.

Regarding geodesic distances, there appears to be a high density of points, except for some isolated data points (*outliers*), as seen in the heatmap. Nonetheless, the matrix is symmetric with a zero diagonal, in agreement with the theoretical foundation. At the end of the day, who am I to judge if the data are too compact in the space...

The problem, after performing dimensionality reduction via MDS and going from 8 to 5 dimensions, is that the error of the new embedded distance is 31 % with respect to the original geodesic distance. At this stage, it becomes difficult to move forward in future iterations of the model with the MDS sub-process, since it has not provided particularly valuable information about the global geometry of the *data space*.

However, the very fact that this difference exists between the original Euclidean distance and the geodesic distance is a first indicator that **curvature does exist**, and that it must be taken into account when performing data analysis.

On the other hand, the fact that the trace of all local metrics is 2 tells us that the normalization was done correctly. This suggests that the high volatility of assets like Bitcoin or Ethereum is not being directly overemphasized: the estimated metric is correctly measuring the deformation of the *data space* locally.

The range of eigenvalues (about 1.24 and 0.75) indicates that the *data space* is not perfectly isotropic and that, indeed, there are directions in the tangent plane that are 63 % more favorable than others, as we can see in the anisotropy ratio. This is by no means a bad thing; on the contrary, it is desirable given the nature of the data, which will hardly be perfectly symmetric or "flat."

The fact that it is somewhat elliptical means there are stronger correlation directions

between certain variables (for example, crypto and tech stocks) than between others (such as gold or the dollar).

The presence, in some cases, of principal curvatures with mixed signs tells us that the geometry is saddle-like. Geometrically, a saddle-type surface appears when the principal directions of variation separate: one grows while the other decreases; this translates into *tension* in the *data space*.

As for the mean curvature H, if we interpret it as the "bending" of the dataset inside the *data space*, we can think of it as extrinsic perturbations. This could be translated into "crises" or "anomalies" in the phenomenon under study. In this case, the peaks seen in the plot can be interpreted as the *data space* entering a phase of global distortion, as we can see in the time interval between 2018 and 2020. When the mean curvature drops and stabilizes (2021–2025), the *manifold* becomes flatter again, meaning that the data are distributed in a more coherent structure, without abrupt changes in correlations.

6.1.2. Economic implications

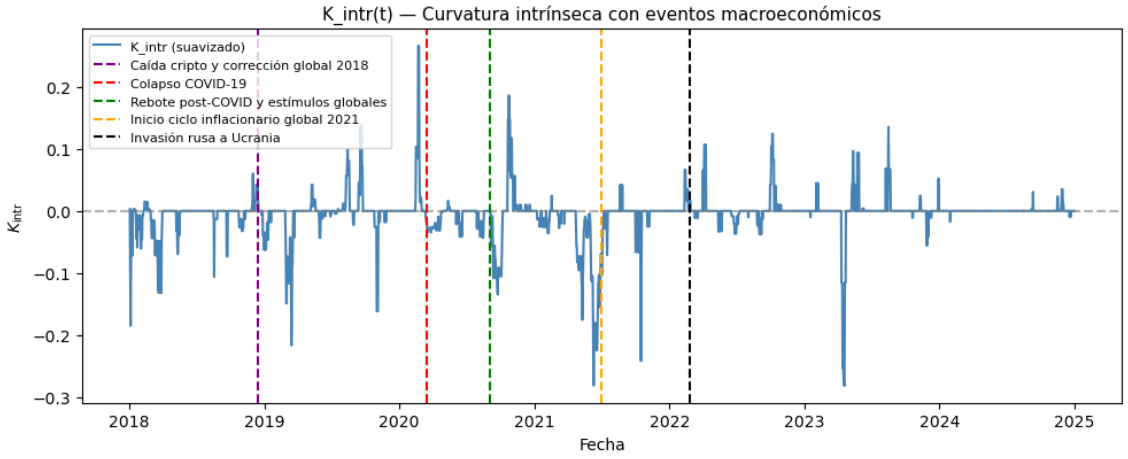
And now, my favorite result. Something I genuinely consider *cool*. I am not sure if I am "allowed" to write that in a thesis, but at this point I honestly do not care much.

Look what happens if we place reference lines for important events on the intrinsic curvature plot.

In some cases, significant economic or geopolitical phenomena can be mapped directly onto the plot, with a slight temporal lag (of a few days or weeks), since the impact on markets is not immediate. Even so, we can observe that for a number of globally relevant events, the curvature of the *data space* of these assets tends to destabilize and change sign, which suggests that the global economic space "feels" transitions earlier than traditional indicators.

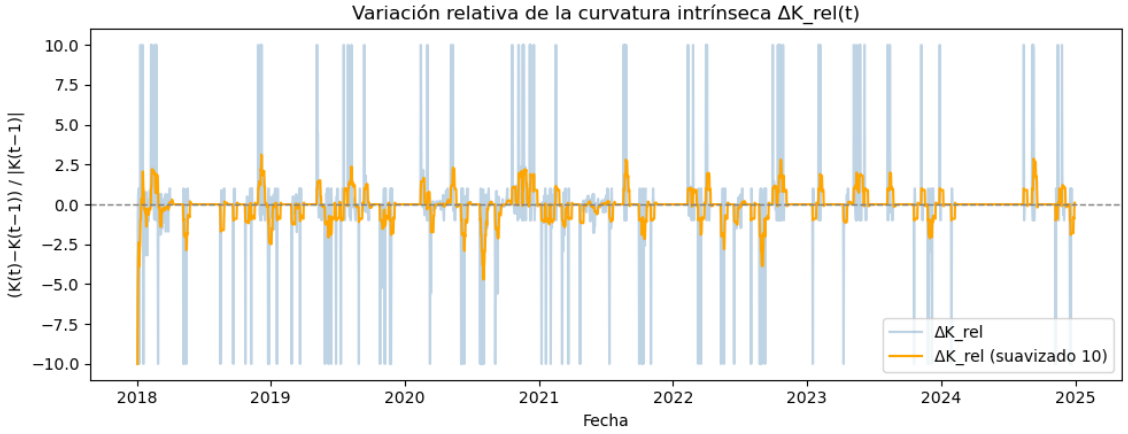
- **2018 — Crypto crash and global correction.**
- **March 2020 — COVID-19 market collapse.**
- **July 2021 — Global inflation surge.**
- **February 2022 — Russian invasion of Ukraine.**

To be honest, I got these events from Google —I am neither a historian nor a sociologist—, and frankly, Engineering Physics demands too much of my attention for me to stay fully on top of world events. Even so, I think this is genuinely interesting: seeing how the *data space* seems to react to major global events.



Conclusion: Taken together, the sign changes in $K_{\text{intr}}(t)$ coincide with **structural transitions of the global economic system**:

We can now analyze the relative variation of intrinsic curvature. This allows us to see whether the peaks we observe are truly peaks. A peak is only a peak if we place it in the context of its neighboring values.



- The Y-axis shows the quantity

$$\Delta K_{\text{rel}}(t) = \frac{K(t) - K(t-1)}{|K(t-1)|}$$

  That is, **how abruptly the curvature of the economic space changes** from one day to the next.

- The X-axis represents time (2018–2025).
- The light blue line is the raw daily variation, and the orange line is a 10-day moving average that captures local trends.

In geometric terms, we are measuring **how the shape of the economic *data space* changes**, not its absolute value.

- When $\Delta K_{\text{rel}} > 0$: the *data space curves more*, which implies stronger coupling or interaction between economic variables (values that move in a more coordinated way).
- When $\Delta K_{\text{rel}} < 0$: the *data space flattens*, reflecting loss of correlation or dispersion (values that decouple or enter a more chaotic regime).
- Values close to zero $\rightarrow$ **geometric stability**, where the interdependencies between variables do not change significantly.

Each peak we see represents a *shock* in the economic state of the world at a given time, a phenomenon that occurs and causes the *data space* to abruptly change shape at that moment. We can interpret this as tension or relaxation of a fabric that underlies this space.

6.1.3. Limits and scope of the geometric approach

The geometric approach proposed in RHEM1 shows that it is possible to characterize structural properties of data beyond traditional statistical methods. Curvature analysis allows us to describe how relationships between variables evolve nonlinearly over time, revealing patterns that would not be detectable through conventional correlations or regressions. Differential geometry applied to financial data suggests that economic phenomena can be interpreted as deformations of the *data space*: crises, periods of synchrony, and recovery phases appear as changes in the curvature of the underlying *manifold*. This suggests a new conceptual framework for the study of complex systems, where the shape of the space contains as much information as the values that inhabit it.

However, the method presents important limitations. First, it relies heavily on the quality and homogeneity of the data: any noise or discontinuity can be amplified when estimating the local metric. Moreover, the results are sensitive to the number of neighbors k and to the chosen embedding dimension, so different configurations can alter the magnitude of the curvatures obtained. The computation of local curvatures also assumes that the *manifold* is sufficiently smooth and dense, something that does not always hold for real economic data. Even with these constraints, the geometric approach opens up a promising path for studying the internal structure of data in a richer and more visual way, offering a new perspective on phenomena that, until now, were described purely from a statistical viewpoint.

6.2. General Discussion

6.3. Review of objectives and achievements

Depending on the day and the time, I can sometimes be a bit hard on myself. Today I am happy and satisfied. The objectives I set —both the romantic and the technical ones— were achieved.

I realized that I love what I do. I love what I study. I am very happy learning and studying... and now applying physics. I was able to apply knowledge from my favorite courses throughout my degree, and to learn even more from them.

In terms of objectives, it was achieved.

First, I managed to reconstruct the local metric of a discontinuous *data space*, and not only that: I formally defined what this *data space* is in a way that is consistent with the work. I discussed the implications of curvature in this *space*, and it is clear that, even though this is far from the most precise result in the history of research projects, the existence of curvature —in the mathematical sense— in these *data spaces* is unquestionable.

It is also clear that differential geometry and physics still have a lot to contribute to the study of economic and socioeconomic phenomena, and to machine learning models in general.

6.4. Limitations of the study

Some clear limitations were the lack of directionality when it came to applying the model to a real-world case. This caused identity conflicts within the work itself, almost as if it lost its initial essence by not having a natural and direct application.

From a technical standpoint, there are also limitations inherent to the model. The problem of generalizing to higher dimensions and the applicability of MDS call for a new iteration of the model, with more geometric quantities to characterize, allowing for a richer geometric analysis.

Another limitation, of course, is the data. The model depends heavily on the data it is fed. The data need to be dense enough to approximate continuity in the *manifold* so the analysis can be performed. This is not a perfect model, and it is certainly not a model that will work for every *data space*.

6.5. Conclusions and Future Work

- Differential geometry is a mathematical tool worthy of consideration for the analysis of high-dimensional data.
- Differential geometry can provide new perspectives beyond dimensionality reduction alone.
- RHEM1 is a model that, at first pass, is able to estimate the local metric of an n-dimensional *data space* and characterize this space through canonical curvatures.
- The results show that the intrinsic curvature of an economic *data space* is a consequence of high-impact global events, and that its peaks can be interpreted as macroeconomic "*shocks*".

6.6. Final reflection

This has been one of the most rewarding experiences of my life. A dream. I have spent almost three years thinking about this, and I finally have something to show for it. Yes, it is imperfect —and what isn't?—, but it is something I am very proud of, and I am sure that the Henry from five years ago, the one who started Engineering Physics, would also be very proud.

The experience of combining different areas of knowledge —such as *Machine Learning*, physics, differential geometry, and economics— is something that enriches a person deeply. It is about exploring with creativity, curiosity, and imagination. It is about taking a wild idea and doing everything in your power to make it happen.

I have grown a lot as a person, from the emotional to the academic. I have also grown professionally, and I had to learn tools that will be useful in the immediate future. I am very glad to have done a technical project that demanded the development of these skills and new knowledge.

I definitely take with me the experience of solitude: the idea of being at peace with it and accepting it as a companion through long nights of study, coding, and frustration, but also as a loyal companion in the joy of partial victories that appear along the way.

Though it is not all solitude... I also carry with me the encouragement of all those who supported me in this process, and I realize that an idea sounds crazier in one's head than when it is shared with others.

This work provides a new perspective on a general problem that appears in many

fields. It advocates for data whose morphology is usually ignored when using conventional analysis methods. It is a piece of work that aims to provide tools that might, in the future, help uncover hidden patterns and unknown clues about relevant phenomena that impact the lives of billions of people.

6.7. Future directions

Based on the results obtained, several lines of extension and methodological refinement can be identified:

- **Topological analysis:** Integrate tools from algebraic topology (such as persistent homology or Mapper) to characterize the global connectivity of the *data space*.
- **Application to dynamic databases:** Extend the pipeline to time series or evolving data streams, tracking the deformation of the *data space* over time.
- **Real-time curvature modeling:** Implement an optimized architecture for incremental curvature computation as new data enter the system.
- **Geometric reinterpretation:** Explore how the results vary when projecting the same *dataset* into spaces with different metrics or structures (Riemannian, hyperbolic, etc.).
- **Implementation in a stable and replicable repository.**

6.8. Closing

"Every science is a form of cartography: we draw the map of the universe and try to describe and reconstruct it with the instruments we have; sometimes we forget that we ourselves are also part of it."

7. Appendix A - Jupyter Lab PoC

In [1]:
```
import numpy as np
print("NumPy:", np.__version__)
```

```
NumPy: 2.3.2
```

In [2]:
```
import sys, os
from pathlib import Path

def attach_src_and_cd():
    here = Path.cwd()
    # Busca 'src' en la carpeta actual y en todos los padres
    for base in [here] + list(here.parents):
        cand = base / "src"
        if cand.is_dir():
            if str(cand) not in sys.path:
                sys.path.append(str(cand))
            # Pon el directorio de trabajo en la raíz del repo (donde vive 'src')
            os.chdir(base)
            print("OK: usando src en", cand)
            print("CWD ahora es:", Path.cwd())
            return True
    print("ERROR: no encontré carpeta 'src' empezando desde", here)
    return False

attach_src_and_cd()
```

```
OK: usando src en C:\Users\henry\economia-no-euclideana\src
CWD ahora es: C:\Users\henry\economia-no-euclideana
```

Out[2]: True

Paso 1: Generamos los puntos sobre una superficie conocida

In [3]:
```
from src.data.synth import generate_sphere_points2

N = 800

X = generate_sphere_points2(N)
print("Shape de X:", X.shape)
print("Primeros 5 puntos:\n", X[:5])
```

```
Shape de X: (800, 3)
Primeros 5 puntos:
 [[ 0.69698348 -1.6142175   0.953161  ]
 [-0.40237888 -1.2538319  -1.50532282]
 [-0.81940403  1.03086882 -1.50528619]
 [ 1.22386981 -1.51348679 -0.45989176]
 [ 1.77480338  0.05127971 -0.92056686]]
```

Sanity Check

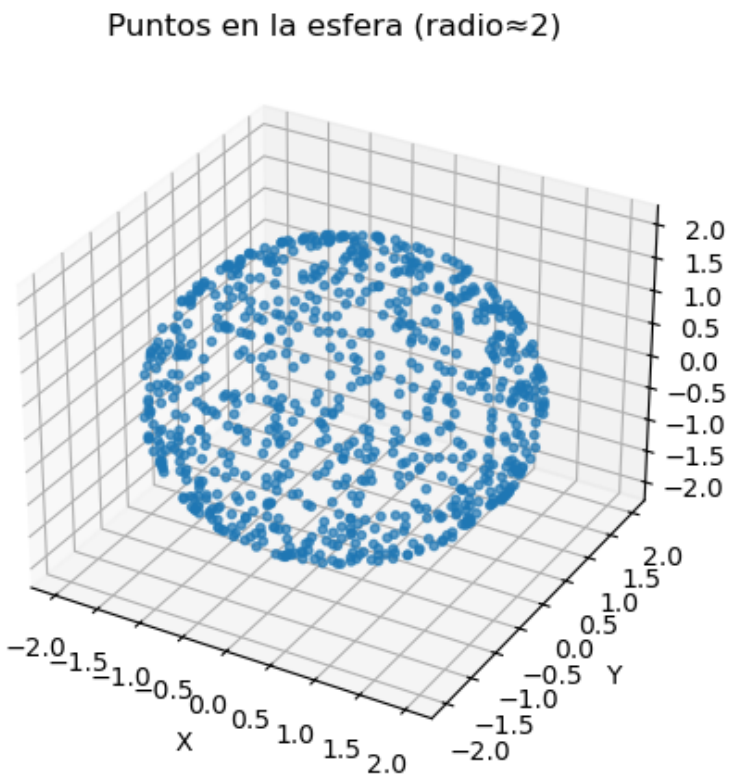
In [4]:
```
import numpy as np

r = np.linalg.norm(X, axis=1)
print("radio medio:", r.mean(), "  desvío:", r.std(), "  min/max:", r.min(), r.max(
```

```
radio medio: 2.0   desvío: 1.0934428697813142e-16   min/max: 1.9999999999999996 2.00
00000000000004
```

Vizuals

Ya tenemos nuestra esfera. Vamos a verla!

In [82]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(5,5))
ax = fig.add_subplot(111, projection="3d")
ax.scatter(X[:,0], X[:,1], X[:,2], s=10, alpha=0.7)

ax.set_xlabel("X"); ax.set_ylabel("Y"); ax.set_zlabel("Z")
ax.set_title("Puntos en la esfera (radio≈2)")
plt.show()
```



KNN

Conectaremos cada punto con su K vecinos mas cercanos

In [8]:
```
import numpy as np
from sklearn.neighbors import NearestNeighbors
from scipy.sparse import csr_matrix, csgraph
import matplotlib.pyplot as plt
```

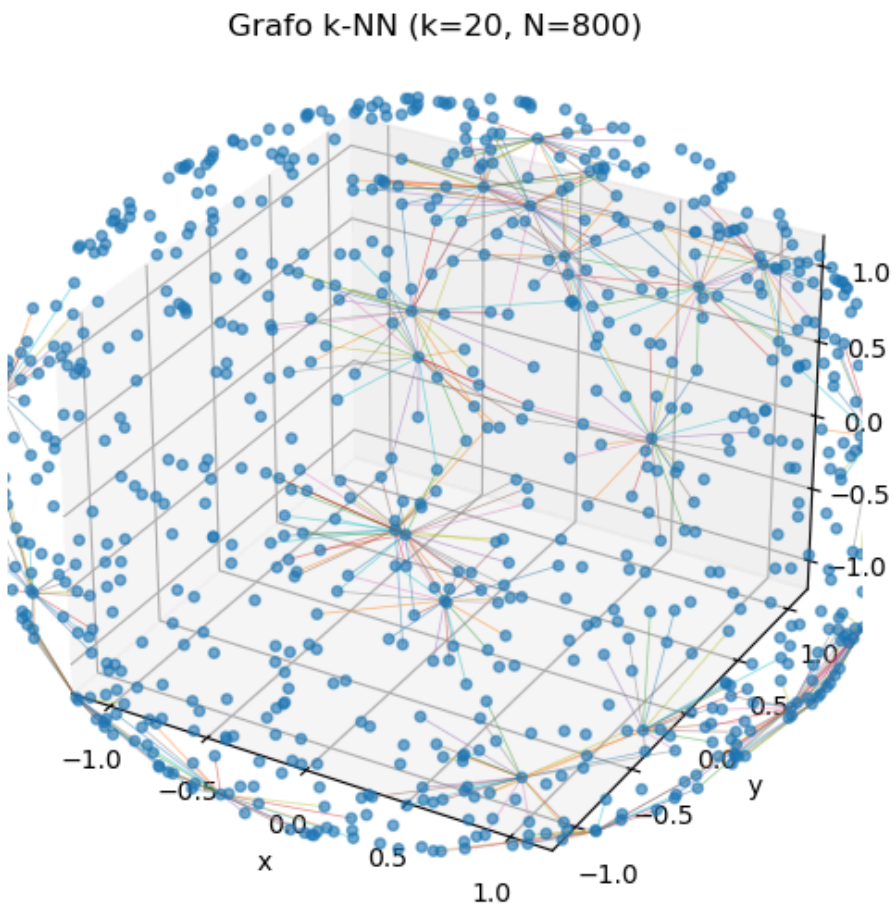
In [9]:
```
from src.graph.knn import build_knn_graph

k = 20
G = build_knn_graph(X, k)

# 3) Revisa que funcione
print(G)
print("Nodos:", G.number_of_nodes())
print("Aristas:", G.number_of_edges())
```

```
Graph with 800 nodes and 8687 edges
Nodos: 800
Aristas: 8687
```

Si: k es muy bajo → analizas la variedad a escalas muy finas. k es muy alto → analizas la variedad a escalas más gruesas (más “promediadas”).

Vizual KNN

In [83]:
```
from src.viz.plots import plot_knn_graph_3d

plot_knn_graph_3d(
    X,
    G,
    max_edges=500,                  # aumenta/reduce según saturación
    outpath="artifacts/knn_3d.png",
    title=f"Grafo k-NN (k={k}, N={N})"
)
```



Geodesics

In [11]:
```
from __future__ import annotations
import numpy as np
import networkx as nx
```

In [12]:
```
from src.geodesic.shortest_paths import compute_geodesic_distances, handle_disconne
```

In [13]:
```
# Calculamos distancias geodesicas
```

In [14]:
```
D = compute_geodesic_distances(G)
print("Matriz de distancias geodésicas:", D.shape)
```

```
Matriz de distancias geodésicas: (800, 800)
```

In [15]:
```
from src.geodesic.shortest_path_FAST import compute_geodesic_distances_fast

# Dijkstra (rápido)
%time D= compute_geodesic_distances_fast(G, method="D")
```

```
CPU times: total: 406 ms
Wall time: 409 ms
```

In [17]:
```
D = handle_disconnections(D, strategy="big_value")
```

In [18]:
```
# sanity checks
```

In [19]:
```
print("¿Simétrica?", is_symmetric(D))
print("¿Diagonal cero?", has_zero_diagonal(D))
```

```
¿Simétrica? True
¿Diagonal cero? True
```

In [20]:
```
# Ejemplo
```

In [21]:
```
print("Distancia entre nodo A y B:", D[19,415])
```

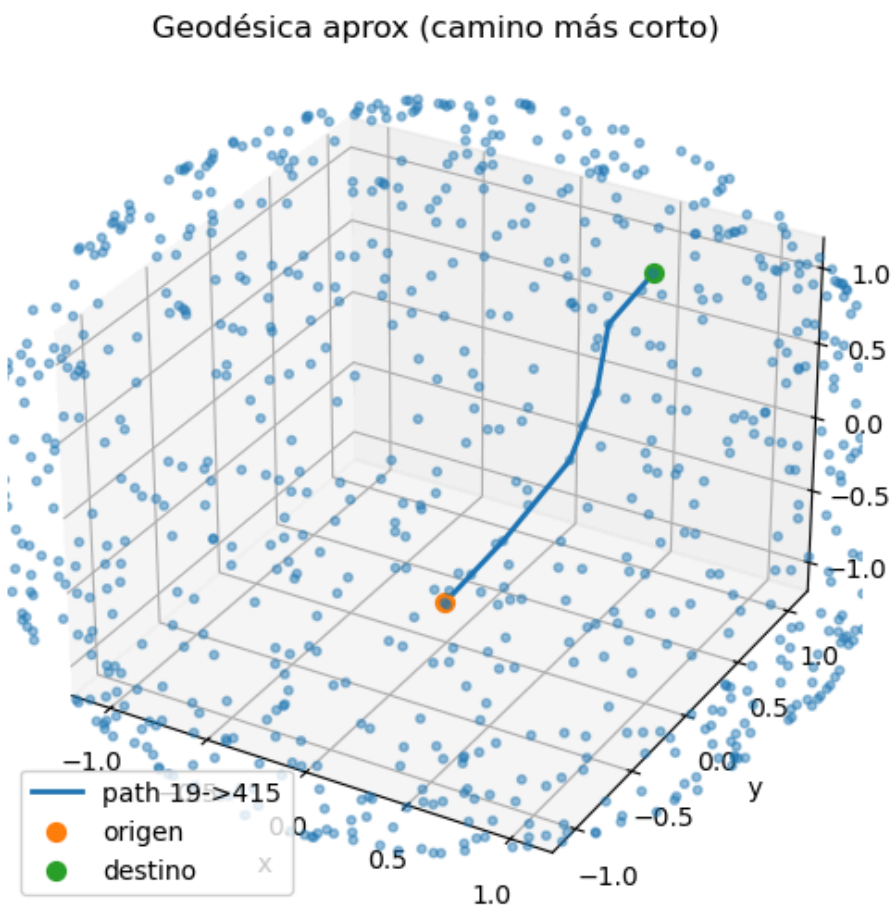
```
Distancia entre nodo A y B: 2.3282830191753905
```

In [22]:
```
# Viz Geodesica
```

In [23]:
```
from src.viz.plots import plot_geodesic_path_3d
```

In [24]:
```
src, dst = 19, 415
```

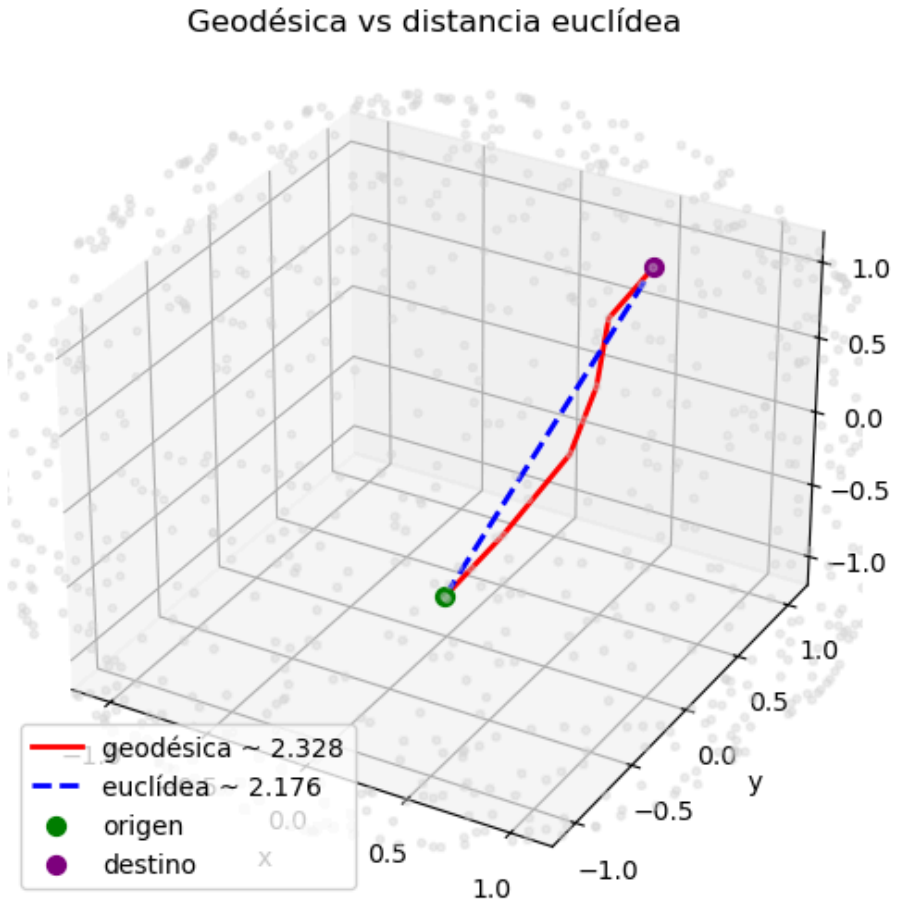
In [25]:
```
plot_geodesic_path_3d(
    X, G, src, dst,
    outpath="artifacts/geodesic_path.png"
)
```



In [26]:
```
# contrastemos!
```

In [27]:
```
from src.viz.plots import plot_geodesic_vs_euclidean
```

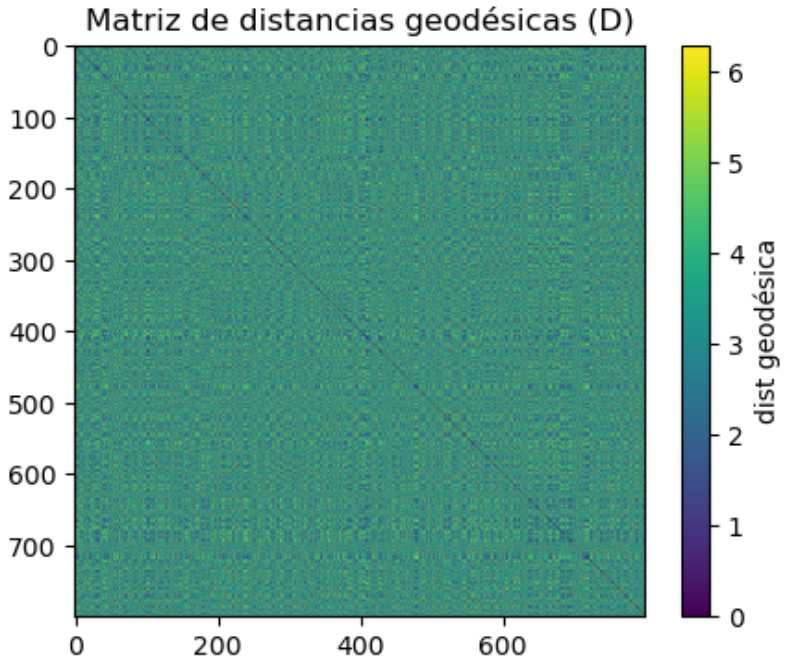
In [28]:
```
plot_geodesic_vs_euclidean(X, G, src, dst, outpath="artifacts/geodesic_vs_euclidean
```



In [29]:
```
#Heatmap
```

In [30]:
```
from src.viz.plots import plot_geodesic_distance_matrix
```

In [31]:
```
plot_geodesic_distance_matrix(
    D
)
```



In [32]:
```
#MDS
```

In [33]:
```
from src.embed.isomap import classical_mds
```

In [34]:
```
#Embeding en 2D
```

In [35]:
```
Y2 = classical_mds(D, n_components=2)
print("Shape embedding 2D:", Y2.shape)
```

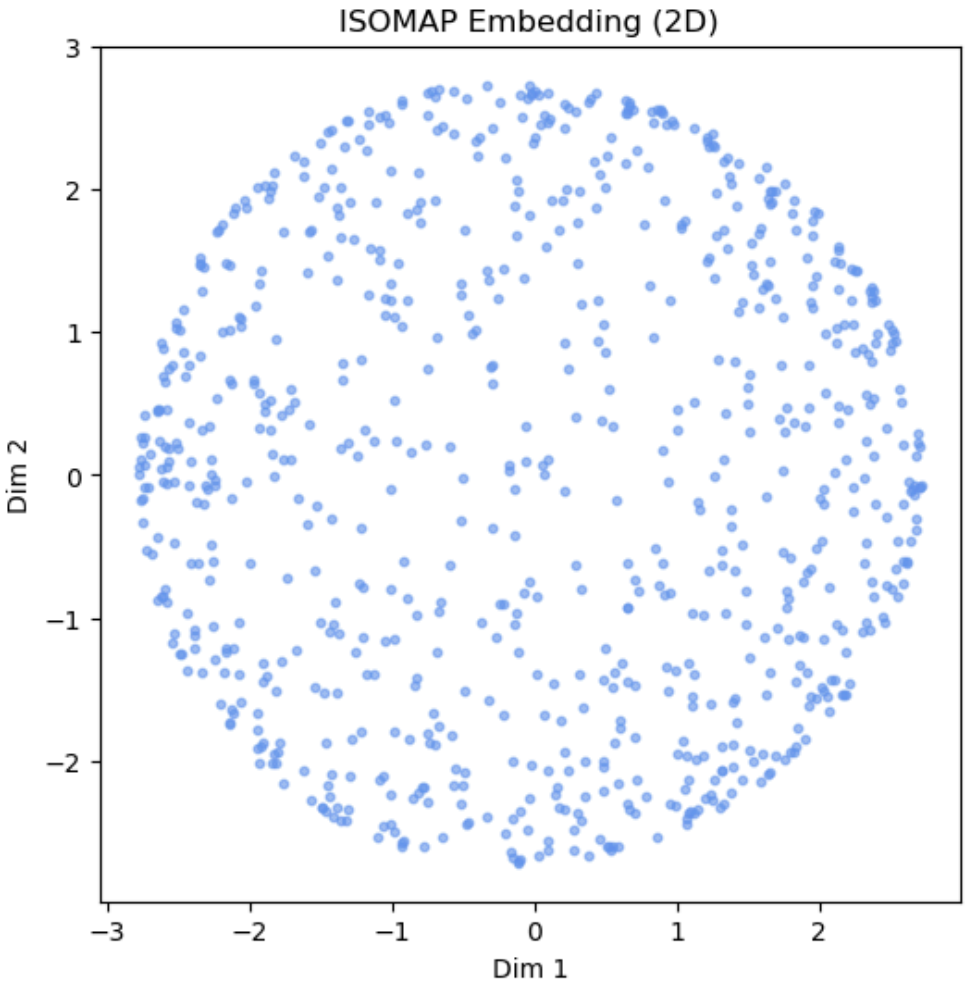
```
Shape embedding 2D: (800, 2)
```

In [37]:
```
# Embedding en 3D
Y3 = classical_mds(D, n_components=3)
print("Shape embedding 3D:", Y3.shape)
```

```
Shape embedding 3D: (800, 3)
```

In [38]:
```
# Viz 2D
```

In [39]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(6,6))
plt.scatter(Y2[:,0], Y2[:,1], s=10, alpha=0.6, c="cornflowerblue")
plt.title("ISOMAP Embedding (2D)")
plt.xlabel("Dim 1"); plt.ylabel("Dim 2")
plt.show()
```



In [41]:
```
#Sanity Check
```

In [42]:
```
# Checar media ~ 0 (centrado)
print("Media de cada columna:", Y2.mean(axis=0))

# Ver primeras distancias preservadas
import numpy as np
print("Dist original geodesica (19,419):", D[19,419])
print("Dist embebida (19,419):", np.linalg.norm(Y2[19]-Y2[419]))
```

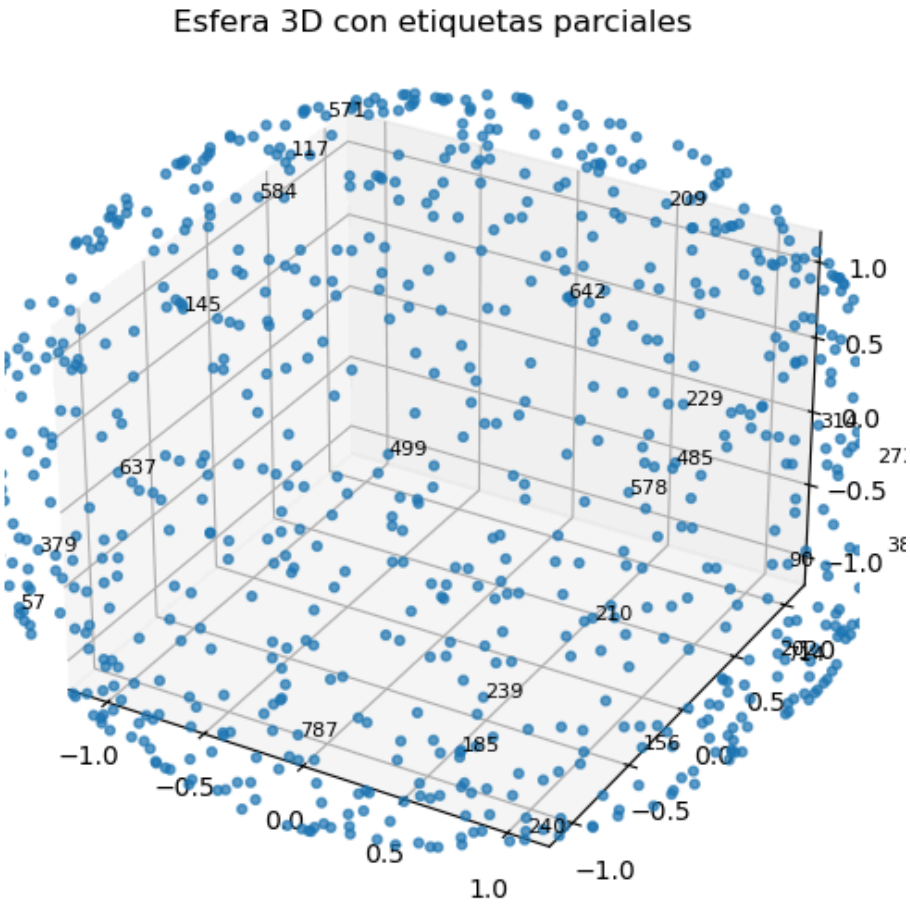
```
Media de cada columna: [-4.27435864e-17  4.16333634e-17]
Dist original geodesica (19,419): 3.167106256781297
Dist embebida (19,419): 0.9426016953619457
```

In [44]:
```
# Etiquetemos los puntos para ver en donde se ubican
```

In [45]:
```
n = X.shape[0]
ids = np.arange(n)  # 0..n-1
```

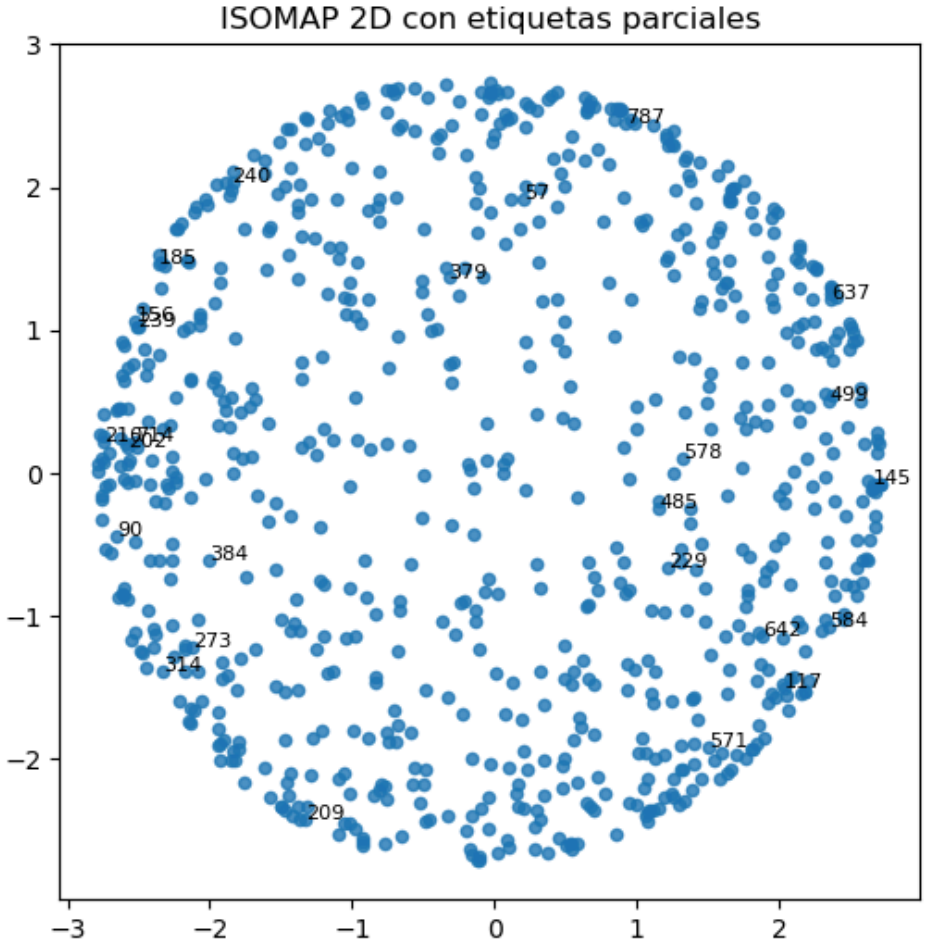
In [46]:
```
# Ejemplo: 25 al azar (reproducible)
np.random.seed(123)
idx_to_label = np.random.choice(n, size=25, replace=False)
```

In [47]:
```
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(6,6))
ax = fig.add_subplot(111, projection="3d")

ax.scatter(X[:,0], X[:,1], X[:,2], s=12, alpha=0.7)

# Etiquetas (solo la muestra)
for i in idx_to_label:
    ax.text(X[i,0], X[i,1], X[i,2], str(i), fontsize=8)

for a in (ax.set_xlim, ax.set_ylim, ax.set_zlim):
    a([-1.2, 1.2])

ax.set_title("Esfera 3D con etiquetas parciales")
plt.savefig("artifacts/fig_sphere_labels.png", dpi=300, bbox_inches="tight")
plt.show()
```



In [48]:
```
plt.figure(figsize=(6,6))
plt.scatter(Y2[:,0], Y2[:,1], s=20, alpha=0.8)

for i in idx_to_label:
    plt.text(Y2[i,0], Y2[i,1], str(i), fontsize=8)

plt.axis("equal")
```

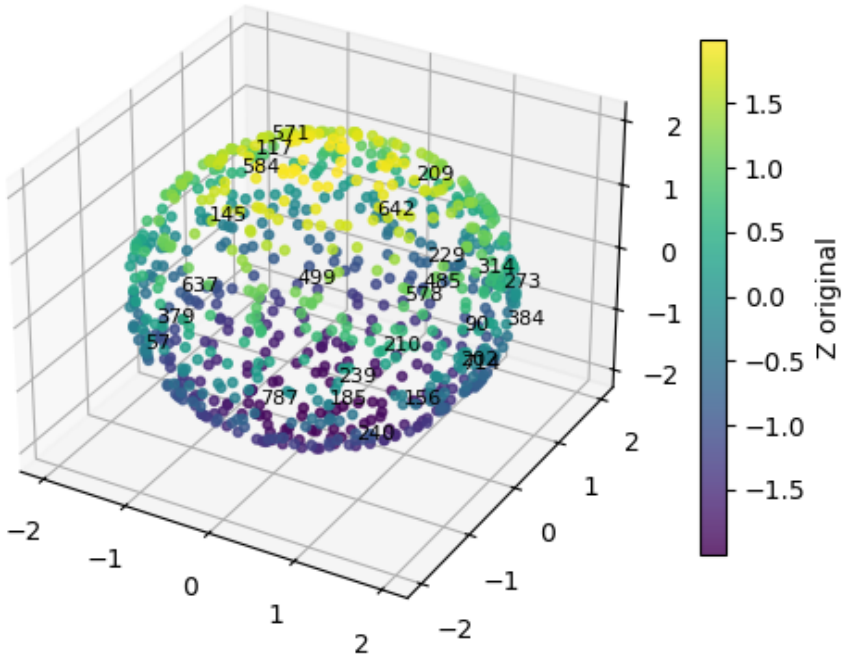
```
plt.title("ISOMAP 2D con etiquetas parciales")
plt.savefig("artifacts/fig_isomap2d_labels.png", dpi=300, bbox_inches="tight")
plt.show()
```

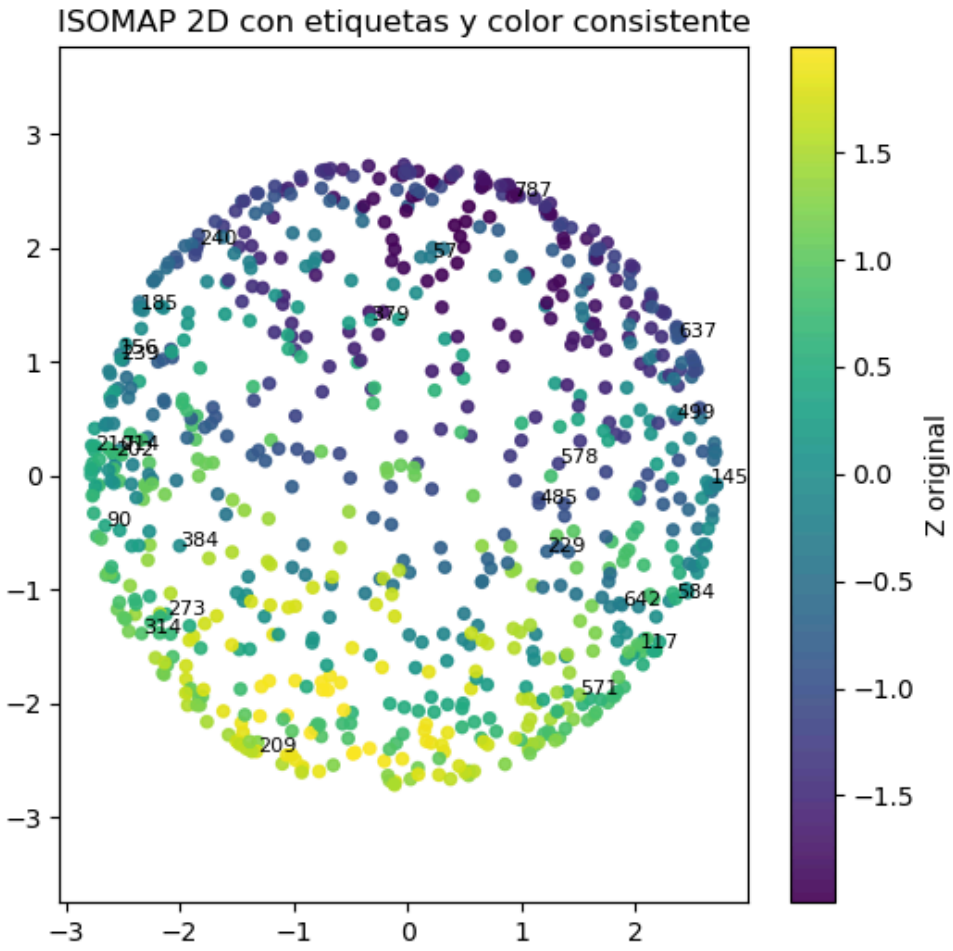


In [50]:

```
c = X[:,2]

# Esfera
fig = plt.figure(figsize=(6,6))
ax = fig.add_subplot(111, projection="3d")
sc = ax.scatter(X[:,0], X[:,1], X[:,2], c=c, s=12, alpha=0.8)
fig.colorbar(sc, ax=ax, shrink=0.6, label="Z original")
for i in idx_to_label:
    ax.text(X[i,0], X[i,1], X[i,2], str(i), fontsize=8)
plt.savefig("artifacts/fig_sphere_labels_colored.png", dpi=300, bbox_inches="tight"
plt.show()

# ISOMAP 2D
plt.figure(figsize=(6,6))
plt.scatter(Y2[:,0], Y2[:,1], c=c, s=20, alpha=0.9)
for i in idx_to_label:
    plt.text(Y2[i,0], Y2[i,1], str(i), fontsize=8)
plt.axis("equal"); plt.colorbar(label="Z original")
plt.title("ISOMAP 2D con etiquetas y color consistente")
```

```
plt.savefig("artifacts/fig_isomap2d_labels_colored.png", dpi=300, bbox_inches="tigh
plt.show()
```





In [51]:
```
# TERMINAMOS ISOMAP PODEMOS AHORA INICIAR LA GEO DESCRIPTIVA
```

LPCA (Local PCA) – Estimación de planos tangentes

La idea es que, en una variedad 2D inmersa en (\mathbb{R}^3), el vecindario de un punto se puede aproximar por un **plano tangente**.

Procedimiento:

1. Tomamos el punto central (x_c) y sus vecinos (x_j).
2. Centramos: (P = X_{\text{vecinos}} - x_c).
3. Hacemos **SVD**: (P = U \Sigma V^\top).
4. En (\mathbb{R}^3):
   - Las **dos primeras columnas de (V)** forman la base del plano tangente (`E` , de tamaño (3,2)).
   - La **última columna** es la aproximación al vector normal (`normal` , de tamaño (3,)).
   - Los **valores singulares principales** (`svals`) indican la variabilidad local en cada dirección tangente.

Ejemplo de salida:

- `E.shape = (3,2)` → dos vectores base del plano tangente.
- `normal.shape = (3,)` → vector normal ortogonal.
- `svals = [1.29, 0.94]` → varianzas locales en las direcciones tangentes.

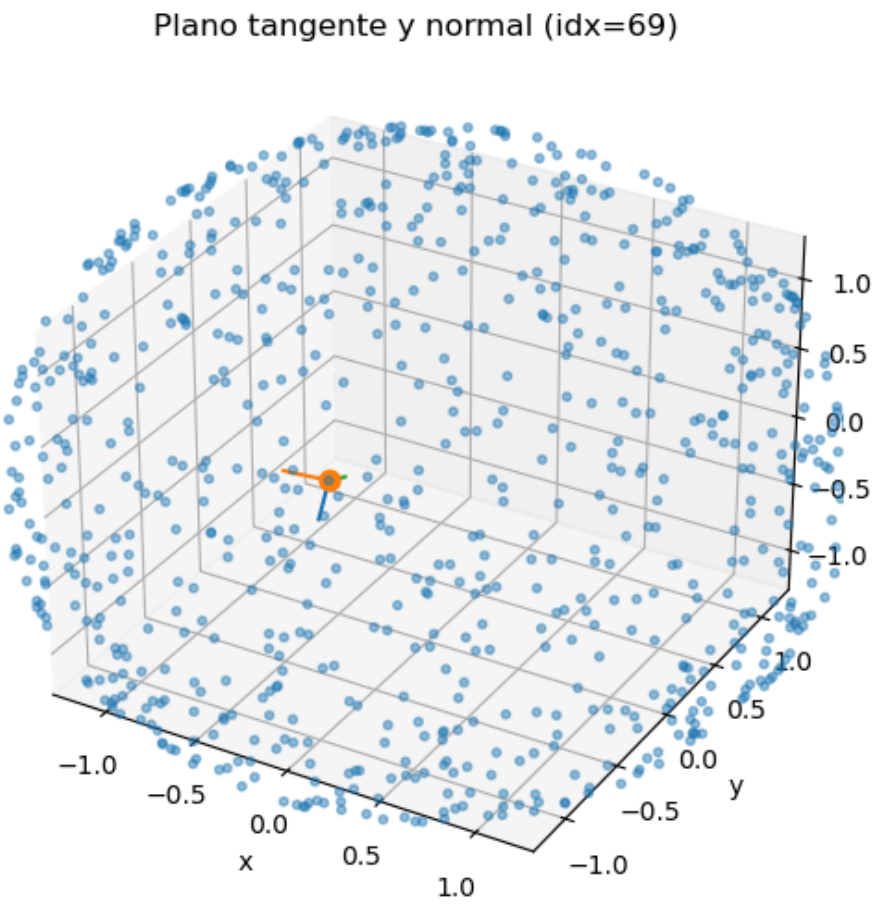
In [52]:
```
import numpy as np
import matplotlib.pyplot as plt


from src.geom.lpca import tangent_plane
from src.viz.plots import plot_tangent_frame_3d
```

In [53]:
```
center_idx = 49                                 # el punto aa analizar
neighbors_idx = np.array(list(G.neighbors(center_idx)))
```

In [54]:
```
center, E, normal, svals = tangent_plane(X, neighbors_idx, center_idx, d=2)
print(E.shape, normal.shape if normal is not None else None, svals)
```

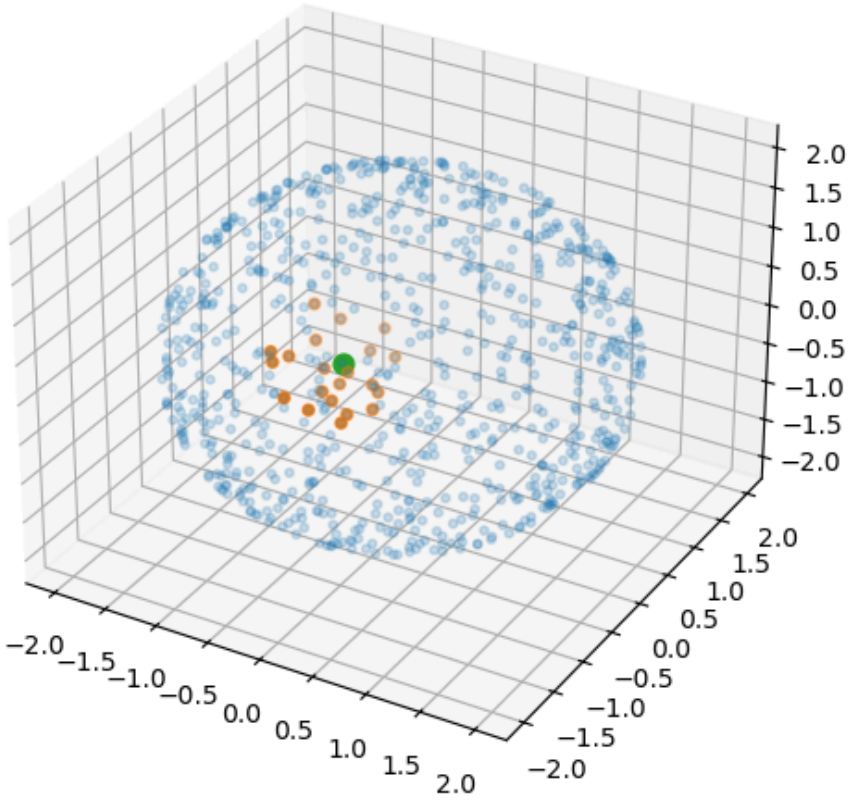
```
(3, 2) (3,) [1.6545423  1.37419352]
```

In [84]:
```
plot_tangent_frame_3d(
    X,
    center_idx=center_idx,
    E=E,
    normal=normal,
    scale=0.25,
    outpath="artifacts/fig_tangent_center42.png",
    title=f"Plano tangente y normal (idx={center_idx})"
)
```



In [85]:
```
#Vecinos usados
```

In [86]:
```
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(6,6))
ax = fig.add_subplot(111, projection="3d")

# todos (suaves)
ax.scatter(X[:,0], X[:,1], X[:,2], s=10, alpha=0.2)

# vecinos resaltados
Nv = X[neighbors_idx]
ax.scatter(Nv[:,0], Nv[:,1], Nv[:,2], s=18)

# centro
c = X[center_idx]
ax.scatter([c[0]], [c[1]], [c[2]], s=60)

plt.savefig(f"artifacts/fig_neighbors_{center_idx}.png", dpi=300, bbox_inches="tigh
plt.show()
```



Coordenadas locales (LPCA → proyección):

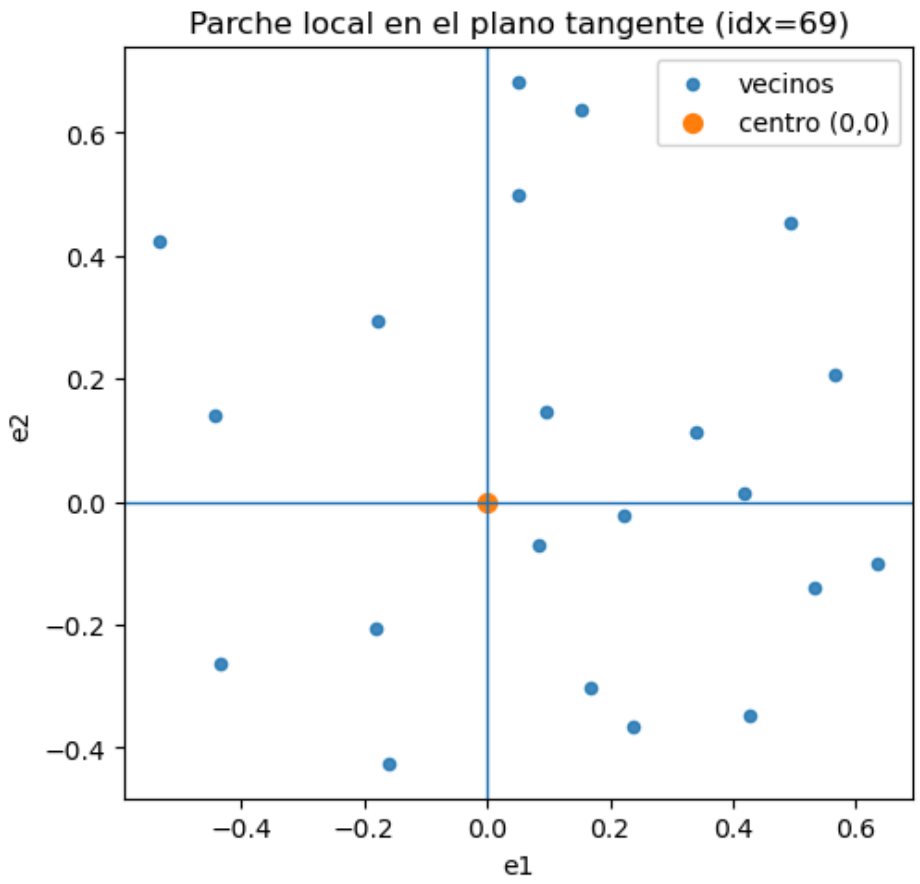
In [87]:
```
from sklearn.decomposition import PCA
```

In [88]:
```
from src.geom.metric import local_coordinates
from src.viz.plots import plot_tangent_frame_3d
```

In [89]:
```
# Vecinos del punto y proyección al plano tangente
Nv = X[neighbors_idx]                          # (m, 3)
coords2 = local_coordinates(Nv, center=center, E=E)  # (m, 2)

# El propio centro en coords locales (debe ser ~ (0,0))
c_local = local_coordinates(center[None, :], center=center, E=E)[0]
print("Centro en coords locales:", c_local)
```

```
Centro en coords locales: [0. 0.]
```

In [90]:
```
plt.figure(figsize=(5.5, 5.5))
plt.scatter(coords2[:,0], coords2[:,1], s=20, alpha=0.85, label="vecinos")
plt.scatter([c_local[0]], [c_local[1]], s=50, label="centro (0,0)")
plt.axhline(0, linewidth=1); plt.axvline(0, linewidth=1)
plt.gca().set_aspect("equal", adjustable="box")
plt.title(f"Parche local en el plano tangente (idx={center_idx})")
plt.xlabel("e1"); plt.ylabel("e2"); plt.legend()
plt.savefig(f"artifacts/fig_local_patch_{center_idx}.png", dpi=300, bbox_inches="ti
plt.show()
```



In [91]:
```
pca2 = PCA(n_components=2).fit(coords2)
print("Varianzas locales (patch 2D):", pca2.explained_variance_)

# razón entre la primera y la segunda varianza
ratio = pca2.explained_variance_[0] / pca2.explained_variance_[1]
print("Razón de anisotropía (var1/var2):", ratio)
```

```
Varianzas locales (patch 2D): [0.1241237  0.10719792]
Razón de anisotropía (var1/var2): 1.1578927868847306
```

In [94]:
```
#Métrica local en el parche (estimate_metric).
```

In [95]:
```
from src.geom.metric import local_coordinates, estimate_metric
from src.viz.plots import plot_local_patch_with_ellipse
```

In [67]:
```
# Parche local en coords
Nv = X[neighbors_idx]                          # (m, 3)
coords2 = local_coordinates(Nv, center=center, E=E)  # (m, 2)
```

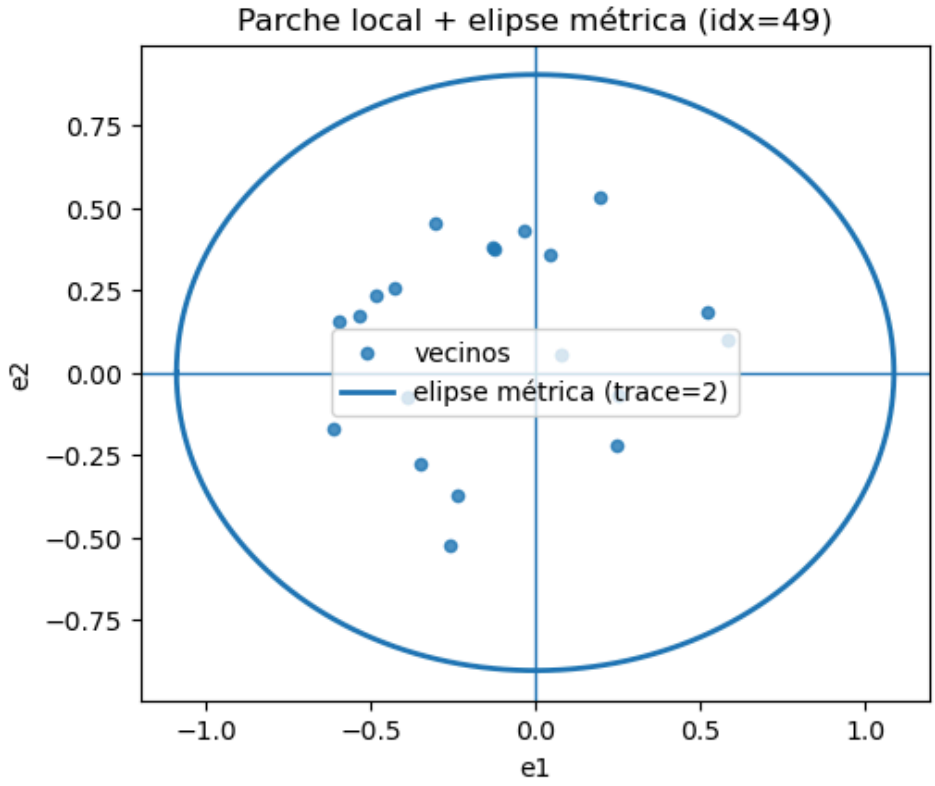
In [68]:
```
# Estimar métrica (normalizada, trace ≈ 2)
g = estimate_metric(coords2, weighted=False)
print("g:\n", g)
print("trace(g) =", np.trace(g))
```

```
g:
 [[ 1.18355327e+00 -3.06000741e-16]
 [-3.06000741e-16  8.16446729e-01]]
trace(g) = 2.0
```

In [69]:
```
#  Diagnóstico rápido: autovalores (anisotropía) y razón
evals, _ = np.linalg.eigh(g)
evals = np.sort(evals)[::-1]  # mayor → menor
print("eigenvals(g) =", evals)
print("razón mayor/menor =", evals[0] / evals[-1])
```

```
eigenvals(g) = [1.18355327 0.81644673]
razón mayor/menor = 1.4496393091460162
```

In [70]:
```
#  Visual del parche + elipse métrica
plot_local_patch_with_ellipse(
    coords2,
    g,
    outpath=f"artifacts/fig_local_metric_{center_idx}.png",
    title=f'Parche local + elipse métrica (idx={center_idx})'
)
```



In [71]:
```
for center_idx in [5, 100, 654, 69]:
    neighbors_idx = np.array(list(G.neighbors(center_idx)))
    center, E, normal, svals = tangent_plane(X, neighbors_idx, center_idx, d=2)
    coords2 = local_coordinates(X[neighbors_idx], center=center, E=E)
    g = estimate_metric(coords2, weighted=True)

    evals, evecs = np.linalg.eigh(g)              # evals ascendente
    lmin, lmax = float(evals[0]), float(evals[1])
    kappa = lmax / lmin
    tr = lmin + lmax

    print(f"{center_idx} → autovalores g: [{lmin:.4f} {lmax:.4f}]  "
          f"razón(lmax/lmin): {kappa:.2f}  trace: {tr:.4f}")
```

```
5 → autovalores g: [0.7731 1.2269]  razón(lmax/lmin): 1.59  trace: 2.0000
100 → autovalores g: [0.8237 1.1763]  razón(lmax/lmin): 1.43  trace: 2.0000
654 → autovalores g: [0.8560 1.1440]  razón(lmax/lmin): 1.34  trace: 2.0000
69 → autovalores g: [0.9166 1.0834]  razón(lmax/lmin): 1.18  trace: 2.0000
```

In [76]:
```
from src.geom.curvature_extrinsic import fit_quadratic_patch, curvature_from_quadra
```

In [77]:
```
#  Plano tangente en el punto
center, E, normal, _ = tangent_plane(X, neighbors_idx, center_idx, d=2)
```

In [78]:
```
#  Parche local (coords 2D en el plano tangente)
Nv = X[neighbors_idx]
coords2 = local_coordinates(Nv, center=center, E=E) # (m,2)
```

In [79]:
```
# 3Alturas respecto a la normal (para el ajuste h(u,v))
heights = (Nv - center) @ normal                    # (m,)
```

In [80]:
```
# Ajuste cuadrático y curvaturas
coeffs = fit_quadratic_patch(coords2, heights, weighted=True)
k1, k2, K, H, R = curvature_from_quadratic(coeffs)
```

In [81]:
```
print("Coeficientes [a, b, c, α, β, γ]:", coeffs)
print("Curvaturas principales:", k1, k2)
print("Curvatura gaussiana K:", K)
print("Curvatura media H:", H)
print("Curvatura escalar R=2K:", R)
```

```
Coeficientes [a, b, c, α, β, γ]: [ 5.15450628e-01  1.16498008e-03  5.12926074e-01 -
7.28128931e-02
 -8.78892020e-02 -3.97697071e-05]
Curvaturas principales: 0.5124706428832692 0.5159060597168834
Curvatura gaussiana K: 0.26438671009048553
Curvatura media H: 0.5141883513000762
Curvatura escalar R=2K: 0.5287734201809711
```

8. Appendix B - Jupyter Lab Real Data

In [84]:
```
import numpy as np
print("NumPy:", np.__version__)
```

```
NumPy: 2.3.2
```

In [19]:
```
import sys, os
from pathlib import Path

def attach_src_and_cd():
    here = Path.cwd()
    # Busca 'src' en la carpeta actual y en todos los padres
    for base in [here] + list(here.parents):
        cand = base / "src"
        if cand.is_dir():
            if str(cand) not in sys.path:
                sys.path.append(str(cand))
            # Pon el directorio de trabajo en la raíz del repo (donde vive 'src')
            os.chdir(base)
            print("OK: usando src en", cand)
            print("CWD ahora es:", Path.cwd())
            return True
    print("ERROR: no encontré carpeta 'src' empezando desde", here)
    return False

attach_src_and_cd()
```

```
OK: usando src en C:\Users\henry\economia-no-euclideana\src
CWD ahora es: C:\Users\henry\economia-no-euclideana
```

Out[19]: True

Paso 1: Generamos los puntos sobre una superficie conocida

In [20]:
```
# === Carga de datos reales: Global Market Curvature (2018–2025)
import pandas as pd
import numpy as np

df = pd.read_csv("global_market_curvature.csv", index_col=0, parse_dates=True)
X = df.to_numpy(dtype=float)
dates = df.index
vars_ = df.columns.tolist()

T, d = X.shape
print(f"Shape de X: {X.shape} → T={T} días, d={d} variables")
print("Fechas:", dates.min().date(), "→", dates.max().date())
print("Variables:", vars_)
print("\nPrimeros 5 puntos:\n", X[:5])
```

```
Shape de X: (2556, 8) → T=2556 días, d=8 variables
Fechas: 2018-01-02 → 2024-12-31
Variables: ['BTC-USD', 'CL=F', 'DX-Y.NYB', 'ETH-USD', 'GC=F', '^GSPC', '^IXIC', '^TN
X']

Primeros 5 puntos:
 [[ 2.57486250e+00 -2.40831544e-03 -1.89678693e-02  2.94087996e+00
  -3.46864694e-02 -2.94868373e-02 -3.19291668e-02 -8.56456819e-03]
 [ 3.85590760e-01  7.35670640e-01  9.62147319e-01  1.84075011e+00
   2.08252679e-01  5.86729484e-01  6.38869281e-01 -2.68183836e-01]
 [ 7.03908160e-01  2.17225275e-01 -1.00008306e+00  3.96825918e-01
   2.75610276e-01  3.58935214e-01  1.09005658e-01  7.81879232e-02]
 [ 3.08949525e+00 -3.32369273e-01  2.97872110e-01  3.58568002e-01
   5.24492671e-02  6.47662226e-01  6.32392899e-01  3.22030015e-01]
 [ 1.35309649e-01 -2.40831544e-03 -1.89678693e-02  9.29768556e-01
  -3.46864694e-02 -2.94868373e-02 -3.19291668e-02 -8.56456819e-03]]
```

KNN

Conectaremos cada punto con su K vecinos mas cercanos

In [21]:
```
import numpy as np
from sklearn.neighbors import NearestNeighbors
from scipy.sparse import csr_matrix, csgraph
import matplotlib.pyplot as plt
```

In [22]:
```
from src.graph.knn import build_knn_graph

k = 40
G = build_knn_graph(X, k)


print(G)
print("Nodos:", G.number_of_nodes())
print("Aristas:", G.number_of_edges())
```

```
Graph with 2556 nodes and 77447 edges
Nodos: 2556
Aristas: 77447
```

Si: k es muy bajo → analizas la variedad a escalas muy finas. k es muy alto → analizas la variedad a escalas más gruesas (más “promediadas”).

In [24]:
```
# KNN indices (para curvatura local)
from sklearn.neighbors import NearestNeighbors
import numpy as np

k_neighbors = 40  # usa 40–60 para estabilidad de parches
nbrs = NearestNeighbors(n_neighbors=k_neighbors+1, metric="euclidean").fit(X)
distances, indices = nbrs.kneighbors(X)
knn_idx  = indices[:, 1:]
knn_dist = distances[:, 1:]
```

```
# sanity rápido
print("knn_idx:", knn_idx.shape, "knn_dist:", knn_dist.shape)
print("dist ordenadas:", np.all(np.diff(knn_dist, axis=1) >= -1e-12))
print("mediana dist al k-ésimo vecino:", np.median(knn_dist[:, -1]))
```

```
knn_idx: (2556, 40) knn_dist: (2556, 40)
dist ordenadas: True
mediana dist al k-ésimo vecino: 1.3719293295586892
```

Geodesics

In [25]:
```
from __future__ import annotations
import numpy as np
import networkx as nx
```

In [26]:
```
from src.geodesic.shortest_paths import compute_geodesic_distances, handle_disconne
```

In [27]:
```
# Calculamos distancias geodesicas
```

In [28]:
```
D = compute_geodesic_distances(G)
print("Matriz de distancias geodésicas:", D.shape)
```

```
Matriz de distancias geodésicas: (2556, 2556)
```

In [29]:
```
from src.geodesic.shortest_path_FAST import compute_geodesic_distances_fast

# Dijkstra (rápido)
%time D= compute_geodesic_distances_fast(G, method="D")
```

```
CPU times: total: 5.52 s
Wall time: 5.74 s
```

In [31]:
```
D = handle_disconnections(D, strategy="big_value")
```

In [32]:
```
# sanity checks
```

In [33]:
```
print("¿Simétrica?", is_symmetric(D))
print("¿Diagonal cero?", has_zero_diagonal(D))
```

```
¿Simétrica? True
¿Diagonal cero? True
```

In [35]:
```
print("Distancia entre nodo A y B:", D[15,2000])
```

```
Distancia entre nodo A y B: 2.358752910491206
```

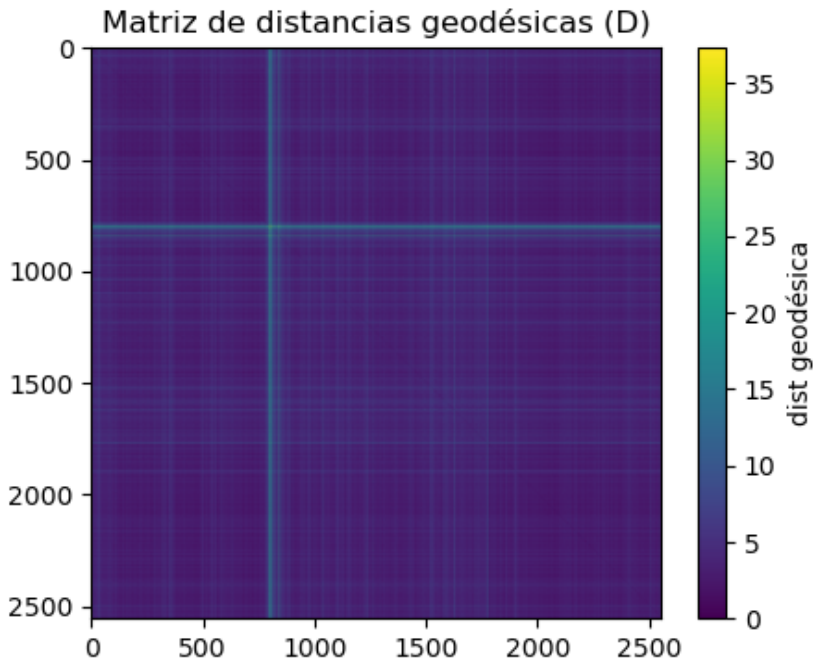
In [37]:
```
from src.viz.plots import plot_geodesic_path_3d
```

In [38]:
```
src, dst = 0, 470
```

In [40]:
```
from src.viz.plots import plot_geodesic_distance_matrix
```

In [41]:
```
plot_geodesic_distance_matrix(
    D
```

```
)
```



In [42]:
```
#MDS
```

In [43]:
```
from src.embed.isomap import classical_mds
```

In [44]:
```
#Embeding en 2D
```

In [45]:
```
Y2 = classical_mds(D, n_components=2)
print("Shape embedding 2D:", Y2.shape)
```

```
Shape embedding 2D: (2556, 2)
```

In [47]:
```
# Embedding en 3D
Y3 = classical_mds(D, n_components=3)
print("Shape embedding 3D:", Y3.shape)
```

```
Shape embedding 3D: (2556, 3)
```

In [48]:
```
# Embedding en 5D
Y5 = classical_mds(D, n_components=5)
print("Shape embedding 5D:", Y5.shape)
```

```
Shape embedding 5D: (2556, 5)
```

In [49]:
```
#Sanity Check
```

In [50]:
```
# Checar media ~ 0 (centrado)
print("Media de cada columna:", Y5.mean(axis=0))

# Ver primeras distancias preservadas
import numpy as np
```

```
print("Dist original geodesica (0,1):", D[0,1])
print("Dist embebida (0,1):", np.linalg.norm(Y5[0]-Y5[1]))
```

```
Media de cada columna: [-6.10275176e-17  2.50842643e-16 -2.23760325e-16 -9.57241589e
-16
 -3.21730123e-16]
Dist original geodesica (0,1): 3.564937681747706
Dist embebida (0,1): 2.460981570402122
```

In [53]:
```
n = X.shape[0]
ids = np.arange(n)  # 0..n-1
```

In [54]:
```
# TERMINAMOS ISOMAP PODEMOS AHORA INICIAR LA GEO DESCRIPTIVA
```

LPCA (Local PCA) – Estimación de planos tangentes

In [55]:
```
import numpy as np
import matplotlib.pyplot as plt


from src.geom.lpca import tangent_plane
from src.viz.plots import plot_tangent_frame_3d
```

In [56]:
```
center_idx = 200
neighbors_idx = knn_idx[center_idx]
```

In [57]:
```
#  — Tangente y normales por PCA; coords tangentes y alturas
from sklearn.decomposition import PCA
import numpy as np

def tangent_and_normals_PCA(X, neighbors_idx, center_idx, dim=2):
    """
    Devuelve: center (D,), Xn (m,D), E (D×dim), F (D×q)
    """
    Xn = X[neighbors_idx]
    center = X[center_idx]
    Xc = Xn - Xn.mean(axis=0)

    D = X.shape[1]
    p = PCA(n_components=D).fit(Xc)
    E = p.components_[:dim].T      # plano tangente
    F = p.components_[dim:].T      # todas las normales
    return center, Xn, E, F

def local_coords_and_heights_all_normals(Xn, center, E, F):
    """
    U: (m,2) coords en el plano tangente. Hs: (m,q) alturas vs cada normal.
    """
    Xc = Xn - center
    U  = Xc @ E
    Fn = F / (np.linalg.norm(F, axis=0, keepdims=True) + 1e-12)
    Hs = Xc @ Fn
    return U, Hs, Fn
```

In [58]:
```
#  — Parche local general (sin suponer 3D)
center_idx = 200
neighbors_idx = knn_idx[center_idx]

center, Xn, E, F = tangent_and_normals_PCA(X, neighbors_idx, center_idx, dim=2)
U, Hs, F = local_coords_and_heights_all_normals(Xn, center=center, E=E, F=F)

print("E shape:", E.shape, "| F shape:", F.shape)   # (D,2) y (D,q)
print("U shape:", U.shape, "| Hs shape:", Hs.shape) # (m,2) y (m,q)
```

```
E shape: (8, 2) | F shape: (8, 6)
U shape: (40, 2) | Hs shape: (40, 6)
```

In [59]:
```
#Vecinos usados
```

Coordenadas locales (LPCA → proyección):

In [60]:
```
#  — Vecinos y proyección al plano (ya lo tenemos)
Nv = X[neighbors_idx]     # (m, D)
coords2 = U               # (m, 2)  ← usa U directo
c_local = np.array([0.0, 0.0])  # el centro en coords locales es (0,0)
print("Centro en coords locales:", c_local)
```

```
Centro en coords locales: [0. 0.]
```

In [61]:
```
pca2 = PCA(n_components=2).fit(coords2)
print("Varianzas locales (patch 2D):", pca2.explained_variance_)

# razón entre la primera y la segunda varianza
ratio = pca2.explained_variance_[0] / pca2.explained_variance_[1]
print("Razón de anisotropía (var1/var2):", ratio)
```

```
Varianzas locales (patch 2D): [0.01276634 0.00973188]
Razón de anisotropía (var1/var2): 1.311805451921592
```

In [62]:
```
#Métrica local en el parche (estimate_metric).
```

In [63]:
```
from src.geom.metric import local_coordinates, estimate_metric
from src.viz.plots import plot_local_patch_with_ellipse
```

In [64]:
```
# Parche local en coords (ya tienes center, E, neighbors_idx, Xn)
Nv = X[neighbors_idx]                          # (m, 3)
coords2 = local_coordinates(Nv, center=center, E=E)  # (m, 2)
```

In [65]:
```
# Estimar métrica (normalizada, trace ≈ 2)
g = estimate_metric(coords2, weighted=False)
print("g:\n", g)
print("trace(g) =", np.trace(g))
```

```
g:
 [[1.13811827 0.19897809]
 [0.19897809 0.86188173]]
trace(g) = 2.0
```

In [66]:
```
#  Diagnóstico rápido: autovalores (anisotropía) y razón
evals, _ = np.linalg.eigh(g)
evals = np.sort(evals)[::-1]  # mayor → menor
print("eigenvals(g) =", evals)
print("razón mayor/menor =", evals[0] / evals[-1])
```

```
eigenvals(g) = [1.24221672 0.75778328]
razón mayor/menor = 1.639277022432253
```

In [67]:
```
# Re-barrido con KNN + PCA local (sin tangent_plane, sin G.neighbors)
for center_idx in [5, 1000, 1291, 2400]:
    neighbors_idx = knn_idx[center_idx]  #fuente única de vecinos

    center, Xn, E, F = tangent_and_normals_PCA(X, neighbors_idx, center_idx, dim=2)
    U, Hs, F = local_coords_and_heights_all_normals(Xn, center=center, E=E, F=F)

    g = estimate_metric(U, weighted=True)       #coords2 → U
    evals = np.linalg.eigvalsh(g)               # ascendente
    lmin, lmax = float(evals[0]), float(evals[-1])
    kappa = lmax / (lmin + 1e-12)
    tr = lmin + lmax

    print(f"{center_idx} -> autovalores g: [{lmin:.4f} {lmax:.4f}] "
          f"razón(lmax/lmin): {kappa:.2f}  trace: {tr:.4f}")
```

```
5 -> autovalores g: [0.3519 1.6481] razón(lmax/lmin): 4.68  trace: 2.0000
1000 -> autovalores g: [0.9240 1.0760] razón(lmax/lmin): 1.16  trace: 2.0000
1291 -> autovalores g: [0.8414 1.1586] razón(lmax/lmin): 1.38  trace: 2.0000
2400 -> autovalores g: [0.3847 1.6153] razón(lmax/lmin): 4.20  trace: 2.0000
```

In [68]:
```
### Curvatura local vía parche cuadrático (Monge + mínimos cuadrados)
```

In [69]:
```
from src.geom.curvature_extrinsic import fit_quadratic_patch, curvature_from_quadra
```

In [70]:
```
# Parche local (coords 2D en el plano tangente)
# Nv = X[neighbors_idx]
coords2 = U
```

In [71]:
```
# Alturas respecto a una normal (puedes iterar sobre todas si quieres)
a = 0
heights = Hs[:, a]
```

In [72]:
```
assert coords2.shape[0] == heights.shape[0], f"mismatch: {coords2.shape[0]} vs {hei
assert coords2.shape[1] == 2
```

In [73]:
```
#  Ajuste cuadrático y curvaturas
coeffs = fit_quadratic_patch(coords2, heights, weighted=True)
k1, k2, K, H, R = curvature_from_quadratic(coeffs)
```

In [74]:
```
print("Coeficientes [a, b, c, α, β, γ]:", coeffs)
print("Curvaturas principales:", k1, k2)
print("Curvatura gaussiana K:", K)
print("Curvatura media H:", H)
print("Curvatura escalar R=2K:", R)
```

```
Coeficientes [a, b, c, α, β, γ]: [ 0.52614907 -0.55384661  0.26854448  0.03905452 -
0.18006734  0.00778955]
Curvaturas principales: -0.17127972480458584 0.9659732702464324
Curvatura gaussiana K: -0.16545163589639475
Curvatura media H: 0.39734677272092334
Curvatura escalar R=2K: -0.3309032717927895
```

In [75]:
```
#Sanity check
```

In [76]:
```
#  Ajuste cuadrático por normal y agregados H, K
import numpy as np

def fit_quad_uv_to_h(U, h, weighted=True):
    u, v = U[:,0], U[:,1]
    A = np.column_stack([u*u, u*v, v*v, u, v, np.ones_like(u)])
    if weighted:
        r2 = u*u + v*v
        s  = np.median(np.sqrt(r2)) + 1e-12
        w  = np.exp(-r2/(2*s*s))
        W  = np.diag(w/(w.sum()+1e-12))
        coeffs = np.linalg.lstsq(W @ A, W @ h, rcond=None)[0]
    else:
        coeffs = np.linalg.lstsq(A, h, rcond=None)[0]
    a,b,c,_,_,_ = coeffs
    H2 = np.array([[2*a, b],[b, 2*c]])
    k1, k2 = np.linalg.eigvalsh(H2)
    return H2, (float(k1), float(k2))

Bs, k_pairs, H_comp = [], [], []
for a in range(Hs.shape[1]):            # loop por todas las normales
    H2, (k1, k2) = fit_quad_uv_to_h(U, Hs[:, a], weighted=True)
    Bs.append(H2); k_pairs.append((k1, k2)); H_comp.append(0.5*np.trace(H2))

H_comp = np.array(H_comp)               # H^α
H_vec  = F @ H_comp                     # vector de curvatura media en ℝ^D
K_intr = float(np.sum([np.linalg.det(B) for B in Bs]))

print("Primeras 3 parejas (k1,k2):", k_pairs[:3], "…")
print("|H| =", float(np.linalg.norm(H_vec)), "   K intrínseca =", K_intr)
```

```
Primeras 3 parejas (k1,k2): [(-0.17127972480536197, 0.9659732702456227), (-1.9618383
64177785, -0.8943418481636513), (-1.3743412376805728, -0.9089876884010434)] …
|H| = 1.9455322663677317    K intrínseca = 2.700897171715264
```

In [77]:
```
import pandas as pd
rows = []
for i in range(X.shape[0]):
    try:
        neighbors_idx = knn_idx[i]
        center, Xn, E, F = tangent_and_normals_PCA(X, neighbors_idx, i, dim=2)
        U, Hs, F = local_coords_and_heights_all_normals(Xn, center, E, F)

        # métrica local (diagnóstico)
        g = estimate_metric(U, weighted=True)
        evals = np.linalg.eigvalsh(g)
        aniso = float(evals[-1] / (evals[0] + 1e-12))
```

```
        # curvaturas por TODAS las normales
        pairs, dets, Hcomp = [], [], []
        for a in range(Hs.shape[1]):
            coeffs = fit_quadratic_patch(U, Hs[:,a], weighted=True)
            a2,b2,c2,_,_,_ = coeffs
            H2 = np.array([[a2, b2],[b2, c2]])
            k1a,k2a = np.linalg.eigvalsh(H2)
            pairs.append((k1a,k2a))
            dets.append(np.linalg.det(H2))
            Hcomp.append(0.5*np.trace(H2))

        H_vec = F @ np.array(Hcomp)
        H_norm = float(np.linalg.norm(H_vec))
        K_intr = float(np.sum(dets))
        k1_mean = float(np.mean([p[0] for p in pairs]))
        k2_mean = float(np.mean([p[1] for p in pairs]))

        rows.append((i, dates[i], H_norm, K_intr, k1_mean, k2_mean, aniso))
    except Exception:
        rows.append((i, dates[i], np.nan, np.nan, np.nan, np.nan, np.nan))

curv_df = pd.DataFrame(rows, columns=["idx","date","H_norm","K_intrinsic","k1_mean"
curv_df.to_csv("curvature_market_full_normals.csv", index=False)
print("✅ Guardado curvature_market_full_normals.csv")
```

```
✅ Guardado curvature_market_full_normals.csv
```

In [78]:

```
# === Sweep temporal: |H|(t) y K_intr(t) ===
import numpy as np
import pandas as pd

rows = []
T = X.shape[0]

for i in range(T):
    try:
        neighbors_idx = knn_idx[i]                          # usa SIEMPRE KNN
        center, Xn, E, F = tangent_and_normals_PCA(X, neighbors_idx, i, dim=2)
        U, Hs, F = local_coords_and_heights_all_normals(Xn, center, E, F)

        # Curvaturas por todas las normales
        dets, Hcomp = [], []
        for a in range(Hs.shape[1]):
            coeffs = fit_quadratic_patch(U, Hs[:, a], weighted=True)
            a2, b2, c2, _, _, _ = coeffs
            H2 = np.array([[a2, b2], [b2, c2]])   # 2ª forma aprox en esa normal
            dets.append(np.linalg.det(H2))        # para K intrínseca
            Hcomp.append(0.5 * np.trace(H2))      # componente de H en esa normal

        H_vec  = F @ np.array(Hcomp)              # vector de curvatura media en R^
        H_norm = float(np.linalg.norm(H_vec))
        K_intr = float(np.sum(dets))

        rows.append((dates[i], H_norm, K_intr))
    except Exception:
```

```
        rows.append((dates[i], np.nan, np.nan))

curv_ts = pd.DataFrame(rows, columns=["date","H_norm","K_intr"]).set_index("date").
curv_ts.to_csv("curvature_timeseries.csv")
print("✅ Guardado: curvature_timeseries.csv  |  shape:", curv_ts.shape)
curv_ts.head()
```

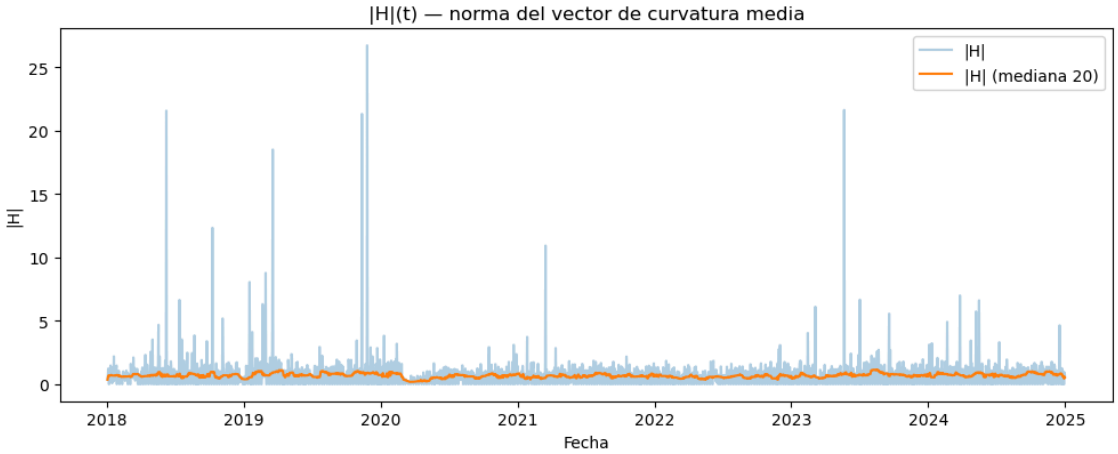
✅ Guardado: curvature_timeseries.csv | shape: (2556, 2)

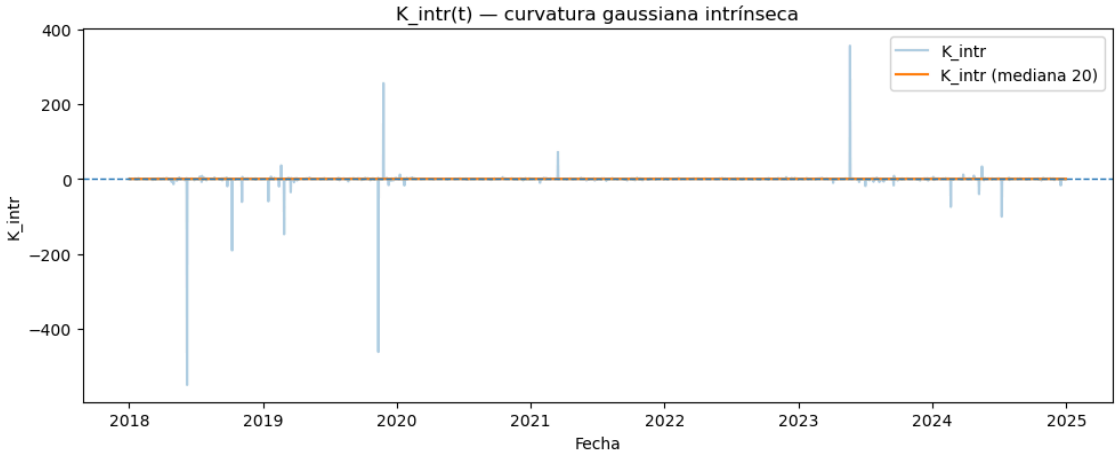
Out[78]:

date	H_norm	K_intr
2018-01-02	0.358638	0.003115
2018-01-03	0.662032	-0.185185
2018-01-04	1.232911	-0.607014
2018-01-05	0.728813	0.006254
2018-01-06	0.000000	0.000000

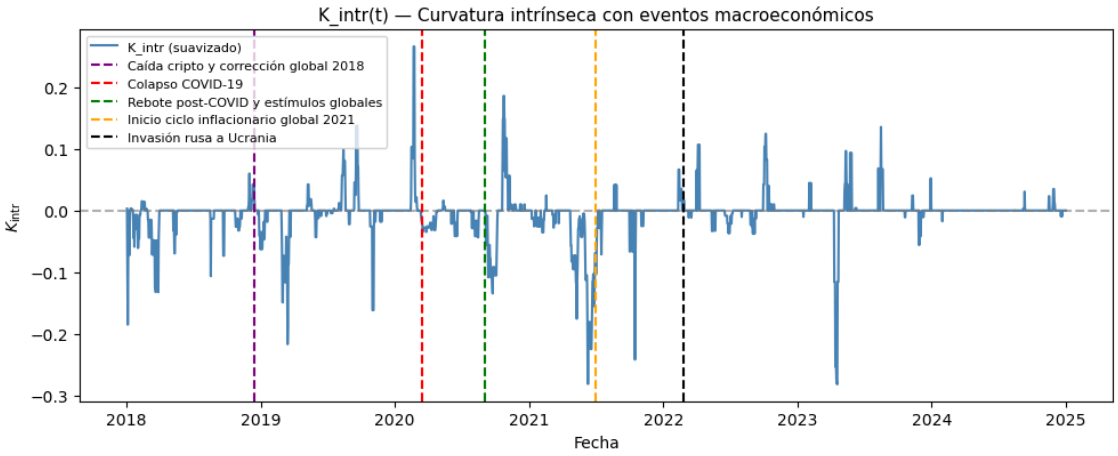
In [79]:

```
import matplotlib.pyplot as plt
ts = curv_ts.copy()

# Suavizado opcional (mueve la ventana a tu gusto)
win = 20
ts["H_norm_s"]  = ts["H_norm"].rolling(win, min_periods=1).median()
ts["K_intr_s"]  = ts["K_intr"].rolling(win, min_periods=1).median()

plt.figure(figsize=(10,4.2))
plt.plot(ts.index, ts["H_norm"],  alpha=0.35, label="|H|")
plt.plot(ts.index, ts["H_norm_s"],           label=f"|H| (mediana {win})")
plt.title("|H|(t) — norma del vector de curvatura media")
plt.xlabel("Fecha"); plt.ylabel("|H|")
plt.legend(); plt.tight_layout(); plt.show()

plt.figure(figsize=(10,4.2))
plt.plot(ts.index, ts["K_intr"],  alpha=0.35, label="K_intr")
plt.plot(ts.index, ts["K_intr_s"],           label=f"K_intr (mediana {win})")
plt.axhline(0, linestyle="--", linewidth=1)
plt.title("K_intr(t) — curvatura gaussiana intrínseca")
plt.xlabel("Fecha"); plt.ylabel("K_intr")
plt.legend(); plt.tight_layout(); plt.show()
```





In [83]:
```
plt.figure(figsize=(10,4.2))
plt.plot(ts.index, ts["K_intr_s"], label="K_intr (suavizado)", color="steelblue")
plt.axhline(0, linestyle="--", color="gray", alpha=0.6)

# --- Eventos principales según la interpretación geométrica ---
plt.axvline(pd.Timestamp("2018-12-15"), color="purple", linestyle="--", label="Caíd
plt.axvline(pd.Timestamp("2020-03-15"), color="red", linestyle="--", label="Colapso
plt.axvline(pd.Timestamp("2020-09-01"), color="green", linestyle="--", label="Rebot
plt.axvline(pd.Timestamp("2021-07-01"), color="orange", linestyle="--", label="Inic
plt.axvline(pd.Timestamp("2022-02-24"), color="black", linestyle="--", label="Invas

plt.legend(fontsize=8, loc="upper left")
plt.title("K_intr(t) — Curvatura intrínseca con eventos macroeconómicos", fontsize=
plt.xlabel("Fecha")
plt.ylabel("$K_{\\text{intr}}$")
plt.tight_layout()
plt.show()
```


K_intr(t) — Curvatura intrínseca con eventos macroeconómicos
K_intr (suavizado)
Caída cripto y corrección global 2018
Colapso COVID-19
Rebote post-COVID y estímulos globales
Inicio ciclo inflacionario global 2021
Invasión rusa a Ucrania
K_{intr}
0.2
0.1
0.0
−0.1
−0.2
−0.3
2018
2019
2020
2021
2022
2023
2024
2025
Fecha

In [ ]:

Referencias

[1] Howard Eves. *A survey of geometry*. Vol. 1. Boston, MA: Allyn y Bacon, 1963.

[2] Noelia Freire. *Axiomas, las normas invisibles que rigen las matemáticas*. National Geographic España. Sep. de 2024. URL: https://www.nationalgeographic.com.es/ciencia/axiomas-normas-invisibles-que-rigen-matematicas_23099.

[3] Euclides. *The thirteen books of the Elements, Vol. 1: Books I–II*. Ed. y trad. por Thomas L. Heath. 2.ª ed. Original work published ca. 300 BCE. New York, NY: Dover Publications, 1956.

[4] Michael Spivak. *A Comprehensive Introduction to Differential Geometry*. 3rd. Vol. 1. Houston, Texas: Publish or Perish, Inc., 1999.

[5] John M. Lee. *Introduction to Riemannian Manifolds*. 2.ª ed. Graduate Texts in Mathematics. Springer, 2018. DOI: 10.1007/978-3-319-91755-9.

[6] *Tecnología de la información — Inteligencia artificial — Conceptos y terminología de la inteligencia artificial*. Primera edición, julio de 2022. Ginebra, Suiza: Organización Internacional de Normalización, 2022.

[7] Richard Bellman. *Dynamic Programming*. A Rand Corporation Research Study, Sixth printing 1972. Princeton, New Jersey: Princeton University Press, 1957. ISBN: 0-691-07951-8.

[8] Andrzej Maćkiewicz y Waldemar Ratajczak. «Principal components analysis (PCA)». En: *Computers Geosciences* 19.3 (1993), págs. 303-342. ISSN: 0098-3004. DOI: https://doi.org/10.1016/0098-3004(93)90090-R. URL: https://www.sciencedirect.com/science/article/pii/009830049390090R.

[9] Laurens van der Maaten y Geoffrey Hinton. «Visualizing data using t-SNE». En: *Journal of machine learning research* 9.Nov (2008), págs. 2579-2605.

[10] Joshua B. Tenenbaum, Vin de Silva y John C. Langford. «A Global Geometric Framework for Nonlinear Dimensionality Reduction». En: *Science* 290.5500 (2000), págs. 2319-2323. DOI: 10.1126/science.290.5500.2319. eprint: https://www.science.org/doi/pdf/10.1126/science.290.5500.2319. URL: https://www.science.org/doi/abs/10.1126/science.290.5500.2319.

[11] Michael A. A. Cox y Trevor F. Cox. «Multidimensional Scaling». En: *Handbook of Data Visualization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, págs. 315-347. ISBN: 978-3-540-33037-0. DOI: 10.1007/978-3-540-33037-0_14. URL: https://doi.org/10.1007/978-3-540-33037-0_14.