El Espectro Modular de π: Unificación Teórica, Isomorfismo DSP y Validación a Exaescala

Arquitectura de Hibridación Algorítmica en $\mathbb{Z}/6\mathbb{Z}$

José Ignacio Peinador Sala
Investigador Independiente
joseignacio.peinador@gmail.com

30 de noviembre de 2025

Resumen

Este artículo presenta la consolidación definitiva del marco teórico "Espectro Modular", que propone una solución a la discontinuidad entre la naturaleza discreta de los enteros y la periodicidad trascendente de las funciones trigonométricas mediante el anillo $\mathbb{Z}/6\mathbb{Z}$. Se demuestra el isomorfismo entre esta estructura aritmética y la **Descomposición Polifase** utilizada en el Procesamiento Digital de Señales (DSP). La investigación culmina con la implementación del "Hiper-Computador Modular" (algoritmo *Hybrid Stride-6*), validado experimentalmente mediante el cálculo de 10^8 dígitos de π en un entorno de recursos restringidos (12GB RAM), logrando una eficiencia de paralelización del 95 % y una velocidad sostenida de 83,000 dígitos/s. Finalmente, se confirma la rigidez espectral de los ceros de Riemann bajo este filtro modular.

1. Introducción: La Crisis del Continuo

La intersección entre la Teoría de Números Computacional y el Análisis Armónico ha buscado históricamente conciliar la naturaleza discreta de $\mathbb{Z}$ con la mecánica ondulatoria continua. La evaluación de funciones como $\sin(n)$ para $n \in \mathbb{Z}$ genera un comportamiento aparentemente caótico debido a la inconmensurabilidad entre la red entera y el período trascendente 2π.

El presente trabajo postula que este caos es aparente y puede ser reordenado mediante un filtro basado en el módulo $m = 6$, expandiendo los hallazgos preliminares sobre estructuras de canales primos presentados en trabajos anteriores [1]. La elección de este módulo no es arbitraria: es el producto de los primeros primos (2×3) y define la red hexagonal, permitiendo capturar simetrías rotacionales que otros módulos pierden.

2. Fundamentos Teóricos: El Filtro $\mathbb{Z}/6\mathbb{Z}$

2.1. Definición de la Descomposición Modular

Todo ángulo $\theta \in \mathbb{Z}$ admite una descomposición única basada en el algoritmo de la división:

$$\theta = 6k + r, \quad \text{donde } r \in \{0,1,2,3,4,5\} \tag{1}$$

Esta transformación convierte la recta numérica en una estructura cilíndrica $\mathbb{Z} \times \mathbb{Z}/6\mathbb{Z}$. Los residuos se clasifican en tres categorías físicas:

- **Canales Nulos** $(0,3)$**:** Puntos de anclaje ($\gcd(r,6) \neq 1$).
- **Canales Primos** $(1,5)$**:** Generadores de alta energía ($\gcd(r,6) = 1$).
- **Canales Compuestos** $(2,4)$**:** Armónicos pares.

2.2. La Identidad de Euler y Aproximaciones Diofánticas

La estabilidad numérica de $e^{i\pi} + 1 = 0$ se analiza bajo la lente modular. Dado que $\pi \approx 3{,}14159$, el entero más cercano es 3. El vector unitario en el canal $r = 3$ es:

$$e^{i3} = \cos(3) + i\sin(3) \approx -0{,}989 + 0{,}141i \tag{2}$$

Este es el punto entero más cercano a -1 en el plano complejo, sugiriendo que la identidad de Euler aprovecha un "atractor de estabilidad" natural en el canal 3.

3. Isomorfismo con Procesamiento Digital de Señales (DSP)

Una de las contribuciones centrales de este estudio es la formalización del vínculo con la teoría de sistemas *multirate*, tal como se describe en la literatura clásica de bancos de filtros [2].

Teorema 3.1 (Isomorfismo Polifase)**.** *La descomposición modular de una serie hipergeométrica $S = \sum a_k$ es matemáticamente equivalente a la Descomposición Polifase de una señal discreta $x[n]$ con factor de decimación $M = 6$.*

En el dominio Z, esto se expresa como:

$$X(z) = \sum_{r=0}^{5} z^{-r} E_r(z^6) \tag{3}$$

Donde $E_r(z)$ son las componentes polifase (los canales modulares). Esto valida la robustez de la arquitectura propuesta, alineándola con algoritmos probados como la FFT de base compuesta y el Algoritmo de Factores Primos (PFA).

4. Arquitectura Algorítmica: Hibridación "Stride-6"

Para superar los cuellos de botella de memoria en el cálculo de π mediante la Serie de Chudnovsky [3], desarrollamos la arquitectura **Hybrid Stride-6 Binary Splitting**.

4.1. El Problema Monolítico

Los algoritmos tradicionales (Binary Splitting estándar) construyen un único árbol de recursión gigantesco. Esto satura el ancho de banda de memoria y provoca condiciones de carrera en sistemas multinúcleo al intentar acceder a los mismos bloques de datos grandes.

4.2. La Solución Modular (Shared-Nothing)

Nuestra implementación descompone la serie en 6 hilos de ejecución totalmente independientes (Canales). Cada canal procesa una sub-serie dispersa:

$$S_r = \sum_{j=0}^{\infty} t_{6j+r} \tag{4}$$

4.2.1. La Hoja de Transición "Stride"

Para hacer esto eficiente, la unidad fundamental de cómputo (la hoja del árbol recursivo) no calcula un solo término, sino que sintetiza la matriz de transición de un bloque completo de 6 términos. Esto permite que el algoritmo "salte" (stride) por la serie, comprimiendo la complejidad local y reduciendo la profundidad del árbol en un factor de $\log_2 6$.

Algorithm 1 Lógica del Worker Modular (Simplificada)

1: **Input:** Residuo r, Rango N
2: Inicializar acumuladores $P = 1, Q = 1$
3: **for** j en Rango Local **do**
4: $k_{start} \leftarrow 6j + r$
5: Calcular Transición Comprimida $(P_{leaf}, Q_{leaf}, B_{val})$ para $k \rightarrow k + 6$
6: Acumular en Árbol Binario Local
7: **end for**
8: Aplicar Corrección de Prefijo (Fase)
9: **return** Suma Racional Parcial S_r

5. Validación Experimental: La Barrera de los 100 Millones

La arquitectura fue sometida a una prueba de estrés extrema: el cálculo de **100,000,000 de dígitos** de π en un entorno de memoria restringida (Google Colab, 2 vCPU, 12GB RAM).

5.1. Resultados del Benchmark

El cálculo se completó exitosamente, generando un archivo de 95 MB y demostrando la estabilidad térmica y de memoria del diseño.

Tabla 1: Métricas de la Ejecución "100M Barrier Run"

Métrica	**Valor Registrado**
Tiempo Fase Paralela	1085.03 s (18.08 min)
Tiempo Total (con I/O)	1194.32 s (19.90 min)
Velocidad Media	**83,729 dígitos/s**
Speedup (vs Secuencial)	1,90× (Eficiencia 95 %)
Consumo de RAM (Pico)	≈ 6,8 GB (Controlado)

5.2. Validación Forense

Se verificó la integridad de la secuencia final comparándola con bases de datos de referencia (y-cruncher records).

Secuencia Final (10^8): `...97572203175207489816116831393751 59`

La coincidencia exacta confirma que la descomposición modular no introduce deriva numérica (*numerical drift*) incluso tras billones de operaciones aritméticas.

6. Implicaciones para la Función Zeta de Riemann

Se aplicó el filtro modular a los ceros no triviales de la función Zeta de Riemann (γ_n), corrigiendo hipótesis preliminares de sesgo.

6.1. Rigidez Espectral Confirmada

El análisis ampliado reveló una distribución altamente uniforme ($p \approx 0{,}98$ en test χ^2) de los ceros módulo 6. Este resultado negativo es significativo: confirma la propiedad de **Rigidez Espectral** predicha por la conexión con Matrices Aleatorias (GUE), estudiada extensamente por Odlyzko [4]. La estructura modular sirve como un tamiz de alta sensibilidad para verificar la "aleatoriedad perfecta" de los ceros.

7. Conclusiones y Futuro: El "Hexa-Core Engine"

La investigación valida el "Espectro Modular de π" como un marco matemáticamente sólido y computacionalmente superior para arquitecturas paralelas.

1. **Isomorfismo:** Se ha demostrado la equivalencia entre la aritmética modular de Chudnovsky y los bancos de filtros polifase en DSP.
2. **Rendimiento:** La arquitectura *Shared-Nothing* elimina los bloqueos de memoria, permitiendo cálculos de escala masiva (10^8 dígitos) en hardware commodity.

Visión de Futuro: Los resultados sugieren que una implementación en hardware dedicado (FPGA/ASIC) con 6 núcleos físicos aislados y memoria HBM distribuida podría escalar linealmente, ofreciendo una eficiencia energética (dígitos/vatio) inalcanzable para las arquitecturas monolíticas actuales.

Agradecimientos

El autor desea expresar su agradecimiento a la comunidad de código abierto, cuyo esfuerzo colectivo permite la democratización de la investigación científica fuera de los entornos académicos tradicionales.

Infraestructura y Software

Este trabajo fue posible gracias a la infraestructura de computación en la nube proporcionada por **Google Colab**, que facilitó el acceso a recursos de cómputo necesarios para la ejecución de los experimentos.

La implementación computacional se desarrolló utilizando el lenguaje de programación **Python**. Agradecemos a los desarrolladores de las siguientes bibliotecas fundamentales para este estudio:

- **gmpy2** y **decimal**: Para la aritmética de precisión arbitraria y operaciones de alto rendimiento con grandes números enteros (`mpz`) y flotantes (`Decimal`), esenciales para la precisión numérica requerida.
- **SciPy** y **NumPy**: Para el cálculo científico avanzado, incluyendo herramientas de procesamiento de señales (`scipy.signal`), funciones especiales (`scipy.special.gamma`) y manipulación matricial eficiente.
- **Pandas**: Por las capacidades para la estructuración, manipulación y análisis de datos tabulares.
- **Matplotlib** e **IPython**: Para la visualización de datos, generación de gráficas y la presentación interactiva de resultados matemáticos.
- **Python Standard Library**: Específicamente los módulos de concurrencia (`multiprocessing`, `concurrent.futures`) que permitieron la ejecución paralela de procesos, así como las funciones matemáticas nativas (`math`, `cmath`) para operaciones en el dominio real y complejo.

Asistencia de Inteligencia Artificial

En consonancia con los principios de transparencia en la investigación, se declara el uso de asistentes basados en Modelos de Lenguaje Grande (LLMs) durante el desarrollo de este manuscrito. Estas herramientas se utilizaron para asistencia bibliográfica, revisión de estilo y depuración de código. La conceptualización teórica, el planteamiento matemático del isomorfismo modular y la interpretación final de los resultados son responsabilidad exclusiva del autor humano.

Disponibilidad de Datos y Código

Con el objetivo de fomentar la reproducibilidad y el avance del conocimiento colectivo, el código fuente completo, los scripts de entrenamiento y los pesos de los modelos generados en esta investigación están disponibles públicamente en el siguiente repositorio:

`https://github.com/NachoPeinador/Arquitectura-de-Hibridacion-Algoritmica-en-Z-6Z`

Licenciamiento

El software se distribuye bajo un modelo de **licenciamiento dual** diseñado para proteger la sostenibilidad de la investigación independiente mientras se fomenta la ciencia abierta:

1. **Uso Académico y No Comercial**: El código fuente está disponible bajo la licencia **PolyForm Noncommercial License 1.0.0**. Esto permite su uso, modificación y distribución gratuita exclusivamente para fines de investigación, educación y proyectos personales sin ánimo de lucro.

2. **Uso Comercial**: Cualquier uso con fines de lucro, incluyendo la integración en productos propietarios, consultoría o servicios SaaS, está estrictamente prohibido sin un acuerdo previo. Para adquirir derechos de explotación comercial, consulte el archivo `LICENSE` o contacte con el autor.

Declaración de Intereses

El autor declara que esta investigación se llevó a cabo de manera independiente, sin recibir financiación externa, subvenciones corporativas ni patrocinios institucionales. El desarrollo de la arquitectura Hex-Ensgine y el marco teórico del isomorfismo modular no presentan conflictos de interés financieros ni comerciales.

Referencias

[1] Peinador Sala, J. I. (2025). The Modular Spectrum of π: From Prime Channel Structure to Elliptic Supercongruences (Versión 1). Zenodo. https://doi.org/10.5281/zenodo.17680024

[2] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*. Prentice Hall, 1993.

[3] D. Chudnovsky, G. Chudnovsky, "Approximations and Complex Multiplication", *Ramanujan Revisited*, 1988.

[4] A. M. Odlyzko, "On the distribution of spacings between zeros of the zeta function", *Mathematics of Computation*, vol. 48, no. 177, pp. 273–308, 1987.