Un Millón de Precisiones
Entorno a π

Cálculo de π con Múltiple Precisión

Horacio Useche Losada
Google Software Developer
horaciouseche@gmail.com

Noviembre de 2018

Diagramación en LaTeX realizada por el autor bajo Linux Fedora

©2018. Todos los derechos reservados.

Contents

1 Introducción

El cálculo de los dígitos de π ha sido siempre una de las tareas más deseadas por los matemáticos de todos los tiempos, siendo la más antigua de todas. El número π se viene calculando desde la edad de hierro, sin exagerar, y en este documento podrá encontrar un resumen de todos esos esfuerzos con más de 5000 años de historia.

Actualmente el record pertenece al físico de partículas suizo Peter Trueb, que en noviembre de 2016, encontró 22 459 157 718 361 números decimales de π, completamente verificados. Estos son 2.2 billones de decimales, una cantidad tan abrumadora que alcanzaría para dar 1.2 vueltas al planeta tierra, por el ecuador, y suponiendo cada decimal del tamaño de las letras que ahora lee.

Muchos lectores se preguntaran para que sirve calcular tantos dígitos decimales si para calcular la circuferencia del universo con un error no superior al radio atómico, bastaría una precisión de 32 decimales. La respuesta es la misma por la cual el ser humano se empeña en reducir el tiempo de recorrido para los 100 metros planos. Es un símbolo de prepotencia y progreso, del cual, el ser humano, no se puede desprender. Una auténtica demostración de cerebro y máquina que presume del alcance de la especie humana.

Para realizar este tipo de esfuerzos, se deben tomar una serie de decisiones concernientes con los algoritmos a usar, esto es, los criterios matemáticos, además de seleccionar las herramientas de software para programar dichos criterios y por último el hardware, o computadores físicos. Todo ello junto, conforma el arsenal de batalla para llevar a cabo hazañas como las de conquistar nuevos records.

Ya se trate de aficionados o matemáticos profesionales, este documento le entrega una revista incremental, desde rústicos y antiguos criterios, hasta los más modernos y sofisticados, usados en la ambiciosa conquista de los dígitos de π, que sin duda, le darán lustre a su saber y habilidad.

Aquí, por lo pronto, nos conformamos con llevar al límite de lo posible, las herramientas de hardware casero, con las cuales el lector podrá hacer uso de las mejores teorías matemáticas para tener una idea muy fresca y fiel, de las tormentas que se desatan en las cumbres borrascosas de la alta matemática.

2 Reseña Histórica

El número π se define, en geometría euclidiana, como la relación entre la longitud de una circunferencia y su diámetro. Es un número irracional[1] y una de las constantes matemáticas más importantes. El valor de π se ha obtenido con diversas aproximaciones a lo largo de la historia, siendo una de las constantes matemáticas que más aparece en las ecuaciones de la física, junto con el número e, o número de Euler.[2]

En esta sección hacemos una breve revista por los hechos históricos más relevantes en torno al valor de π.[3]

2.1 Antiguo Egipto

Su estimación data del año 1800 A.C.,[4] con la relación:

$$\pi = \frac{256}{81} = 3.16049$$

2.2 Mesopotamia

Su estimación data del año 1900-1600 A.C., basado en la relación:

$$\pi = 3 + \frac{1}{8} = 3.125$$

2.3 Antigüedad clásica

Arquímedes (siglo III, A.C.):

$$3\frac{10}{71} < \pi < 3\frac{1}{7}$$
$$3.1408 < \pi < 3.1428$$

También Claudio Tolomeo (siglo II) encontró:

$$\pi = \frac{377}{120} = 3.1416$$

[1] Que tiene infinitos decimales
[2] Véase Wikipedia, número Pi.
[3] La mayoría de estos apuntes provienen de Wikipedia, la popular enciclopedia en línea.
[4] Descrito en el papiro Rhind

2.4 Antigua China

Hacia el año 120, el astrónomo chino Zhang Heng (78-139) fue uno de los primeros en usar la aproximación:

$$\sqrt{10} = 3.1622$$

Un siglo después, el astrónomo Wang Fang lo estimó en:

$$\pi = \frac{142}{45} = 3.1555$$

Hacia 263, el matemático Liu Hui fue el primero en sugerir que $\pi \simeq 3.14$.

A finales del siglo V, el matemático chino Zu Chongzhi calculó el valor de π en 3.141592, al que dio dos aproximaciones racionales de π,

$$\begin{aligned} \pi &= \frac{22}{7} = 3.142857 \\ &= \frac{355}{113} = 3.141592 \end{aligned}$$

2.5 Matemática India

El matemático indio Aryabhata estimó (siglo V) el valor de π en 3.1416. A mediados del siglo VII, Brahmagupta calcula $\pi \simeq \sqrt{10} \simeq 3.1622$. Hacia 1400 Madhava obtiene una aproximación exacta hasta 10 dígitos (3.14159265359), siendo el primero en emplear series para realizar la estimación.

2.6 Matemática islámica

En el siglo IX Al-Jwarizmi, en su Álgebra, hace notar que el hombre práctico usa $\pi \simeq 22/7$. El matemático persa Ghiyath al-Kashi fue capaz de calcular el valor aproximado de π con nueve dígitos, empleando una base numérica sexagesimal, lo que equivale a una aproximación de 16 dígitos decimales: $2\pi = 6.2831853071795865$.

2.7 Renacimiento europeo

Fibonacci (1170-1250), en su Practica Geometriae, amplifica el método de Arquímedes, proporcionando un intervalo más estrecho. Algunos matemáticos del siglo XVII, como Viète, usaron polígonos de hasta 393 216 lados

para aproximarse con buena precisión a 3.141592653. En 1593 el flamenco Adriaan van Roomen (Adrianus Romanus) obtiene una precisión de 16 dígitos decimales usando el método de Arquímedes.

2.8 Época moderna

En 1610 el matemático Ludolph van Ceulen calculó los 35 primeros decimales de π. Se dice que estaba tan orgulloso de esta hazaña que lo mandó grabar en su lápida.

En 1665 Isaac Newton desarrolla la serie:

$$\arcsin x = x + \frac{1}{2}\frac{x^3}{3} + \frac{1\cdot 3}{2\cdot 4}\frac{x^5}{5} + \frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\frac{x^7}{7}$$

en la cual, haciendo $x = \frac{1}{2}$, obtuvo:

$$\arcsin \frac{1}{2} = \frac{\pi}{6}$$

El matemático inglés John Wallis desarrolló en 1655 la conocida serie Producto de Wallis:

$$\frac{\pi}{2} = \frac{2}{1}\cdot\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\frac{6}{7}\cdot\frac{8}{7}\cdot\frac{8}{9}\cdots$$

En 1699, a sugerencia de Edmond Halley, el matemático inglés Abraham Sharp (1651-1742) calculó π con una precisión de 71 dígitos decimales usando la serie de Gregory:

$$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \ldots$$

con $x = \frac{1}{\sqrt{3}}$ se obtiene una serie para:

$$\arctan\left(\frac{1}{\sqrt{3}}\right) = \frac{\pi}{6}$$

Para alcanzar la precisión obtenida, debió usar alrededor de trescientos términos en la serie, según el relato que hace la enciclopedia Wikipedia sobre este hecho.

En el siglo XVII, los matemáticos empezaron a usar series infinitas e identidades trigonométricas en busca de π. El cuadro 1 presenta un resumen de los esfuerzos hechos durante esta época, anterior al desarrollo de la informática moderna.

Año	**Autor**	**Dígitos exactos de π**
1610	Ludolph van Ceulen	32
1621	Willebrord Snell	35
1630	Christoph Grienberger	38
1699	Abraham Sharp	71
1706	Jhon Machin	100
1719	Thomas Fanted de Lagry	100
1789	Jurij Vega	112
1794	Jurij Vega	136
1841	William Rutherford	152
1841	Zacarias Dase	200
1847	Thomas Clausen	248
1853	Lehmann	261
1855	Richter	500
1873	William Shanks (*)	527
1946	D. F. Ferguson	620
1947	D. F. Ferguson (**)	808

Table 1: Cálculos de π en la época moderna precomputacional

- (*) William Shanks tardó 15 años en obtener 527 decimales correctos.
- (**) En 1947 Ferguson recalculó π con 808 decimales con la ayuda de una calculadora mecánica.

2.9 Cálculos computarizados

Durante la segunda mitad del siglo XX, se desarrolla la informática moderna computacional, en la cual y por supuesto, se comenzaron los esfuerzos por calcular π haciendo uso de cada avance que se obtenía en esta materia. El cuadro 1 resume la actividad computacional en este periodo.

Comparando las tablas 1 y 2 se nota claramente la disparada que se pegó este asunto con la ayuda de los más modernos y poderosos ordenadores e impulsados también por el prestigio que conlleva para los fabricantes de ordenadores el hecho de que su marca aparezca en la lista de los guiness récords.

Año	**Autor**	**Dígitos exactos de π**
1949	G.W. Reitwiesner y otros	2037
1959	Guilloud	16167
1973	Guillord y Bouyer	1 001 250
1981	Miyoshi y Kanada	2 000 036
1982	Guilloud	2 000 050
1986	Bailey	29 360 111
1986	Kanada y Tamura	67 108 839
1987	Kanada, Tamura, Kobo y otros	134 217 700
1988	Kanada y Tamura	201 326 000
1989	Hermanos Chudnovsky	480 000 000
1989	Hermanos Chudnovsky	1 011 196 691
1991	Hermanos Chudnovsky	2 260 000 000
1994	Hermanos Chudnovsky	4 044 000 000
1995	Kanada y Takahashi	6 442 450 000
1997	Kanada y Takahashi	51 539 600 000
1999	Kanada y Takahashi	68 719 470 000
1999	Kanada y Takahashi	206 158 430 000
2002	Kanada y otros	1 241 100 000 000
2009	Daisuke Takahashi	2 576 980 370 000
2009	Fabrice Bellard	2 699 999 990 000
2010	Shigeru Kondo	5 000 000 000 000
2011	Shigeru Kondo	10 000 000 000 000
2012	A. J. Yee	5×10^{12}
2012	S. Kondo and A. J. Yee	10×10^{12}
2013	A. J. Yee and S. Kondo	12.1×10^{12}
2016	Peter Trueb	22.459.157.718.361

Table 2: Cálculos de π en la época moderna computacional. Tomado de Wikipedia.

3 Cálculo de π

Después de esta breve reseña histórica, ha llegado el momento de entrar en materia y sentarnos a calcular los dígitos de π, para lo cual se debe tener en cuenta, básicamente dos aspectos, cuya elección determinará el éxito de la operación:

1. Un algoritmo eficiente y

2. Un lenguaje rápido para desarrollar el algoritmo

La primera condición implica hacer uso del mejor progreso teórico en la materia, ya que no es lo mismo, como se verá enseguida, utilizar un criterio, como el método de Leibniz, que utilizar el método de Ramanujan. Los criterios matemáticos desarrollados a la fecha tienen diferentes tiempos de computo (eficiencia) y por tanto, algunos serán más rápidos y por ello buenos candidatos para ser explotados en este tipo de "empresas".

De otra parte, tenemos la elección de las herramientas de software con la cual se debe desarrollar el algoritmo elegido en el punto (1). Aquí surgen nuevas inquietudes que nos hacen detener un poco en su elección. Por ejemplo, podríamos elegir un lenguaje de alto nivel como Java, Visual Basic, Ruby, Phiton, Perl, etc. Estos ambientes son más fáciles de usar, amigables con el usuario, pero tienen el grave inconveniente de que por naturaleza, no soportan tipos de datos de alta precisión y aquí se requiere una precisión "endemoniada", de mucha exigencia. Ruby, Phiton y Java tiene soporte a tipos de precisión arbitraria ya sea de forma nativa como Ruby o, a través de bibliotecas especializadas como es el caso de Java y Phiton. No obstante, al ser lenguajes de alto nivel son más lentos y menos eficientes para el computo científico. Además, la precisión de estos ambientes es "aparente"ya que en la práctica introducen un redondeo en los datos que resulta intolerable para los objetivos que se persiguen. En resumen, son ambientes de trabajo apropiados para propósitos pedagógicos y aficionados, pero de ninguna manera para asuntos más profesionales como intentar un "record guiness"o algo similar.

Los lenguajes de Google, Go, Dart, y Kotlin, son una buena elección para desarrollos orientados a la web y a dispositivos móviles, pero para ir más lejos comparten problemas como los comentados anteriormente, en relación con la precisión y la velocidad y lo mismo puede decirse de las herramientas introducidas por Facebook (hack) y Apple, esto para no mencionar a Microsoft con su Visual C++ que, además de lento (por su modelo de clases), son ambientes de software propietario que limitan considerablemente su uso legal en este tipo de proyectos.

Herramientas como Mathematica o MathLab, no valen, primero son software propietario de uso restringido por su tipo de licencia, hay que ser legales! y segundo, estos paquetes ya tienen "prefabricado"los objetivos que se per-

siguen, son ideales para propósitos pedagógicos[5] pero no para investigación.

Habiendo hecho esta breve aclaración, la pregunta obligada es, cuál es la elección viable ? En mi modesta opinión y experiencia, se requiere un ambiente de programación que satisfaga las siguientes condiciones:

- Debe ser "open source", o software libre.
- Debe ser muy rápido y
- Deber ser muy profesional

El candidato ideal, cumpliendo con las exigencias, es el uso de la biblioteca GMP con C++ y será la herramienta utilizada en lo que sigue para los cálculos que nos esperan.

3.1 Método de Leibniz

El método de Leibniz se basa en la expresión:

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = 1 - \frac{1}{3} + \frac{1}{5} - \cdots = \frac{\pi}{4}$$

y su código asociado es:

```
mpf_class PiLeibniz(unsigned long int k)
{
 // calcula Pi por Leibniz
 unsigned long int i;
 mpf_class a(0.0,PREC);
 mpf_class s(0.0,PREC);
 mpf_class r(0.0,PREC);
 mpf_class sum(0.0,PREC);
 for(i=0;i<k;i++)
 {
  a=2*i+1;
  s=1/a;
  if(i%2==0)
```

[5]En colegios y universidades donde se pueden dar le lujo de invertir miles de dólares en licencias.

```
  {
   sum=sum+s;
  }
  else
  {
   sum=sum-s;
  }
 }
 r=4*sum;
 return r;
}
```

Salida:

```
Método de Leibniz con 1000000 términos en la sumatoria ...
3.14159165358979323871264338327919038419717035250010 58
Begin time: Fri Nov  2 10:09:25 2018
End time:   Fri Nov  2 10:09:25 2018
```

como podemos ver el método de Leibniz es bastante ineficiente, pues necesitó un millón de términos en la sumatoria para obtener tan solo 5 dígitos decimales de precisión.

Para correr esta rutina hemos usado una función `main()` similar a:

```
#include <stdio.h>
#include <iostream>
#include <gmpxx.h>
#include <time.h>
#include <math.h>

using namespace std;

const int PREC=1000;

int main()
{
 time_t tm1,tm2;
 struct tm *ptr;
 tm1=time(NULL);
```

```
 tm2=time(NULL);
 ptr=localtime(&tm1);
 mpf_class a(0.0,PREC);
 int iTerms=1000000;
 a=PiLeibniz(iTerms);
 printf("Método de Leibniz con %d términos
            en la sumatoria ...\n",iTerms);
 gmp_printf("%.100Ff\n",a);
 ptr=localtime(&tm2);
 cout<<"Begin time: "<<asctime(ptr)<<"\n";
 cout<<"End time:"<<asctime(ptr)<<"\n";
 return 0;
}
```

La función `main()` hace una cita a la función `PiLeibniz()`, la cual toma como argumento el número de términos que se usaran en la sumatoria. Una cita a la función `gmp_printf()` imprime el resultado del cálculo en pantalla y los objetos `tm1,tm2` sirven para tomar el tiempo de inicio y finalización de la rutina que, en este caso, están en el mismo segundo, lo cual significa que el cálculo tomó apenas algunas centésimas de segundo.

Para todos los ejercicios restantes se debe usar una función `main()` similar a la anterior, por tal motivo, no se volverá a citar esta función y supondremos que el lector entiende el uso de esta metodología. Esto para no caer en redundancias de código que nada aportan a la discusión.

3.2 Método de Nilakantha

El método de Nilakantha se basa en la expresión:

$$\pi = 3 + \sum_{n=1}^{\infty} \frac{4(-1)^{n+1}}{(2n) \times (2n+1) \times (2n+2)}$$

el código para este método es:

```
mpf_class PiNilakantha(unsigned long int k)
{
 // calcula Pi por Nilakantha
 unsigned long int i;
 mpf_class a(12.0,PREC);
```

```
 mpf_class s(0.0,PREC);
 mpf_class sum(3.0,PREC);
 for(i=1;i<=k;i++)
 {
  a=(2*i)*(2*i+1)*(2*i+2);
  s=4/a;
  if(i%2==0)
  {
   sum=sum-s;
  }
  else
  {
   sum=sum+s;
  }
 }
 return sum;
}
```

Salida:

```
Método de Nilakantha con 1000000 términos en la sumatoria ...
3.14159265358979323821264413327831538513466938375109018 9
Begin time: Fri Nov  2 10:33:13 2018
End time:   Fri Nov  2 10:33:13 2018
```

No de Términos	**Dígitos exactos**	**Estimación de π**
10	2	3.141839618929402211
100	6	3.141592903558552764
1000	9	3.141592653839792925
10000	10	3.141592653590043238
100000	15	3.141592653589793488
1000000	18	3.14159265358979323821

Table 3: Eficiencia y exactitud del método de Nilakantha.

El método es mucho más eficiente que el de Leibniz, el cuadro 3 muestra el número de dígitos exactos que se calculan versus el número de términos en la sumatoria del método.

3.3 Fórmula de Machin

La fórmula de Machin para estimar el valor de π es:

$$\frac{\pi}{4} = 4\arctan\frac{1}{5} - \arctan\frac{1}{239}$$

La fórmula entrega un valor de π con 14 dígitos verdaderos. No obstante, en este trabajo solo citaremos el uso de la fórmula modificada de Machin, como se muestra a continuación.

3.4 Fórmula modificada de Machin

La fórmula modificada de Machin para estimar el valor de π es:

$$\frac{\pi}{4} = 44\arctan\frac{1}{57} + 7\arctan\frac{1}{239} - 12\arctan\frac{1}{682} + 24\arctan\frac{1}{12943}$$

en código GMP/C++ se escribe:

```
mpf_class PiMachinMod(unsigned long int k)
{
 // pi por la formula de Machin
 mpf_class a(0.0,PREC);
 mpf_class sum(0.0,PREC);
 a=1.0/57.0;
 sum=44*ArcTan(a,k);
 a=1.0/239.0;
 sum=sum+7*ArcTan(a,k);
 a=1.0/682.0;
 sum=sum-12*ArcTan(a,k);
 a=1.0/12943.0;
 sum=sum+24*ArcTan(a,k);
 a=4.0*sum;
 return a;
}
```

Salida:

```
Método de Machin modificado con 1000 términos
                                          en la sumatoria ...
```

```
3.14159265358979305692442579676614890653644377459194230622017
Begin time: Fri Nov  2 10:44:56 2018
End time:   Fri Nov  2 10:44:56 2018
```

la fórmula produce un valor de π con 15 dígitos verdaderos.

3.5 Método de Ramanujan

El método de Ramanujan se basa en la expresión:

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{9801} \sum_{k=0}^{\infty} \frac{(4k)!(1103 + 26390k)}{(k!)^4 396^{4k}}$$

el código para este método es:

```
mpf_class PiRamanujan(unsigned long int k)
{
 // calcula Pi por Ramanujan
 unsigned long int i;
 mpf_class r(0.0,PREC);
 mpf_class a(12.0,PREC);
 mpf_class b(640320.0,PREC);
 mpf_class c(0.0,PREC);
 mpf_class d(0.0,PREC);
 mpf_class e(0.0,PREC);
 mpf_class s(0.0,PREC);
 mpf_class num(0.0,PREC);
 mpf_class den(0.0,PREC);
 mpf_class sum(0.0,PREC);
 for(i=0;i<=k;i++)
 {
  a=Factorial(4*i);
  b=1103+26390*i;
  num=a*b;
  c=Factorial(i);
  d=Pow(c,4);
  e=Pow(396,4*i);
  den=d*e;
  s=num/den;
```

```
  sum=sum+s;
 }
 a=Sqrt(2);
 b=2*a;
 c=b/9801;
 s=sum*c;
 r=1/s;
 return r;
}
```

Salida:

```
Método de Ramanujan con 10 términos en la sumatoria ...
3.14159265358979323846264338327950288419716939937511
Begin time: Fri Nov  2 10:49:26 2018
End time:   Fri Nov  2 10:49:26 2018
```

No Términos	**Dígitos exactos**
1	7
2	15
3	23
4	31
5	39
6	47
7	53
8	63
9	70
10	77

Table 4: Eficiencia del método de Ramanujan.

El cuadro 4 muestra que el método de Ramanujan[6] es más eficiente que el de Nilakantha, Machin, y otros. El método produce, aproximadamente,

[6]Srinivasa Ramanujan Iyengar o simplemente como Ramanujan; Erode, 22 de diciembre de 1887 - Kumbakonam, 26 de abril de 1920) fue un matemático autodidacta indio que, con una mínima educación académica en matemáticas puras, hizo contribuciones extraordinarias al análisis matemático, la teoría de números, las series y las fracciones continuas. Ramanujan desarrolló inicialmente su propia investigación matemática en forma aislada, que fue rápidamente reconocida por los matemáticos indios. Cuando sus habi-

8 nuevos dígitos en cada iteración!n y fue el método más eficiente conocido durante la primera mitad del siglo XX.

3.6 Método de Chudnovsky

El método de Chudnovsky se basa en la expresión:

$$\frac{1}{\pi} = 12 \sum_{k=0}^{\infty} \frac{(-1)^k (6k)!(13591409 + 545140134k)}{(3k)!(k!)^3 640320^{3k+\frac{3}{2}}}$$

el código para este método es:

```
mpf_class PiChudnovsky(unsigned long int k)
{
 // calcula Pi por Chudnovsky
 unsigned long int i;
 mpf_class r(0.0,PREC);
 mpf_class a(12.0,PREC);
 mpf_class b(640320.0,PREC);
 mpf_class c(0.0,PREC);
 mpf_class d(0.0,PREC);
 mpf_class e(0.0,PREC);
 mpf_class s(0.0,PREC);
 mpf_class sum(0.0,PREC);
 for(i=0;i<=k;i++)
 {
  s=FactorK(i);
  if(i%2==0)
  {
   sum=sum+s;
  }
  else
  {
   sum=sum-s;
```

lidades se hicieron evidentes para una comunidad matemática más amplia, centrada en Europa en ese momento, comenzó su famosa colaboración con el matemático británico G. H. Hardy. Redescubrió teoremas conocidos previamente, además de formular numerosas nuevas proposiciones. Tomado de [20].

```
  }
 }
 e=Pow(b,3);
 c=Sqrt(e);
 d=a/c;
 a=d*sum;
 r=1/a;
 return r;
}
mpf_class FactorK(unsigned long int k)
{
 // calcula el factor K
 // en la fórmula de Chudnovsky
 mpf_class r(0.0,PREC);
 mpf_class a(0.0,PREC);
 mpf_class b(0.0,PREC);
 mpf_class c(0.0,PREC);
 mpf_class d(0.0,PREC);
 mpf_class e(0.0,PREC);
 mpf_class f(0.0,PREC);
 mpf_class g(0.0,PREC);
 a=Factorial(6*k);
 b=a*(545140134*k+13591409);
 c=Factorial(k);
 e=Pow(c,3);
 d=Factorial(3*k);
 f=e*d;
 g=f*Pow(640320,3*k);
 r=b/g;
 return r;
}
```

Salida:

```
Método de Chudnovsky con 10 términos en la sumatoria ...
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812
Begin time: Fri Nov  2 11:15:45 2018
```

```
End time:   Fri Nov  2 11:15:45 2018
```

No Términos	Dígitos exactos
1	14
2	27
3	41
4	55
5	69
6	84
7	98
8	112
9	126
10	141

Table 5: Eficiencia del método de los hermanos Chudnovsky.

El algoritmo publicado aquí,[7] de hecho, cumple con todo para establecer o al menos, intentar establecer un nuevo record en esta materia, solo haría falta el acceso a un equipo de cómputo lo suficientemente poderoso para acometer la hazaña.

3.6.1 Eficiencia del algoritmo de Chudnovsky

No Dígitos	Tiempo
1000	$1''$
10000	$20''$
100000	$1^h36'$
1000000	más de un día

Table 6: Eficiencia del método de Chudnovsky.

El cuadro 6 muestra el consumo de tiempo de acuerdo con el número de dígitos calculados para el algoritmo de Chudnovsky. Este método, implementado por los hermanos Chudnosky en 1989, se lleva todos los aplausos

[7]Por cierto, quizá sea el único que se consiga en la web de forma abierta, con estas prestaciones.

hasta el momento. Es el método más eficiente conocido,[8] lo usan programas como Mathematica, MathLab, etc. También es el método utilizado para establecer todos los "guiness records"desde 1889. Los propios hermanos, lo usaron para establecer 4 records, dos en 1989, uno en 1991 y otro en 1994 (Ver cuadro 5).

3.6.2 Historia de los recods de los hermanos Chudnovsky

"Galileo[9] no tenía la posibilidad de ir a una tienda y comprarse un telescopio. En 1606, el matemático italiano se fabricó su propio instrumento astronómico para observar la Luna. Lo cierto es que el barbudo científico sabía de la existencia de un dispositivo fabricado en Holanda". Quien pone este ejemplo a Teknautas es otro matemático, esta vez ucraniano y con el rostro libre de vello, que sigue el mismo principio que el astrónomo. "Cuando no tienes instrumentos que te sirvan, los fabricas", nos dice David Chudnovsky, que trabaja codo con codo con su hermano Gregory, también matemático, en el Instituto de Matemáticas y Supercomputación Avanzada de la Universidad de Nueva York. Además de ser colegas científicos, David asiste a su hermano, que padece miastenia gravis, una enfermedad neuromuscular autoinmune. Juntos desarrollaron en 1987 un algoritmo para calcular el número π. Y también juntos, a principios de los noventa, construyeron una máquina que les permitió batir un récord mundial al calcular π con más de 2000 millones de decimales. No lo hicieron en ningún laboratorio, aunque por aquel entonces eran investigadores de la Universidad de Columbia, sino en el apartamento que Gregory poseía en la Gran Manzana. Alojar la máquina en la facultad podía traerles problemas si acababa fundiéndose y, por otro lado, el proyecto era sumamente caro. "Intentamos construir diferentes sistemas desde finales de los años 70 porque ninguna de las herramientas a las que teníamos acceso servían para nuestros propósitos", asegura David. "Los centros de computación eran instalaciones masivas, con mucha refrigeración, a las que resultaba difícil acceder", describe el ucraniano. Los científicos prácticamente tenían que "suplicar"para poder aprovechar su capacidad de

[8]El más eficiente de los métodos seguros. Aclaramos que le rinde más al algoritmo de Spigot visto en la sección 3.7, sin embargo, dicho algoritmo, que computa dígito por dígito los decimales de π, no entrega fidelidad absoluta en cuanto al resultado obtenido como si lo hace el método de Chudnovsky. Por tal motivo el algoritmo de Spigot no se utiliza en el tema de conseguir nuevos records.

[9]Sección tomada de [3].

cálculo durante un cierto tiempo, había prolongadas esperas y "muchos se ponían hostiles". Además, alojar la máquina en la facultad podía traerles problemas si acababa fundiéndose y, por otro lado, el proyecto era sumamente caro. Así que "lo más fácil era construirla nosotros". Bautizaron a su enorme retoño M zero. La letra hace referencia a "machine"y el número al éxito en su fabricación (lo intentaron varias veces). Ocupaba toda una habitación. "Tuvimos que pedir asesoramiento financiero en Wall Street para conseguir el dinero", indica David. Compraron la mayoría de los componentes a diferentes proveedores, pero no todos estaban disponibles en el mercado. "Diseñamos algunos de los chips y tuvimos que pagar mucho para que Toshiba nos los fabricara en Japón", recuerda.

Figure 1: Los hermanos Chudnovsky, creadores de un método eficiente con el cual se lograron varios records mundiales.

Un gigante difícil de guardar. El principal problema, aparte de la economía, era la refrigeración, "y más todavía por tratarse de dispositivos ópticos con componentes de silicio", puntualiza el matemático. De hecho, el primer dispositivo de la estirpe terminó fundiéndose. En el caso de M zero utilizaron, esencialmente, ventiladores para evitar la catástrofe. El principal objetivo de la computación no es el resultado, sino entender qué implica este. La creación de este superordenador tenía solo fines matemáticos: "El principal objetivo de la computación no es el resultado, sino entender qué

implica este", explica David. Por una parte, su objetivo era probar la eficacia de la fórmula que habían creado y, por otra, aplicarla para comprobar los resultados. Porque esta es una de las particularidades del algoritmo Chudnovsky: permite detectar fallos de cálculo. Aún se utiliza, además de como un estándar para estimar π, para revisar procesos en superordenadores. Pero ¿por qué tanto ajetreo por un número? El atractivo de la misteriosa cifra radica en su naturaleza. El matemático alemán Ferdinand von Lindemann demostró en 1882 que π es un número trascendente. Este tipo de expresiones "no satisfacen ninguna ecuación finita", explica David. "π es generado por una especie de proceso infinito". A partir de mediados del siglo XX, los cálculos comenzaron a realizarse en grandes máquinas. En 1949, el científico George Reitwiesner consiguió obtener 237 decimales gracias a un ENIAC, uno de los primeros ordenadores digitales electrónicos, instalado en el Laboratorio de Investigación en Balística de Maryland. Una década después, Daniel Shanks y John Wrench, lograron unos 100 000 decimales con un ordenador IBM 7094. El número siguió creciendo durante los años 80 gracias al trabajo de otros científicos. El antecesor del récord de los Chudnovsky fue el japonés Yasumasa Kanada, que junto a su equipo estimó más de 200 millones de dígitos. "Su ordenador era miles de veces mayor que el nuestro y costó cientos de veces más, los japoneses eran líderes en hardware", asegura David. Pero, según el ucraniano, sus conocimientos matemáticos les dieron ventaja. Un logro triplicado. En 1989, los hermanos anunciaron que habían conseguido calcular 408 millones de decimales y continuaron hasta obtener más de 2000 millones en 1991, dos de los récords mundiales que han batido. Hicieron los cálculos en tres ordenadores: para los primeros, utilizaron un Cray 2 instalado en el Centro de Supercomputación de Minnesota y el IBM 3090-VF del Centro de Investigación T. J. Watson en Nueva York. En el último, usaron M zero. Así, se aseguraban de que el resultado era fiable, y comparando el tiempo que tardaban unos y otros, podían deducir la capacidad de su retoño de silicio. Cualquier "smartphone"actual tiene más potencia que nuestra M zero. "Era un sistema muy complicado, que equivaldría probablemente a unos 56 procesadores, pero era una idea totalmente diferente", explica Chudnovsky. El superordenador funcionaba con un sistema diseñado en paralelo, de manera que todos los procesadores ejecutaban simultáneamente el mismo problema, aumentando la rapidez del proceso. Aun así, tardaron varios meses en ver resultados. Hoy, M zero ha pasado a la historia. "Hace tiempo que la desmontamos", señala el matemático ucraniano. "Probablemente cualquier smartphone actual tiene más potencia que

ella". Desde entonces han construido otras máquinas, "aunque ninguna en casa", bromea. Una de ellas les sirvió para calcular 4000 millones de dígitos (su último récord). Sin embargo, para ellos el "hardware"es algo secundario. Desde su punto de vista, un ordenador no es otra cosa que "una colección de algoritmos cableados y organizados en chips de silicio". "Ni los ordenadores ni los móviles actuales existirían sin las matemáticas", asegura David.

3.6.3 Los primeros mil dígitos de π

En esta etapa nos complace hacer uso del método de Chudnovsky para computar los primeros 1000 dígitos de π que a continuación citamos:

```
Primeros 1000 dígitos de Pi con el Método de Chudnovsky
3.14159265358979323846264338327950288419716939937510582097494459
23078164062862089986280348253421170679821480865132823066470938
44609550582231725359408128481117450284102701938521105559644622 9
48954930381964428810975665933446128475648233786783165271201909 1
45648566923460348610454326648213393607260249141273724587006606 3
15588174881520920962829254091715364367892590360011330530548820 4
66521384146951941511609433057270365759591953092186117381932611 7
93105118548074462379962749567351885752724891227938183011949129 8
33673362440656643086021394946395224737190702179860943702770539 2
17176293176752384674818467669405132000568127145263560827785771 3
42757789609173637178721468440901224953430146549585371050792279 6
89258923542019956112129021960864034418159813629774771309960518 7
07211349999998372978049951059731732816096318595024459455346908 3
02642522308253344685035261931188171010003137838752886587533208 3
81420617177669147303598253490428755468731159562863882353787593 7
51957781857780532171226806613001927876611195909216420198 9
```

```
Begin time: Fri Nov  2 11:40:41 2018
End time:   Fri Nov  2 11:40:41 2018
```

Nótese que el computo no tardó ni siquiera un segundo!

El equipo usado para este cálculo es un computador de escritorio con las siguientes características:

Característica	Detalle
PC	Intel Core 2 Quad
Procesador	Core 2 Quad, 4 núcleos
CPU	Q 6600 2.4 GHz
Tecnología	64 bits
RAM	4 GBytes
ROM	520 GBytes
Sistema	Fedora 28
Kernel linux	4.18.12-200.fc28.x86_64
KDE	5.13.5
Qt	5.11.1

Table 7: Equipo de computo usado en este trabajo.

3.7 El algoritmo de Spigot

El algoritmo de Spigot tiene la particular circunstancia de calcular los dígitos de π, dígito a dígito, por lo cual no requiere el uso de tipos de datos especiales de múltiple precisión, es decir, podemos en esta ocasión prescindir del uso de la biblioteca GMP de GNU.

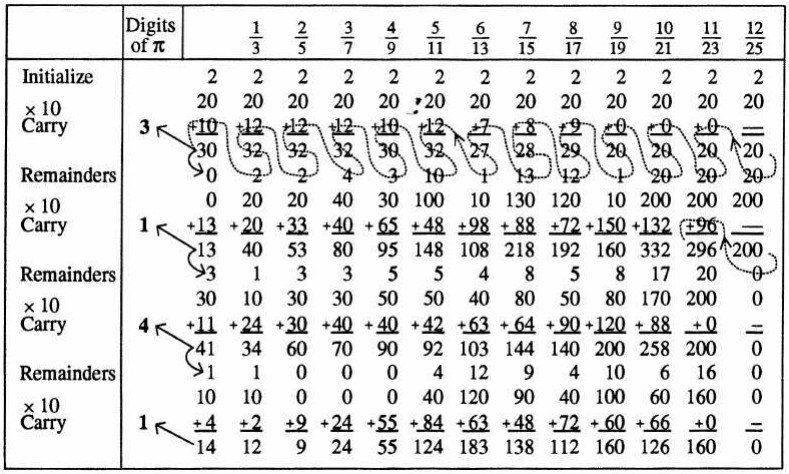

Figure 2: Resumen del método spigot.

Las líneas punteadas en la figura 2 indican los pasos a seguir, comenzando desde la derecha, las entradas son reducidas por el módulo del denominador

de la fila de encabezado (25, 23, 21, etc.), con los cocientes, después de multiplicar por el numerador (12, 11 , 10, etc.), acarreado el resultado en la columna izquierda (3, 1, 4, 1, etc).

Para más detalles consultar [16].

El algoritmo de Spigot es más eficiente que el algoritmo de los hermanos Chudnovsky, como se puede apreciar en el cuadro 8 que resume los consumos de tiempo para las dos técnicas. No obstante, como se advirtió anteriormente, el algoritmo de Spigot no es del todo seguro, en el sentido de que no entrega una fidelidad absoluta en relación con los dígitos computados.

En efecto, observemos los primeros decimales con este método a continuación:

```
03141592653589793238462643383279402884197169399375105820974944 5
```

y comparemos este resultado con el que se obtiene por el método de Chudnovsky:

```
3.14159265358979323846264338327950288419716939937510582097494 45
```

si nos fijamos bien, notaremos que hay "pequeñas"discrepancias en el resultado, en el caso citado, Spigot entrega ... 794028 ... mientras Chudnovsky arroja ... 795028 ... y sabemos, por muchas pruebas, que el resultado correcto es el que corresponde a Chudnosvky. Este detalle hace que el método de Spigot, aunque interesante, se descalifica para el ejercicio de buscar records mundiales, entre otros propósitos.

No de Dígitos	**Spigot**	**Chudnovsky**
1000	$0.1''$	$1''$
10000	$8''$	$20''$
100000	$10'54''$	$1^h36''$
1000000	$26^h42'$	N.D.

Table 8: Consumo de tiempo para los métodos de Spigot y Chudnovsky. N.D.= no determinado.

Se puede apreciar como la diferencia es grande, tanto que para el caso de un millón de dígitos, no resultaba práctico estimar el tiempo con el algoritmo de Chudnovsky, ya que en este caso, la máquina habría de estar trabajando por varios días continuos.

El software para el algoritmo Spigot es como sigue:

```
#include <stdio.h>
#include <time.h>

const int n=100000;
const int dim=((10*n)/3);

int main (void)
{
 struct tm *tm_ptr;
 time_t tmb,tme;
 int i,j,k, resto, digant, nueves, aux;
 int *pi;
 pi=new int[3500000];
 // maneja el tiempo
 (void) time(&tmb);
 tm_ptr=gmtime(&tmb);
 for (i=1;i<=dim;i++) pi[i]=2;
 nueves=0; digant=0;
 for (i=1;i<=n;i++)
 {
  resto=0;
  for (j=dim;j>=1;j--)
  {
   aux=10*pi[j]+resto*j;
   pi[j]=aux % (2*j-1);
   resto=aux/(2*j-1);
  }
 pi[1]=resto % 10;
 resto=resto/10;
 if (resto==9) nueves++;
 else if (resto==10)
 {
  printf("%i",digant);
  for (k=1;k<=nueves;k++) printf("0");
   digant=0;
   nueves=0;
  }
 else
```

```
 {
  printf("%i",digant);
  digant=resto;
  if (nueves!=0)
  {
   for(k=1;k<=nueves;k++) printf("9");
   nueves=0;
  }
 }
}
printf("Begin Hour: %02d:%02d:%02d\n",tm_ptr->tm_hour,
                        tm_ptr->tm_min,tm_ptr->tm_sec);
(void) time(&tme);
tm_ptr=gmtime(&tme);
printf("End Hour: %02d:%02d:%02d\n",tm_ptr->tm_hour,
                        tm_ptr->tm_min,tm_ptr->tm_sec);
}
```

3.8 Un millón de precisiones en torno a π

Y llegó la hora de poner a prueba llevando al límite el hardware usado en este trabajo cuyas características técnicas se pueden consultar en el cuadro 7.

A continuación hacemos, literalmente, 1 000 000 de precisiones, es decir, calculamos π con ese número de dígitos decimales. Este cálculo define el límite práctico de decimales de π que se pueden calcular usando un hardware casero. Para ir más lejos se requiere, o bien, un equipo de computo mucho más poderoso, tal como un "mainframe", o bien, un sistema de computo distribuido, que básicamente consiste en hacer que el software se ejecute en diferentes equipos comunes, un enjambre de computadores trabajando todos al unisono para un mismo objetivo.

Por obvias razones no podemos citar todos esos dígitos en el cuerpo de este documento, pero para cualquiera que resulte interesado, puede pedir el archivo de texto con el cálculo completo por la vía del correo electrónico. Citamos los primeros y los últimos dígitos de este cálculo:

```
3.14159265358979323846264338327940288419716939937510582097494459
923078164062862089986280348253421170679821480865132823066470938
```

```
4460955058223172535940812848111745028410270193852110555964462 29
  .
  .
  .
8345094730945343661590728416319368307571979806823153573715557 18
1612215678793642501388711702327555577930226678580319993081083 05
7630765233205074001393909580790163771762925928376487479017727 41
2567819055556218050487674699114083997791937654232062337471732 47
0336976335792589151526031561403332127284919441843715069655208 75
4245059895678796130331164628399634646042209010610577945815 1
```

References

[1] Bailey, D. H. *The Computation of pi to 29360000 Decimal Digit using Borwein's Quartically Convergent Algorithm.* Math. Comput. 50, 283-296, 1988.

[2] Borwein, J. M. Borwein, P. B. and Bailey, D. H. *Ramanujan, Modular Equations, and Approximations to Pi, or How to Compute One Billion Digits of Pi.* Amer. Math. Monthly 96, 201-219, 1989.

[3] Caballero, Lucía. *Los hermanos que batieron el récord de dígitos de pi con un superordenador casero.* https://adservice.google.com.co/adsid/integrator.sync.js?domain=d-4014651662326578436 6.ampproject.net

[4] Peterson, I. *Pi by the Billions.* Sci. News 156, 255, Oct. 16, 1999.

[5] Peterson, I. *A Passion for Pi.* In Mathematical Treks: From Surreal Numbers to Magic Circles. Washington, DC: Math. Assoc. Amer., 2001.

[6] Rabinowitz, S. and Wagon, S. *A Spigot Algorithm for the Digits of pi.* Amer. Math. Monthly 102, 195-203, 1995.

[7] Wrench, J. W. Jr. *The Evolution of Extended Decimal Approximations to pi.* Math. Teacher 53, 644-650, 1960.

SITIOS WEB

[8] Agencia Sinc. *Pi, el número que rebasa a las matemáticas*.http://blogs.ciencia.unam.mx/cienciamundo/2018/03/21/-pi-el-numero-que-rebasa-a-las-matematicas/feed/

[9] Borwein, J. M. *Talking About Pi.* http://www.cecm.sfu.ca/personal/-jborwein/pi_cover.html.

[10] GMP. *Computing billions of ? digits using GMP* https://gmplib.org/download/misc/gmp-chudnovsky.c

[11] Gourdon, X. and Sebah, P. *PiFast: The Fastest Program to Compute Pi.* http://numbers.computation.free.fr/Constants/PiProgram/pifast.html.

[12] Kanada, Y. *New World Record of Pi: 51.5 Billion Decimal Digits.* http://www.cecm.sfu.ca/personal/jborwein/Kanada_50b.html.

[13] Kanada, Y. *Our Latest Record.* Sep. 20, 1999. ftp://www.cc.u-tokyo.ac.jp/README.our_latest_record

[14] Kanada, Y. *Sample Digits for Decimal Digits of Pi.* Jan. 18, 2003. http://www.super-computing.org/pi-decimal_current.html.

[15] Peterson, I. *MathTrek: A Trillion Pieces of Pi.* Dec. 14, 2002. http://www.sciencenews.org/20021214/mathtrek.asp.

[16] Rabinowitz, S. & Wagon, S. *A Spigot Algorithm for the Digits of* π. The American Mathematical Monthly. 2015. 10 p.

[17] Smith, H. J. *Computing Pi.* http://www.geocities.com/-hjsmithh/Pi.html.

[18] Vélez, Ana Cristina. *La insondable sabiduría de Pí.* https://blogs-elespectador-com.cdn.ampproject.org/

[19] Wikipedia. Número π. https://es.m.wikipedia.org/w/load.php?

[20] Wikipedia, la enciclopedia libre. *Srinivasa Aiyangar Ramanujan.* https://es.m.wikipedia.org/w/load.php?

[21] Yee, A. J. *y-cruncher - A Multi-Threaded Pi-Program.* http://www.numberworld.org/y-cruncher/.

[22] *Pi Digits.* http://mathworld.wolfram.com/css/mathworldv2.css