2 Programa de la asignatura de Cálculo

El enfoque que propongo para la asignatura de Cálculo está basado en los siguientes principios:

- Respeto y fidelidad al temario aprobado en el plan de estudios de la FIUM
- Búsqueda de un enfoque propio de las "Matemáticas concretas" al estilo propuesto por D.E. Knuth y su escuela de pensamiento.
- Búsqueda de motivación permanente, basada en las aplicaciones que los contenidos de la asignatura pudieran tener en la vida profesional de los estudiantes (tanto durante sus estudios de grado y post-grado (master y doctorado, cursos de especialización, etc.) como en su vida laboral futura.
- Uso de SAGE como herramienta para la visualización y la experimentación.

El temario es el siguiente:

- Tema 1: Inducción y Polinomio de Taylor
- Tema 2: Sucesiones
- Tema 3: Integrales
- Tema 4: Series

2.1 Tema 1: Inducción y Polinomio de Taylor

2.1.1. La teoría

2.1.1.1. Inducción

En la primera clase se presenta la asignatura, explicando la metodología que se va a seguir en clase, cómo se van a coordinar los contenidos y los mecanismos de evaluación entre los distintos grupos que existen, la evaluación, las tutorías, etc., haciendo especial énfasis en que la asignatura es impartida por el departamento y por tanto involucra a varios profesores (normalmente, tres, pero depende del número de alumnos matriculados) que se coordinan y que deciden todos los aspectos relacionados con la docencia de manera conjunta. En particular, todas las pruebas que van

a calificarse se proponen entre todos los profesores que participan de la asignatura. A continuación se explica a los alumnos que existen ciertos conocimientos que se les suponen conocidos, como son los aspectos básicos del cálculo diferencial (continuidad y derivabilidad de funciones de una variable, así como el uso de la derivada para representar gráficamente las funciones y para resolver problemas sencillos de optimización). Todos estos contenidos se pueden consultar en cualquier libro de la asignatura de Matemáticas II de bachillerato. Si un alumno tiene dudas puede, por supuesto, hacer uso de las tutorías para reforzar estos conocimientos básicos y será bienvenido. En cualquier caso, pienso que se deberían organizar algunos seminarios (dos o tres por curso académico) en los que se explicaría a los alumnos que deseen asistir buena parte de las herramientas que se suponen dadas. Esto incluye, entre otros posibles temas que puedan surgir, los siguientes tópicos:

- Cálculos algebraicos sencillos -con números y con expresiones algebraicas- y algunas identidades notables, como el Teorema del Binomio y las identidades trigonométricas básicas.
- Funciones elementales: polinomios, trigonométricas -y sus inversas-, exponencial y logaritmo, funciones racionales, etc. Todas estas funciones deben ser representadas gráficamente. El uso de SAGE permitiría:
  (a) Introducir el entorno SAGE para futuros usos en la asignatura.
  (b) Visualizar de forma rápida y eficaz las funciones elementales (y sus inversas). Es importante (esencial) que un alumno sepa identificar ciertos tipos de curvas a primera vista, como las parábolas, cúbicas, los senos y cosenos, la tangente, etc.
  (c) Visualizar la relación que existe entre las propiedades de una función (crecimiento, decrecimiento, concavidad y convexidad, puntos críticos, etc.) en términos de las propiedades de sus derivadas.
- Números complejos. Incluida la identidad de Euler y el cálculo de raíces n-ésimas de la unidad (estas raíces son esenciales en muchos algoritmos que utilizan la FFT, por ejemplo).
- Construcción del algunos modelos matemáticos sencillos y resolución de problemas relacionados con ellos -en particular, problemas de optimización de funciones-

A continuación se les explica el método de inducción matemática. Concretamente, se recuerda que el conjunto $\mathbb{N} = \{1, \cdots, \}$ de los números naturales puede definirse de forma axiomática (mediante los llamados axiomas de Peano) y que uno de dichos axiomas, al cual nos referimos a partir de ahora como axioma de inducción, garantiza que si $S \subseteq \mathbb{N}$ satisface que $1 \in S$ y

$$n \in S \Rightarrow n+1 \in S,$$

entonces $S = \mathbb{N}$. Esta propiedad sirve para realizar cierto tipo de demostraciones matemáticas. Concretamente, si deseamos probar una propiedad que es posible formular como un predicado $P(n)$ que depende del parámetro n, siendo este un número natural, entonces al considerar el conjunto $S_P = \{n : P(n)\}$ de los números naturales para los que dicha propiedad es cierta, vemos que afirmar la veracidad de $P(n)$ para todo número natural n equivale a afirmar que $S_P = \mathbb{N}$, por lo que el axioma de inducción nos permite deducir que, si $1 \in S_P$ (es decir, si $P(1)$ es cierta) y $n \in S_P \Rightarrow n+1 \in S_P$ (es decir, si $P(n+1)$ se sigue de $P(n)$, sea quien sea $n \in \mathbb{N}$), entonces $S_P = \mathbb{N}$, por lo que $P(n)$ es cierto para todo $n \in \mathbb{N}$. No parece que haya un gran misterio en este tipo de demostraciones. La clave es, simplemente, que para alcanzar un número $m \in \mathbb{N}$ cualquiera -por muy grande que este sea- dando saltos de una unidad y comenzando en el número 1, siempre bastará dar una cantidad finita de saltos (exactamente, $m-1$), por lo que la iteración de aplicar la propiedad $n \in S_P \Rightarrow n+1 \in S_P$, aplicada primero al caso $n = 1$, a continuación al caso $n = 2$, etc., siempre alcanza el caso m. Lo maravilloso de esta idea es que, en vez de vernos obligados a una prueba de $P(n)$ para cada uno de los posibles valores de n -lo cual requeriría comprobar infinitos casos y, por tanto, sería una tarea imposible- los matemáticos descubrimos hace tiempo que basta cambiar ligeramente de estrategia para reducir una tarea aparentemente titánica a una tarea realmente menor: comprobar el caso $n = 1$ (lo cual obviamente debe ser generalmente sencillo) y sustituir la propiedad $P(n)$ por otra de naturaleza diferente: $P(n) \Rightarrow P(n+1)$.

En este momento alguien podría argumentar contra nuestra exposición, afirmando que si llamamos $Q(n)$ a la nueva propiedad $P(n) \Rightarrow P(n+1)$, entonces estaríamos nuevamente en la situación original: necesitamos probar $Q(n)$ para todo n. Esto es ciertamente el caso, pero en nuestra defensa hay que decir que con frecuencia una prueba directa de $Q(n)$ -que pasa por abstraerse del valor concreto de n e intentar realizar algún argumento que relacione los predicados $P(n)$ y $P(n+1)$ de un modo directo- es más sencillo que estudiar -con idéntico nivel de abstracción- directamente la propiedad $P(n)$. Y, en efecto, esto es lo que sucede en numerosos ejemplos, de los cuales debemos mostrar varios de forma inmediata. Así, por ejemplo, se exponen las demostraciones por inducción de las siguientes identidades:

- $1 + 2 + \cdots n = \frac{n(n+1)}{2}$
- $1 + r + \cdots + r^n = \frac{r^{n+1}-1}{r-1}$ (para $r \neq 1$)
- $1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$
- $1^3 + 2^3 + \cdots + n^3 = \frac{n^2(n+1)^2}{4} = (1 + 2 + 3 + \cdots + n)^2$

A continuación, se proponen diversos ejercicios sencillos en los que el alumno podrá practicar este método de demostración. Existen diversas razones para justificar la introducción del método de inducción a estos alumnos. Una de ellas es que en el estudio de problemas de complejidad de algoritmos es frecuentemente necesario establecer una fórmula para el coste computacional de un algoritmo o incluso de un problema (son conceptos distintos, de hecho) y la demostración de dicha fórmula -que generalmente depende del tamaño n de los datos introducidos en el algoritmo- pasa con frecuencia por utilizar el método de inducción. Más adelante, cuando se haya avanzado un poco más en la asignatura (concretamente, en el tema dedicado a las sucesiones) se pondrán algunos ejemplos concretos de este tipo de situaciones. En este momento se da solo una explicación genérica de este hecho, dedicada fundamentalmente a motivar a los alumnos, y se les avisa de que habrá ejemplos concretos en el futuro que se explicarán en detalle.
Lo cierto es que el método de demostración por inducción es extremadamente útil en ciencias de la computación. De hecho, en su libro [38] sobre "Matemáticas para las ciencias de la computación", los profesores E. Lehman y T. Leighton llegaron a afirmar que "El método de inducción es, con diferencia, la técnica de demostración más importante que existe en ciencias de la computación". Una de las razones podría ser que en numerosos procesos en los que es necesario hacer algún tipo de recuento (de los cuales el cálculo del costo de un algoritmo es solo un caso muy particular) el establecimiento firme de la correspondiente fórmula pasa por utilizar el método de inducción. Otra razón podría ser que a menudo es necesario, para probar que un determinado algoritmo funciona adecuadamente, demostrar que cierta propiedad se mantiene invariante durante la aplicación del mismo. Esto se puede formular, evidentemente, como una propiedad $P(n)$ en la que n representa el n-ésimo paso realizado por el algoritmo y, en consecuencia, resulta natural intentar demostrar esta propiedad por inducción sobre el número de pasos realizados por el algoritmo. Esto se aplica también al estudio de determinados juegos matemáticos, que obviamente pueden ser programados. Por ejemplo, es bien conocido que en el Juego del 15, que consiste en mover horizontalmente y verticalmente las piezas de un puzzle de tamaño 4×4 al que se ha quitado una pieza para permitir el movimiento, existen ciertas posiciones imposibles. Concretamente, se sabe que en dicho juego (si etiquetamos cada casilla móvil con un número que va del 1 al 15 y el lugar vacío, que permite el movimiento de las piezas, con un asterisco $*$) es imposible pasar de la posición

$$\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & * \end{array}$$

a la posición

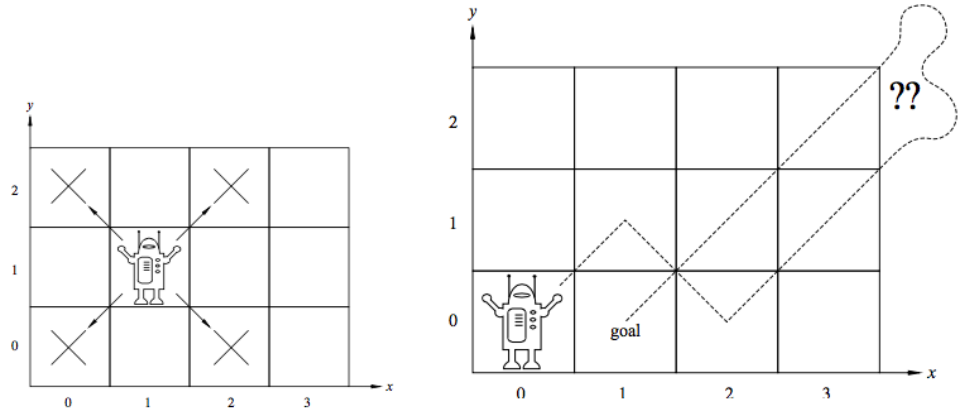
$$\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 15 & 14 & * \end{array}$$

Lo que no es -quizás- tan conocido es que la prueba de esta afirmación se puede realizar por inducción sobre el número de movimientos realizados a partir de la posición inicial, y que la clave para aplicar el método de inducción en este problema es observar que existe una propiedad que permanece invariante cuando se avanza en el juego, pero la posición final sugerida no posee dicha propiedad.
Una manera quizás más formal de explicar esta idea es a través de las máquinas de estados, que no es otra cosa que un conjunto (llamado conjunto de estados) E dotado de una cierta relación binaria $R \subseteq E \times E$. Así, si p, q son dos estados, denotamos por $p \to q$ la propiedad $(p, q) \in R$ y decimos que la máquina puede pasar del estado p al estado q. También escribimos $(E, \to)$ para denotar la máquina de estados anterior. En esencia, los algoritmos se pueden representar utilizando máquinas de estados y el principio de inducción se puede reformular en este contexto del siguiente modo:
Dada una máquina de estados $(E, \to)$, decimos que disponemos de una ejecución de la misma si conocemos una sucesión de estados (posiblemente infinita) $S = \{q_0, q_1, \cdots\}$ tales que, si $q_n, q_{n+1} \in S$ entonces $q_n \to q_{n+1}$. Un estado $q \in E$ se dice alcanzable a partir del "estado inicial" q_0 si existe una ejecución $S = \{q_0, q_1, \cdots\}$ de la máquina tal que $q \in S$. Finalmente, decimos que una propiedad P de los estados de una máquina $(E, \to)$ es invariante si del hecho de que q posea dicha propiedad y $q \to r$ se deduce que también r posee dicha propiedad. El principio de inducción puede entonces formularse como el siguiente principio de invarianza:

Teorema 2.1 (Principio de invarianza)

Supongamos que $(E, \to)$ es una máquina de estados y P es un invariante para la máquina. Si $q_0 \in E$ verifica P entonces todos los estados alcanzables a partir de q_0 también satisfacen P.

Este principio fue formulado por R. W. Floyd (1936-2001) en un influyente artículo de 1967 que llevaba el sugerente título "Asignando significados a los programas". Floyd lo utilizó de forma muy inteligente (y persistente) en el area de verificación de programas, trabajo por el cual recibió el Premio Turing (equivalente en informática del premio Nobel) en 1978. Un ejemplo muy sencillo, pero ilustrativo, de la aplicación del principio de invarianza es la prueba de que si un robot que salta aleatoriamente de la posición (n, m) a alguna de las posiciones $\{(n+1, m+1), (n-1, m-1), (n-1, m+1), (n-1, m-1)\}$ comienza en el origen de coordenadas, entonces nunca llegará al punto $(1, 0)$.



La demostración es tan sencilla como observar que si (n, m) es el estado inicial del robot y $(n, m) \to (a, b)$, entonces los números $n + m$ y $a + b$ son necesariamente ambos pares o ambos impares, pues

$$a + b \in \{n + m + 2, n + m - 2, n + m\}.$$

Como el estado inicial asumido es $(0,0)$, que arroja un valor $0+0=0$ que es par, el estado $(1,0)$ nunca será alcanzado por el robot, pues $1+0=1$ es impar. Obviamente, esta misma prueba sirve para demostrar que el robot nunca alcanzará ninguno de los puntos $(1,2),(-3,10),(9,24),(10123,-2300)$, etc. Es más, esta información podría usarse para, en el diseño de un juego, colocar dos objetos móviles en posiciones iniciales $(0,0)$ y $(-3,10)$ (por ejemplo) y dejar que el juego avance según las reglas explicadas inicialmente, para ambos objetos, con la tranquilidad plena de que nunca colisionarán. El resultado mencionado sobre el Juego del 15 es también consecuencia del principio de invarianza de Floyd, aunque el invariante concreto asociado es más complejo de definir.
Otro contexto fundamental en el que la inducción matemática es una herramienta esencial es la teoría de la recursividad, que es una parte esencial de la teoría de la computación y sirve para construir todo tipo de objetos (y algoritmos) basados en procesos recursivos, así como para estudiar las propiedades de los objetos así construidos. Vale la pena explicar brevemente algunas de estas ideas a los alumnos porque con ellas se sentirán motivados para asimilar la técnica de demostración por inducción, valorándola como algo que ciertamente necesitarán en el futuro.

2.1.1.2. Polinomio de Taylor

En el siguiente item se explica la construcción de los polinomios de Taylor de una función (suficientemente derivable) en un punto. Concretamente, se explica que la fórmula

$$P_{f,x_0,N}(x)=\sum_{k=0}^{N}\frac{f^{(k)}(x_0)}{k!}(x-x_0)^k$$

determina al único polinomio $P(x)=a_0+a_1+\cdots+a_Nx^N$ de grado $\leq N$ que verifica la siguiente cadena de igualdades:

$$P^{(k)}(x_0)=f^{(k)}(x_0);\ \text{para } k=0,1,\cdots,N.$$

Que estas igualdades identifican a un único polinomio es un cálculo sencillo de probar. Lo interesante (y, de hecho, lo sorprendente y el aspecto sobre el que se incide más en el aula) es que este polinomio realmente resulta útil, en una amplia gama de casos, para aproximar a la función f en las cercanías del punto x_0. De hecho, se observa que $P_{f,_{x0},1}(x)=f(x_0)+f'(x_0)(x-x_0)$ es la ecuación de la recta tangente al gráfico descrito por la función f en el punto $(x_0,f(x_0))$ (aquí se aprovecha para recordar a los alumnos el concepto de derivada y el cálculo de la recta tangente a una curva en un punto) y a continuación se muestra como los polinomios de Taylor de orden superior van aproximándose cada vez mejor a la curva que describe la función $f(x)$, lo cual se comprueba para varias elecciones de $f(x)$ y, en particular, para las funciones e^x, $\sin x$, $\cos x$, $\sqrt{x}$ y $\arctan x$. Además, estos cálculos se hacen en pizarra -para que el alumno aprenda a calcular a mano el polinomio de Taylor de una función- y en SAGE -para visualizar el resultado.
A continuación se explica la fórmula de acotación del error

$$|f(x)-P_{f,x_0,N}(x)|\leq\frac{\|f^{(N+1)}\|_{[x_0,x]}}{(N+1)!}|x-x_0|^{N+1}$$

y se muestran varios ejemplos en los que el miembro de la derecha de la desigualdad anterior puede acotarse superiormente por un número tan pequeño como se desee si se elige N suficientemente elevado.
Se observa que a menudo es fundamental utilizar el método de inducción para el cálculo de una expresión explícita de $f^{(n)}(x)$, lo cual es un paso necesario para hallar el término general $\frac{f^{(k)}(x_0)}{k!}(x-x_0)^k$ del polinomio de Taylor y, posteriormente, para estimar $\|f^{(N+1)}\|_{[x_0,x]}$ en la fórmula del error. (Y esta era una buena razón para explicar la inducción en el apartado anterior).
Por otra parte, se explica que los polinomios de Taylor sirven, en informática, para la construcción de una librería permanente de funciones. Esto se logra combinando las propiedades específicas que tienen las funciones elementales, como sus simetrías o propiedades de adición, etc., para reducir el cálculo de las mismas a conocerlas en un intervalo

finito concreto, donde éstas son sustituidas por un polinomio de Taylor de grado adecuado que garantice, a lo largo de todo el intervalo, un número de cifras significativas prefijado. Todos estos cálculos se realizan una sola vez y posteriormente se guarda un algoritmo que los ejecuta cada vez que llamamos a la función elegida.

2.1.2. Las prácticas

En las clases de prácticas se mezcla la explicación de problemas en pizarra con el uso de SAGE para resolver problemas de todo tipo. Así, por ejemplo, a la vez que mostramos cómo se realiza la prueba por inducción de una identidad como

$$1^3 + 2^3 + \cdots + n^3 = \frac{n^2(n+1)^2}{4} = (1 + 2 + 3 + \cdots + n)^2,$$

lo cual conduce a demostrar que

$$1^3 + 2^3 + \cdots + n^3 + (n+1)^3 = \frac{n^2(n+1)^2}{4} + (n+1)^3 = \frac{(n+1)^2(n+2)^2}{4}$$

(y esto se logra realizando las operaciones indicadas en la suma de fracciones $\frac{n^2(n+1)^2}{4} + (n+1)^3$ y en la fracción $\frac{(n+1)^2(n+2)^2}{4}$ y comprobando que ambas son iguales (coinciden con $\frac{1}{4}n^4 + \frac{3}{2}n^3 + \frac{13}{4}n^2 + 3n + 1$)), los mismos cálculos se comprueban con SAGE mediante la siguiente secuencia de órdenes:

```
sage: num=var('num')
sage: (num^2*(num+1)^2/4+(num+1)^3-(num+1)^2*((num+1)+1)^2/4).is_zero()
True
```

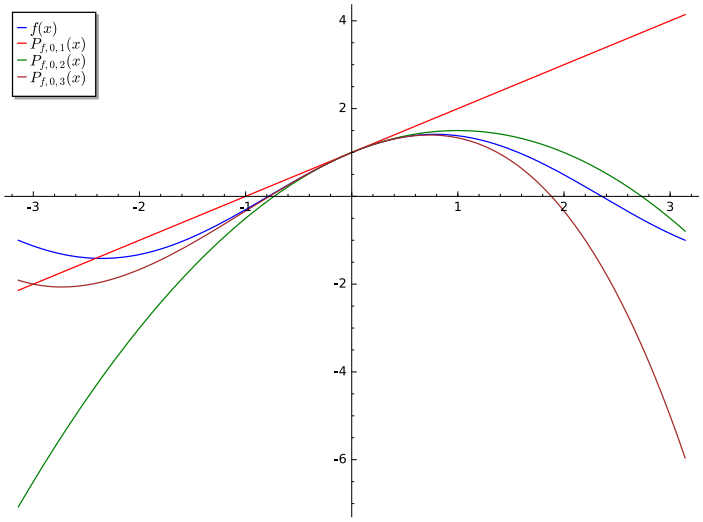
o bien con las órdenes:

```
sage: num=var('num')
sage: (num^2*(num+1)^2/4+(num+1)^3).expand()
1/4*num^4 + 3/2*num^3 + 13/4*num^2 + 3*num + 1
sage: ((num+1)^2*((num+1)+1)^2/4).expand()
1/4*num^4 + 3/2*num^3 + 13/4*num^2 + 3*num + 1
```

Para hallar el polinomio de Taylor de una función -como, por ejemplo, $f(x) = \sin(x) + \cos(x)$- así como para dibujar ambas funciones en el mismo gráfico, se usan las siguientes órdenes (aquí hemos dibujado los polinomios de Taylor de orden 1,2 y 3, cada uno de un color):

```
sage: f(x)=sin(x)+cos(x)
sage: p1(x)=taylor(f(x),x,0,1)
sage: p2(x)=taylor(f(x),x,0,2)
sage: p3(x)=taylor(f(x),x,0,3)
sage: plot(f(x),(x,-pi,pi),color='blue',legend_label='$f(x)$')+plot(p1(x),(x,-pi,pi)
    ,color='red',legend_label='$P_{f,0,1}(x)$')+plot(p2(x),(x,-pi,pi),color='green',
    legend_label='$P_{f,0,2}(x)$')+plot(p3(x),(x,-pi,pi),color='brown',legend_label='
    $P_{f,0,3}(x)$')
Graphics object consisting of 4 graphics primitives
```

Lo cual da lugar a la siguiente gráfica:



Para decidir el grado del polinomio de Taylor que garantice que $|f(x) - P_{f,x_0,N}(x)| \leq 10^{-m}$ para cierto valor de m y cierto punto x, se plantea la desigualdad

$$\frac{M_n}{(n+1)!}|x - x_0|^{n+1} \leq 10^{-m},$$

donde M_n es una cota superior de la función $|f^{(n+1)}|$ en el intervalo $[x_0, x]$. Por ejemplo, si $f(x) = \sin(x) + \cos(x)$ resulta evidente que podemos tomar $M_n = 2$ para todo x, x_0 y todo n. En tal caso, la desigualdad adquiere la forma

$$\frac{2}{(n+1)!}|x - x_0|^{n+1} \leq 10^{-m}$$

Tomando logaritmos a ambos lados y despejando, llegamos a la siguiente expresión equivalente:

$$\log_{10}(n+1)! - (n+1)\log_{10}|x - x_0| \geq m + \log_{10} 2$$

Así, por ejemplo, si $x = 0{,}1$ y $x_0 = 0$, $m = 15$, debemos encontrar el valor de n para el cual

$$\log_{10}(n+1)! - (n+1)\log_{10} 0{,}1 \geq 15 + \log_{10} 2$$

Esto se logra de forma sencilla con SAGE: Primero hallamos el valor $15 + \log_{10} 2$

```
sage: (15+log(2,10)).n()
15.3010299956640
```

y a continuación hacemos un bucle que nos permita conocer el valor de m para el cual la desigualdad es cierta. Esto se puede hacer con la orden

```
sage: for m in range(1,20): print 'Para m=',m, ', log((m+1)!)-(m+1)*log0.1 =', (log(
    factorial(m+1),10)-(m+1)*log(0.1,10)).n()
```

o con la orden

```
sage: m=1;
sage: while log(factorial(m+1),10)-(m+1)*log(0.1,10)< 15.3010299956640: m=m+1
sage: m
```

```
9                                                                                21
```

En ambos casos, el valor arrojado es $m = 9$.

2.2 Tema 2: Sucesiones

2.2.1. La teoría

2.2.1.1. Conceptos generales. Sucesiones recurrentes. Límites. Propiedades sencillas de las sucesiones

Este tema se inicia en la segunda semana de clases. Lo primero que se hace es definir una sucesión como una aplicación $a : \mathbb{N} \to \mathbb{R}$ (aunque aquí cabría usar también $\mathbb{C}$, $\mathbb{R}^n$, etc. como conjuntos imagen, si la sucesión que se va a considerar está formada por números complejos, vectores, etc.) y explicar la notación usual: $a_n = a(n)$ denota el término general de la sucesión y ésta se denota generalmente como $\{a_n\}$ o incluso (a_n). Las sucesiones de números reales aparecen con frecuencia cuando buscamos resolver de forma aproximada todo tipo de problemas -cuya solución exacta puede ser difícil de hallar por diversos motivos-. Por ejemplo, más adelante explicaremos varios métodos para encontrar de forma aproximada la solución de una ecuación, y dichos métodos se basan en la construcción de ciertas sucesiones que "convergen" al número buscado, de modo que a menudo podemos dar tantas cifras significativas del mismo como deseemos. Otras veces no se persigue resolver una ecuación sino que se busca un valor óptimo para cierta magnitud. Esto es especialmente relevante en las aplicaciones en informática porque, como se comentó en el capítulo anterior, la búsqueda de máximos y/o mínimos es parte esencial de teorías tan importantes como el Machine Learning o la Inteligencia Artificial, entre otras. Aunque típicamente se da especial importancia a la convergencia de sucesiones (y su uso para demostrar la continuidad de funciones), en los estudios de informática son también fundamentales las sucesiones divergentes porque a menudo ellas representan en coste computacional (o en tiempo) de un algoritmo y, en tales casos, resulta de interés comparar las diferentes velocidades a las que estas sucesiones divergen porque ellas representan la velocidad a la que un problema cuya solución depende de n datos se va alejando de ser resoluble computacionalmente. Así, se recuerda en clase que en Algorítmica (o en Teoría de la computación) llamarán "función costo" a lo que nosotros hemos llamado "sucesión". Más adelante se introducirán ejemplos y se expondrá el lenguaje básico usado en este contexto. Otra motivación fundamental para el estudio de las sucesiones es que la digitalización del mundo pasa precisamente por sustituir objetos analógicos como las funciones continuas $f(t)$ por discretizaciones suyas, que se obtienen simplemente tomando muestras de las mismas con cierto periodo de muestreo T, lo cual significa que cambiamos la función analógica $f(t)$ por su versión discreta, $\{f(nT)\}$, que es una sucesión. Resulta sorprendente que esto se pueda hacer sin perder esencialmente información (si el periodo de muestreo es un número suficientemente pequeño), pero existen razones profundas para ello -matemáticas y físicas-. Concretamente esto es consecuencia del famoso teorema del muestreo de Shannon, Whitaker y Kotelnikov. A los alumnos se les explica el hecho en sí -no la demostración- y este se relaciona con la capacidad que tienen los ordenadores de, analizando datos discretos como las sucesiones, guardar toda la información relevante en numerosas aplicaciones y además obtener a partir de estos datos digitalizados todo tipo de aplicaciones en las distintas areas del conocimiento. Otro contexto en el que aparecen sucesiones como objeto fundamental de trabajo es el estudio de las series temporales -como la evolución de precios, datos poblacionales, e incluso las muestras tomadas en estudios clínicos de todo tipo-.

Una vez se ha introducido el concepto de sucesión y se han propuesto varias motivaciones para su estudio, se introducen varios ejemplos y, entre ellos, se explica también el concepto de sucesión recurrente. Las sucesiones recurrentes

que van a trabajar son, generalmente, del tipo

$$\left\{\begin{array}{lcl} a_0 & = & \alpha \\ a_{n+1} & = & f(a_n) \end{array}\right.$$

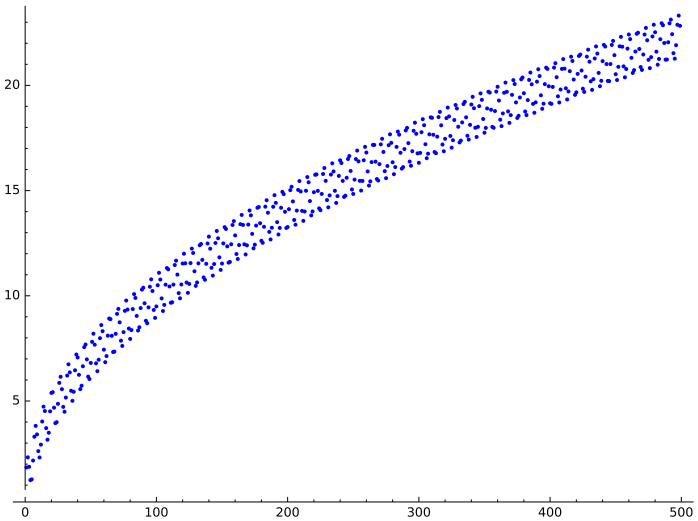
aunque, evidentemente, se explica que pueden construirse sucesiones recurrentes más generales, como

$$\left\{\begin{array}{lcll} a_k & = & \alpha_k & k = 0, 1, \cdots, s \\ a_{n+s} & = & f(a_n, a_{n+1}, \cdots, a_{n+s-1}) & \end{array}\right.$$

y que en el contexto de la algorítmica es frecuente encontrar sucesiones "recurrentes" del tipo:

$$\left\{\begin{array}{lcl} a_1 & = & \alpha \\ a_n & = & f(a_{n/2}, n) \end{array}\right.$$

(Por ejemplo, la táctica de "divide y vencerás", tan útil para mejorar la eficiencia de muchos algoritmos, da con frecuencia lugar a considerar sucesiones del tipo anterior con $f(x, n) = \alpha x + \beta n$).
Los primeros ejemplos de sucesiones que se muestran son, evidentemente, expresados con el término general y se visualizan diversos comportamientos. Por ejemplo, $a_n = 1/n$, $a_n = 1 - 1/n$, $a_n = \sin(n)/n$, $a_n = \log n$, $a_n = n^2$, $a_n = (-1)^n/\sqrt{n+1}$, etc., son todas sucesiones que tienen un comportamiento bien definido cuando $n \to \infty$ en el sentido de que o bien convergen a un cierto número real o bien divergen hacia infinito. Por otra parte, sucesiones como $(-1)^n$ o $\sqrt{n}+(-1)^n n$ tienen un comportamiento "bipolar" -se descomponen en dos sucesiones que "convergen" hacia sitios diferentes- Se muestra también el ejemplo $a_n = \sqrt{n} + \sin n$ para que observen cómo una fórmula sencilla puede producir un comportamiento bastante "indomable".



Con estos ejemplos a mano se motiva la necesidad de un concepto para "sucesión convergente", otro para "sucesión divergente" y otros dos para "sucesión divergente hacia ∞" y "sucesión divergente hacia $-\infty$". Todos ellos se introducen formalmente y todos ellos se visualizan en el plano.

Definición 2.1

Se dice que la sucesión (a_n) converge a $L \in \mathbb{R}$, lo cual se denota por $\lim a_n = L$, si, para cada $\varepsilon > 0$ existe un $N_0 = N_0(\varepsilon) \in \mathbb{N}$ tal que $|a_n - L| < \varepsilon$ siempre que $n \geq N_0$. Se dice que la sucesión (de números reales) (a_n) es convergente si converge a L para algún $L \in \mathbb{R}$. Se dice que la sucesión (de números reales) (a_n) es divergente si no es convergente.

Definición 2.2

Se dice que la sucesión (a_n) diverge a ∞, lo cual se denota por $\lim a_n = \infty$, si, para cada $M > 0$ existe un $N_0 = N_0(M) \in \mathbb{N}$ tal que $a_n > M$ siempre que $n \geq N_0$.

Definición 2.3

Se dice que la sucesión (a_n) diverge a $-\infty$, lo cual se denota por $\lim a_n = -\infty$, si, para cada $M < 0$ existe un $N_0 = N_0(M) \in \mathbb{N}$ tal que $a_n < M$ siempre que $n \geq N_0$.

Una vez se han introducido estos conceptos se estudian rigurosamente varios ejemplos de cálculo de límites. Los primeros ejemplos se hacen directamente a partir de la definición para mostrar cómo esta técnica conduce rápidamente a cálculos difíciles que involucran un buen manejo de desigualdades clásicas, como la desigualdad triangular. Conviene, pues, disponer de herramientas matemáticas que faciliten el cálculo de límites y nos ayuden a desembarazarnos de este tipo de trabajo, que puede llegar a ser inabordable. Esta idea justifica la formulación explícita de las propiedades elementales de los límites:

- Límite de la suma: Si $\lim a_n = L$ y $\lim b_n = S$, entonces $\lim(a_n + b_n) = L + S$.
- Límite del producto: Si $\lim a_n = L$ y $\lim b_n = S$, entonces $\lim(a_n b_n) = LS$.
- Límite del cociente: Si $\lim a_n = L$ y $\lim b_n = S \neq 0$, entonces $\lim \frac{a_n}{b_n} = \frac{L}{S}$.
- Límites y desigualdades: Si (a_n), (b_n) son sucesiones convergentes y $a_n \leq b_n$ para todo n, entonces $\lim a_n \leq \lim b_n$
- Producto de una sucesión nula y otra acotada: Si $\lim a_n = 0$ y $|b_n| \leq M$ para cierta constante $M > 0$ y todo n, entonces $\lim(a_n b_n) = 0$.
- Propiedad del Sandwich: Si $a_n \leq b_n \leq c_n$ para todo n y, además, $\lim a_n = \lim c_n = L$, entonces $\lim b_n = L$ (Aquí también vale $L \in \{+\infty, -\infty\}$)

Mediante el uso de las propiedades anteriores se puede calcular el límite de muchas sucesiones. Sin embargo, conviene disponer de algunos resultados más que nos permitan abordar el límite de ciertas sucesiones cuyo estudio se escapa a la capacidad de cálculo que proporcionan dichas propiedades. En particular, para el cálculo de límites de sucesiones para las que se conoce el término general a_n conviene disponer de la regla de L'Hospital,

Teorema 2.2 (Regla de L'Hospital)

Supongamos que $\lim_{x\to\infty} f(x) = \lim_{x\to\infty} g(x) = \alpha \in \{0, \infty, -\infty\}$. Si existe $\lim_{x\to\infty} \frac{f'(x)}{g'(x)} = L$, entonces también existe $\lim_{x\to\infty} \frac{f(x)}{g(x)} = L$.

y para sucesiones dadas en forma recurrente son importantes los siguientes resultados:

Teorema 2.3 (Sucesiones monótonas)

Si (a_n) es una sucesión (de números reales) monótona y acotada, entonces existe $L \in \mathbb{R}$ tal que $\lim a_n = L$. Además, si la sucesión (a_n) es monótona no acotada, entonces $\lim a_n = +\infty$ si la sucesión es creciente y $\lim a_n = -\infty$ si la sucesión es decreciente.

Teorema 2.4 (Del punto fijo, Banach)

Supongamos que

$$\begin{cases} a_0 & = & c \\ a_{n+1} & = & f(a_n) \quad (n \geq 0) \end{cases}$$

Si $f : [a, b] \to [a, b]$ es continua y su primera derivada es tal que $|f'(t)| \leq K < 1$ para todo $t \in [a, b]$, entonces:

- $f(x) = x$ tiene una única solución $\alpha \in [a, b]$
- $\lim a_n = \alpha$

Además,

$$|a_n - \alpha| \leq K^n \operatorname{máx}\{|a - c|, |b - c|\} \ \ (n \in \mathbb{N}).$$

Se exponen diversos ejemplos a los que se pueden aplicar los resultados anteriores. Entre los mismos cabría destacar los siguientes:

- Si $a_n = \frac{\log n}{n}$ entonces $a_n = f(n)/g(n)$ con $f(x) = \log x$, $g(x) = x$. Aplicando L'Hospital a f/g, obtenemos que $\lim_{x\to\infty} \frac{\log x}{x} = \lim_{x\to\infty} \frac{1/x}{1} = 0$, de modo que $\lim_{n\to\infty} \frac{\log n}{n} = 0$.
- Si $a_n = r^n$ y $0 < r < 1$ entonces $a_n > 0$ para todo n, $a_{n+1} = ra_n < a_n$ para todo n. Por tanto a_n decrece y es una sucesión acotada, por lo que tiene límite. Además, si $L = \lim a_n$ entonces $L = \lim a_{n+1} = \lim ra_n = r \lim a_n = rL$, por lo que $(1 - r)L = 0$ y, por tanto, $L = 0$. Obsérvese que en este caso también podríamos haber aplicado el teorema del punto fijo de Banach para demostrar que $\lim r^n = 0$ cuando $0 < r < 1$.
- Si $a_n = r^n$ y $1 < r$ entonces $a_{n+1} = ra_n > a_n$ para todo n. Por tanto a_n es creciente. Veamos que no es una sucesión acotada, por lo que tiene límite, que vale ∞. En efecto, $a_{n+1} - a_n = (r - 1)a_n \geq (r - 1)a_1 = (r - 1)r = c > 0$, por lo que

  $$a_n = (a_n - a_{n-1}) + (a_{n-1} - a_{n-2}) + \cdots + (a_2 - a_1) + a_1 > (n - 1)c + a_1 = (n - 1)c + r,$$

  que es una cantidad que tiende a ∞ porque $c > 0$.

Sin entrar en demasiados detalles sobre la demostración del teorema del punto fijo de Banach (también llamado teorema de la aplicación contractiva), vale la pena explicar el significado de las hipótesis del mismo. Concretamente, se recuerda el teorema del valor medio, que aplicado a la función f del teorema del punto fijo, nos dice que si $x, y \in [a, b]$ entonces $\frac{f(x)-f(y)}{x-y} = f'(\xi)$ para cierto $\xi \in [x, y]$ y, por tanto,

$$|f(x) - f(y)| = |f'(\xi)||x - y| < K|x - y|.$$

Es decir, la hipótesis $|f'(t)| < K < 1$ para todo $t \in [a, b]$ nos garantiza que la función f acerca los puntos del intervalo $[a, b]$ (es decir, la distancia entre $f(x)$ y $f(y)$ es menor que la distancia entre x e y y la razón con la que dicha distancia se ve recortada es precisamente la constante K). Es por esto que llamamos "contractiva" a toda función $f : [a, b] \to [a, b]$ que verifique $\sup_{x\in[a,b]} |f'(x)| = K < 1$. Obviamente, la iteración de la función f a ambos puntos tendrá entonces la propiedad de atraer las imágenes cada vez más. En particular, si $\alpha \in [a, b]$ satisface $f(\alpha) = \alpha$, entonces para cualquier otro valor $c \in [a, b]$ tendremos que

$$|\alpha - a_n| = |f(\alpha) - f(a_{n-1})| < K|\alpha - a_{n-1}| < K^2|\alpha - a_{n-2}| < \cdots K^{n-1}|\alpha - a_1| < K^n|\alpha - c| \leq K^n \,\text{máx}\{|a - c|, |b - c|\}$$

Esta es la clave del teorema en cuestión.

N Obsérvese que podemos deshacernos de la constante $\text{máx}\{|a - c|, |b - c|\}$ simplemente obteniendo una estimación de $|\alpha - c|$ que no dependa del tamaño del intervalo $[a, b]$. Concretamente,

$$|\alpha - c| = |\alpha - f(c)| + |f(c) - c| = |f(\alpha) - f(c)| + |f(c) - c| \leq K|\alpha - c| + |f(c) - c|$$

de modo que

$$|\alpha - c| \leq \frac{1}{1 - K}|f(c) - c|$$

y, por tanto,

$$|\alpha - a_n| \leq \frac{K^n}{1 - K}|f(c) - c|$$

Usamos SAGE (aunque en clase de teoría, a menudo basta realizar un boceto en pizarra) para visualizar la construcción de sucesiones recurrentes del tipo $a_{n+1} = f(a_n)$ mediante la implementación de un algoritmo que dibuja la poligonal

$$(a_0, 0), (a_0, a_1), (a_1, a_1), (a_1, a_2), \cdots, (a_{n-1}, a_{n-1}), (a_{n-1}, a_n), (a_n, a_n), \cdots$$

superpuesta a los gráficos de la función $f(x)$ y la recta bisectriz $y = x$. Por ejemplo, para visualizar la sucesión

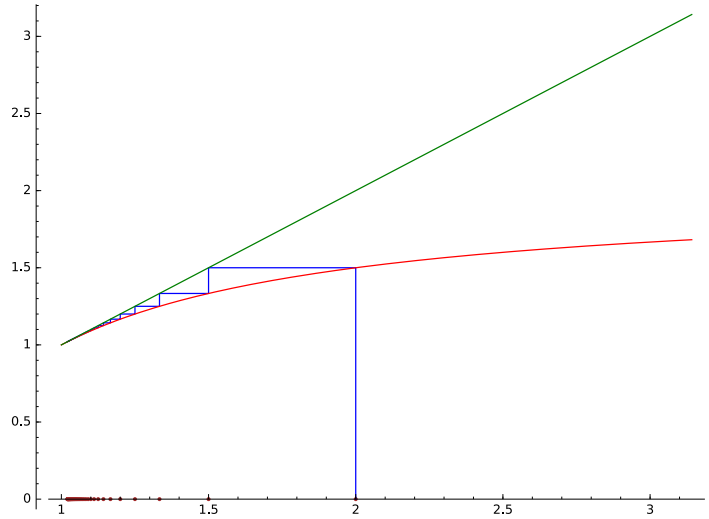
$$\begin{cases} a_1 &= 2 \\ a_{n+1} &= 2 - \frac{1}{a_n} \end{cases}$$

que se obtiene de iterar la función racional $f(x) = 2 - 1/x$, construimos el siguiente programa:

```
def a(n):
      if n==1:
         return 2
      else:
         return 2-1/a(n-1)
p=[(a(1),0)]
for k in [1..50]:
```

```
    p=p+[(a(k),a(k+1))]+[(a(k+1),a(k+1))]
line(p)+plot(2-1/x,(x,1,pi),color='red')+plot(x,(x,1,pi),color='green') \
 + list_plot([(a(n),0) for n in range(1,50)],color='brown')
```

cuya salida es el gráfico:



En este caso concreto el gráfico nos muestra que la sucesión (a_n) es monótona decreciente y acotada, por lo que converge. Obviamente, la visualización solo es una ayuda y en clase se exponen los argumentos concretos que demuestran que esta sucesión tiene las propiedades que se acaban de mencionar. Sin embargo, en este punto concreto se insiste en mostrar a los alumnos que la ayuda de un software científico como SAGE permite realizar una conjetura certera sobre la situación matemática que se está estudiando, lo cual a menudo allana el camino hacia una demostración formal. De hecho, el software científico se usa frecuentemente en investigación matemática para "sondear"la verdad que se persigue por la vía de la experimentación.

2.2.1.2. Solución numérica de ecuaciones. Teorema de Bolzano. Método de Newton

Como era de esperar, para funciones continuas $f(x)$, la búsqueda de límites para sucesiones recurrentes del tipo $a_{n+1} = f(a_n)$ está íntimamente ligada a la búsqueda de los puntos fijos de la función $f(x)$ por la sencilla razón de que si $\alpha = \lim a_n$ entonces

$$\alpha = \lim a_{n+1} = \lim f(a_n) = f(\lim a_n) = f(\alpha).$$

El problema de encontrar los puntos fijos de una función continua es, de hecho, equivalente a la búsqueda de las soluciones de ecuaciones $g(x) = 0$ con $g(x)$ continua. Para verlo, basta observar que $f(x) = x$ si y solo si $f(x) - x = 0$, que es una ecuación del tipo $g(x) = 0$ con $g(x) = f(x) - x$. Además, si $h(x) = 0$ para cierta función $h(x)$ entonces tendremos que $f(x) = x$ cuando $f(x) = h(x) + x$. Las ecuaciones son, además, uno de los temas más antiguos de las matemáticas y, probablemente también uno de los más importantes. En el contexto de la informática es frecuentemente necesario localizar los puntos extremos (máximos y mínimos) de una función, uno de los cuales se tomará como nueva entrada (mejorada respecto de algún dato obtenido previamente o proporcionado por el algoritmo) a un algoritmo que resuelve algún problema concreto, y esto -la localización de los puntos donde se alcanza un valor extremal- suele traducirse en la resolución de una o varias ecuaciones. Vale, por tanto, la pena

añadir a lo expuesto hasta ahora algún método que sirva para resolver ecuaciones y sea aplicable a un abanico de situaciones lo más amplio posible.
Como la solución exacta de una ecuación se va a obtener sólo en contadas ocasiones (por ejemplo, sabemos de la imposibilidad de resolver la ecuación general (algebraica) de grado 5 o superior, por no hablar de ecuaciones que mezclan funciones algebraicas con funciones trascendentes, como $x + \sin(x) = 0$, entre otras muchas situaciones que se pueden encontrar con cierta facilidad en la práctica), el problema de resolver ecuaciones se plantea como una cuestión numérica. Se trata de probar, por un lado, que cierta ecuación se puede resolver -y a ser posible, detectar cuántas soluciones admite y una cierta zona que delimite unívocamente cada una de ellas- y, por otro lado, encontrar cada una de las soluciones de la ecuación con un grado de precisión tan elevado como se desee. Esto se logra construyendo una sucesión que converge a la solución, así como algún tipo de estimación del error que se comete cuando se toma uno de los elementos de la sucesión. Una vez se ha motivado este problema concreto, explicamos a los alumnos dos resultados que sirven para resolver ecuaciones con el grado de precisión que se desee. Concretamente, exponemos la prueba del teorema de Bolzano basada en el uso del método de bisección y, para ecuaciones definidas a partir de funciones diferenciables, exponemos el método de Newton.
Concretamente, se enuncia el teorema de Bolzano:

Teorema 2.5 (Bolzano)

Supongamos que $f : [a, b] \to \mathbb{R}$ es una función continua y $f(a)f(b) < 0$. Entonces existe al menos un valor $\alpha \in]a, b[$ tal que $f(\alpha) = 0$.

La prueba de este teorema se explica brevemente porque tiene carácter constructivo y, además, sirve para ilustrar el razonamiento de reducción al absurdo, tan útil en matemáticas. La idea es bien sencilla. Tomamos $a_0 = a$, $b_0 = b$ y $c_0 = c$. Si $f(c_0) = 0$ entonces habremos terminado, pues nuestro valor α será precisamente $\alpha = c_0$. Suponemos, pues, que $f(c_0) \neq 0$. Como $f(a)f(b) < 0$, entonces $f(c_0)$ tendrá necesariamente el signo de $f(a_0)$ (en cuyo caso $f(c_0)f(b_0) < 0$ y la función verificará las hipótesis del teorema en el intervalo $[a_1, b_1] = [c_0, b_0]$, que tiene longitud igual a $\frac{b-a}{2}$), o tendrá el signo de $f(b_0)$ (en cuyo caso $f(a_0)f(c_0) < 0$ y la función verificará las hipótesis del teorema en el intervalo $[a_1, b_1] = [a_0, c_0]$, que tiene longitud igual a $\frac{b-a}{2}$). Este proceso podemos repetirlo, partiendo esta vez del intervalo recién calculado, $[a_1, b_1]$, y tomando $c_1 = \frac{a_1+b_1}{2}$, lo cual nos proporcionará bien directamente una raíz de la ecuación $f(x) = 0$ o bien un nuevo intervalo, que notaremos $[a_2, b_2]$, donde la función cambia el signo. Evidentemente, este nuevo intervalo tendrá longitud $\frac{b-a}{2^2}$. Si reiteramos el argumento de forma indefinida, obtendremos una sucesión de intervalos encajados

$$[a, b] = [a_0, b_0] \supset [a_1, b_1] \supset \cdots \supset [a_n, b_n] \supset \cdots$$

que, además, verifican las siguientes propiedades:

- $b_n - a_n = \frac{b-a}{2^n} \to 0$
- $f(a_n)f(b_n) < 0$ para todo n.

(si la sucesión se detuviera sería porque $f(c_n) = 0$ para un cierto valor de n, en cuyo caso no habría nada que probar). Como cada vez que cambiamos el valor de a_n por a_{n+1} resulta que o bien $a_n = a_{n+1}$ (si $f(a_n)f(c_n) < 0$) o bien $a_n < a_{n+1} = c_n = \frac{a_n+b_n}{2}$ (si $f(a_n)f(c_n) > 0$, lo cual conduce necesariamente a que $f(c_n)f(b_n) < 0$ y, por tanto, $[a_{n+1}, b_{n+1}] = [c_n, b_n]$) queda claro que la sucesión (a_n) es monótona creciente y está acotada, por quedar siempre dentro del intervalo $[a, b]$. Se sigue que $\lim a_n$ necesariamente existe. El mismo tipo de argumento nos dice que (b_n) decrece y está acotada, por lo que también existe $\lim b_n$. Además, como $b_n - a_n = \frac{b-a}{2^n} \to 0$, ambos límites son iguales a cierto número $\alpha \in [a, b]$. La desigualdad $a_n \leq c_n \leq b_n$ unida al principio del sandwich nos dice que $\lim a_n = \lim c_n = \lim b_n = \alpha$.

Veamos, por reducción al absurdo, que $f(\alpha) = 0$. En efecto, si fuera $f(\alpha) \neq 0$, tendríamos que, para cierto $\delta > 0$, la imagen a través de f del intervalo $[\alpha - \delta, \alpha + \delta]$ estaría dentro del intervalo $[f(\alpha) - \frac{|f(\alpha)|}{2}, f(\alpha) - \frac{|f(\alpha)|}{2}]$, que no contiene al 0. En particular, todos los puntos de dicha imagen tienen necesariamente el mismo signo. Sin embargo, si n es lo bastante grande, tendremos que $a_n, b_n \in [\alpha - \delta, \alpha + \delta]$ y $f(a_n)f(b_n) < 0$, lo cual es imposible. Esto finaliza la prueba.
El proceso de prueba utilizado se denomina método de bisección y tiene la bonita propiedad de que puede ser fácilmente implementado en forma de algoritmo (de hecho, lo he explicado como un algoritmo). Además, la sencilla observación de que $\alpha \in [a_n, b_n]$ y que c_n es el punto medio de dicho intervalo, garantiza que

$$|\alpha - c_n| \leq \frac{b_n - a_n}{2} = \frac{b - a}{2^{n+1}}.$$

En particular, si se desea calcular una aproximación de α con un error menor o igual a $\varepsilon > 0$, bastará forzar la desigualdad $\frac{b-a}{2^{n+1}} \leq \varepsilon$, lo cual conduce a

$$n \geq \log_2 \frac{b - a}{\varepsilon} - 1.$$

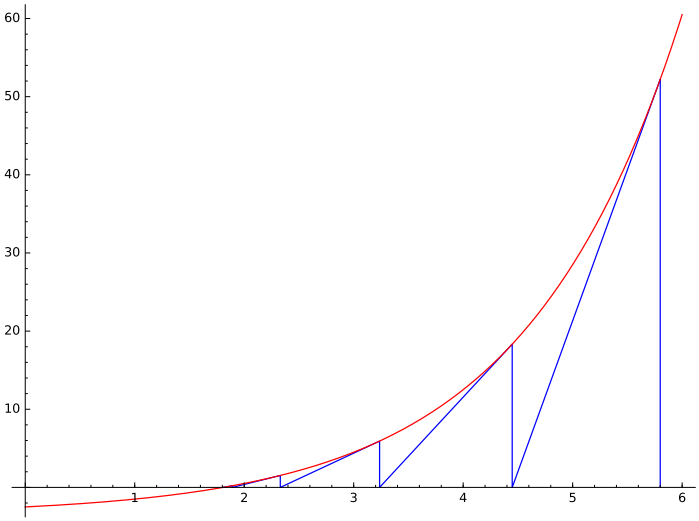
El método de bisección tiene la particularidad de que es aplicable a una clase muy amplia de funciones: las funciones continuas. Por otra parte, no utiliza apenas otra cosa de la función que su continuidad, lo cual tiene el efecto de que su velocidad de convergencia a la solución de la ecuación sea relativamente lenta. Existen métodos que, al aplicarse a clases más restrictivas de funciones, permiten utilizar la función en particular para lanzar, en cada iteración del algoritmo, una aproximación de la solución que se adapta a la función concreta que define la ecuación. Tal es el caso del método de Newton, que asume, entre otras cosas, que la función es derivable en todos los puntos del intervalo elegido y que la derivada no se anula en dicho intervalo. Si hemos obtenido una aproximación a_n de la solución de la ecuación $f(x) = 0$, entonces calculamos a_{n+1} intersecando la recta tangente al gráfico de f en el punto $(a_n, f(a_n))$ (que está dada por la ecuación $y = f(a_n) + f'(a_n)(x - a_n)$) con el eje de abscisas $y = 0$, lo cual arroja el valor

$$a_{n+1} = a_n - \frac{f(a_n)}{f'(a_n)}$$

Para garantizar la convergencia del algoritmo, así como que el límite es un cero de la función $f(x)$, es necesario imponer ciertas condiciones sobre la función $f(x)$, pero el método converge muy rápido para aquellas funciones que verifican dichas condiciones, que son, después de todo, muchas.
El siguiente programa de SAGE sirve para mostrar el comportamiento del método en un ejemplo concreto:

```
f(x)=2^x-3.5
g(x)=diff(f,x,1)
eps=10^(-9)
N=50 # Nº máximo de iteraciones
c= 5.8 #Punto de inicio
p=[(c,0)]
for n in [1..N]:
    d=float(c-f(c)/g(c))
    p=p+[(c,f(c))]+[(d,0)]
    if abs(d-c)<eps:
        c=d
        break
    c=d
line(p)+plot(f,(x,0,6),color='red')
```

y produce como salida el siguiente gráfico:



En clase se expone el siguiente resultado de convergencia global para el método de Newton y se muestra cómo utilizarlo en varios ejemplos:

Teorema 2.6

Supongamos que $f : [a, b] \to \mathbb{R}$ es una función derivable que satisface las siguientes condiciones:

- $f(a)f(b) < 0$.
- $f'(x) \neq 0$ para todo $x \in [a, b]$
- $f''(x)$ no cambia el signo en $[a, b]$
- $\text{máx}\{\frac{|f(a)|}{|f'(a)|}, \frac{|f(b)|}{|f'(b)|}\} \leq b - a$.

Entonces la ecuación $f(x) = 0$ tiene una única solución $\alpha \in [a, b]$ y, si consideramos la sucesión recurrente

$$\begin{cases} a_0 & = & c \in]a, b[\\ a_{n+1} & = & a_n - \frac{f(a_n)}{f'(a_n)} \end{cases}$$

entonces $\lim a_n = \alpha$.

N Es importante observar que existen métodos híbridos entre el de Newton y el de bisección en los que la convergencia global queda garantizada y, además, la velocidad de convergencia está mejorada respecto de la que corresponde al método de bisección puro. Alguno de estos métodos se implementa de forma efectiva en las prácticas. También es interesante explicar el método de la secante, cuyo orden de convergencia es superior al de la bisección pero inferior al de Newton. También se explica en las prácticas el método de aceleración de la convergencia de sucesiones de Aitken.

N Otra de las ventajas del método de Newton frente al algoritmo de bisección -que deberíamos mencionar en clase, aunque no dispongamos de tiempo para exponerla de forma detallada- es que este se puede implementar de forma sencilla para funciones de variable compleja y para sistemas de ecuaciones definidas por funciones funciones de varias variables.

2.2.1.3. Sucesiones divergentes. Órdenes de magnitud. Aplicaciones al estudio de los algoritmos

Si bien lo típico en un primer curso de Cálculo es prestar especial atención a las sucesiones convergentes y relacionarlas con la continuidad de las funciones, en el caso que nos ocupa es imprescindible prestar especial atención a las sucesiones divergentes a infinito e incluso clasificarlas según su orden de magnitud. Esto es así porque, como ya se avanzó en la introducción del tema, el costo de un algoritmo cuya entrada es un objeto de "tamaño" n (por ejemplo, una lista de n datos, un vector de $\mathbb{R}^n$, una matriz cuadrada de orden n, etc.), suele definirse a través de una sucesión que normalmente diverge cuando n tiende a infinito y, como es natural, resulta apropiado estudiar la velocidad con la que esta sucesión crece. Los conceptos relevantes aquí son los siguientes:

Definición 2.4

Dadas dos sucesiones $(a_n), (b_n)$, decimos que

- $a_n = \mathbf{O}(b_n)$ si existe una constante $C > 0$ (que no depende del valor n) tal que $|a_n| \leq C|b_n|$ para $n = 0, 1, \cdots$.
- $a_n = \mathbf{\Omega}(b_n)$ si existe una constante $C > 0$ (que no depende del valor n) tal que $|a_n| \geq C|b_n|$ para $n = 0, 1, \cdots$.
- (a_n) es "mucho menor que" (b_n) (lo cual denotamos por $a_n << b_n$) si $\lim \frac{a_n}{b_n} = 0$.
- $(a_n) \sim (b_n)$ si $\lim \frac{|a_n|}{|b_n|} = L \neq 0$ para cierto $L \in \mathbb{R}$.

Las sucesiones se dice que tienen el mismo orden de magnitud cuando $(a_n) \sim (b_n)$.

Es usual utilizar la siguiente notación: Si $f, g : \mathbb{N} \to \mathbb{R}$ son sucesiones,

$$f \in \mathbf{O}(g) \Leftrightarrow f(n) = \mathbf{O}(g(n))$$

$$f \in \mathbf{\Omega}(g) \Leftrightarrow f(n) = \mathbf{\Omega}(g(n))$$

$$f \in \mathbf{\Theta}(g) \Leftrightarrow (f(n)) \sim (g(n)).$$

Definición 2.5

Supongamos que $T(n)$, $S(n)$ definen los costes de dos algoritmos $\mathcal{A}, \mathcal{B}$, respectivamente, que se aplican a ciertos problemas de tamaño n. Diremos que ambos algoritmos están en la misma clase de complejidad si $t_n = T(n) \sim s_n = S(n)$. Diremos que el algoritmo $\mathcal{A}$ es mucho más eficiente que el algoritmo $\mathcal{B}$ si $t_n << s_n$.

N A la hora de calcular el coste (o la clase de complejidad) que tiene un algoritmo, el cual podemos asumir se ha especificado con un pseudocódigo, es útil tener en cuenta las siguientes asociaciones:

- El coste de cada una de las operaciones aritméticas elementales (suma, resta, multiplicación y división) es una constante.
- El coste de realizar una comparación entre dos números o datos (e.g., si un número es mayor que otro) también es constante.
- El coste de una asociación del tipo **x:=e** es bien constante o bien el coste de la evaluación de **e**. Esto se puede aplicar también a asignaciones múltiples.
- El coste de una sucesión de instrucciones del tipo $\mathbf{I_1}; \mathbf{I_2}$ es la suma de los costes correspondientes a $\mathbf{I_1}$ e $\mathbf{I_2}$, respectivamente.
- El coste de una instrucción del tipo **Si** B**, entonces** I_1**, si no**B**, entonces** I_2 es la suma del coste de evaluar la condición booleana B más el coste de ejecutar la orden I_1 **o** I_2.

Obsérvese que para que la comparación entre algoritmos introducida en la definición anterior tenga utilidad lo razonable es que ambos algoritmos $\mathcal{A}$, $\mathcal{B}$ resuelvan el mismo problema y, además, es importante tener en cuenta la forma en la que se ha calculado el coste de cada algoritmo. Concretamente, un mismo algoritmo puede tener asociadas varias funciones que miden su costo computacional. Por ejemplo, podemos disponer de una estimación del costo del algoritmo para el peor de los casos posibles (el tiempo que tarda en ejecutarse el algoritmo puede depender de ciertas propiedades de los datos de entrada como, por ejemplo, si están ordenados, etc.) y otras bien distintas para el mejor de los casos y para el caso promedio. Entonces no tendría sentido comparar dichas funciones costo para dos algoritmos diferentes, a no ser que ambas funciones representen el mismo tipo de problema. Así pues, si disponemos de dos algoritmos diferentes que resuelven el mismo problema y calculamos sus funciones costo pueden darse varias posibilidades. Por ejemplo, podría suceder que el costo de uno de los algoritmos, aplicado al peor de los casos sea mucho menor que el costo del otro algoritmo (aplicado, también al peor de los casos) pero que, cuando se considera el caso promedio el primer algoritmo resulte en un coste mucho mayor que el correspondiente al mismo caso promedio en el segundo algoritmo. Si esto sucede, la decisión de elegir uno u otro algoritmo dependerá de la naturaleza del problema estudiado. También es importante estudiar la complejidad de los problemas, lo cual implica a menudo la búsqueda de cotas inferiores y superiores para el costo de cualquier algoritmo que pueda resolver el problema en cuestión.
Un ejemplo concreto de problema relevante en informática para el que existen algoritmos diferentes cuyo coste computacional varía de forma sensible es el problema de ordenar una lista de datos. Estos datos pueden ser de cualquier tipo (nombres, números reales, etc.) y, evidentemente, la tarea de ordenarlos es fundamental en el trabajo informático por numerosas razones:

- Facilita la lectura de los mismos por humanos. Por ejemplo, la utilidad de los diccionarios, las guías telefónicas, los censos electorales, etc. se debe precisamente a que son listas ordenadas y, por tanto, la búsqueda de un dato concreto se puede realizar muy rápidamente -en algunos de estos casos, sería una tarea descomunal, casi imposible, para una persona, buscar un nombre concreto si la correspondiente lista estuviera desordenada.
- Cuando un ordenador va a realizar una búsqueda en una lista, esta también se ve simplificada -y se reduce de forma sensible en términos de consumo de tiempo- si la lista está previamente ordenada.
- En una lista ordenada se detectan de forma sencilla las repeticiones (y esto sirve, por ejemplo, para eliminarlas). Comparar dos listas ordenadas para saber si coinciden o no es mucho más sencillo si ambas están ordenadas -por el mismo criterio-.

Esta tarea es tan importante que se estima que más de la cuarta parte del tiempo de cálculo de los ordenadores se dedica a ordenar datos. Precisamente por eso existen numerosos algoritmos que se ocupan del problema de la ordenación. El más sencillo de programar, llamado algoritmo burbuja, consiste en lo siguiente: Si disponemos de n datos desordenados formando una lista $x_1, x_2, \cdots, x_n$, iniciamos el algoritmo comparando los términos de dos en dos del siguiente modo: primero comparamos x_1 y x_2 y los recolocamos de modo que el mayor de ambos quede a la derecha en la lista. A continuación realizamos la misma comparación entre el término que ahora ocupa la segunda posición (que será el mayor de x_1, x_2) y el tercer término, x_3, dejando nuevamente a la derecha el mayor de ambos. Se sigue del mismo modo avanzando posiciones a la derecha hasta realizar la correspondiente comparación con x_n. Decimos que hemos realizado en este punto una primera pasada (o vuelta) sobre los datos y, obviamente, el resultado de dicha operación es colocar a la derecha al mayor de los elementos de la lista. Repetimos ahora el proceso con los primeros $n-1$ términos de la lista resultante, luego con los primeros $n-2$ términos de la lista que surge tras la segunda pasada, etc., hasta que se hayan realizado n pasadas. Al finalizar este algoritmo (que se puede programar con un doble bucle) tendremos ante nosotros la lista ordenada de menor a mayor. En la primera pasada se realizan $n-1$ comparaciones, en la segunda, $n-2$ comparaciones, etc., hasta que en la última pasada se realiza una única comparación, por lo que el número total de comparaciones es $1+2+\cdots+(n-1) = \frac{n(n-1)}{2} = \mathbf{O}(n^2)$. Además, el número de intercambios realizados en el algoritmo es también $\mathbf{O}(n^2)$.
Un ejemplo de ordenación por el algoritmo burbuja que acabamos de describir es el siguiente (Representamos solamente los resultados de la primera vuelta, que colocan a la derecha al máximo de la lista de números elegida, $5, 1, 3, 9, 7, 6$. Para finalizar el algoritmo sería necesario realizar -en el peor de los casos- cinco vueltas más):

Dato inicial:	5	1	3	9	7	6
Primera comparación:	1	5				
Segunda comparación:		3	5			
Tercera comparación:			5	9		
Cuarta comparación:				7	9	
Quinta comparación:					6	9
Fin de la primera vuelta:	1	3	5	7	6	9

Existen algoritmos mucho más eficientes que el que acabamos de mostrar. Uno de ellos es el método de ordenación por inserción, que consiste en lo siguiente: Se parte de la lista inicial $L_1 = \{x_1\}$ y se van insertando los demás datos x_k, $k = 2, \cdots, n$ en el orden que les corresponde en listas sucesivas de tamaño $2, 3, \cdots$, hasta finalizar el proceso. Este algoritmo, aplicado a la lista de números anterior, produce la siguiente salida:

Dato inicial:			5	1	3	9	7	6
Primer paso:	L_1	=	5					
Segundo paso:	L_2	=	1	5				
Tercer paso:	L_3	=	1	3	5			
Cuarto paso:	L_4	=	1	3	5	7		
Quinto paso:	L_5	=	1	3	5	6	7	
Sexto -y último- paso:	L_6	=	1	3	5	6	7	9

En el k-esimo paso debemos colocar el valor x_k en el lugar que le corresponde de la lista $L_k = L_{k-1} \cup \{x_k\}$ que contiene los primeros k elementos. Esto se puede hacer comparando de forma ordenada con el resto de elementos, lo cual podría conllevar hasta $k-1$ comparaciones y, por tanto, un total de $1+2+\cdots+(n-1) = \mathbf{O}(n^2)$ comparaciones. Ahora bien, realizamos la inserción fijándonos primero en qué mitad de la lista L_{k-1} debe estar, luego hacemos lo mismo con la mitad de la lista elegida, etc., entonces, si el número de elementos de L_k está entre 2^{r-1} y 2^r, averiguaremos su posición exacta en a lo sumo r pasos (aquí se usa fuertemente que la lista L_{k-1} ya está ordenada). Es decir, utilizando este mecanismo de inserción se necesitan a lo sumo $\log_2 k$ comparaciones en el paso k, lo cual nos da un total del orden de

$$\log_2 2 + \log_2 3 + \cdots + \log_2 n \leq n \log_2 n$$

comparaciones. Esto supone una mejora fundamental respecto del algoritmo de burbuja. Sin embargo, en el algoritmo de inserción se requieren (en el peor de los casos) $\mathbf{O}(n^2)$ intercambios. Sobre este tema se pueden decir, de hecho, muchas más cosas. Vale la pena mencionar a los alumnos que es posible demostrar que resolver el problema de ordenar una lista de n elementos siempre conlleva $\mathbf{O}(n\log_2 n)$ comparaciones y, además, existen algoritmos que logran realizar esta tarea en $\mathbf{O}(n\log_2 n)$ comparaciones y con un total de intercambios también del orden $\mathbf{O}(n\log_2 n)$. Uno de ellos, relativamente sencillo, llamado algoritmo de Willians, aparece en el libro de Biggs [8, Capítulo 9].
Si se piensa en la idea con la que hemos acelerado el proceso de inserción anterior, vemos que esta se corresponde fielmente con el método que solíamos usar en la infancia para reducir lo máximo posible el número de preguntas necesarias en el juego de adivinar un número que se ha elegido entre dos dados. De hecho, ese juego es equivalente a la búsqueda de la posición exacta en la que debemos insertar x_k en la lista L_{k-1} para formar la nueva lista ordenada L_k. Es más, esta idea de 'romper el problema en dos mitades del mismo tamaño" responde a una táctica general que funciona bastante bien para reducir el coste computacional de muchos algoritmos y que recibe el nombre de "divide y vencerás". Si asumimos que conocemos un algoritmo $\mathcal{A}$ que se aplica a una lista de n datos con coste computacional $T(n)$ y que, en caso de tener resuelto el problema para dos listas de $n/2$ elementos disponemos de un algoritmo $\mathcal{B}$ que "funde"los resultados del algoritmo aplicado a ambas listas para recuperar una solución del algoritmo para la lista formada por la unión de ambas, con coste computacional $S(n)$, entonces podemos sustituir nuestro algoritmo original por el que resulta de realizar el siguiente proceso:

- Dividir la lista de n elementos en dos mitades de $n/2$ elementos y aplicar el algoritmo $\mathcal{A}$ a ambas listas.
- Fundir los resultados obtenidos en el paso anterior, mediante la aplicación del algoritmo $\mathcal{B}$.
- Aplicar los pasos anteriores de forma iterada a cada uno de los subproblemas que se obtienen de dividir la lista inicial en dos mitades.

Si hacemos esto, el costo computacional del nuevo algoritmo -que denotamos nuevamente por $T(n)$- satisface la relación de recurrencia

$$T(n) = 2T(n/2) + S(n)$$

Por ejemplo, en el caso de ordenar una lista, es fácil comprobar que, si se dispone de dos listas ordenadas de tamaño $n/2$ el proceso de fundirlas en una única lista ordenada de tamaño n requiere cn comparaciones. Esto se logra, por ejemplo, con el siguiente algoritmo: Dadas dos listas ordenadas $x_1, x_2, \cdots, x_m$, $y_1, y_2, \cdots, y_s$, se comparan los dos primeros elementos de ambas (en nuestro caso, x_1, y_1) y se elige el menor, que será el primer elemento de la nueva lista. Además, las listas originales se sustituyen por dos listas nuevas en las que ya no está el elemento elegido (de modo que una de ellas reduce su tamaño en una unidad). Luego se comparan los dos segundos elementos y se vuelve a tomar el menor de ambos y a generar dos nuevas listas, etc. El proceso termina cuando una de las dos listas haya quedado vacía. Como mucho, se requieren $m + s$ comparaciones. Esto implica que, para nuestro algoritmo $\mathcal{B}$, $S(n) = cn$ para una cierta constante $c > 0$ -que representa el costo computacional de hacer una sola comparación y que puede depender de la máquina que lo haga- y por tanto

$$T(n) = 2T(n/2) + cn$$

Aplicado a un número del tipo $n = 2^m$ obtenemos que

$$\begin{aligned}
T(n) = T(2^m) &= 2T(2^{m-1}) + 2^m c \\
&= 2(2T(2^{m-2}) + 2^{m-1}c) + 2^m c = 2^2 T(2^{m-2}) + 2^m c + 2^m c = 2^2 T(2^{m-2}) + 2\cdot 2^m c \\
&= 2^2(2T(2^{m-3}) + 2^{m-2}c) + 2\cdot 2^m c = 2^3 T(2^{m-3}) + 3\cdot 2^m c \\
&\vdots \\
&= 2^m T(1) + m2^m c \\
&= cn\log_2 n
\end{aligned}$$

pues $T(1) = 0$ (no hay que hacer nada para ordenar una lista de un solo dato, por lo que $T(1) = 0$ necesariamente). No es difícil ahora comprobar (por ejemplo, aplicando el método de inducción) que $T(n) = cn\log_2 n$ satisface la fórmula $T(n) = 2T(n/2) + cn$ con $T(1) = 0$ para todo $n \in \mathbb{N}$.
A veces la técnica de "divide y vencerás"da lugar a relaciones de recurrencia distintas, más difíciles de resolver. Por ejemplo, en el problema de multiplicar dos números enteros x, y de n cifras, que implica $\mathbf{O}(n^2)$ multiplicaciones cuando el producto se hace con el algoritmo usual, se puede proceder del siguiente modo: Asuminos primero que $n = 2^m$. Se descomponen los números como $x = a10^{n/2} + b$ e $y = c10^{n/2} + d$. Entonces los números a, b, c, d tienen a lo sumo $n/2$ cifras cada uno. Además,

$$\begin{aligned} xy &= (a10^{n/2} + b)(c10^{n/2} + d) = ac10^n + (ad + bc)10^{n/2} + bd \\ &= ac10^n + ((a + b)(c + d) - ac - bd)10^{n/2} + bd \end{aligned}$$

Esto significa que realizando tres productos de números de $n/2$ cifras, ac, bd y $(a + b)(c + d)$, podemos recuperar el producto xy. Si el coste computacional del algoritmo para multiplicar dos números de n cifras es $T(n)$, tendremos entonces que

$$\begin{cases} T(1) &= k \\ T(n) &= 3T(n/2) + cn \end{cases}$$

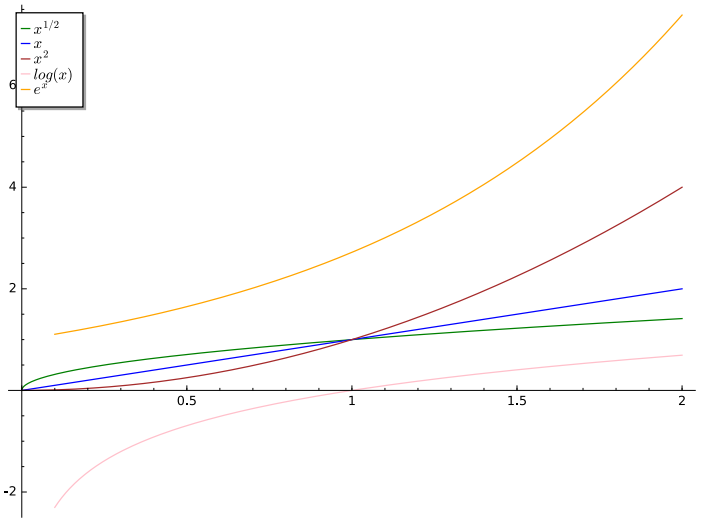
(porque multiplicar dos números de una cifra conlleva cierto coste, que hemos denotado por k, y porque las sumas que aparecen en el lado derecho de la descomposición de xy que se ha mostrado más arriba conllevan un coste del orden $\mathbf{O}(n)$). En este caso se puede demostrar que la función $T(n)$ satisface $T(n) = \mathbf{O}(n^{\log_2 3})$, lo cual es una mejora puesto que $\log_2(3) = 1{,}58496250072116...$, que es estrictamente menor que 2.
En clase explicamos los siguientes resultados sobre órdenes de magnitud de sucesiones:

Teorema 2.7 (Escala básica de los órdenes de magnitud)

Supongamos que $0 < c < d < \infty$, $1 < r < s < \infty$. Se tiene que

$$1 << \log n << n^c << n^d << r^n << s^n << n! << n^n$$

La explicación del teorema anterior conviene acompañarla de la visualización de la siguiente gráfica:



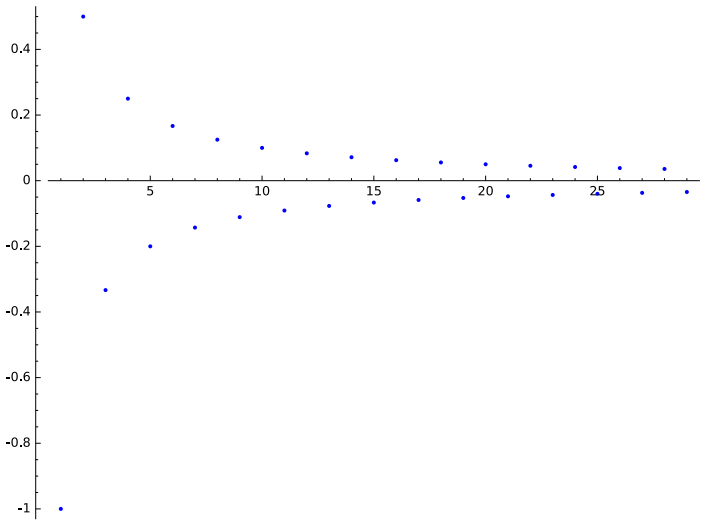
Es importante incluir numerosos ejemplos de sucesiones para las que se deben calcular sus órdenes de magnitud asociados. Así, un ejercicio típico de los que se exponen en clase consiste en introducir varias sucesiones y pedir al alumno que: (a) encuentre sus distintos órdenes de magnitud, y (b) ordene las distintas sucesiones en términos de su orden de magnitud, de menor a mayor. Para que los alumnos afronten este tipo de cuestiones con mayor facilidad e incorporando una cierta "mecánica" para los cálculos, se les explican algunas propiedades algebraicas de los órdenes de magnitud. Así, por ejemplo, se utilizan los siguientes criterios:

- Si $a_n \neq 0$ para $n \geq N_0$ y $\alpha \neq 0$, entonces $(a_n) \sim (|a_n|) \sim (\alpha a_n)$.
- Si $(a_n) \sim (c_n)$ y $(b_n) \sim (d_n)$, entonces $(a_n b_n) \sim (c_n d_n)$ y $(a_n/b_n) \sim (c_n/d_n)$.
- Si $(a_n) << (b_n)$, entonces $(a_n + b_n) \sim (b_n)$.
- Si $(a_n) \sim (c_n)$ y $(b_n) \sim (d_n)$, entonces $a_n << b_n \Leftrightarrow c_n << d_n$.

2.2.2. Las prácticas

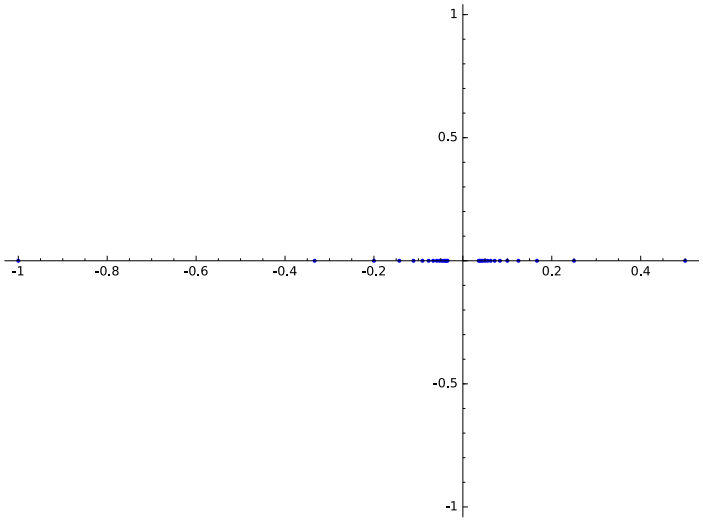
En las prácticas de este tema, además de aprender a manejar sucesiones desde un punto de vista analítico -que suele involucrar el uso de recurrencias y desigualdades, así como diversos tipos de manipulaciones algebraicas-, se persigue que los alumnos visualicen las sucesiones y aprendan a distinguir los distintos órdenes de magnitud con los que se van a encontrar en el futuro con mayor probabilidad. SAGE permite visualizar sucesiones de varias formas. Por ejemplo, la siguiente orden sirve para dibujar sucesiones en el plano:

```
sage: list_plot([(n,(-1)^n/n) for n in range(1,30)])                    22
Graphics object consisting of 1 graphics primitive                      23
```



Por otra parte, la orden:

```
sage: list_plot([((-1)^n/n,0) for n in range(1,30)])                    24
Graphics object consisting of 1 graphics primitive                      25
```



sirve para dibujar una sucesión de puntos tal como van apareciendo en la recta real -abstrayéndonos del orden en el que estos puntos aparecen-
Con las órdenes del tipo:

```
def a(n):
    if n==1:
        return 2
    else:
        return (2*a(n-1)/n)
```

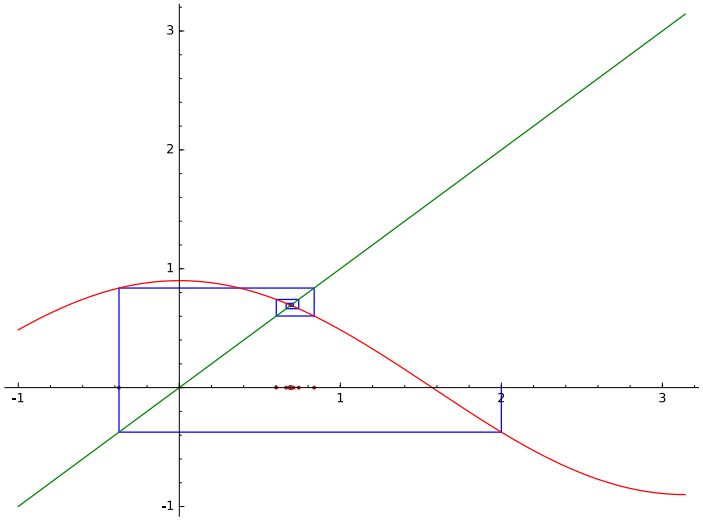
se pueden definir sucesiones recurrentes. Una vez definida la sucesión con las órdenes anteriores, esta puede evaluarse para los valores de n que queramos:

```
sage: a(1)
2
sage: a(2)
2
sage: a(5)
4/15
sage: a(13)
8/6081075
```

Si la sucesión recurrente se obtiene como resultado de iterar una función $f(x)$ a partir de un valor inicial $a_0 = c$, su evolución puede dibujarse sobre la gráfica de f del siguiente modo:

```
def a(n):
    if n==0:
        return 2
    else:
        return 0.9*cos(a(n-1))
p=[(a(0),0)]
for k in [0..50]:
    p=p+[(a(k),a(k+1))]+[(a(k+1),a(k+1))]
```

```
line(p)+plot(0.9*cos(x),(x,-1,pi),color='red')+plot(x,(x,-1,pi),color='green') \
+list_plot([(a(n),0) for n in range(0,50)],color='brown')
```



Estas sucesiones, cuando convergen, lo hacen hacia un punto fijo de la función $f(x)$ o, lo que es lo mismo, hacia alguna de las soluciones de la ecuación $f(x) = x$. Sabemos, además, por el Teorema del Punto fijo de Banach, que cuando $f(x)$ es derivable, $f([a,b]) \subseteq [a,b]$ y la derivada satisface $\|f'\|_{[a,b]} = \sup_{x\in[a,b]} |f(x)| = K < 1$, la ecuación $f(x) = x$ tiene un único punto fijo α en el intervalo $[a.b]$ y la sucesión recurrente dada por $a_0 = c$ y $a_{n+1} = f(a_n)$ para $n = 0, 1, \cdots$, converge a dicho punto fijo y, además,

$$|a_n - \alpha| \leq K^n \,\text{máx}\{|c-a|, |c-b|\}.$$

Esto permite, para cada $\varepsilon > 0$, anticipar el valor $n \in \mathbb{N}$ a partir del cual $|a_n - \alpha| < \varepsilon$. En efecto, tomando el logaritmo en base K a ambos lados de la desigualdad

$$K^n \,\text{máx}\{|c-a|, |c-b|\} \leq \varepsilon$$

obtenemos que

$$n + \log_K(\text{máx}\{|c-a|, |c-b|\}) \geq \log_K \varepsilon$$

(obsérvese que la función $\log_K(x) = \frac{\log x}{\log K}$ es decreciente cuando $0 < K < 1$, pues $\log K < 0$) y, por tanto:

$$n \geq \log_K \varepsilon - \log_K(\text{máx}\{|c-a|, |c-b|\})$$

Por ejemplo, en el caso que hemos dibujado más arriba la función es $f(x) = 0{,}9 * \cos(x)$, el intervalo considerado es $[-1, \pi]$ y $c = 2$. Obviamente, en este caso $K = 0{,}9 < 1$ y $\text{máx}\{|c-a|, |c-b|\} = \pi - 2$, por lo que la siguiente orden de SAGE nos proporciona el valor de n que garantiza un error inferior a $\varepsilon = 10^{-7}$:

```
sage: float(log(10^(-7),0.9) -log(pi-2,0.9))                                    34
154.237286155                                                                   35
```

El número de pasos necesarios es en este caso elevado debido a que $K = 0{,}9$ es un valor próximo a 1. Si hubiéramos considerado, en el mismo intervalo, las iteraciones que se obtienen para la función $\frac{1}{2}\cos(x)$, la constante sería $K = 1/2$ y el valor de n a partir del cual el error es inferior a 10^{-7} vendría dado por

```
sage: float(log(10^(-7),0.5) -log(pi-2,0.5))
23.4445446201
```

que es, obviamente, mucho menor.
En las clases de prácticas se realizan ejercicios de SAGE relacionados con resolución de ecuaciones basados en los métodos de Bisección, Punto fijo -vía el teorema de la aplicación contractiva de Banach- y Newton. En particular, se enseña a los alumnos cómo programar dichos métodos y se resuelven distintos ejemplos de ecuaciones no lineales que, de otra forma, serían inabordables. En pizarra se resuelven, además, numerosos problemas sobre convergencia de sucesiones, cálculo de órdenes de magnitud y ordenación de sucesiones en base a sus órdenes de magnitud.
Para programar el método de Newton se utiliza el siguiente código:

```
f(x)=2^x-3.5
g(x)=diff(f,x,1)
eps=10^(-9)
N=50 # N° máximo de iteraciones
c= 5.8 #Punto de inicio
p=[(c,0)]
for n in [1..N]:
    d=float(c-f(c)/g(c))
    p=p+[(c,f(c))]+[(d,0)]
    if abs(d-c)<eps:
        c=d
        break
    c=d
line(p)+plot(f,(x,0,6),color='red')
```

En él se utiliza como criterio de parada que $|x_{n+1} - x_n| \leq \varepsilon$ para cierto valor $\varepsilon > 0$ que nos sirve de tolerancia (con el añadido de que, si se supera un número máximo de iteraciones -que en este caso es $N = 50$- también paramos el algoritmo). Hay una razón fundamental que justifica este criterio de parada, y es la siguiente: se sabe que si α es una raíz simple de la ecuación (es decir, si $f(\alpha) = 0$ y $f'(\alpha) \neq 0$), entonces el método de Newton converge cuadráticamente cuando se parte de una aproximación inicial cercana a la raíz α. Esto significa que para datos iniciales x_0 cercanos a α, la sucesión $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ satisface la relación

$$\lim_{n\to\infty} \frac{|x_{n+1} - \alpha|}{|x_n - \alpha|^2} = \gamma \neq 0$$

para cierto valor γ. En particular, $|x_{n+1} - \alpha| << |x_n - \alpha|$ y, por tanto,

$$\alpha - x_n = x_{n+1} - x_n + (\alpha - x_{n+1}) \simeq x_{n+1} - x_n.$$

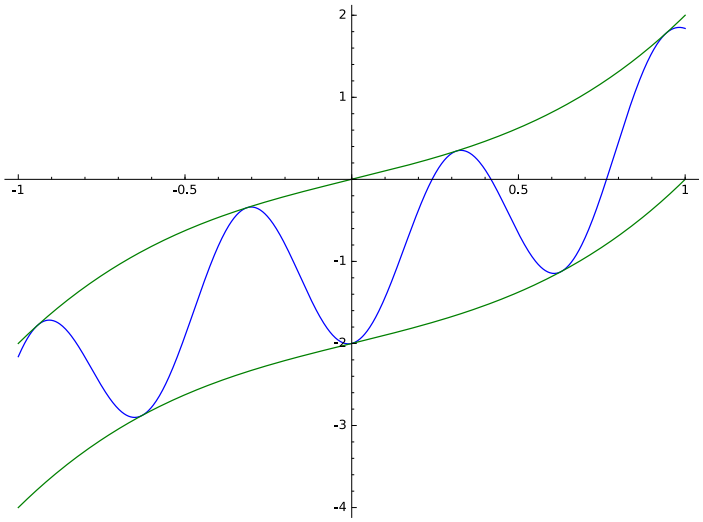
Es decir, $|x_{n+1} - x_n|$ es una buena aproximación del error $|\alpha - x_n|$.
Por otro lado, en el caso del método de bisección el código que se utiliza es del tipo

```
f(x)=x^3+x-1-cos(10*x) #Ejemplo de función a la que aplicamos disección.
a=0.7
b=0.8 #Estos son los extremos del intervalo a considerar.
epsilon=10^(-6)
N=log((b-a)/epsilon,2)-1
contador=0
a0=a
```

```
b0=b
c0=(a0+b0)/2
while f(c0)<>0 and contador<N:
    if f(a0)*f(c0)<0:
        b0=c0
        c0=(a0+b0)/2
    else:
        a0=c0
        c0=(a0+b0)/2
    contador=contador+1
    c0.n()
```

que produce la salida: $0{,}7641929626464484$. Obviamente, antes de aplicar el algoritmo a esta función con los extremos $a = 0{,}7, b = 0{,}8$, ha sido necesario separar la raíz en dicho intervalo, lo cual se puede hacer visualizando la gráfica de la función.

```
sage: plot(f,(x,-1,1))+plot(x^3+x-1+1,(x,-1,1),color='green')+plot(x^3+x-1-1,(x
   ,-1,1),color='green')
Graphics object consisting of 3 graphics primitives
```



El criterio de parada es, en este caso, no sobrepasar un determinado número de iteraciones N. Este criterio se justifica porque el valor de N que garantiza un error absoluto inferior a una cantidad prefijada ε se puede determinar a priori gracias a la fórmula:

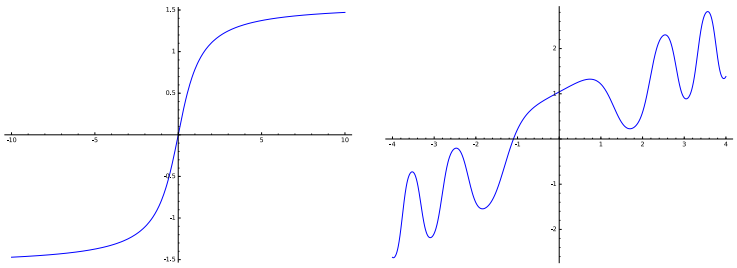
$$N \geq \log_2 \frac{b-a}{\varepsilon} - 1,$$

donde $[a, b]$ es el intervalo en el que está definida la función.
Ya hemos comentado que es posible desarrollar métodos híbridos entre bisección y Newton, que garantizan la convergencia global aunque disminuyan la velocidad de convergencia cuando se les compara con el método de Newton puro (pero la mejoran frente a bisección pura). Un ejemplo interesante es el siguiente algoritmo:

Algoritmo híbrido Newton-Bisección:

- Datos iniciales: x_0, $TOL = \varepsilon > 0$ y $f(x)$ (que debe ser derivable). Comenzamos con $k = 0$.
- Paso 1: Criterio de parada: Si $|f(x_k)| < TOL$, paramos.
- Paso 2: Calcular $x_+ = x_k - \frac{f(x_k)}{f'(x_k)}$. (Esta es la parte del algoritmo que emula al método de Newton)
- Paso 3: Mientras $|f(x_+)| \geq |f(x_k)|$, redefinir $x_+ = \frac{x_k + x_+}{2}$. (Esta es la parte del algoritmo que emula al método de bisección)
- Paso 4: $k = k + 1$, $x_k = x_+$. Ir al Paso 1.

Asociado a cada método iterativo para la solución de ecuaciones y a cada función $f(x)$ (cuyos ceros pretendemos hallar), podemos definir la región de convergencia del método como el conjunto de valores iniciales x_0 para los que la sucesión obtenida tras aplicar el método elegido es convergente. Así, podemos comparar dos métodos (para una función dada) simplemente comparando sus respectivas regiones de convergencia. El estudio de estas regiones de convergencia, para el método de Newton y para funciones de variable compleja, iniciado por Fatou y Julia a principios del S.XX, daría lugar, tras la aparición de los ordenadores y de manos de Mandelbrot, a la creación de los objetos matemáticos que hoy llamamos "fractales". En las prácticas proponemos calcular la region de convergencia contenida en el intervalo $[-4, 4]$ para las funciones $f(x) = \arctan(x)$ y $g(x) = \frac{x}{2} + \sin(\cos(x^2)) + 0{,}2$, para el método de Newton y el método híbrido de Newton-Bisección. Esto se logra pintando de rojo los puntos del intervalo donde el método converge y dejando sin pintar el resto.



Por ejemplo, es fácil comprobar que el método de Newton no converge para $g(x) = \frac{x}{2} + \sin(\cos(x^2)) + 0{,}2$ cuando tomamos como dato inicial $x_0 = 3$. Sin embargo, si aplicamos el método híbrido, el método converge. Esto se comprueba fácilmente haciendo correr el algoritmo. Así, si imponemos un error absoluto inferior a 10^{-9}, el algoritmo se implementa en SAGE mediante la siguiente secuencia de instrucciones:

```
f(x)=x/2+sin(cos(x^2))+0.2
g(x)=diff(f,x,1)

eps=10^(-9) # "Error absoluto" máximo permitido

N=50 # Nº máximo de iteraciones
c= -0.3
for n in [1..N]:
    d=c-f(c)/g(c)
    if abs(d-c)<eps:
       c=d
       break
```

```
        for k in [1..20]:
            while abs(f(d))>=abs(f(c)) and k<20:
             d=(d+c)/2

        c=d

    print 'La solucion aproximada es c[',n,'] ='
    print float(c)
```

y proporciona la salida -1.09824182056807, que es correcta.

2.3 Tema 3. Integración

2.3.1. La teoría

En este tema explicamos los rudimentos básicos de cálculo integral en una variable, tanto desde la perspectiva formal del cálculo de primitivas como desde la perspectiva de la aproximación numérica de integrales definidas. Para lo segundo, se hace necesario introducir antes la técnica de aproximación de funciones mediante interpolación polinómica e interpolación con funciones definidas a trozos mediante polinomios.

2.3.1.1. Integración de funciones de una variable

En la primera clase se introduce el concepto de integral definida como el área encerrada por una curva. Obviamente, dicho área se obtiene como límite de las áreas encerradas por ciertas funciones escalonadas (funciones que son constantes en intervalos que definen una partición del intervalo de partida), cosa que se concreta introduciendo el concepto de integral de Riemann. Así, si $f : [a, b] \to \mathbb{R}$ es nuestra función -la cual debemos asumir que es acotada- y queremos determinar el area encerrada por la curva $y = f(x)$ y el eje de abscisas, entre los valores $x = a$ y $x = b$, tiene sentido elegir una partición arbitraria $P = \{a = t_0 < t_1 < \cdots < t_{n-1} < t_n = b\}$ del intervalo $[a, b]$ y considerar las sumas superior e inferior siguientes:

$$S(f, P) = \sum_{k=0}^{n-1} [(t_{k+1} - t_k) \sup_{t \in [t_k, t_{k+1}]} f(f)]$$

y

$$s(f, P) = \sum_{k=0}^{n-1} [(t_{k+1} - t_k) \inf_{t \in [t_k, t_{k+1}]} f(f)]$$

(En clase, se asume de partida que la función es continua y se utilizan, para la definición anterior, el máximo y el mínimo en vez del supremo y el ínfimo, porque el Teorema de Weierstrass garantiza que toda función continua en un intervalo cerrado alcanza dichos valores, y esto permite evitar introducir los conceptos de supremo e ínfimo, que son quizás demasiado sutiles para los alumnos de un primer curso de informática). Es evidente que, si denotamos por A el area buscada, entonces $s(f, P) \leq A \leq S(f, P)$ sea cuál sea la partición. Además, si P, Q son particiones de $[a, b]$ y Q es más fina que P, en el sentido de que todos los nodos t_k de P son también nodos de Q, cosa que denotamos por $P \preceq Q$, entonces $S(f, Q) \leq S(f, P)$ y $s(f, Q) \geq s(f, P)$. Finalmente, si P, Q son dos particiones arbitrarias, entonces podemos formar, a partir de ambas, una más fina que ambas simplemente considerando como nodos todos los nodos

de ambas particiones, la cual podemos denotar por $P \sqcup Q$. Esto nos lleva a afirmar que

$$s(f,P) \leq s(f,P \sqcup Q) \leq S(f,P) \leq S(f,Q).$$

Estas propiedades permiten garantizar que siempre existen las cantidades

$$\overline{\int_a^b} f = \inf_{P \text{ partición de } [a,b]} S(f,P)$$

y

$$\underline{\int_a^b} f = \sup_{P \text{ partición de } [a,b]} s(f,P)$$

que toman el nombre de integral superior e inferior de f, respectivamente, en el intervalo $[a,b]$ y que verifican, además, la desigualdad:

$$\underline{\int_a^b} f \leq \overline{\int_a^b} f.$$

Diremos que la función f es integrable de Riemann en $[a,b]$ si ambos valores coinciden, en cuyo caso llamados integral de la función a dicho valor común que, por supuesto, representa el area buscada:

$$\int_a^b f = \underline{\int_a^b} f = \overline{\int_a^b} f = A.$$

En intervalos semi-infinitos del tipo $[a,\infty)$ o $(-\infty,b]$ la integral se define como un límite:

$$\int_a^\infty f = \lim_{b\to\infty} \int_a^b f$$

$$\int_{-\infty}^b f = \lim_{a\to-\infty} \int_a^b f,$$

y si consideramos integrales definidas sobre toda la recta real, estas se calculan como

$$\int_{-\infty}^\infty f = \int_{-\infty}^0 f + \int_0^\infty f$$

(aunque podríamos sustituir el valor 0 por cualquier número real a).
Una vez se ha introducido el concepto de integral (en el sentido de Riemann), se enuncia (con un esbozo breve de la demostración) el siguiente resultado:

Teorema 2.8 (Teorema Fundamental del Cálculo)

Toda función continua $f : [a,b] \to \mathbb{R}$ es integrable en el sentido de Riemann. Es más, si definimos $F(x) = \int_a^x f$, entonces $F' = f$. En particular, si $G' = f$ para cierta función G, entonces $\int_a^b f = G(b) - G(a)$.

Este teorema explica el importante vínculo que existe entre los procesos de derivación e integración de funciones. Se remarca de forma insistente que dicho nexo fue un descubrimiento fundamental que debemos a varios matemáticos del siglo XVII, incluyendo Newton y Leibniz, pero cuya prueba detallada tuvo que esperar mucho tiempo, pues para su desarrollo fue necesario introducir un concepto apropiado de integral y profundizar en la naturaleza del conjunto de los números reales y del conjunto de las funciones continuas definidas sobre un intervalo, cuestiones

ambas que forman parte esencial de los mismísimos fundamentos del análisis y que no pudieron determinarse con precisión hasta bien entrado el siglo XIX. Sin embargo, ya Newton y Leibniz se dieron cuenta de la importancia de este resultado, que conecta dos conceptos geométricos aparentemente alejados, como son el área encerrada por una curva y la recta tangente a una curva en un punto. Además, no cabe duda que Newton valoraba también los aspectos físicos del tema, pues su principal interés, a la hora de redactar sus "Principios Matemáticos de Filosofía Natural" estaba en lo que hoy denominamos Física. Si la derivada sirve para describir magnitudes físicas como la velocidad y la aceleración, la integral se puede utilizar para describir otras magnitudes físicas, como el flujo, el centro de masa, etc. Es más: el teorema fundamental del cálculo -y sus versiones para funciones de varias variables, que son los llamados teoremas integrales del cálculo vectorial y que incluyen los teoremas de Gauss, Green y Stokes- justifica que los procesos de derivación e integración son en cierto sentido inversos y que, por tanto, también se puedan emplear integrales para explicar fenómenos físicos que se describen de forma natural utilizando derivadas. Así, por ejemplo, las ecuaciones de Maxwell para el electromagnetismo pueden expresarse tanto en forma diferencial como en forma integral y es cuestión de gustos o del tipo de problema al que deseemos enfrentarnos que se decida utilizar unas u otras.

El interés de las integrales en los estudios de informática no está tan definido como en el caso de la física o incluso otras ingenierías como la ingeniería civil o la ingeniería de telecomunicaciones, pero sí que podemos motivarlo aunque sea de una forma un poco colateral, si tenemos en consideración la importancia cada vez mayor que tiene la Estadística y, de forma muy particular, los procesos estocásticos, en el diseño de todo tipo de aplicaciones informáticas.

Así, por ejemplo, cuando deseamos realizar simulaciones de procesos estocásticos, es a menudo necesario estimar la esperanza matemática de una variable aleatoria,

$$E(X) = \int_{-\infty}^{\infty} t f_X(t) dt,$$

que es una integral sobre la recta real. Frecuentemente este tipo de integrales se aproximan aplicando el método de Monte Carlo -introducido por S. Ulam a mediados del siglo pasado-, el cual se sustancia mediante la generación de muestras de la variable aleatoria X a partir de muestras de una variable aleatoria uniforme. Concretamente, denotemos por U la variable aleatoria uniforme en $[0,1]$. Es obvio que $F_U(t) = t$ para $t \in [0,1]$. Si $F_X(t) = P[X \leq t]$ es la función de distribución de la variable aleatoria X -de la cual deseamos obtener algunas muestras-, entonces $F_X : \mathbb{R} \to [0,1]$ está definida por $F_X(t) = P[X \leq t]$ y, en una amplia gama de casos, se trata de una función estrictamente creciente. En particular, existe un intervalo $]a,b[\subseteq \mathbb{R}$ (que coincide con el rango de la variable aleatoria X y podría ocupar toda la recta real) tal que $F_X(t) = 0$ para $t \leq a$, $F_X(t) = 1$ para $t \geq b$, por lo que podemos considerar su función inversa $F_X^{-1} : [0,1] \to [a,b]$. Además, dicha función será también monótona creciente, por lo que podemos garantizar que

$$F_X(x) = F_U(F_X(x)) = P[U \leq F_X(x)] = P[F_X^{-1}(U) \leq F_X^{-1}(F_X(x))] = P[F_X^{-1}(U) \leq x],$$

de donde obtenemos que la variable aleatoria $X = F_X^{-1}(U)$ sigue la misma distribución que la variable aleatoria X. Así, si sabemos simular U, podremos simular X tan pronto como sepamos calcular la función F_X^{-1}. En particular, conviene conocer $F_X(t) = \int_{-\infty}^{t} f_X(s) ds$, que es una integral. Es importante observar que la simulación de variables aleatorias forma parte esencial de numerosos algoritmos que un informático puede necesitar programar en contextos de lo más diversos. En teoría de la probabilidad se definen a menudo medidas, como los momentos, la varianza, etc., cuyo cálculo involucra integrales, muchas de ellas, impropias de primera especie (es decir, definidas sobre intervalos infinitos).

Otro contexto en el que aparecen las integrales de forma natural es el análisis asintótico de algoritmos.

Una vez hemos explicado el TFC (teorema fundamental del cálculo), se introducen varias técnicas para determinar el valor de una primitiva o integral indefinida, que se considera un paso previo a la aplicación de la fórmula $\int_a^b f = F(b) - F(a)$ (donde $F' = f$, es decir, F es cualquiera de las primitivas de f). Los métodos de cálculo de primitivas que

se exponen son los más sencillos que existen: cambio de variables, integración por partes e integración de funciones racionales y funciones racionales trigonométricas. Se insiste en que en determinados casos, como por ejemplo cuando $f(x) = e^{-x^2}$, el cálculo explícito de una primitiva es imposible y, por tanto, para hallar integrales como $\int_a^b e^{-x^2}$ es necesario utilizar algún método de aproximación numérica.
Algunos cambios de variable que aparecen con frecuencia en el cálculo de integrales merecen una explicación específica. En particular, considero de interés motivar el cambio de variables que se da de forma habitual para convertir el cálculo de integrales de funciones racionales trigonométricas en integrales de funciones racionales, que viene especificado por las expresiones:

$$\begin{cases} t & = & \tan(x/2) \\ \sin(x) & = & \frac{2t}{1+t^2} \\ \cos(x) & = & \frac{1-t^2}{1+t^2} \end{cases}$$

y se basa en el hecho notable de que la circunferencia unidad admite, además de la parametrización usual basada en funciones trigonométricas, $(\cos(x), \sin(x))$, la parametrización racional $\left(\frac{2t}{1+t^2}, \frac{1-t^2}{1+t^2}\right)$.

2.3.1.2. Interpolación Polinómica

El siguiente punto a tratar en este tema es la interpolación polinómica y la interpolación polinómica a trozos. Este tema está motivado por la necesidad de buscar sustitutos sencillos para aquellas funciones f que no admiten una primitiva elemental. Así, si no podemos hallar $\int_a^b f$ porque la búsqueda de primitivas para la función f fracasa, sería deseable sustituir la función f por un polinomio P (o una función polinómica a trozos P) que tenga la propiedad de permanecer, a lo largo de todo el intervalo $[a, b]$, uniformemente cerca de la función f. Así, si $\|f - P\|_{[a,b]} := \sup_{t\in[a,b]} |f(t) - P(t)| \leq \varepsilon$, entonces $\left|\int_a^b (f - P)\right| \leq (b - a)\varepsilon$ y, por tanto, para $\varepsilon > 0$ suficientemente pequeño, $I\{f\} := \int_a^b P$ será una buena aproximación de $\int_a^b f$. Pues bien: el mecanismo que utilizamos para elegir la función P (a la que impondremos que sea un polinomio o, en el peor de los casos, una función polinómica definida a trozos), es interpolar a la función original f en un cierto conjunto (finito) de puntos, a los que llamamos nodos de interpolación. Por otra parte, el proceso de sustituir el cálculo de $\int_a^b f$ por $\int_a^b P$ es lo que llamamos 'integración numérica".

N Hay otra motivación fundamental para utilizar los procesos de interpolación, y es la siguiente: la experiencia dice que en multitud de situaciones prácticas no es posible disponer de expresiones analíticas cerradas para las funciones con las que se está trabajando -que pueden ser, por ejemplo, magnitudes físicas-. Aún así, sí es cierto que en general podremos medir dichas funciones (o señales) para diferentes valores de la variable que las define. Con esta información a mano, los procesos de interpolación nos permiten encontrar funciones que son sencillas de manejar y que aproximan, siguiendo algún criterio fijado a antemano, a la función de la que se han tomado varias muestras. Vamos a utilizar polinomios o funciones polinómicas a trozos para calcular nuestras aproximaciónes, y, como criterio de aproximación, vamos a exigir la total coincidencia con la función de partida en cierto conjunto de puntos (que llamamos nodos). Primero demostraremos la existencia y unicidad de dichos polinomios (o funciones polinómicas a trozos) y, posteriormente, estudiaremos el error cometido al utilizar dichas funciones como aproximación, según otro tipo de criterios (como la convergencia uniforme, etc.) Es importante observar, además, que la interpolación con funciones polinómicas a trozos se utiliza sistemáticamente en el procesado de imágenes digitales y tiene especial relevancia cuando éstas se utilizan para aplicaciones en medicina así como en la creación de programas que manejan entornos gráficos, vídeos y fotos, como googlemaps.

El primer resultado importante que exponemos es, por tanto, el teorema de interpolación de Lagrange:

Teorema 2.9 (Teorema de interpolación de Lagrange)

Sea $\{x_k\}_{k=0}^n \subseteq \mathbb{R}$ un conjunto de $n+1$ puntos de la recta que son distintos dos a dos. Entonces para cada conjunto $\{y_k\}_{k=0}^n \subseteq \mathbb{R}$ existe un único polinomio $P(x) = a_0 + a_1 x + \cdots + a_n x^n$ de coeficientes reales y con grado $\leq n$ tal que $P(x_k) = y_k$ para $k = 0, 1, \cdots, n$.

Hay varias demostraciones de este resultado. Quizás la más sencilla pasa por estudiar los casos más simples para los que el problema puede resolverse y deducir, a partir de ellos, la solución general. En efecto, es bien sabido que un polinomio de grado $\leq n$ tiene a lo sumo n ceros -a no ser que se trate del polinomio constantemente nulo, que tiene infinitos ceros. Así, queda claro que $P = 0$ es la única solución del sistema de ecuaciones $P(x_k) = 0$, $k = 0, 1, \cdots, n$, que se corresponde con la exigencia de que todos los valores y_k se anulen. Además, el mismo argumento vale para demostrar que, en el caso general, caso de existir solución al problema planteado, esta es única porque si P, P^* fueran dos soluciones del problema, entonces $(P - P^*)(x_k) = P(x_k) - P^*(x_k) = 0$ para $k = 0, 1, \cdots, n$ y, al ser $P - P^*$ de nuevo un polinomio de grado $\leq n$, se tiene que necesariamente es el polinomio nulo, por lo que $P = P^*$.
El siguiente caso a considerar es que solo uno de los y_k sea diferente de cero. Por ejemplo, si asumimos que $y_0 = 1$ pero $y_1 = \cdots = y_n = 0$, es claro que $P(x)$ tendrá necesariamente la forma $P(x) = C \cdot (x - y_1)(x - y_2) \cdots (x - y_n)$ para cierta constante $C \neq 0$. Esto es así por la combinación de los siguientes dos argumentos: Si $P(\alpha) = 0$ para cierto valor α, entonces al dividir $P(x)$ por $x - \alpha$ obtendremos que $P(x) = (x - \alpha)Q(x) + r$ para cierta constante r, pero como $P(\alpha) = 0$, sustituyendo $x = \alpha$ en el segundo miembro de la igualdad anterior , obtenemos que $r = 0$ y por tanto $x - \alpha$ divide a $P(x)$. Por otro lado, P tiene grado n, por lo que, al ser P un múltiplo de todos los polinomios $x - x_k$ para $k = 1, \cdots, n$, no queda más remedio que $P(x) = C \cdot (x - y_1)(x - y_2) \cdots (x - y_n)$. Por otra parte, $P(x_0) = 1$ garantiza que $1 = C \cdot (x_0 - y_1)(x_0 - y_2) \cdots (x_0 - y_n)$, por lo que podemos despejar la constante C para obtener

$$P(x) = \frac{(x - y_1)(x - y_2) \cdots (x - y_n)}{(x_0 - y_1)(x_0 - y_2) \cdots (x_0 - y_n)}.$$

Denotemos este polinomio por $\ell_0(x)$. Es obvio que $\ell_0(x_k) = 0$ para $k = 1, \cdots, n$ y $\ell_0(x_0) = 1$. Con los mismos argumentos podemos obtener la única solución $\ell_k(x)$ del problema

$$\ell_k(x_i) = \begin{cases} 1 & \text{, si } i = k \\ 0 & \text{, si } i \neq k \end{cases}$$

que viene determinada por el polinomio

$$\ell_k(x) = \frac{(x - y_0)(x - y_1) \cdots (x - y_{k-1})(x - y_{k+1}) \cdots (x - y_n)}{(x_0 - y_0)(x_0 - y_1) \cdots (x_0 - y_{k-1})(x_0 - y_{k+1}) \cdots (x_0 - y_n)}.$$

Una vez hemos resuelto estos "problemas elementales", la solución del problema general viene determinada de forma inmediata por la expresión

$$P(x) = \sum_{k=0}^{n} y_k \ell_k(x).$$

Obsérvese que, si f es una función definida sobre un intervalo que contiene a los nodos $\{x_k\}_{k=0}^n$, entonces el teorema de interpolación de Lagrange garantiza la existencia de un único polinomio $P(x)$ de grado $\leq n$ (cosa que denotamos en lo sucesivo por $P \in \Pi_n$) tal que

$$f(x_k) = P(x_k) \text{ para } k = 0, 1, \cdots, n.$$

Es habitual denotar a dicho polinomio por $P = L_n(f)$.

Segunda demostración del teorema de interpolación de Lagrange:
Buscamos un polinomio $L_n(f)(x) = a_0 + a_1x + \cdots + a_nx^n \in \Pi_n$ tal que $L_n(f)(x_i) = f(x_i)$ $(i = 0, 1, \cdots, n)$. Ahora bien, este problema se puede intepretar como la búsqueda de los coeficientes a_i, $i = 0, \cdots, n$ para los que se satisface la siguiente ecuación lineal:

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & & \ddots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{pmatrix}$$

El determinante de la matriz que define la ecuación anterior se puede calcular de la siguiente forma: primero tratamos el nodo x_0 como una variable independiente (i.e., hacemos $x_0 = t$). Si desarrollamos el correspondiente determinante utilizando la primera fila de la matriz, obtenemos un polinomio de grado $\leq n$:

$$V(t) = V(t, x_1, \cdots, x_n) = \det \begin{pmatrix} 1 & t & t^2 & \cdots & t^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & & \ddots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix} = B_0 + B_1t + \cdots + B_nt^n$$

Por otra parte, $V(x_1) = \cdots = V(x_n) = 0$. Esto implica que $V(t)$ admite una descomposición de la forma:

$$V(t) = c(t - x_1)(t - x_2)\cdots(t - x_n)$$

Si $c \neq 0$ entonces $V(x_0) = c(x_0 - x_1)(x_0 - x_2)\cdots(x_0 - x_n) \neq 0$ y, por tanto, el sistema tiene una única solución (que es lo que deseamos probar). Tenemos, pues, que demostrar que $c \neq 0$. Ahora bien: esto se puede hacer por inducción sobre el orden de la matriz. De hecho,

$$c = B_n = (-1)^n \det \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & & \ddots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{pmatrix} = (-1)^n V(x_1, \cdots, x_n) \neq 0.$$

Esto concluye la prueba. □
Evidentemente, la demostración que acabamos de dar para la existencia (y unicidad) del polinomio de interpolación de Lagrange $L_n(f)$ no proporciona un buen método para el cálculo de dicho polinomio, ya que las matrices que aparecen son complicadas (contienen demasiados términos no nulos). De hecho, es posible demostrar que se trata de sistemas mal condicionados. En un lenguaje propio del Álgebra Lineal, podemos decir que la base escogida $\{1, x, x^2, \cdots, x^n\}$ para representar polinomios $p(x) \in \Pi_n$ no es precisamente la mejor posible de cara a resolver nuestro problema. Es más, si atendemos a la primera demostración que se ofreció de este teorema, en ella se demuestra que los polinomios fundamentales de Lagrange $\{\ell_0(x), \ell_1(x), \cdots, \ell_n(x)\}$ forman una base mucho mejor adaptada para el cálculo de $L_n(f)$, pues

$$L_n(f)(x) = \sum_{k=0}^{n} f(x_k)\ell_k(x).$$

En términos de Álgebra Lineal, lo que sucede es que al utilizar los polinomios fundamentales de Lagrange como base del espacio vectorial Π_n, se ha logrado "diagonalizar" el sistema lineal de ecuaciones que surge al tratar de buscar el polinomio de interpolación de Lagrange. Hay, sin embargo, también algunas objeciones al método que acabamos de describir. La más importante consiste en que la base $\{\ell_i(x)\}_{i=0}^n$ no está preparada para añadir más información. Es decir, si queremos utilizar el valor de la función en un nuevo nodo x_{n+1} entonces tendremos que modificar completamente todos los polinomios fundamentales de Lagrange.

Si tomamos como base de Π_n los polinomios

$$\{1, x-x_0, (x-x_0)(x-x_1), \cdots, (x-x_0)(x-x_1)\cdots(x-x_{n-1})\}$$

en el caso de nodos equiespaciados, $x_0 = 0$; $x_i = x_0 + ih$, $i \leq n$ se satisface la siguiente fórmula:

$$\begin{aligned} L_n(f)(x) &= \sum_{k=0}^{n} \frac{(\Delta^k L_n(f))(0)}{k!} x(x-h)(x-2h)\cdots(x-(k-1)h) \\ &= \sum_{k=0}^{n} \frac{(\Delta^k f)(0)}{k!} x(x-h)(x-2h)\cdots(x-(k-1)h) \end{aligned}$$

¿Qué pasará para elecciones arbitrarias de nodos?. Se tiene el siguiente resultado:

Teorema 2.10 (Newton)

$$L_n(f) = \sum_{k=0}^{n} f[x_0, x_1, ..., x_k] w_k(x)$$

donde

$$w_k(x) = \begin{cases} (x-x_0)(x-x_1)...(x-x_{k-1}) & \text{si } k > 0 \\ 1 & \text{si } k = 0 \end{cases}$$

y

$$\begin{cases} f[x_i] &= f(x_i) \\ f[x_{i_0}, x_{i_1}, ..., x_{i_k}] &= \frac{f[x_{i_0}, x_{i_1}, ..., x_{i_{k-1}}] - f[x_{i_1}, x_{i_1}, ..., x_{i_k}]}{x_{i_0} - x_{i_k}} \end{cases}$$

N Obsérvese que la notación empleada para los coeficientes $f[x_0, x_1, ..., x_k]$ que aparecen en la expansión de $L_n(f)$ indica que $f[x_0, x_1, ..., x_k]$ sólo depende del valor de f en los nodos $\{x_0, x_1, \cdots, x_k\}$. Además, si truncamos la suma en el k-ésimo término, obtenemos el polinomio de interpolación de Lagrange para los nodos $\{x_0, x_1, \cdots, x_k\}$ y si añadimos la información en un nuevo nodo x_{n+1}, el nuevo polinomio de interpolación, que denotamos por $L_{n+1}(f)$, se escribe como $L_{n+1}(f) = L_n(f) + f[x_0, \cdots, x_{n+1}]w_{n+1}(x)$ y, por tanto, su cálculo sólo requiere el cálculo de un nuevo coeficiente $f[x_0, \cdots, x_{n+1}]$, y la información anterior se aprovecha completamente (sin cambios). Llamamos diferencia dividida de f en los nodos $\{x_0, x_1, ..., x_n\}$ al valor $f[x_0, \cdots, x_n]$.

Demostración del Teorema de Newton. Ahora vamos a abordar la prueba del teorema. Para ello, comenzamos observando que, si $p_{0,1,...,n-1}(t)$ denota el polinomio de interpolación a f en los nodos $\{x_0, x_1, ..., x_{n-1}\}$ y $p_{1,...,n}(t)$ denota el polinomio de interpolación a f en los nodos $\{x_1, ..., x_n\}$, entonces se tiene la siguiente identidad (demostrada por Aitken):

$$L_n(f)(t) = \frac{(x_n - t)p_{1,...,n}(t) - (x_0 - t)p_{0,1,...,n-1}(t)}{x_n - x_0}.$$

Esto es fácil de probar. Basta observar que

$$\begin{aligned}&(x_n - x_k)p_{1,\dots,n}(x_k) - (x_k - x_0)p_{0,1,\dots,n-1}(t) = \\ &= \begin{cases} (x_n - x_0)f(x_0) & k = 0 \\ (x_n - x_k - x_0 + x_k)f(x_k) & 1 \leq k \leq n-1 \\ -(x_0 - x_n)f(x_n) & k = n \end{cases} \\ &= (x_n - x_0)f(x_k),\ k = 0, 1, ..., n.\end{aligned}$$

Ahora, el significado de la fórmula de Aitken es que podemos reducir el cálculo del polinomio de interpolación en $n+1$ nodos cualesquiera, al cálculo de dicho polinomio sobre n nodos cualesquiera. En particular, vamos a probar que existe una forma recurrente para el cálculo de las diferencias divididas $f[x_0, \cdots, x_k]$, $k = 0, 1, ..., n$.
Es evidente que existen coeficientes $\{\alpha_0, \alpha_1, ..., \alpha_n\} \subset \mathbb{R}$ tales que

$$L_n(f) = \sum_{k=0}^{n} \alpha_k w_k(x),$$

pues $\{w_k(x)\}_{k=0}^n$ es una base de Π_n. Lo que no resulta completamente obvio es que el coeficiente $\alpha_k (= f[x_0, \cdots, x_k])$, dependa exclusivamente de la tabla de valores $\{(x_i, f(x_i))\}_{i=0}^k$, o que se pueda calcular con un cierto algoritmo. Ahora bien, si nos fijamos precisamente en el coeficiente α_n, nos damos en seguida cuenta de que

$$\begin{aligned}L_n(f) &= \alpha_0 + \alpha_1(t - x_0) + \cdots + \alpha_n(t - x_0)(t - x_1)\cdots(t - x_{n-1}) \\ &= \alpha_n t^n + (\text{pol. grado } \leq n-1)\end{aligned}$$

y, por tanto, podemos definir $f[x_0, \cdots, x_n] = \alpha_n$ como el coeficiente leader del polinomio de interpolación de f en los nodos $\{x_0, x_1, ..., x_n\}$. Evidentemente, esta definición implica que $f[x_0, \cdots, x_n] = f[x_{i_0}, \cdots x_{i_n}]$ para cualquier permutación $(i_0, i_1, ..., i_n)$ de $\{0, 1, ..., n\}$ (pues el polinomio de interpolación no depende de la ordenación de los nodos). Por otra parte, se sigue de la fórmula de Aitken, que

$$\begin{aligned}&L_n(f)(t) = \\ &= \tfrac{(x_n - t)(f[x_1, x_2, \dots, x_n]t^{n-1} + (pol.grad \leq n-2)) - (x_0 - t)(f[x_0, x_1, \dots, x_{n-1}]t^{n-1} + (pol.grad \leq n-2))}{x_n - x_0} \\ &= \tfrac{f[x_1, x_2, \dots, x_n] - f[x_0, x_1, \dots, x_{n-1}]}{x_n - x_0} t^n + (\text{pol. grado } \leq n-1))\end{aligned}$$

y, por tanto,

$$f[x_0, x_2, ..., x_n] = \frac{f[x_1, x_2, ..., x_n] - f[x_0, x_1, ..., x_{n-1}]}{x_n - x_0}.$$

Esto concluye la prueba del teorema. □
Como hemos visto, el polinomio $L_n(f)$ admite varias representaciones distintas. Esto implica que un método numérico cuyo análisis (i.e. el análisis de su estabilidad, convergencia, etc...) está basado en una representación concreta (e.g. representación de Newton) puede ser implementado usando otra representación diferente (e.g. representación de Lagrange). Por tanto, son interesantes las diferentes formas de escribir el polinomio de interpolación de Lagrange $L_n(f)$.

Estimación del error
Consideremos la representación de Newton del polinomio de interpolación de Lagrange:

$$L_n(f, \{x_k\}_{k=0}^n)(x) = \sum_{k=0}^{n} f[x_0, x_1, ..., x_k]w_k(x).$$

Fijado un nuevo valor $x^* \in \mathbb{R} \setminus \{x_k\}_{k=0}^n$, podemos afirmar que se satisface la siguiente identidad:

$$\begin{aligned}L_{n+1}(f)(x) &= L_{n+1}(f, \{x_k\}_{k=0}^n \cup \{x^*\})(x) \\ &= L_n(f, \{x_k\}_{k=0}^n)(x) + f[x_0, x_1, ..., x_n, x^*](x - x_1)\cdots(x - x_n)\end{aligned}$$

Además,

$$\begin{aligned} 0 &= f(x^*) - L_{n+1}(f)(x^*) \\ &= f(x^*) - L_n(f)(x^*) - f[x_0,x_1,...,x_n,x^*](x^* - x_1)(x^* - x_2)\cdots(x^* - x_n). \end{aligned}$$

Por tanto,

$$f(x^*) - L_n(f)(x^*) = f[x_0,x_1,...,x_n,x^*](x^* - x_1)(x^* - x_2)\cdots(x^* - x_n).$$

Como x^* fue fijado de forma arbitraria, hemos demostrado el siguiente resultado:

Teorema 2.11

Sea $L_n(f)$ el polinomio de interpolación de Lagrange a la función f en los nodos $\{x_k\}_{k=0}^n$. Entonces se satisface la relación

$$f(x) - L_n(f)(x) = f[x_0,x_1,...,x_n,x](x - x_1)(x - x_2)\cdots(x - x_n)$$

para todo $x \in \mathbb{R}$.

Para funciones diferenciables se tiene el siguiente resultado:

Teorema 2.12

Supongamos que la función es de clase $n+1$. Entonces

$$\|f - L_n(f)\|_{\mathbf{C}[a,b]} \leq \frac{\left\|f^{(n+1)}\right\|_{\mathbf{C}[a,b]}}{(n+1)!} \|\pi_n(x)\|_{\mathbf{C}[a,b]}, \text{ donde } \pi_n(x) = \prod_{k=0}^{n}(x - x_k).$$

Demostración. Por definición de $L_n(f)$, sabemos que

$$R(x) = f(x) - L_n(f)(x)$$

se anula en los nodos $\{x_k\}_{k=0}^n$. Aplicando el teorema de Rolle, obtenemos que

$$R'(x) = f'(x) - L_n(f)'(x)$$

tiene al menos n ceros distintos en el intervalo $(\text{mín}\{x_k\}_{k=0}^n, \text{máx}\{x_k\}_{k=0}^n) \subset (a,b)$. El mismo argumento sirve para demostrar que $R''(x)$ se anula en al menos $n-1$ puntos en el m ismo intervalo. Aplicando esto reiteradamente, obtenemos que $R^{(n)}(\xi) = 0$ para cierto valor $\xi \in [a,b]$. Ahora bien:

$$\begin{aligned} & R^{(n)}(\xi) = f^{(n)}(\xi) - L_n(f)^{(n)}(\xi) \\ = \; & f^{(n)}(\xi) - (q_{n-1}(x) + f[x_0,x_1,...,x_n]x^n)^{(n)}(\xi), \text{ para cierto } q_{n-1} \in \Pi_{n-1} \\ = \; & f^{(n)}(\xi) - n!f[x_0,x_1,...,x_n] \end{aligned}$$

Esto demuestra que si f es de clase al menos n en el intervalo $[a,b]$, entonces

$$f[x_0,x_1,...,x_n] = \frac{f^{(n)}(\xi)}{n!}$$

para cierto $\xi \in (a,b)$. En particular, si $f \in \mathbf{C}^{(n+1)}[a,b]$ y $x \in [a,b]$, entonces

$$f[x_0,x_1,...,x_n,x] = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}$$

para cierto $\xi = \xi_x \in (a,b)$. Ahora, teniendo en cuenta la fórmula del error obtenida en el teorema anterior, se ve que

$$\begin{aligned} R(x) &= f(x) - L_n(f)(x) = f[x_0, x_1, ..., x_n, x]\pi_n(x) \\ &= \frac{f^{(n+1)}(\xi_x)}{(n+1)!}\pi_n(x) \end{aligned}$$

y, por tanto,

$$\|f - L_n(f)\|_{\mathbf{C}[a,b]} \leq \frac{\left\|f^{(n+1)}\right\|_{\mathbf{C}[a,b]}}{(n+1)!} \|\pi_n(x)\|_{\mathbf{C}[a,b]},$$

que es lo que queríamos demostrar. □

Se deduce del resultado anterior que para la estimación del error de interpolación no sólo es importante estimar el tamaño de las derivadas sucesivas de la función a interpolar sino que, además, existe una (importante) influencia de la elección de los nodos de interpolación. De hecho, la estimación de la norma uniforme $\|\pi_n(x)\|_{\mathbf{C}[a,b]}$ es difícil incluso en casos sencillos. Por ejemplo, si tomamos nodos equiespaciados, se tiene que

$$\|\pi_n(x)\|_{\mathbf{C}[a,b]} = \max_{x\in[a,b]} \prod_{k=0}^{n} |x - a - kh|,$$

donde $h = (b-a)/n$.

Si tomamos un intervalo $[a,b]$ de longitud $b - a < 1$, entonces para elecciones arbitrarias de nodos $\{x_i\}_{i=0}^n \subset [a,b]$ se tiene que

$$\|\pi_n(x)\|_{\mathbf{C}[a,b]} = \max_{x\in[a,b]} \prod_{k=0}^{n} |x - x_k| \leq (b-a)^{n+1} \to 0, (n \to \infty)$$

y por tanto incluso para funciones f cuyas derivadas sucesivas crecen en norma infinito muy rápido, es posible garantizar la convergencia uniforme de la sucesión de polinomios de interpolación a la función de partida.

Una pregunta importante es la siguiente: ¿qué elección de nodos $\{x_i\}_{i=0}^n$ minimiza el tamaño de $\|\pi_n(x)\|_{\mathbf{C}[a,b]}$?. Para responder a esta cuestion es necesario introducir la siguiente familia de polinomios:

Llamamos a $T_n(x) = \cos(n \arccos x)$ el n-ésimo polinomio de Tchebychev.

N La función $T_n(x)$ definida arriba es un polinomio de grado exactamente n. De hecho,

$$\begin{cases} T_0(x) = 1, T_1(x) = x \\ T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \quad \text{para } n \geq 1. \end{cases}$$

(estas identidades son fáciles de comprobar) y, por tanto,

$$T_n(x) = 2^{n-1}x^n + \{\text{polinomio de grado } \leq n-1\}, n = 1, 2, \cdots$$

Definimos el n-ésimo polinomio mónico de Tchebychev como

$$T_n^*(x) = \frac{1}{2^{n-1}} T_n(x), n = 1, 2, \cdots$$

Se satisface el siguiente resultado:

Teorema 2.13

Para cada $n \in \mathbb{N}$ se tiene que

$$\frac{1}{2^{n-1}} = \|T_n^*(x)\|_{[-1,1]} = \min_{q(x)\in\Pi_{n-1}} \|x^n - q(x)\|_{[-1,1]}\,.$$

Además, si $[a,b] \subset \mathbb{R}$ es un intervalo cerrado (arbitrario), entonces:

$$\min_{q(x)\in\Pi_{n-1}} \|x^n - q(x)\|_{[a,b]} = \left(\frac{b-a}{2}\right)^n \frac{1}{2^{n-1}}$$

Demostración. Si tomamos $x_k = \cos(\frac{\pi k}{n})$, entonces

$$\begin{aligned} T_n^*(x_k) &= \frac{1}{2^{n-1}} T_n(x_k) = \frac{1}{2^{n-1}} \cos(n \arccos\cos(\frac{\pi k}{n})) \\ &= \frac{1}{2^{n-1}} \cos(\pi k) = \frac{(-1)^k}{2^{n-1}},\ k = 0,1,2,...,n. \end{aligned}$$

Si $p(t)$ es un polinomio mónico de grado n tal que $||q(t)||_{[-1,1]} < \frac{1}{2^{n-1}} = \|T_n^*(x)\|_{[-1,1]}$, entonces

$$\mathbf{sign}(T_n^*(x_k) - q(x_k)) = \mathbf{sign}(T_n^*(x_k)) = (-1)^k,\ k = 0,1,...,n$$

Pero esto implica obviamente que $h(t) = T_n^*(t) - q(t)$ tiene al menos n ceros y, por tanto, $h(t)$ es identicamente nula (al ser un polinomio de grado a lo sumo $n-1$): una contradicción.
El caso de un intervalo general sale de observar que $T(t) = \frac{b-a}{2}t + \frac{a+b}{2}$ es una biyección que envía el intervalo $[-1,1]$ al intervalo $[a,b]$. Por tanto, todo polinomio $p(x) = x^n - h(x)$ con $h \in \Pi_{n-1}$ se puede representar como

$$p(x) = p\left(\frac{b-a}{2}t + \frac{a+b}{2}\right) = \left(\frac{b-a}{2}t + \frac{a+b}{2}\right)^n - h\left(\frac{b-a}{2}t + \frac{a+b}{2}\right) = \left(\frac{b-a}{2}\right)^n t^n - s(t)$$

para cierto $s \in \Pi_{n-1}$. Por tanto,

$$\begin{aligned} \min_{h\in\Pi_{n-1}} \|x^n - h(x)\|_{[a,b]} &= \min_{s\in\Pi_{n-1}} \| \left(\frac{b-a}{2}\right)^n t^n - s(t)\|_{[-1,1]} \\ &= \left(\frac{b-a}{2}\right)^n \min_{s\in\Pi_{n-1}} \|t^n - s(t)\|_{[-1,1]} \\ &= \left(\frac{b-a}{2}\right)^n \frac{1}{2^{n-1}} \end{aligned}$$

□

Algunos comentarios sobre convergencia
Para definir un proceso de interpolación con expectativas de estudiar su convergencia, necesitamos una matriz triangular inferior infinita cuyas entradas son tales que la fila $(n+1)$-ésima de la matriz, viene dada por $\{x_0^{(n)}, x_1^{(n)}, ..., x_n^{(n)}\}$, los nodos del n-ésimo polinomio interpolador $L_n(f)$. Es decir,

$$X = \begin{bmatrix} x_0^{(0)} & & & \\ x_0^{(1)} & x_1^{(1)} & & \\ \vdots & & \ddots & \\ x_0^{(n)} & \cdots & & x_n^{(n)} \\ & & & & \ddots \end{bmatrix}$$

y considerar la sucesión de operadores lineales $L_n : \mathbf{C}[a,b] \to \mathbf{C}[a,b]$, $f \to L_n(f)$. Podemos entonces acotar

$$\|f - L_n(f)\|_{\mathbf{C}[a,b]} \leq (\|L_n\| + 1)E_n(f)$$

en términos de las normas $\Lambda_n = \|L_n\|$ de dichos operadores y los errores de mejor aproximación en norma uniforme $E_n(f) = \min_{q \in \Pi_n} ||f - q||$. De hecho, las constantes Λ_n se pueden calcular

Teorema 2.14

$$\Lambda_n = \Lambda_n(X) = \max_{x \in [a,b]} \sum_{i=0}^{n} | \prod_{j=0,\ j \neq i}^{n} \frac{x - x_j}{x_i - x_j} |$$

Teorema 2.15 (Faber, 1914)

$$\Lambda_n(X) \geq \frac{2}{\pi^2} \log n - 1$$

Corolario 2.1 (Faber, 1914) Para cualquier matriz infinita de nodos X, existen funciones continuas $f \in \mathbf{C}[a,b]$ para las que $\{L_n(f)\}_{n=0}^{\infty}$ no converge a f.

Por otra parte, también es cierto el siguiente resultado:

Teorema 2.16 (Marcinkiewicz, 1937)

Dada $f \in \mathbf{C}[a,b]$ siempre existe una matriz de nodos X tal que la correspondiente sucesión de polinomios interpoladores $L_n(f)$ converge uniformemente en $[a,b]$ a f.

También se han estudiado algunos ejemplos que responden a fenómenos de convergencia intermedia, como el siguiente ejemplo debido a Runge:

Ejemplo 2.1 (Runge). Consideremos nodos equiespaciados sobre el intervalo $[-1,1]$ y sea $\{L_n(f)\}_{n=0}^{\infty}$ la correspondiente sucesión de polinomios de interpolación para la función $f(t) = (1 + 25t^2)^{-1}$. Entonces existe una constante $\alpha \in (0,1)$ (cuyos primeros dígitos son $\alpha = 0{,}7266...$) tal que

$$\lim_{n \to \infty} |f(t) - L_n(f)(t)| = \begin{cases} 0 & |t| < \alpha = 0{,}7266... \\ \infty & |t| > \alpha = 0{,}7266... \end{cases}$$

Ejemplo 2.2 (Bernstein). Tomamos también nodos equiespaciados en $[-1,1]$ para la interpolación de $f(t) = |t|$. Entonces

$$\lim_{n \to \infty} |f(t) - L_n(f)(t)| = \begin{cases} 0 & t \in \{-1, 0, 1\} \\ \infty & \text{otro caso.} \end{cases}$$

Sorprendentemente si utilizamos los nodos de Tchebychev entonces es posible demostrar la convergencia de la correspondiente sucesión de polinomios de interpolación si suponemos que las funciones son derivables de clase uno. Más precisamente, se puede demostrar (aunque nosotros no lo haremos aquí) el siguiente resultado:

Teorema 2.17

Si

$$X = \begin{bmatrix} 0 & & & & \\ 1 & -1 & & & \\ \vdots & & \ddots & & \\ \cos(\frac{\pi 0}{n}) & \cos(\frac{\pi}{n}) & \cdots & \cos(\frac{\pi n}{n}) & \\ & & & & \ddots \end{bmatrix}$$

y $f \in \mathbf{C}^1[-1,1]$, entonces $\lim_{n\to\infty} ||f(t) - L_n(f)(t)|\,|_{[-1,1]} = 0$.

Finalmente, es posible obtener un resultado análogo al dado por el ejemplo de Bernstein mencionado anteriormente. Se trata del siguiente[1]

Teorema 2.18 (Ërdos-Turán)

Para cualquier elección de la tabla de nodos X, (con la única restricción de que $x_i^{(n)} \in [a,b]$ para todo $i \leq n$ y todo $n \in \mathbb{N}$), existe una función f tal que $L_n(f)$ diverge en un subconjunto denso de $[a,b]$.

2.3.1.3. Interpolación trigonométrica

De la misma forma que es posible utilizar polinomios algebraicos para la interpolación de funciones, también podemos usar polinomios trigonométricos. En particular, no es difícil comprobar que las funciones

$$t_i(t) = \prod_{j=0;j\neq i}^{2n} \frac{\sin\frac{1}{2}(t-x_j)}{\sin\frac{1}{2}(t_i-x_j)}$$

juegan un papel similar al de los polinomios fundamentales de Lagrange en el caso algebraico. Así pues, podemos afirmar que el polinomio

$$T_n(f) = \sum_{i=0}^{2n} f(x_i)t_i(t) \in \mathbf{span}\{\cos kt, \sin kt\}_{k=0}^{n}$$

interpola a $f(x)$ en los nodos $\{x_i\}_{i=0}^{2n} \subset [0, 2\pi)$.
Por otra parte, si tenemos en cuenta la conocida fórmula de Euler, $e^{\mathbf{i}\theta} = \cos\theta + \mathbf{i}\sin\theta$, todo polinomio trigonométrico de grado $\leq n$,

$$p_n(t) = a_0 + \sum_{k=1}^{n}(a_k\cos(kt) + b_k\sin(kt))$$

admite una expresión de la forma

$$p_n(t) = \sum_{k=-n}^{n} c_k e^{\mathbf{i}kt};$$

donde

$$c_0 = a_0,\ c_{-k} = \overline{c_k},\ c_k = \frac{a_k + \mathbf{i}b_k}{2};\ k = 1, 2, ..., n.$$

[1] En realidad, el Teorema de Ërdos-Turán, garantiza divergencia casi por doquier, pero este resultado no se puede enunciar en un curso de este nivel, pues no se han introducido conceptos de teoría de la medida.

Ahora, si hacemos $z = e^{\mathbf{i}t}$, entonces $P_n(z) := p_n(t) = \sum_{k=-n}^{n} c_k z^k$ y, por tanto, $z^n P_n(z) \in \Pi_{2n}$. Esto implica que podemos utilizar el teorema de interpolación de Lagrange para demostrar la existencia y unicidad del polinomio de interpolación trigonométrica.

N En el caso de nodos equiespaciados, $x_k = \frac{2\pi k}{2n+1}$, $k = 0, 1, ..., 2n$, podemos hacer uso, para el cálculo de $T_n(f)$, del siguiente resultado:

Lema 2.1

Sea $j \in \mathbb{Z}$ y supongamos que $x_k = \frac{2\pi k}{2n+1}$, $k = 0, 1, ..., 2n$. Entonces

$$\sum_{k=0}^{2n} e^{\mathbf{i}jx_k} = \begin{cases} 2n+1 & \text{si } j \in (2n+1)\mathbb{Z} \\ 0 & \text{otro caso.} \end{cases}$$

Ahora, para el cálculo del polinomio de interpolación trigonométrica $T_n(f)$, podemos hacer uso de que éste admite una expresión de la forma

$$T_n(f)(t) = \sum_{k=-n}^{n} c_k e^{\mathbf{i}kt},$$

y del Lema anterior. Así pues, si forzamos las identidades

$$T_n(f)(x_j) = \sum_{k=-n}^{n} c_k e^{\mathbf{i}kx_j} = f(x_j);\ j = 0, 1, ..., 2n,$$

multiplicamos la ecuación j-ésima anterior por $e^{-\mathbf{i}mx_j}$ y sumamos todas las ecuaciones, obtenemos que:

$$\sum_{j=0}^{2n} \sum_{k=-n}^{n} c_k e^{\mathbf{i}(k-m)x_j} = \sum_{j=0}^{2n} e^{-\mathbf{i}mx_j} f(x_j)$$

para $m \in \{-n, -n+1, ..., 0, 1, ..., n\}$. Intercambiando el orden de sumación en el primer miembro de la igualdad, se tiene que

$$\begin{aligned} \sum_{j=0}^{2n} e^{-\mathbf{i}mx_j} f(x_j) &= \sum_{k=-n}^{n} \left(c_k \sum_{j=0}^{2n} e^{\mathbf{i}(k-m)x_j} \right) \\ &= \sum_{k=-n}^{n} c_k \begin{cases} 2n+1 & \text{si } k = m \\ 0 & \text{otro caso.} \end{cases} \\ &= (2n+1)c_m \end{aligned}$$

Se sigue que

$$c_k = \frac{1}{2n+1} \sum_{j=0}^{2n} e^{-\mathbf{i}kx_j} f(x_j);\ k = -n, -n+1, ..., 0, 1, ..., n.$$

Por tanto, otra expresión de $T_n(f)$ es:

$$T_n(f)(t) = \frac{1}{2n+1} \sum_{k=-n}^{n} \left(\sum_{j=0}^{2n} e^{-\mathbf{i}kx_j} f(x_j) \right) e^{\mathbf{i}kt}.$$

La aplicación $f \to \mathcal{F}(f) = (c_{-n}(f), c_{-n+1}(f), ..., c_0(f), c_1(f), ..., c_n(f))$, donde

$$c_k(f) = \frac{1}{2n+1} \sum_{j=0}^{2n} e^{-\mathbf{i}kx_j} f(x_j),\ -n \leq k \leq n,$$

recibe el nombre de transformada discreta de Fourier de la función 2π-periódica $f(t)$. Esta transformada es especialmente importante para las aplicaciones en teoría de señales y, por tanto, su cálculo mediante algoritmos que reduzcan la complejidad computacional, es un tema ampliamente estudiado. De hecho, existen varios de estos algoritmos que reciben el nombre genérico de Transformadas Rápidas de Fourier[2]

2.3.1.4. Interpolación de Hermite

El problema de interpolación de Hermite consiste en la búsqueda de un polinomio $H_n(f)$ (de grado $\leq 2n+1$) que coincida con la función f y con su primera derivada en un conjunto de $n+1$ nodos $\{x_0, x_1, \cdots, x_n\}$.
¿Por qué deberíamos utilizar derivadas de funciones, para su aproximación por polinomios?. En principio, experimentalmente las derivadas de una función son difíciles de obtener y, por tanto, su uso en fórmulas de interpolación debería resultar un tanto inútil. Sin embargo, si se conoce una expresión matemática de la función, entonces sus derivadas son fáciles de obtener. Por supuesto que dichas derivadas responden a fórmulas cada vez más largas, que crecen y crecen de tamaño conforme derivamos la función (al menos, esta es nuestra experiencia cuando derivamos funciones elementales): sólo en ciertos casos es posible encontrar una expresión cerrada para la derivada n-ésima de una función. Sin embargo, este problema no es importante si tenemos en cuenta la capacidad de cálculo que tienen los computadores modernos. Otra propiedad importante de la derivación es la siguiente: en general, derivar una función elemental (un polinomio, una función trigonométrica, una raiz, etc) no implica la aparición de funciones nuevas (i.e., funciones muy diferentes de la inicial), sino la ejecución de nuevas operaciones sobre la misma función. Esto es importante, si se tiene en cuenta que, en general, la evaluación de la función inicial en cada punto suele ser computacionalmente costosa (puede llevar 50 o 60 pasos en un algoritmo interno del ordenador). Por tanto, si almacenamos el valor de la función en cada nodo y lo utilizamos para el cálculo de las derivadas sucesivas de ésta en dichos nodos, entonces ganaremos en tiempo de ejecución de los algoritmos, en comparación con la evaluación de la función en otros nodos nuevos.
Para resolver el problema de interpolación de Hermite hacemos un planteamiento similar al propuesto por Lagrange para la interpolación simple de funciones. Es decir, nos preocupamos antes de resolver los problemas más sencillos dados por:

- Encontrar polinomios $\sigma_i \in \Pi_{2n+1}$ tales que

$$\sigma_i(x_j) = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases} \text{ y } \sigma_i'(x_j) = 0 \text{ para todo } j \leq n$$

- Encontrar polinomios $\tau_i \in \Pi_{2n+1}$ tales que

$$\tau_i(x_j) = 0 \text{ para todo } j \leq n \text{ y } \tau_i'(x_j) = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases}$$

Evidentemente, si pudiésemos resolver los problemas anteriores entonces la solución general del problema de interpolación de Hermite estará dada por:

$$H_n(f) = \sum_{i=0}^{n} f(x_i)\sigma_i(x) + \sum_{i=0}^{n} f(x_i)\tau_i(x)$$

[2]Nótese que llamamos 'transformada' a lo que en realidad es un algoritmo para su cálculo.

ya que, si $H_n(f)$ está definido por la fórmula anterior, entonces

$$\begin{aligned} H_n(f)(x_j) &= \sum_{i=0}^n f(x_i)\sigma_i(x_j) + \underbrace{\sum_{i=0}^n f'(x_i)\tau_i(x_j)}_{\text{Esta parte se anula, pues } \tau_i(x_j)=0} \\ &= f(x_j) \end{aligned}$$

y

$$\begin{aligned} H_n'(f)(x_j) &= \underbrace{\sum_{i=0}^n f(x_i)\sigma_i'(x_j)}_{\text{Esta parte se anula, pues } \sigma_i'(x_j)=0} + \sum_{i=0}^n f'(x_i)\tau_i'(x_j) \\ &= f'(x_j) \end{aligned}$$

¿Cómo calculamos los polinomios fundamentales de Hermite σ_i, τ_i $(i \leq n)$?. Veamos cómo es el cálculo de los polinomios $\sigma_i(x)$. Tomamos, para empezar, un polinomio de la forma

$$p_i(x) = \pi_i(x)^2(x-b), \text{ donde } \pi_i(x) = \prod_{j\leq n;\ (j\neq i)} (x-x_j) \text{ y } b \notin \{x_i\}_{i=0}^n.$$

Entonces $p_i(x_j) = 0$ si $j \neq i$ y $p_i(x_i) \neq 0$. Además,

$$p_i'(x) = 2\pi_i(x)\pi_i'(x)(x-b) + \pi_i(x)^2.$$

Queremos determinar el valor de b de modo que $0 = p_i'(x_i) = 2\pi_i(x_i)\pi_i'(x_i)(x_i-b) + \pi_i(x_i)^2$, $i = 1, 2, \cdots, n$. La ecuaciones se simplifican si dividimos por $p_i(x_i)$, ya que

$$\frac{\pi_i'(x_i)}{\pi_i(x_i)} = \sum_{j\leq n;\ j\neq i} \frac{1}{x_i - x_j}.$$

Buscamos b tal que

$$\begin{aligned} 0 &= \frac{p_i'(x_i)}{p_i(x_i)} = \frac{2\pi_i(x_i)\pi_i'(x_i)(x_i-b) + \pi_i(x_i)^2}{\pi_i(x_i)^2(x_i-b)} \\ &= 2\frac{\pi_i'(x_i)}{\pi_i(x_i)} + \frac{1}{x_i-b} \\ &= 2\sum_{j\leq n;\ j\neq i} \frac{1}{x_i-x_j} + \frac{1}{x_i-b} \end{aligned}$$

Es decir,

$$b = x_i + \frac{1}{2\sum_{j\leq n;\ j\neq i} \frac{1}{x_i-x_j}}$$

Así, el polinomio

$$p_i(x) = \pi_i(x)^2 \left(x - x_i - \frac{1}{2\sum_{j\leq n;\ j\neq i} \frac{1}{x_i-x_j}} \right)$$

satisface

$$p_i(x_j) = \begin{cases} \neq 0 & j = i \\ 0 & j \neq i \end{cases} \text{ y } p_i'(x_j) = 0 \text{ para todo } j \leq n.$$

Esto implica que

$$\sigma_i(x) = \frac{p_i(x)}{p_i(x_i)}, i = 0, 1, 2, \cdots, n$$

son los polinomios que buscábamos. (Para los polinomios τ_i es posible encontrar una expresión utilizando argumentos similares: ¡Ejercicio!).

Estimación del error

La estimación del error en la fórmula de interpolación de Hermite es análoga a la estimación obtenida para el problema de interpolación de Lagrange. Veámoslo:
Evidentemente, $R(x) = f(x) - H_n(x)$ es una función que posee ceros dobles en los nodos $\{x_i\}_{i=0}^n$. Por tanto, podemos escribir

$$R(x) = f(x) - H_n(x) = w_n(x)^2 K(x),$$

donde $w_n(x) = (x - x_0) \cdots (x - x_n)$ y $K(x)$ es una función desconocida. Tomamos $x^* \notin \{x_i\}_{i=0}^n$ arbitrario. Entonces $f(x^*) = H_n(x^*) + w_n(x^*)^2 K(x^*)$ y la función

$$\Phi(x) = f(x) - H_n(x) - w_n(x)^2 K(x^*)$$

posee un cero doble en cada uno de los nodos $\{x_i\}_{i=0}^n$ y también se anula en el punto x^*. Por tanto, $\Phi'(x)$ posee al menos $2n + 2$ ceros, $\Phi^{''}(x)$ posee al menos $2n + 1$ ceros, etc. Finalmente, la derivada

$$\Phi^{(2n+2)}(x) = f^{(2n+2)}(x) - (2n + 2)! K(x^*)$$

posee al menos un cero $\overline{x}$. Esto implica que

$$K(x^*) = \frac{f^{(2n+2)}(\overline{x})}{(2n + 2)!}$$

para cierto valor $\overline{x}$. Como x^* se tomó fijo pero arbitrario, podemos afirmar que

$$R(x) = f(x) - H_n(x) = w_n(x)^2 \frac{f^{(2n+2)}(\overline{x})}{(2n + 2)!}.$$

Esta es la expresión del error para la fórmula de interpolación de Hermite. Una consecuencia evidente es que

$$|R(x)| \leq w_n(x)^2 \frac{\left\| f^{(2n+2)} \right\|_{[a,b]}}{(2n + 2)!} \text{ , para todo } x \in [a, b].$$

Una de las razones por las que las fórmulas anteriores son de interés es que $w_n(x)^2 \geq 0$ para todo $x \in \mathbb{R}$.

Interpolación de Birkhoff

El problema de interpolación de Hermite generalizado (también llamado interpolación de Birkhoff) consiste en la búsqueda de un polinomio (de grado adecuado) que coincida con la función f y con algunas de sus derivadas hasta un orden $k_i < \infty$ (que puede depender del nodo x_i) para unos cuantos nodos $\{x_0, x_1, \cdots, x_n\}$. Por supuesto, en algunos nodos podría interpolar sólo a la función f (y no a las derivadas). Un caso particular (extremo) es la interpolación de Lagrange -que no usa información sobre las derivadas de f-. Otro caso extremo es el polinomio de Taylor, $T_n(f) = \sum_{k=0}^n \frac{f^{(k)}(a)}{k!}(x - a)^k$ (que sólo usa información de un nodo).

Interpolación lineal a trozos

Consideremos una función f definida en el intervalo $[a, b]$ y supongamos que $a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$ es una partición de dicho intervalo.

Definición 2.6

Llamamos función interpolante lineal a trozos de la función f en los nodos $\{x_k\}$ (donde $a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$) a la función $\ell(x)$ que satisface lo siguiente:

- $\ell_{|[x_k,x_{k+1}]}(x)$ coincide con el polinomio de interpolación de Lagrange de f en los nodos $\{x_k, x_{k+1}\}$, $k = 0, 1, \cdots, n-1$.

El siguiente resultado es ahora fácil de demostrar -basta conocer la estimación del error para el polinomio de interpolación de Lagrange de grado ≤ 1 y aplicarlo en cada uno de los intervalos $[x_k, x_{k+1}]$.

Teorema 2.19

Sea $\ell(x)$ la función interpolante lineal a trozos de la función f en los nodos $\{x_k\}$ (donde $a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$). Si $f \in \mathbf{C}^2[a, b]$, entonces

$$|f(x) - \ell(x)| \leq \frac{1}{8}\|f''\|_{[a,b]} \max_{k=0,\cdots,n-1}(x_{k+1} - x_k)^2.$$

En particular, para nodos equiespaciados $x_k = a + k\frac{b-a}{n}$, se tiene que

$$|f(x) - \ell(x)| \leq \frac{1}{8n^2}\|f''\|_{[a,b]}(b-a)^2.$$

2.3.1.5. Integración Numérica mediante fórmulas de cuadratura

Desventajas del Teorema Fundamental de Cálculo

Comenzamos recordando el teorema fundamental del cálculo y algunas desventajas que posee el método de calcular integrales definidas basado en la búsqueda de una primitiva de la función que queremos integrar y, a continuación aplicar la regla de Barrow.

Evidentemente, el cálculo de primitivas (que, a veces, toman el nombre de "integrales indefinidas") tiene interés por sí mismo, aunque en la inmensa mayoría de los casos el objetivo principal es calcular una integral definida. Sin embargo:

• En general es difícil identificar las primitivas.

• A veces es imposible calcular una primitiva por métodos elementales (i.e., cambios de variables, integración por partes, etc.).

• Aunque finalmente obtengamos una primitiva F de la función f, es usual que en la evaluación de $F(b) - F(a)$ se produzcan errores de redondeo. Es decir, es posible que f sea sencilla de evaluar mientras que F sea difícil de evaluar. Un ejemplo sencillo de esto es el siguiente:

$$\int_a^b \frac{1}{x}dx = \log\frac{b}{a}$$

Además, a esta dificultad hay que añadir la posibilidad de que encontrar la primitiva F sea realmente complicado (cosa que no sucede en el ejemplo anteriormente expuesto).

• A veces es imprescindible evaluar $\int_a^b f$ a pesar de que no se pueda hallar una primitiva de f por métodos elementales. Por ejemplo, en Cálculo de Probabilidades es imprescindible disponer de una tabla de valores para las

probabilidades asociadas a la distribución normal $\mathbf{N(0,1)}$,

$$\mathbf{P(N(0,1)\leq a)} = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\mathbf{a}} \exp\left(-\mathbf{x}^2\right)\mathbf{dx}$$

Todo esto nos lleva a la necesidad de métodos numéricos para el cálculo aproximado de integrales definidas.

Regla de los Trapecios y Regla de los Trapecios Repetida

Una primera idea para atacar el cálculo de integrales definidas sin utilizar la regla de Barrow es acudir a la interpretación geométrica del concepto de integral definida. La integral $\int_a^b f$ representa el área encerrada por el gráfico de f, el eje de absisas, y las rectas $x=a$ y $x=b$.

Si queremos aproximar dicho área, podemos en principio utilizar el trapecio de vértices $(a,0)$, $(a,f(a))$, $(b,f(b))$, y $(b,0)$. Es decir:

$$\int_a^b f \simeq \frac{b-a}{2}(f(a)+f(b)) = \mathbf{RT}\{f\}$$

Esta fórmula toma el nombre de "Regla de los Trapecios" por razones obvias. Evidentemente, para que la regla de los trapecios sea útil, el intervalo $[a,b]$ debe ser muy pequeño o, en otro caso, la función f debe ser muy plana. Si ninguna de estas cosas sucede, es claro que la aproximación va a ser mala.

Ahora bien, como

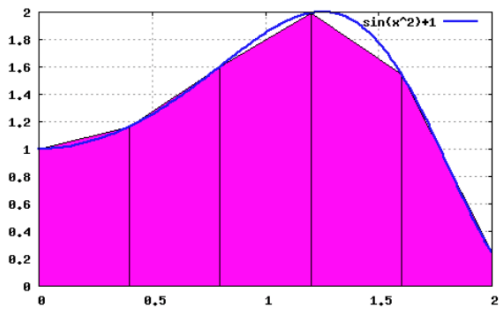
$$\int_a^b f = \int_a^c f + \int_c^b f$$

para cualquier valor $c\in[a,b]$, podemos descomponer el cálculo de $\int_a^b f$ en el cálculo de $\int_{x_i}^{x_{i+1}} f$ para una sucesión de nodos $\{x_i\}$ que formen una partición de $[a,b]$. Si los intervalos $[x_i,x_{i+1}]$ son lo bastante pequeños, entonces es claro que podremos utilizar (con éxito) la regla de los trapecios para el cálculo aproximado de cada una de las integrales $\int_{x_i}^{x_{i+1}} f$ y, por tanto, para el cálculo aproximado de $\int_a^b f$.

Una elección sencilla de los nodos $\{x_i\}_{i=0}^n$, es $x_i = a+ih$, donde $h=(b-a)/n$ (nodos equiespaciados) y la correspondiente fórmula,

$$\begin{aligned}
&\int_a^b f \\
\simeq\ & \frac{h}{2}\left(\underbrace{f(a)+f(a+h)}+\underbrace{f(a+h)+f(a+2h)}+\cdots+\underbrace{f(a+(n-1)h)+f(b)}\right) \\
=\ & \frac{h}{2}\left(f(a)+2\sum_{i=1}^{n-1} f(a+ih)+f(b)\right) = \mathbf{RTR}_n\{f\}
\end{aligned}$$

toma el nombre de Regla de los Trapecios Repetida.



Es evidente que

$$\lim_{n\to\infty} \mathbf{RTR}_n\{f\} = \int_a^b f$$

Definición de fórmula de cuadratura. El problema de la convergencia de fórmulas de cuadratura.
Se deduce del apartado anterior, que una buena idea para atacar el cálculo aproximado de integrales podría ser la substitución del cálculo de primitivas de f por una evaluación (pesada) de la función f en cierto conjunto de nodos $\{x_i\} \subset [a,b]$. De forma más precisa, podemos dar la siguiente definición:
Una fórmula de cuadratura es una expresión de la forma

$$\mathbf{I}_n\{f\} = \sum_{i=0}^{n} A_i f(x_i)$$

que va a ser utilizada para el cálculo aproximado de $\mathbf{I}\{f\} = \int_a^b f$, para funciones f más o menos arbitrarias. Los puntos x_i se llaman nodos de la fórmula de cuadratura y los números A_i se llaman pesos de la fórmula de cuadratura. En la definición anterior no se ha dicho nada sobre la localización de los nodos $\{x_i\}$ aunque, en principio, se suponen contenidos en el intervalo $[a,b]$ (sin embargo, a veces tiene sentido estudiar fórmulas de cuadratura con nodos fuera del intervalo de integración). La idea es muy simple. Para fijar una fórmula de cuadratura deben darnos los nodos y los pesos. Si nos dan una tabla infinita de nodos y pesos (en forma de dos matrices traingulares inferiores infinitas cuya fila n-ésima es el conjunto de nodos $\{x_i^{(n)}\}_{i=0}^n$ -para el caso de la matriz de nodos $\mathbf{X}$- y el conjunto de pesos $\{A_i^{(n)}\}_{i=0}^n$ - para el caso de la matriz de pesos $\mathbf{A}$), entonces nos interesa saber si las correspondientes fórmulas de cuadratura

$$\mathbf{I}_n\{f\} = \sum_{i=0}^{n} A_i^{(n)} f(x_i^{(n)})$$

convergen al valor de la integral para alguna clase $\mathcal{C}$ de funciones (i.e., ¿es cierto que $\lim_{n\to\infty} \mathbf{I}_n\{f\} = \int_a^b f$ para todas las funciones $f \in \mathcal{C}$?).
Si profundizamos un poco más en el problema de la convergencia de fórmulas de cuadratura, nos interesará conocer estimaciones buenas de los errores

$$\mathbf{E}_n\{f\} = \mathbf{I}\{f\} - \mathbf{I}_n\{f\}$$

Por ejemplo, en el caso de las fórmulas $\mathbf{RTR}_n\{f\}$, podemos actuar de la siguiente forma:
$\rightarrow$ Primero hallamos el error de la regla trapezoidal simple, lo que se concreta en el siguiente teorema:

Teorema 2.20

Supongamos que $f \in C^2[a,b]$. Entonces

$$\mathbf{E}\{f\} = \mathbf{I}\{f\} - \mathbf{RT}\{f\} = \frac{-1}{12}(b-a)^3 f''(\eta)$$

para cierto $\eta \in]a,b[$.

Demostracion (Esbozo) La idea de la prueba es utilizar el Teorema del valor medio para integrales,

Lema 2.2 (Teorema del valor medio para integrales)

Supongamos que la función peso $w(x)$ no cambia de signo en el intervalo $[a,b]$. Entonces para toda función continua $f \in C[a,b]$ se tiene que existe un valor $\xi \in]a,b[$ tal que

$$\int_a^b f(x)w(x)dx = f(\xi)\int_a^b w(x)dx$$

Como $\mathbf{RT}\{f\} = \mathbf{I}\{L_1(f,\{a,b\})\}$, entonces

$$\begin{aligned}\mathbf{E}\{f\} &= \mathbf{I}\{f\} - \mathbf{RT}\{f\} \\ &= \mathbf{I}\{f - L_1(f,\{a,b\})\} \\ &= \int_a^b \frac{f''(\xi(x))}{2}(x-a)(x-b)dx\end{aligned}$$

y, utilizando el lema (lo que se puede hacer porque $w(x) = (x-a)(x-b)$ no cambia de signo en $[a,b]$, evidentemente), se tiene que

$$\begin{aligned}\mathbf{E}\{f\} &= \frac{f''(\eta)}{2}\int_a^b (x-a)(x-b)dx \\ &= \frac{-1}{12}(b-a)^3 f''(\eta)\end{aligned}$$

para cierto valor $\eta \in]a,b[$. □

A continuación, utilizamos el error obtenido para la regla trapezoidal simple en el cálculo del error de la regla trapezoidal repetida, de la siguiente forma:

$$\begin{aligned}\mathbf{E}_n\{f\} &= \mathbf{I}\{f\} - \mathbf{RTR}_n\{f\} \\ &= \sum_{k=0}^{n-1}\int_{a+kh}^{a+(k+1)h} f - \sum_{k=0}^{n-1}\mathbf{RT}\{f,[a+kh,a+(k+1)h]\} \\ &= \sum_{k=0}^{n-1}\left\{\int_{a+kh}^{a+(k+1)h} f - \mathbf{RT}\{f,[a+kh,a+(k+1)h]\}\right\} \\ &= \frac{-1}{12}h^3\sum_{k=0}^{n-1} f''(\eta_k) \\ &= n\frac{-1}{12}h^3\left\{\frac{1}{n}\sum_{k=0}^{n-1} f''(\eta_k)\right\} \\ &= n\frac{-1}{12}h^3 f''(\eta) \\ &= \frac{-1}{12}h^2 f''(\eta)(b-a) \text{ (ya que } h = (b-a)/n)\end{aligned}$$

De modo que hemos probado el siguiente teorema:

Teorema 2.21 (Error de la Regla Trapezoidal repetida)

Si $f \in \mathbf{C}^2[a,b]$, entonces

$$\mathbf{E}_n\{f\} = \mathbf{RTR}_n\{f\} - \mathbf{I}\{f\} = \frac{1}{12}h^2 f''(\eta)(b-a) = \frac{(b-a)^3}{12n^2}f''(\eta)$$

para cierto valor $\eta \in]a,b[$. En particular,

$$|\mathbf{RTR}_n\{f\} - \mathbf{I}\{f \mid \leq \frac{(b-a)^3}{12n^2}\|f''\|_{[a,b]}$$

Grado de precisión. Fórmulas de tipo interpolatorio. Fórmulas de Newton-Cotes

Ya hemos observado que la regla trapezoidal se puede interpretar como el resultado de integrar, en vez de la función f, su polinomio de interpolación en los extremos del intervalo, $L_1(f,\{a,b\})$. En general, podemos plantear como método de integración numérica, substituir el integrando f por un polinomio de interpolación $L_n(f,\{x_i\})$. Esto tiene sentido ya que

a) Los polinomios de interpolación (si se eligen de forma adecuada) son buenas aproximaciónes para clases de funciones bastante generales

b) Gracias a la fórmula de interpolación de Lagrange, podemos escribir el polinomio de interpolación $L_n(f)$ como $L_n(f) = \sum_{i=0}^n f(x_i)l_i(x)$ y, por tanto,

$$\begin{aligned}\mathbf{I}_n(f) &= \int_a^b L_n(f)dx = \int_a^b \left\{\sum_{i=0}^n f(x_i)l_i(x)\right\}dx \\ &= \sum_{i=0}^n \left(\int_a^b l_i(x)dx\right) f(x_i)\end{aligned}$$

representa claramente una fórmula de cuadratura (ver la definición).

Este tipo de fórmulas de cuadratura reciben el nombre de **fórmulas de tipo interpolatorio**. Cuando los nodos se toman de forma equiespaciada, la fórmula de tipo interrogatorio asociada recibe el nombre especial de **fórmula de Newton-Cotes**.

Como consecuencia de la unicidad del polinomio de interpolación de una función, es fácil demostrar que si $f \in \Pi_n$, entonces $L_n(f,\{x_i\}_{i=0}^n) = f$ para cualquier conjunto de $n+1$ nodos $\{x_i\}_{i=0}^n$, y, por tanto,

$$\mathbf{I}_n(f) = \int_a^b L_n(f)dx = \int_a^b f dx$$

Decimos que la fórmula de cuadratura es exacta en f si $\mathbf{I}_n(f) = \mathbf{I}\{f\}$. Decimos que la fórmula es exacta en cierta clase $\mathcal{C}$ de funciones, si es exacta en f para todo $f \in \mathcal{C}$.

Hemos probado, pues, que toda fórmula de tipo interpolatorio basada en $n+1$ nodos $\{x_i\}_{i=0}^n$, es exacta en $\mathbf{\Pi}_n$ (otra forma de expresar esto es decir que la fórmula tiene grado de precisión $\geq n$). De hecho, esta propiedad caracteriza las fórmulas de tipo interpolatorio:

Teorema 2.22

La fórmula de cuadratura $\mathbf{I}_n\{f\} = \sum_{i=0}^n A_i f(x_i)$ es de tipo interpolatorio si y solo si tiene grado de precisión $\geq n$.

N Como $\{\mathbf{1}, \mathbf{x}, \mathbf{x}^2, ..., \mathbf{x}^m\}$ es una base de Π_m (como espacio vectorial) para cada $m \in \mathbb{N}$, y tanto $f \rightsquigarrow \mathbf{I}_n\{f\}$ como $f \rightsquigarrow \mathbf{I}\{f\}$ son aplicaciones lineales, es claro que

$$\left(\begin{array}{c}\text{La fórmula de cuadratura}\\ \mathbf{I}_n\{f\} \text{ es exacta en } \Pi_m\end{array}\right) \Leftrightarrow \left(\mathbf{I}_n\{\mathbf{x}^k\} = \mathbf{I}\{\mathbf{x}^k\},\, k = 0, 1, ..., m\right)$$

Es decir, $\mathbf{I}_n\{f\} = \sum_{i=0}^n A_i f(x_i)$ es exacta en Π_m si y solo si

$$\sum_{i=0}^n A_i x_i^k = \frac{b^{k+1} - a^{k+1}}{k+1},\, k = 0, 1, ..., m \tag{2.1}$$

Esto último es un conjunto de $m+1$ ecuaciones (no lineales) en el que desconocemos $2n+2$ parámetros (los nodos $\{x_i\}_{i=0}^n$ y los pesos $\{A_i\}_{i=0}^n$).
Podemos, pues, preguntarnos por el máximo grado de precisión alcanzable por la fórmula $\mathbf{I}_n\{f\} = \sum_{i=0}^n A_i f(x_i)$.
En principio, parece razonable pensar que dicho grado de precisión sea $2n+2$. Sin embargo, dicho grado de precisión nunca se alcanza, como queda demostrado por el siguiente argumento: Dada la fórmula de cuadratura $\mathbf{I}_n\{f\} = \sum_{i=0}^n A_i f(x_i)$, tomamos $P(x) = \prod_{i=0}^n (x - x_i)^2$. Es claro que $P \in \Pi_{2n+2}$ y $\mathbf{I}(P) > 0$ (por ser $P \geq 0$, $P \neq 0$). Sin embargo,

$$\mathbf{I}_n\{P\} = \sum_{i=0}^n A_i P(x_i) = 0,$$

pues $P(x_i) = 0$ para todo i.
Por otra parte, también sabemos que, si se fijan los nodos, entonces tenemos que resolver un conjunto de m ecuaciones lineales. El mejor m esperado será, pues, $m = n+1$, ya que disponemos de exactamente $n+1$ pesos a elegir. Además, sabemos que esto se puede hacer (basta tomar la correspondiente fórmula de tipo interpolatorio). De aquí se deduce que, una vez fijados los nodos, los mejores pesos están dados por la fórmula de tipo interpolatorio asociada. Entonces, ¿existirán conjuntos de nodos privilegiados, para los que el grado de precisión se acerque lo máximo posible a la cota que hemos calculado -que es $2n+1$- ?
La respuesta, sorprendente, es Sí y se debe a Gauss:

Teorema 2.23 (Gauss)

Para una elección adecuada de los nodos $\{x_i\}$, podemos conseguir grado de precisión exactamente $2n+1$.

La demostración de éste teorema en un contexto un poco más general es el objetivo de la próxima sección. Esta sección la terminamos con algunas observaciones sobre fórmulas de tipo interpolatorio.

Fórmulas de cuadratura Gaussiana

Para explicar la demostración del teorema de Gauss es necesario antes recordar el concepto de sucesión de polinomios ortogonales. Veamos por qué:
Supongamos que disponemos de una sucesión de polinomios $\{p_n\}_{n=0}^\infty$ tales que $\deg p_n = n$ para todo n y

$$\int_a^b q(x) p_n(x) dx = 0$$

para todo $q \in \Pi_{n-1}$ y todo n (i.e., $\{p_n\}$ es una sucesión de polinomios ortogonales). Si tomamos $P \in \Pi_{2n+1}$(arbitrario), entonces al aplicar el algoritmo de división Euclídea, podemos escribir $P = qp_{n+1} + r$ para ciertos polinomios $q, r \in \Pi_n$ y, por tanto,

$$\mathbf{I}\{P\} = \mathbf{I}\{qp_{n+1}\} + \mathbf{I}\{r\} = \mathbf{I}\{r\}.$$

Si tomamos los ceros de p_{n+1} como nodos y construimos la correspondiente fórmula de cuadratura, entonces

$$\begin{aligned} \mathbf{I}_n\{P\} &= \mathbf{I}_n\{qp_{n+1}\} + \mathbf{I}_n\{r\} \\ &= \sum_{i=0}^{n} A_i q(x_i) p_{n+1}(x_i) + \mathbf{I}_n\{r\} \\ &= \mathbf{I}_n\{r\} = \mathbf{I}\{r\} \text{ (pues la fórmula es exacta en } \Pi_n) \end{aligned}$$

y, por tanto, $\mathbf{I}\{P\} = \mathbf{I}_n\{P\}$. □

De hecho, la idea de utilizar polinomios ortogonales es bastante sorprendente a primera vista, si se tiene en cuenta que nuestro objetivo inicial es elegir una sucesión de valores $\{x_i\} \subset [a, b]$ de modo que el número de ecuaciones de la forma (2.1) que se puedan resolver sea el máximo posible (un problema dificil, evidentemente). Sin embargo, la idea es sencilla:

Algunas cosas hay que observar:

• Habría que demostrar que existe una sucesión de polinomios ortogonales. ¿Cuántas existen?.

• Habría que demostrar que los ceros del polinomio p_n son todos distintos (i.e., son simples) y que todos pertenecen al intervalo$[a, b]$ (lo que es necesario para poder elegirlos como nodos dela fórmula de cuadratura).

Resulta que existe esencialmente una única sucesión de polinomios ortogonales (en el sentido de que si $\{p_n\}$ y $\{q_n\}$ son dos sucesiones de polinomios ortogonales, entonces existe una sucesión de escalares no nulos $\{\alpha_n\} \subset \mathbb{R} \setminus \{0\}$ tal que $p_n(x) = \alpha_n q_n(x)$ para todo n), si tomamos la definición que hemos introducido. Por tanto, sólo podemos construir una fórmula de cuadratura con grado de precisión máximo, mediante la construcción anterior.

Sin embargo, podemos introducir un concepto de sucesión ortogonal, ligeramente más general, y ver de qué forma podemos utilizarlo para construir fórmulas de integración numérica.

Así, diremos que la sucesión de polinomios $\{p_n\}$ con $p_n \in \Pi_n$, para todo n, es una sucesión de polinomios ortogonales respecto de la función peso[3] $w(x)$ si y solo si $\int_a^b p_n(x)p_m(x)w(x)dx = 0$ siempre que $n \neq m$. Fijada una función peso w, es fácil demostrar que existe esencialmente una única sucesión de polinomios ortogonales respecto de w, y que los ceros del correspondiente n-ésimo polinomio ortogonal son simples y están contenidos en $[a, b]$ para todo n).

Para cada función peso w podemos plantearnos el problema de la aproximación numérica de integrales de la forma

$$\mathbf{I}_w(f) = \int_a^b f(x)w(x)dx.$$

Este problema es evidentemente una generalización del problema de integración numérica que hemos tratado hasta ahora (para el que simplemente teníamos $w(x) \equiv 1$). También es claro que podemos hablar de fórmulas de cuadratura, grado de precisión y fórmulas de tipo interpolatorio, y que las fórmulas de tipo interpolatorio están caracterizadas igualmente por el simple hecho de que su grado de precisión es $\geq n$ (numéro de nodos $= n + 1$).

---

[3] Decimos que $w(x)$ es una función peso si $w \geq 0$ en $[a, b]$, $w \neq 0$, y las integrales $\int_a^b w(x)x^k dx$ existen para todo $k \in \mathbb{N}$.
Fijada w una función peso,

$$< f, g >_w = \int_a^b f(x)g(x)w(x)dx$$

define un producto interior sobre el espacio vectorial $L_2(w) = \{f : \int_a^b f(x)^2 w(x)dx < \infty\}$.

Teorema 2.24 (Gauss)

Para cada función peso w, y para cada $n \in \mathbb{N}$, existe una única fórmula de tipo interpolatorio

$$\mathbf{I}_n\{f\} = \sum_{i=0}^{n} A_i f(x_i) \simeq \mathbf{I}_w\{f\} = \int_a^b f(x)w(x)dx$$

con grado de precisión exactamente $2n+1$ (que es el máximo grado de precisión alcanzable por este tipo de fórmulas). Además, dicha fórmula está dada por la fórmula de tipo interpolatorio con peso w cuyos nodos son los ceros del $(n+1)$-ésimo polinomio ortogonal asociado a w.

Este tipo de fórmulas toman el nombre genérico de **fórmulas de cuadratura Gaussiana**.

Sobre el cálculo de los ceros de polinomios ortogonales

Para utilizar una fórmula de cuadratura Gaussiana, es necesario antes calcular los nodos de la fórmula, que son (como sabemos) los ceros del polinomio ortogonal mónico de grado $n+1$. ¿Existen algoritmos sencillos (o, al menos, precisos) para el cálculo de dichos ceros?.

Vamos a explicar un truco que nos permite convertir nuestro problema en uno de Álgebra Lineal Numérica. En concreto, vamos a transformar el problema del cálculo de dichos ceros en el del cálculo de los autovalores de cierta matriz. Evidementemente, ambos problemas responden al cálculo de los ceros de cierto polinomio (en el caso de los autovalores, se trata del polinomio característico de la matriz), pero tenemos la suerte de que la matriz que aparece en nuestra reducción es tridiagonal, y para estas matrices existen potentes algoritmos que calculan sus autovalores.

La conexión entre ambos problemas se basa en el siguiente conocido resultado de la teoría de polinomios ortogonales:

Teorema 2.25

Supongamos que tenemos definido un producto escalar $(,)$ sobre el espacio de los polinomios de coeficientes reales, tal que

$$(tp(t), q(t)) = (p(t), tq(t))$$

para todo $p(t), q(t) \in \Pi$. Entonces los polinomios ortogonales mónicos respecto de este producto escalar, $\{q_k(t)\}_{k=0}^{\infty}$ satisfacen la siguiente relación de recurrencia a tres términos:

$$q_{k+1}(t) = (t - \alpha_k)q_k(t) - \beta_k q_{k-1}(t); k = 0, 1, ..., n-1,$$

donde $q_{-1}(t) = 0$, $q_0(t) = 1$, y

$$\alpha_k = \frac{(tq_k(t), q_k(t))}{(q_k(t), q_k(t))};\ \beta_k = \frac{(q_k(t), q_k(t))}{(q_{k-1}(t), q_{k-1}(t))}$$

N Favard demostró que si la sucesión de polinomios mónicos $\{q_k(t)\}_{k=0}^{\infty}$ está dada por una relación de recurrencia a tres términos como la anterior (donde se supone que los coeficientes α_k, β_k están dados de antemano, y $\beta_k > 0$ para todo k), entonces es la sucesión de polinomios ortogonales mónicos respecto de cierto producto escalar.

La relación de recurrencia a tres términos anterior es una herramienta muy útil para el estudio de polinomios ortogonales (y, por tanto, para las fórmulas Gaussianas). El primer hecho que podemos observar es, obviamente, que dicha relación permite el cálculo eficiente de la sucesión de polinomios ortogonales mónicos, sin utilizar el proceso de ortogonalización de Gram-Smidth. A este algoritmo se le conoce como 'algoritmo de Stieltjes'. Vamos a establecer, además, la importante conexión que existe con el problema del cálculo de los ceros de $q_{n+1}(t)$.
Definimos la función vectorial $\mathbf{q}(t) = (q_0(t), q_1(t), \cdots, q_n(t))^T \in \mathbb{R}^{n+1}$. Si $\mathbf{J}_n \in \mathbf{M}_{n+1}(\mathbb{R})$ denota la matriz tridiagonal

$$\mathbf{J}_n = \begin{pmatrix} \alpha_0 & 1 & 0 & & \cdots & 0 \\ \beta_1 & \alpha_2 & 1 & & \cdots & 0 \\ 0 & \beta_2 & \alpha_3 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \beta_{n-1} & \alpha_{n-1} & 1 \\ 0 & 0 & \cdots & 0 & \beta_n & \alpha_n \end{pmatrix}$$

entonces la relación de recurrencia a tres términos se puede reescribir como:

$$\mathbf{J}_n \mathbf{q}(t) = (tq_0(t), tq_1(t), \cdots, tq_n(t) - q_{n+1}(t))^T$$

y, por tanto, $q_{n+1}(\xi) = 0$ si y sólo si $\mathbf{J}_n \mathbf{q}(t) = \xi \mathbf{q}(t)$. Esto reduce el problema del cálculo de los ceros de $q_{n+1}(t)$ al problema de la búsqueda de los autovalores de la matriz $\mathbf{J}_n$. Ahora bien: existen algoritmos eficientes y robustos para el cálculo de autovalores de matrices tridiagonales, un hecho del que podemos, pues, obtener beneficio. De hecho, es un hecho aceptado que esta forma de atacar el problema es más estable que la solución directa de la ecuación $q_{n+1}(t) = 0$ por métodos iterativos. Aún así, todavía encontramos el problema del cálculo de los coeficientes α_k, β_k ya que éstos dependen del cálculo de ciertas integrales (en general, los productos escalares que se utilizan están dados por la elección de cierta función peso $w(t)$, y son de la forma $(f(t), g(t)) = \int_a^b f(t)g(t)w(t)dt$). Sin embargo, en todos los casos clásicos -que son los más interesantes- se conocen expresiones explícitas para α_k y β_k. En particular, esto sucede para los polinomios de Jacobi, que son la sucesión de polinomios ortogonales mónicos $\{P_n^{\alpha,\beta}(t)\}_{n=0}^{\infty}$ respecto de la función peso $w(t) = (1-t)^\alpha (1+t)^\beta$, con exponentes $\alpha, \beta > -1$. Para esta elección, se satisface que

$$\begin{cases} \alpha_k &= \frac{\beta^2 - \alpha^2}{(2k+\alpha+\beta)(2k+\alpha+\beta+2)} \\ \beta_k &= \frac{4k(k+\alpha)(k+\beta)(k+\alpha+\beta)}{(2k+\alpha+\beta-1)(2k+\alpha+\beta)^2(2k+\alpha+\beta+1)} \end{cases}$$

Otros casos interesantes son los dados por los polinomios mónicos de Legendre, de Laguerre, etc. (Ver Bibliografía).

Estimación del error en fórmulas de cuadratura de tipo interpolatorio
En algunos casos excepcionales -como la fórmula de los trapecios- se puede aplicar el teorema del valor medio para integrales para estimar el error. Este teorema no sirve para fórmulas de tipo interpolatorio de grado > 1, pues en tal caso

$$E_n(f)(x) = f(x) - L_n(f)(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi_x) w_n(x),$$

donde $w_n(x) = (x - x_0) \cdots (x - x_n)$ es una función que cambia de signo dentro del intervalo $[a, b]$.
Sin embargo, podemos utilizar la anterior igualdad para acotar el error:

$$\begin{aligned} |\mathbf{E}_n\{f\}| &= |\mathbf{I}\{f - L_n(f)\}| = |\mathbf{I}\{E_n(f)(x)\}| \\ &= \left| \int_a^b \frac{1}{(n+1)!} f^{(n+1)}(\xi_x) w_n(x) dx \right| \\ &\leq \frac{1}{(n+1)!} \left\| f^{(n+1)} \right\|_{\mathbf{C}[a,b]} \int_a^b |(x - x_0) \cdots (x - x_n)| \, dx \end{aligned}$$

o, simplificando mucho, se obtiene la acotación

$$|\mathbf{E}_n\{f\}| \leq \frac{(b-a)^{n+1}}{(n+1)!} \left\| f^{(n+1)} \right\|_{\mathbf{C}[a,b]}$$

N En el caso de que la fórmula sea de tipo Gaussiano, se puede aprovechar que el grado de precisión es $2n+1$ para mejorar la fórmula del error, dándose el caso de que

$$\mathbf{E}_n\{f\} = \int_a^b f(t)w(t)dt - I_n\{f\} = \frac{f^{(2n)}(\xi)}{(2n)!}\langle P_n^*, P_n^*\rangle,$$

donde $P_n^*(t) = t^n + q_{n-1}(t)$ representa el $(n+1)$-ésimo polinomio ortogonal mónico respecto de la función peso $w(t)$ en el intervalo $[a,b]$, y $\xi \in (a,b)$. En particular, para $w(t) = 1$ (que es el caso de la integración usual), se tiene que

$$\int_a^b f(t)dt - I_n\{f\} = \frac{(b-a)^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3} f^{(2n)}(\xi)$$

Representación de Peano del error en fórmulas de quadratura de tipo interpolatorio.
Supongamos que la función f es de clase al menos $n+1$ en un entorno de a. Entonces podemos usar la expresión del resto de la fórmula de Taylor

$$f(x) - T_n(x) = \frac{1}{(n+1)!}\int_a^x f^{(n+1)}(\xi)(x-\xi)^n d\xi; \; T_n(f) = \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!}(x-a)^k$$

para estimar el resto de las fórmulas de cuadratura. Si definimos

$$(x-\xi)_+^n = \begin{cases} (x-\xi)^n & x \geq \xi \\ 0 & x \leq \xi \end{cases}$$

y $b > a$ entonces es claro que

$$\int_a^x f^{(n+1)}(\xi)(x-\xi)^n d\xi = \int_a^b f^{(n+1)}(\xi)(x-\xi)_+^n d\xi \text{ para todo } x \in [a,b].$$

Por tanto,

$$f(x) - T_n(f) = \frac{1}{(n+1)!}\int_a^b f^{(n+1)}(\xi)(x-\xi)_+^n d\xi.$$

Supongamos que nuestra fórmula de cuadratura

$$\mathbf{I}_n\{f\} = \sum_{i=0}^{n} A_i f(x_i) \simeq \mathbf{I}_w\{f\} = \int_a^b f(x)w(x)dx$$

es de tipo interpolatorio. Entonces $\mathbf{I}_n\{T_n(f)\} = \mathbf{I}_w\{T_n(f)\}$ y, por tanto,

$$\begin{aligned} \mathbf{I}_n\{f\} &= \mathbf{I}_n\{T_n(f)\} + \mathbf{I}_n\{\frac{1}{(n+1)!}\int_a^b f^{(n+1)}(\xi)(x-\xi)_+^n d\xi\} \\ &= \mathbf{I}_w\{T_n(f)\} + \mathbf{I}_n\{\frac{1}{(n+1)!}\int_a^b f^{(n+1)}(\xi)(x-\xi)_+^n d\xi\} \end{aligned}$$

Por otra parte,

$$\mathbf{I}_w\{f\} = \mathbf{I}_w\{T_n(f)\} + \mathbf{I}_w\{\frac{1}{(n+1)!}\int_a^b f^{(n+1)}(\xi)(x-\xi)_+^n d\xi\}$$

y, por tanto,

$$\begin{aligned}\mathbf{I}_n\{f\} - \mathbf{I}_w\{f\} &= (\mathbf{I}_n - \mathbf{I}_w)\{\frac{1}{(n+1)!}\int_a^b f^{(n+1)}(\xi)(x-\xi)_+^n d\xi\} \\ &= \int_a^b f^{(n+1)}(\xi)(\mathbf{I}_n - \mathbf{I}_w)\{\frac{(x-\xi)_+^n}{(n+1)!}\}d\xi \\ &= \int_a^b f^{(n+1)}(\xi)G_n(\xi)d\xi,\end{aligned}$$

donde

$$\begin{aligned}G_n(\xi) &= (\mathbf{I}_n - \mathbf{I}_w)\{\frac{(x-\xi)_+^n}{(n+1)!}\} \\ &= \int_a^b \left(L_n(\frac{(x-\xi)_+^n}{(n+1)!}, \{x_i\}_{i=0}^n)(x) - \frac{(x-\xi)_+^n}{(n+1)!}\right) w(x)dx\end{aligned}$$

es la llamada función de influencia (o núcleo de Peano) asociada a la fórmula de cuadratura.
Por supuesto, las propiedades de convergencia de la fórmula de cuadratura dependen de las propiedades de la función de influencia $G_n(\xi)$. Una primera propiedad de la función de influencia es la siguiente:

Proposición 2.1

$$\int_a^b G_n(\xi)d\xi = \frac{\mathbf{I}_n\{x^{n+1}\} - \mathbf{I}_w\{x^{n+1}\}}{(n+1)!} := \frac{\mathbf{E}_{n+1}}{(n+1)!}.$$

Demostración. Tomamos $f(x) = x^{n+1}$. Entonces $f^{(n+1)}(\xi) = (n+1)!$ y, por tanto,

$$\mathbf{E}_{n+1} = \mathbf{I}_n\{x^{n+1}\} - \mathbf{I}_w\{x^{n+1}\} = \int_a^b (n+1)!G_n(\xi)d\xi.$$

Esto termina la prueba. □

Corolario 2.2 Si la función de influencia $G_n(\xi)$ tiene signo constante, entonces el error de la fórmula de cuadratura viene dado por

$$\mathbf{I}_n\{f\} - \mathbf{I}_w\{f\} = \frac{\mathbf{E}_{n+1}}{(n+1)!}f^{(n+1)}(\overline{x})$$

para cierto valor $\overline{x} \in [a,b]$.

Demostración. Como $G_n(\xi)$ tiene signo constante, podemos usar el teorema integral del valor medio para la expresión del error:

$$\begin{aligned}\mathbf{I}_n\{f\} - \mathbf{I}_w\{f\} &= \int_a^b f^{(n+1)}(\xi)G_n(\xi)d\xi \\ &= f^{(n+1)}(\overline{x})\int_a^b G_n(\xi)d\xi \\ &= \frac{\mathbf{E}_{n+1}}{(n+1)!}f^{(n+1)}(\overline{x}).\ \square\end{aligned}$$

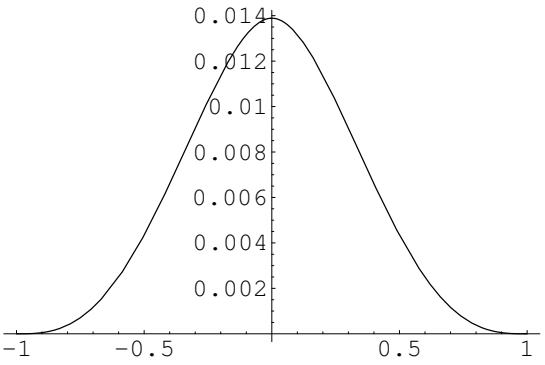
Algunos casos en los que la función de influencia posee signo constante son bien conocidos. Por ejemplo, consideremos la fórmula de Simpson,

$$\mathbf{RS}\{f\} := \frac{1}{3}\left(f(-1)+4f(0)+f(1)\right) \simeq \mathbf{I}\{f\} = \int_{-1}^{1} f(t)dt.$$

No es difícil comprobar que se trata de una fórmula de tipo Gaussiano con 3 nodos por lo que es exacta en $\Pi_{2\cdot 2-1} = \Pi_3$ y, si queremos aplicar las expresiones anteriores, tendremos que tomar $n = 3$. Por tanto, la función de influencia viene dada por

$$\begin{aligned}
G_3(\xi) &= (\mathbf{RS}-\mathbf{I})\{\frac{(x-\xi)_+^3}{(n+1)!}\} \\
&= \frac{1}{3}\left(\frac{(-1-\xi)_+^3}{3!}+4\frac{(0-\xi)_+^3}{3!}+\frac{(1-\xi)_+^3}{3!}\right)-\int_{-1}^{1}\frac{(x-\xi)_+^3}{3!}dx \\
&= \frac{1}{18}\left((-1-\xi)_+^3+4(0-\xi)_+^3+(1-\xi)_+^3\right)-\frac{1}{6}\frac{(1-\xi)_+^4-(-1-\xi)_+^4}{4} \\
&= \begin{cases} 0 & \text{si } |\xi|>1 \\ \frac{(1-\xi)^3}{18}-\frac{(1-\xi)^4}{24} & \text{si } \xi\in[0,1) \\ \frac{4(-\xi)^3+(1-\xi)^3}{18}-\frac{1}{6}\frac{(1-\xi)^4}{4} & \text{si } \xi\in(-1,0] \end{cases} \\
&= \begin{cases} 0 & \text{si } |\xi|>1 \\ -\frac{1}{72}\left(3\xi+1\right)\left(\xi-1\right)^3 & \text{si } \xi\in[0,1) \\ -\frac{1}{72}\left(3\xi-1\right)\left(\xi+1\right)^3 & \text{si } \xi\in(-1,0] \end{cases}
\end{aligned}$$

El siguiente gráfico muestra $G_3(\xi)$ para $\xi \in [-1,1]$.



Esto demuestra que $G_3(\xi)$ tiene signo constante y, por tanto, podemos afirmar que

$$(\mathbf{RS}-\mathbf{I})\{f\} = \frac{\mathbf{E}_4}{4!}f^{(4)}(\overline{x}) = \frac{f^{(4)}(\overline{x})}{90}.$$

En general, para un intervalo $[a,b]$ arbitrario, la regla de Simpson viene dada por

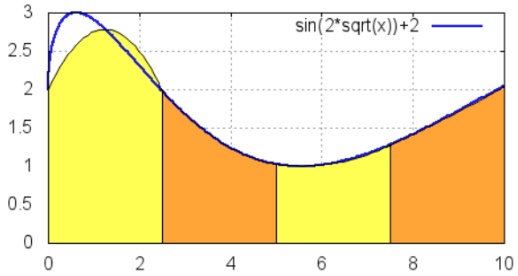
$$\mathbf{RS}\{f\} = \frac{b-a}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right)$$

y el error cometido viene dado por:

$$\mathbf{RS}\{f\} - \int_a^b f = \left(\frac{b-a}{2}\right)^5 \frac{1}{90}f^{(4)}(\xi),$$

para cierto valor de $\xi \in (a,b)$. Evidentemente, la regla de Simpson admite una versión "repetida" tal como se estudió con la regla del trapecio, y viene dada por la expresión

$$\mathbf{RSR}_n\{f\} = \frac{b-a}{6n}\left(f(a) + 4\sum_{k=1}^{n} f(a+(2k-1)h) + 2\sum_{k=1}^{n-1} f(a+2kh) + f(b)\right);\ \text{donde } h = \frac{b-a}{2n}.$$



A continuación, utilizamos el error obtenido para la regla de Simpson simple en el cálculo del error de la regla de Simpson repetida:

$$\begin{aligned}
\mathbf{E}_n\{f\} &= \mathbf{I}\{f\} - \mathbf{RSR}_n\{f\} \\
&= \sum_{k=0}^{n-1}\int_{a+2kh}^{a+2(k+1)h} f - \sum_{k=0}^{n-1}\mathbf{RS}\{f,[a+2kh,a+2(k+1)h]\} \\
&= \sum_{k=0}^{n-1}\left\{\int_{a+2kh}^{a+2(k+1)h} f - \mathbf{RS}\{f,[a+2kh,a+2(k+1)h]\}\right\} \\
&= \sum_{k=0}^{n-1}\left(\frac{2h}{2}\right)^5\frac{-1}{90}f^{(4)}(\eta_k) \\
&= n\frac{-1}{90}h^5\left\{\frac{1}{n}\sum_{k=0}^{n-1} f^{(4)}(\eta_k)\right\} \\
&= n\frac{-1}{90}h^5 f^{(4)}(\eta) \\
&= \frac{-1}{180}h^4 f^{(4)}(\eta)(b-a) \text{ (ya que } h = \frac{(b-a)}{2n}\text{)}
\end{aligned}$$

De modo que hemos probado el siguiente teorema:

Teorema 2.26 (Error de la Regla de Simpson repetida)

Si $f \in \mathbf{C}^4[a,b]$, entonces

$$\mathbf{E}_n\{f\} = \mathbf{RSR}_n\{f\} - \mathbf{I}\{f\} = \frac{1}{180}h^4 f^{(4)}(\eta)(b-a) = \frac{(b-a)^5}{2880n^4}f^{(4)}(\eta)$$

para cierto valor $\eta \in]a,b[$. En particular,

$$|\mathbf{RSR}_n\{f\} - \mathbf{I}\{f\}| \leq \frac{(b-a)^5}{2880n^4}\|f^{(4)}\|_{[a,b]}$$

2.3.2. Las prácticas

Los contenidos del tema 3 se pueden representar gráficamente y, en consecuencia, se pueden comprender con mayor profundidad, con la ayuda de SAGE. De hecho, los 4 exámenes de prácticas de SAGE se realizan atendiendo exclusivamente al contenido de los temas 1, 2 y 3, siendo el caso que los temas 1 y 2 tienen un examen asociado cada uno mientras que el tema 3 tiene dos exámenes asociados: uno para la parte de interpolación y otro para la parte dedicada a integración numérica, por lo que el peso de las prácticas de este tema es mayor que el del resto de temas del curso.

En clase de prácticas se explica cómo utilizar SAGE para calcular el polinomio de interpolación de Lagrange de una función f en una tabla de nodos $\{x_i\}_{i=0}^n$, así como representar en un solo gráfico la función y el polinomio, y en otro gráfico, el error. Además, se explica cómo calcular el paso h que garantice que el error cometido en la aproximación de la función f al aproximarla con el polinomio de interpolación de Lagrange en nodos equiespaciados con paso h se mantenga inferior a una cantidad dada. Por otro lado, en la parte dedicada a integración numérica se da especial importancia a las fórmulas de Newton-Cotes repetidas, concretamente a la regla de los trapecios repetida y la regla de Simpson repetida. Se explica a los alumnos cómo utilizar SAGE para implementar dichas reglas de integración bajo la petición extra de que el error de aproximación sea inferior a una cantidad dada, y se muestran diversos ejemplos en los que estas reglas se comparan entre sí. Se trata de mostrar ejemplos en los que el cálculo directo de una primitiva resulta inabordable. De este modo, los alumnos aprenden de forma práctica la utilidad del uso de fórmulas de cuadratura.

Veamos algunos ejemplos concretos de cómo se logran estos objetivos utilizando SAGE:

Ejemplo 2.3 Sea $P(x)$ el polinomio interpolador de Lagrange de la función $f(x) = 1/(1+x^2)$ en los nodos

$$-2, -1{,}5, , -1, \ldots, 1{,}5, 2.$$

(a) Calcula una cota superior teórica para el error $|f(x) - P(x)|$ en el intervalo $[-2, 2]$.

(b) Calcula $P(x)$ y represéntalo gráficamente junto con $f(x)$ en el intervalo $[-2, 2]$.

(c) Representa gráficamente la "función error" $|f(x) - P(x)|$ en el intervalo $[-2, 2]$, comprobando que es menor que la cota obtenida en el apartado (a).

Para poder trabajar simbólicamente con polinomios se requiere comenzar utilizando la siguiente orden de SAGE:

```
sage: poli=RR['x']                                                    40
```

Lo primero que debemos tener en cuenta es la acotación del error del polinomio de interpolación de Lagrange:

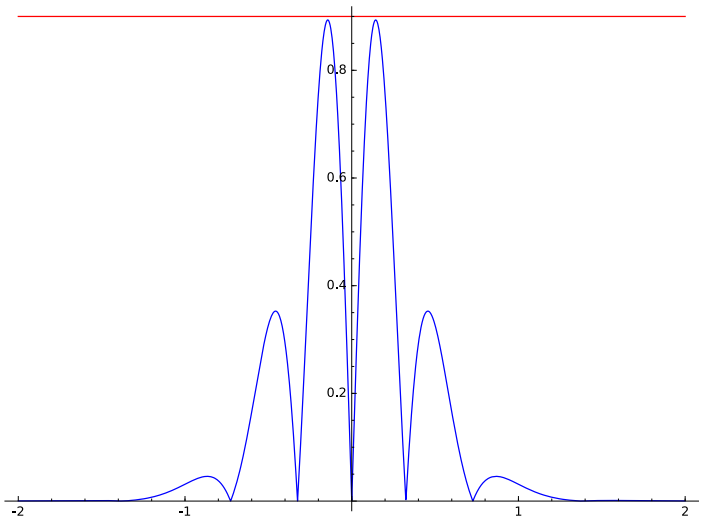
$$|f(x) - P(x)| \leq \frac{\|f^{(n+1)}\|}{(n+1)!}|w_n(x)|,$$

donde $w_n(x) = \prod_{k=0}^{n+1} |x - x_k|$.

Ahora bien, aquí tenemos nodos equiespaciados: $x_k = -2 + k/2$, $k = 0, 1, \cdots, 8$.

Se sigue que $n = 8$ y debemos acotar $f^{(9)}/9!$ en $[-2, 2]$, para lo cual representamos gráficamente dicha función, que previamente debemos calcular:

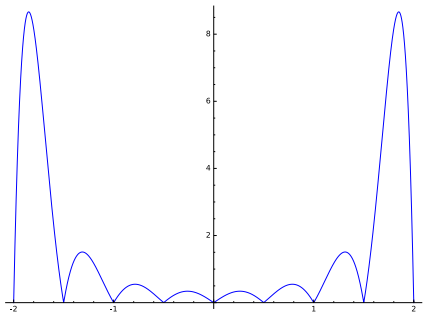
```
sage: f(x)=1/(1+x^2);                                                 41
sage: g(x)=abs(diff(f(x),x,9))                                        42
```



Está claro que la cota de error teórica es:

$$|f(x) - P(x)| \leq E(x) = 0{,}9 \cdot |w_8(x)|$$

```
sage: E(x)=abs(0.9*(x+2)*(x+1.5)*(x+1)*(x+0.5)*x*(x-0.5)*(x-1)*(x-1.5)*(x-2));  43
```



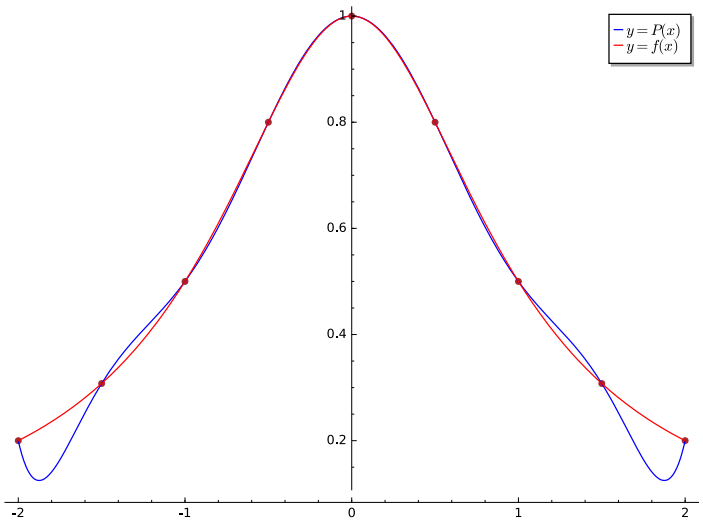
Calculamos ahora el polinomio de interpolación. Para ello, comenzamos definiendo la tabla de puntos en los que dicho polinomio debe coincidir con el gráfico de la función f.

```
sage: n=8;                                                                  44
sage: a=-2;                                                                 45
sage: b=2;                                                                  46
sage: X=[a+(b-a)/n*k for k in [0..n]];                                      47
sage: XY=[(X[j],float(f(X[j]))) for j in srange(0,len(X))]; XY              48
```

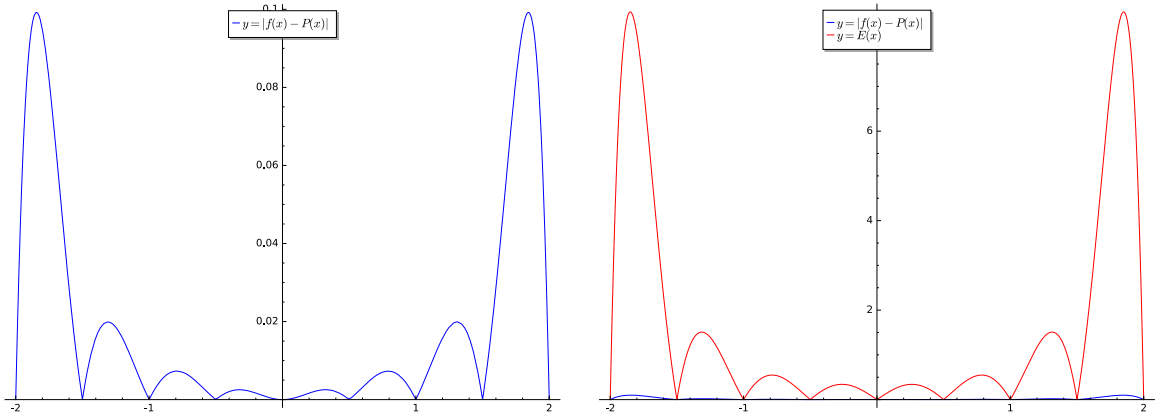
Usamos ahora el operador propio de SAGE para obtener el polinomio interpolador de Lagrange en los puntos dados, al cual llamamos "P":

```
sage: P=poli.lagrange_polynomial(XY);                                       49
sage: P                                                                     50
0.0246153846153846*x^8 - 6.93889390390723e-18*x^7 - 0.209230769230769*x^6 -  51
   1.38777878078145e-16*x^5 + 0.629230769230769*x^4 - 0.944615384615384*x^2 +
   1.11022302462516e-16*x + 1.00000000000000
```

Ahora podemos dibujar en un único gráfico conjunto tanto al polinomio P como a la función f, lo cual nos permite hacernos una idea de la bondad de dicho polinomio para aproximar a la función elegida.



Finalmente, dibujamos el error cometido $f(x) - P(x)$ y, en el siguiente gráfico, lo comparamos con el error estimado -que sale mayor, evidentemente.



Ejemplo 2.4 Se desea aproximar la función $f(x) = \sin(x^2)$ en el intervalo $[-\pi, \pi]$ mediante la función interpoladora lineal a trozos $\ell_n(x)$ en $n+1$ nodos equiespaciados

$$-\pi = x_0 < x_1 < \ldots < x_n = \pi.$$

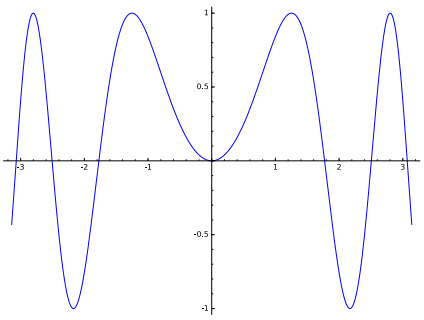
(a) Hallar el valor de n que garantiza que $|f(x) - \ell_n(x)| < 0{,}1$ para todo $x \in [-\pi, \pi]$

(b) Para dicho valor de n, utiliza la función $\ell(x)$ para aproximar $f(0{,}75)$ y comprueba que el error cometido es menor que $0{,}1$.

Comenzamos definiendo y dibujando la función $f(x)$:

```
sage: f(x)=sin(x^2);
```
52

```
sage: plot(f,-pi,pi)
Graphics object consisting of 1 graphics primitive
```

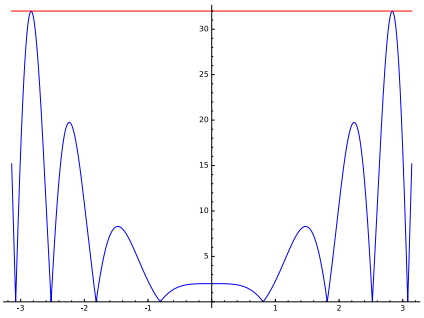


A continuación definimos y dibujamos la función $g(x) = |f''(x)|$, pues sabemos que la acotación del error viene dada por

$$|f(x) - \ell(x)| \leq \frac{1}{8n^2}\|f''\|_{[a,b]}(b-a)^2$$

y, por tanto, necesitamos conocer una constante $M \geq \|f''\|_{[a,b]}$ para, a partir de ella, determinar, dado $\varepsilon > 0$, el valor de n a partir del cual $\frac{1}{8n^2}M(b-a)^2 \leq \varepsilon$, que es $N_0 = \left[(b-a)\sqrt{\frac{M}{8\varepsilon}}\right] + 1$.

```
sage: g(x)=abs(diff(f(x),x,2));
sage: plot(g,-pi,pi)
Graphics object consisting of 1 graphics primitive
```



Como se puede apreciar, $M = 32$ sirve en nuestro caso, por lo que el valor de n debe superar a

```
sage: float(sqrt(32/(8*0.1))*2*pi)
39.7383530632
```

Es decir, $n = 40$ nos sirve para garantizar un error inferior a $0{,}1$. Con esta información a mano, ya podemos proceder a calcular la función $\ell(x)$ y dibujarla conjuntamente con f y, en gráfica separada, dibujar el error cometido $E(x) = |f(x) - \ell(x)|$.
Primero calculamos los puntos en los que la función $\ell(x)$ va a interpolar a $f(x)$

```
sage: X=[-pi+k*2*pi/40 for k in [0..40]];
sage: XY=[(X[j],float(f(X[j]))) for j in srange(0,len(X))]; XY
```

Ahora definimos la función $\ell(x)$. Para ello, definimos previamente la recta $r(x)$ que interpola a $f(x)$ en dos nodos c, d:

```
sage: r(c,d,x)=(f(d)-f(c))/(d-c)*(x-c)+f(c)                                    62
```

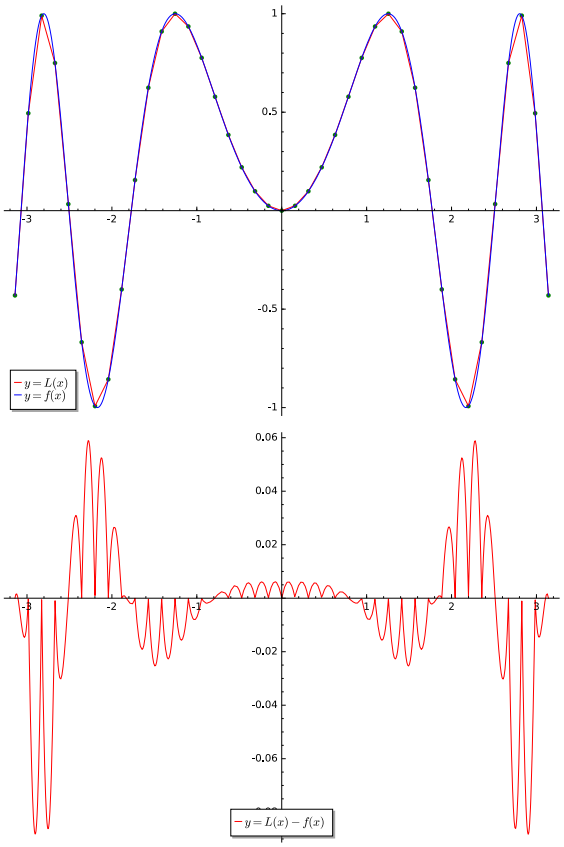
Esto nos permite definir el interpolante lineal a trozos como sigue:

```
sage: L=Piecewise([[(X[j],X[j+1]),r(X[j],X[j+1],x)] for j in srange(0,len(X)-1)],x);  63
sage: L                                                                        64
Piecewise defined function with 40 parts, [[(-pi, -19/20*pi), x |--> 20*(pi + x)*(   65
    sin(361/400*pi^2) - sin(pi^2))/pi + sin(pi^2)], [(-19/20*pi, -9/10*pi), x |-->
    -(19*pi + 20*x)*(sin(361/400*pi^2) - sin(81/100*pi^2))/pi + sin(361/400*pi^2)],
    [(-9/10*pi, -17/20*pi), x |--> -2*(9*pi + 10*x)*(sin(81/100*pi^2) - sin(289/400*
    pi^2))/pi + sin(81/100*pi^2)], [(-17/20*pi, -4/5*pi), x |--> -(17*pi + 20*x)*(sin
    (289/400*pi^2) - sin(16/25*pi^2))/pi + sin(289/400*pi^2)], [(-4/5*pi, -3/4*pi), x
     |--> -4*(4*pi + 5*x)*(sin(16/25*pi^2) - sin(9/16*pi^2))/pi + sin(16/25*pi^2)],
    [(-3/4*pi, -7/10*pi), x |--> -5*(3*pi + 4*x)*(sin(9/16*pi^2) - sin(49/100*pi^2))/
    pi + sin(9/16*pi^2)], [(-7/10*pi, -13/20*pi), x |--> -2*(7*pi + 10*x)*(sin
    (49/100*pi^2) - sin(169/400*pi^2))/pi + sin(49/100*pi^2)], [(-13/20*pi, -3/5*pi),
     x |--> -(13*pi + 20*x)*(sin(169/400*pi^2) - sin(9/25*pi^2))/pi + sin(169/400*pi
    ^2)], [(-3/5*pi, -11/20*pi), x |--> -4*(3*pi + 5*x)*(sin(9/25*pi^2) - sin
    (121/400*pi^2))/pi + sin(9/25*pi^2)], [(-11/20*pi, -1/2*pi), x |--> -(11*pi + 20*
    x)*(sin(121/400*pi^2) - sin(1/4*pi^2))/pi + sin(121/400*pi^2)], [(-1/2*pi, -9/20*
    pi), x |--> -10*(pi + 2*x)*(sin(1/4*pi^2) - sin(81/400*pi^2))/pi + sin(1/4*pi^2)
    ], [(-9/20*pi, -2/5*pi), x |--> -(9*pi + 20*x)*(sin(81/400*pi^2) - sin(4/25*pi^2)
    )/pi + sin(81/400*pi^2)], [(-2/5*pi, -7/20*pi), x |--> -4*(2*pi + 5*x)*(sin(4/25*
    pi^2) - sin(49/400*pi^2))/pi + sin(4/25*pi^2)], [(-7/20*pi, -3/10*pi), x |-->
    -(7*pi + 20*x)*(sin(49/400*pi^2) - sin(9/100*pi^2))/pi + sin(49/400*pi^2)],
    [(-3/10*pi, -1/4*pi), x |--> -2*(3*pi + 10*x)*(sin(9/100*pi^2) - sin(1/16*pi^2))/
    pi + sin(9/100*pi^2)], [(-1/4*pi, -1/5*pi), x |--> -5*(pi + 4*x)*(sin(1/16*pi^2)
    - sin(1/25*pi^2))/pi + sin(1/16*pi^2)], [(-1/5*pi, -3/20*pi), x |--> -4*(pi + 5*x
    )*(sin(1/25*pi^2) - sin(9/400*pi^2))/pi + sin(1/25*pi^2)], [(-3/20*pi, -1/10*pi),
     x |--> -(3*pi + 20*x)*(sin(9/400*pi^2) - sin(1/100*pi^2))/pi + sin(9/400*pi^2)],
     [(-1/10*pi, -1/20*pi), x |--> -2*(pi + 10*x)*(sin(1/100*pi^2) - sin(1/400*pi^2))
    /pi + sin(1/100*pi^2)], [(-1/20*pi, 0), x |--> -(pi + 20*x)*sin(1/400*pi^2)/pi +
    sin(1/400*pi^2)], [(0, 1/20*pi), x |--> 20*x*sin(1/400*pi^2)/pi], [(1/20*pi,
    1/10*pi), x |--> -(pi - 20*x)*(sin(1/100*pi^2) - sin(1/400*pi^2))/pi + sin(1/400*
    pi^2)], [(1/10*pi, 3/20*pi), x |--> -2*(pi - 10*x)*(sin(9/400*pi^2) - sin(1/100*
    pi^2))/pi + sin(1/100*pi^2)], [(3/20*pi, 1/5*pi), x |--> -(3*pi - 20*x)*(sin
    (1/25*pi^2) - sin(9/400*pi^2))/pi + sin(9/400*pi^2)], [(1/5*pi, 1/4*pi), x |-->
    -4*(pi - 5*x)*(sin(1/16*pi^2) - sin(1/25*pi^2))/pi + sin(1/25*pi^2)], [(1/4*pi,
    3/10*pi), x |--> -5*(pi - 4*x)*(sin(9/100*pi^2) - sin(1/16*pi^2))/pi + sin(1/16*
    pi^2)], [(3/10*pi, 7/20*pi), x |--> -2*(3*pi - 10*x)*(sin(49/400*pi^2) - sin
    (9/100*pi^2))/pi + sin(9/100*pi^2)], [(7/20*pi, 2/5*pi), x |--> -(7*pi - 20*x)*(
    sin(4/25*pi^2) - sin(49/400*pi^2))/pi + sin(49/400*pi^2)], [(2/5*pi, 9/20*pi), x
    |--> -4*(2*pi - 5*x)*(sin(81/400*pi^2) - sin(4/25*pi^2))/pi + sin(4/25*pi^2)],
    [(9/20*pi, 1/2*pi), x |--> -(9*pi - 20*x)*(sin(1/4*pi^2) - sin(81/400*pi^2))/pi +
     sin(81/400*pi^2)], [(1/2*pi, 11/20*pi), x |--> -10*(pi - 2*x)*(sin(121/400*pi^2)
     - sin(1/4*pi^2))/pi + sin(1/4*pi^2)], [(11/20*pi, 3/5*pi), x |--> -(11*pi - 20*x
```

```
    )*(sin(9/25*pi^2) - sin(121/400*pi^2))/pi + sin(121/400*pi^2)], [(3/5*pi, 13/20*
    pi), x |--> -4*(3*pi - 5*x)*(sin(169/400*pi^2) - sin(9/25*pi^2))/pi + sin(9/25*pi
    ^2)], [(13/20*pi, 7/10*pi), x |--> -(13*pi - 20*x)*(sin(49/100*pi^2) - sin
    (169/400*pi^2))/pi + sin(169/400*pi^2)], [(7/10*pi, 3/4*pi), x |--> -2*(7*pi -
    10*x)*(sin(9/16*pi^2) - sin(49/100*pi^2))/pi + sin(49/100*pi^2)], [(3/4*pi, 4/5*
    pi), x |--> -5*(3*pi - 4*x)*(sin(16/25*pi^2) - sin(9/16*pi^2))/pi + sin(9/16*pi
    ^2)], [(4/5*pi, 17/20*pi), x |--> -4*(4*pi - 5*x)*(sin(289/400*pi^2) - sin(16/25*
    pi^2))/pi + sin(16/25*pi^2)], [(17/20*pi, 9/10*pi), x |--> -(17*pi - 20*x)*(sin
    (81/100*pi^2) - sin(289/400*pi^2))/pi + sin(289/400*pi^2)], [(9/10*pi, 19/20*pi),
     x |--> -2*(9*pi - 10*x)*(sin(361/400*pi^2) - sin(81/100*pi^2))/pi + sin(81/100*
    pi^2)], [(19/20*pi, pi), x |--> (19*pi - 20*x)*(sin(361/400*pi^2) - sin(pi^2))/pi
     + sin(361/400*pi^2)]]
```

Y ahora pintamos las correspondientes gráficas:

```
sage: a=-pi;b=pi;                                                              66
sage: plot(lambda x:L(x),(x,a,b),legend_label='$y=L(x)$',color='red')+point2d(XY,  67
   size=20,color='green')+plot(f(x),(x,a,b),color='blue',legend_label='$y=f(x)$')
Graphics object consisting of 3 graphics primitives                            68
sage: plot(lambda x:L(x)-f(x),(x,a,b),legend_label='$y=L(x)-f(x)$',color='red')  69
Graphics object consisting of 1 graphics primitive                             70
```



Veamos ahora cómo se trataría un ejemplo de integración numérica. Concretamente, exponemos un caso en el que comparamos la eficacia de la regla de los trapecios repetida con la regla de Simpson repetida.

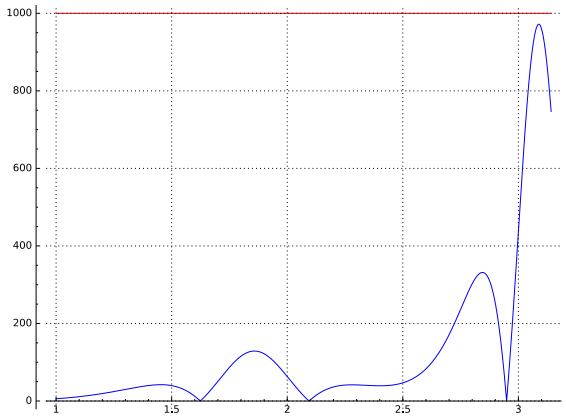
Ejemplo 2.5 Calcular aproximadamente la integral $\int_1^\pi \log(4+\cos(x^2))dx$ mediante la fórmula de cuadratura de Simpson repetida, con error absoluto menor que 10^{-4}. Hacer lo propio con la fórmula de los trapecios repetida.

Antes de proceder a calcular la integral de forma aproximada, nos cercionamos de que esta no se puede calcular de forma inmediata -aplicando Barrow- porque no tiene una integral elemental. Para verlo, pedimos a SAGE que la calcule:

```
sage: f(x)=log(4+cos(x^2))                                                    71
sage: integral(f,x)                                                           72
x |--> -x*log(2) + 1/2*x*log(2*(8*cos(x^2) + 1)*cos(2*x^2) + cos(2*x^2)^2 + 64*cos(x  73
    ^2)^2 + sin(2*x^2)^2 + 16*sin(2*x^2)*sin(x^2) + 64*sin(x^2)^2 + 16*cos(x^2) + 1)
    + integrate(-4*(4*x^2*cos(2*x^2)*sin(x^2) - 4*x^2*sin(x^2) - (4*x^2*cos(x^2) + x
    ^2)*sin(2*x^2))/(2*(8*cos(x^2) + 1)*cos(2*x^2) + cos(2*x^2)^2 + 64*cos(x^2)^2 +
    sin(2*x^2)^2 + 16*sin(2*x^2)*sin(x^2) + 64*sin(x^2)^2 + 16*cos(x^2) + 1), x)
```

Comenzamos con Simpson. Lo primero que hacemos es determinar el número n de parábolas a usar: Buscamos (gráficamente) una constante $M > 0$ que cumpla $|f^{(4)}(x)| \leq M$ para todo $x \in [1, \pi]$:

```
sage: plot(abs(derivative(f(x),x,4)),(x,1,pi),gridlines=True)+plot(1000,(x,1,pi),   74
    color='red')
Graphics object consisting of 2 graphics primitives                           75
```



En la gráfica anterior vemos que se puede tomar $M = 1000$. Como

$$\left| \mathbf{RSR}_n(f) - \int_a^b f(x)\,dx \right| \leq \frac{(b-a)^5}{2880n^4} \cdot M \text{ para todo } n \in \mathbb{N}$$

entonces para determinar el número n de parábolas que debemos utilizar para alcanzar la precisión deseada, buscamos un n que cumpla

$$\frac{(\pi-1)^5}{2880\,n^4} \cdot M < 10^{-4} \iff n > \sqrt[4]{\frac{10^4(\pi-1)^5}{2880} \cdot M}$$

Lo que nos conduce a estimar:

```
sage: float((10^4/2880*1000*(pi-1)^5)^(1/4))
19.8871689633
```

Por tanto, vamos a aplicar el método de Simpson con $n = 20$ parábolas. Calculamos los nodos:

```
sage: p=[1+k*float((pi-1))/(2*n) for k in [0..2*n]] # Nodos
sage: p
[1.0, 1.133849540849362, 1.2676990816987241, 1.4015486225480862, 1.5353981633974483,
 1.6692477042468103, 1.8030972450961724, 1.9369467859455345, 2.0707963267948966,
 2.204645867644259, 2.3384954084936207, 2.4723449493429825, 2.606194490192345,
 2.740044031041707, 2.873893571891069, 3.007743112740431, 3.141592653589793]
```

Aplicamos ahora la regla de Simpson repetida:

```
sage: simpson=(pi-1)/(6*n)*(f(p[0])+4*sum(f(p[2*k-1]) for k in srange(1,n+1))+2*sum(
    f(p[2*k]) for k in srange(1,n))+f(p[2*n]))
sage: float(simpson)
2.85090369185
```

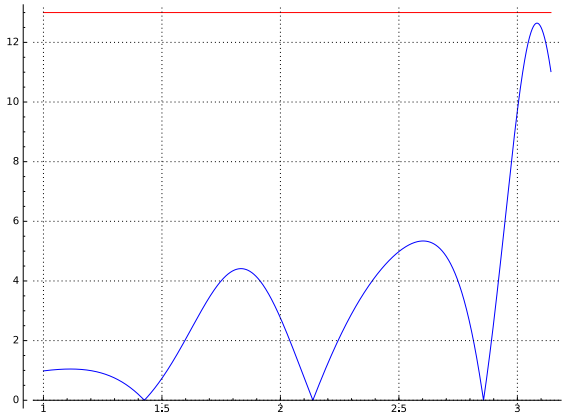
Conclusión: $\mathbf{RSR}_{20}\{f\} = 2{,}85090369185\ldots$ es una aproximación de $\displaystyle\int_1^{\pi} log(4+cos(x^2))\,dx$ con error absoluto menor que 10^{-4}.

¿Qué hubiese sucedido si hubiéramos intentado aplicar la regla de los trapecios repetida para aproximar esta misma integral con el mismo error absoluto? Como sabemos, la acotación del error es en este caso

$$\left|\mathbf{RTR}_n(f) - \int_a^b f(x)\,dx\right| \leq \frac{(b-a)^3}{12n^2}\cdot \|f''\|_{[a,b]} \text{ para todo } n \in \mathbb{N},$$

por lo que debemos primero estimar el tamaño de la segunda derivada:

```
sage: plot(abs(derivative(f(x),x,2)),(x,1,pi),gridlines=True)+plot(13,(x,1,pi),color
    ='red')
Graphics object consisting of 2 graphics primitives
```



Esto confirma que $\|f''\|_{[1,\pi]} \leq 13 = M$ y, por tanto, la desigualdad que debemos resolver -para buscar el valor apropiado de n- es $\frac{(\pi-1)^3}{12n^2} \cdot 13 \leq 10^{-4}$. Es decir, se tiene que $n \geq \sqrt{\frac{10^4(\pi-1)^3 \cdot 13}{12}}$

```
sage: float(sqrt(10^4*(pi-1)^3*13/12))
326.20180192
```

Es decir, debemos tomar al menos $n = 327$ para garantizar dicha acotación. Esto supone evaluar la función en 328 puntos, que es mucho más costoso que evaluarla en los 41 puntos requeridos por la regla de Simpson repetida.
Por supuesto, en las clases de prácticas también se abordan problemas de cálculo integral en el sentido clásico, por ejemplo para hallar el área encerrada por varias curvas, como se hace en el siguiente ejemplo:

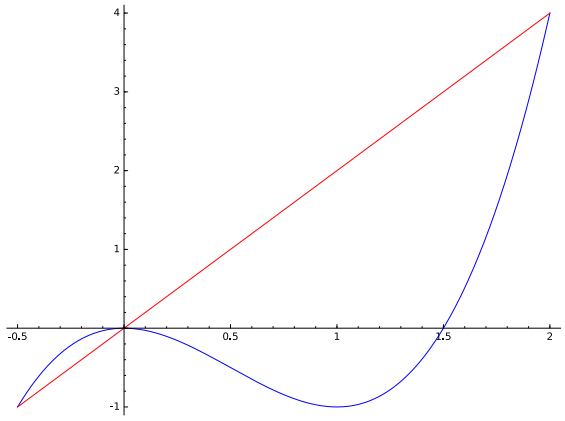
Ejemplo 2.6 Calcular el área de la región del plano limitada por la gráfica de la función $f(x) = 2x^3 - 3x^2$ y la recta $y = 2x$.

Lo primero que hacemos es calcular los puntos de corte de ambas funciones, para así poder determinar lel intervalo de la recta en el que debemos dibujar ambas funciones:

```
sage: solve(2*x^3-3*x^2-2*x==0,x)
[
x == (-1/2),
x == 2,
x == 0
]
```

Esto nos conduce a considerar el intervalo $[-1/2, 2]$:

```
sage: plot(2*x^3-3*x^2,(x,-2,2))+plot(2*x,(x,-2,2),color='red')
Graphics object consisting of 2 graphics primitives
```



Se sigue que el área a determinar viene dada por la suma

$$\int_{-1/2}^{0} (2x^3 - 3x^2 - 2x)dx + \int_{0}^{2} (2x - 2x^3 + 3x^2)dx$$

que calculamos como sigue:

```
sage: integral(2*x^3-3*x^2-2*x,(x,-1/2,0))+integral(2*x-2*x^3+3*x^2,(x,0,2))
131/32
```

2.4 Tema 4: Series

2.4.1. La teoría

Los objetos que vamos a estudiar en este tema ya han aparecido varias veces a lo largo del curso. Se trata de las series numéricas. Por definición, una serie $\sum_{k=0}^{\infty} a_k$ es una sucesión de sumas $S_N = \sum_{k=0}^{N} a_k$, donde los a_k son números reales (o complejos), y el objetivo principal de la teoría de series numéricas es la descripción del comportamiento asintótico de la sucesión $(S_N)_{N=0}^{\infty}$. Si dicha sucesión converge a un cierto valor $S < \infty$, diremos que la serie $\sum_{k=0}^{\infty} a_k$ es convergente y que su suma vale S, lo que se expresa brevemente como

$$\sum_{k=0}^{\infty} a_k = S.$$

En otras palabras, $\sum_{k=0}^{\infty} a_k = \lim_{N\to\infty} S_N$, donde $S_N = \sum_{k=0}^{N} a_k$. Si el límite de las sumas parciales no existe o vale ∞, diremos que la serie diverge. Algunos autores hablan de series divergentes a $\pm\infty$ si $\lim_{N\to\infty} S_N = \pm\infty$.
La primera clase se dedica a explicar el concepto de serie numérica y su convergencia. Las series numéricas aparecen frecuentemente en computación, especialmente en el análisis asintótico de algoritmos (eficiencia). En estos casos lo habitual es estudiar series divergentes a $+\infty$ (como el coste computacional de un algoritmo cuya entrada tiene N datos) y, en tal caso, se analiza el orden de magnitud de las sumas parciales de la serie. Por otra parte, las series convergentes son también útiles, por ejemplo cuando se calculan áreas o volúmenes, y deben estudiarse también detenidamente. Cuando la serie depende de un parámetro, esta puede analizarse para los distintos valores del parámetro, lo cual conlleva a menudo a que la misma serie sea a veces convergente (para ciertos valores del parámetro que estamos considerando) mientras que, en otras ocasiones (para otros valores del parámetro) es divergente, en cuyo caso se estudia el orden de magnitud de sus sumas parciales.
Un ejemplo típico es la serie geométrica $\sum_{k=0}^{\infty} x^k$. Otro, la serie tipo armónica $\sum_{n=1}^{\infty} \frac{1}{n^p}$. En el primer caso, el parámetro es la variable x y, en el segundo, la variable p. Ambas series aparecen repetidas veces en este tema, pues tienen un papel vertebrador en torno a las ideas que se desarrollan aquí.
Así, utilizando la identidad

$$1 + x + \cdots + x^n = \frac{1 - x^{n+1}}{1 - x},$$

se obtiene que la suma parcial de la serie geométrica viene dada por $S_N(x) = \sum_{k=0}^{N} x^k = \frac{1-x^{N+1}}{1-x}$, por lo que el estudio de $\lim_{N\to\infty} S_N(x)$ se reduce al estudio de $\lim_{N\to\infty} x^N$. En particular, la fórmula anterior implica que

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1-x} \text{ cuando } |x| < 1$$

y que la suma no converge en el resto de casos. Es más, cuando $|x| > 1$ las sumas parciales divergen a ∞ con un orden de magnitud conocido, pues

$$|S_N(x)| = \frac{|1 - x^{N+1}|}{|1 - x|} = \mathbf{O}(|x|^{N+1}).$$

La serie geométrica aparece con frecuencia en todo tipo de aplicaciones y, además, representa un paradigma de comportamiento para todas las series de potencias. Se suele utilizar como primer ejemplo de serie convergente y es fácil encontrar enunciados de problemas relacionados con los intereses de los alumnos en los que aparece esta serie. Por ejemplo, podemos comenzar la primera clase formulando la siguiente cuestión: "Supongamos que deseamos comprar un ordenador. Supongamos que en el primer año de vida del ordenador consumimos 10 Gb. de la memoria del disco duro y que cada año consumimos el $80\,\%$ de la memoria que consumimos el año anterior. ¿Qué capacidad debe tener el disco duro, si deseamos permanezca útil de de por vida y nunca borramos nada?" Obviamente, se trata de averiguar el valor de la serie

$$\sum_{k=0}^{\infty} 10\left(\frac{8}{10}\right)^k = 10\sum_{k=0}^{\infty}\left(\frac{8}{10}\right)^k = 10\frac{1}{1-\frac{8}{10}} = 50Gb,$$

que converge porque $0 \leq 8/10 < 1$.
Un ejemplo sorprendente, que quizás interese a los alumnos, en el que aparece de forma natural la serie geométrica -y proporciona la solución a un problema aparentemente complicado- es el el juego de la rana saltarina: En un papel cuadriculado, con cuadros lo bastante grandes como para que puedan contener una moneda en su interior (por ejemplo, un euro), trazamos una línea gruesa, horizontal, que deje exactamente 5 filas de cuadros por encima suyo. (Ver la figura). A continuación colocamos, en las filas que quedan bajo la linea fronteriza que hemos marcado, tantas monedas como queramos. Una vez colocadas las monedas, estas pueden moverse a lo largo del tablero siguiendo las siguientes reglas:

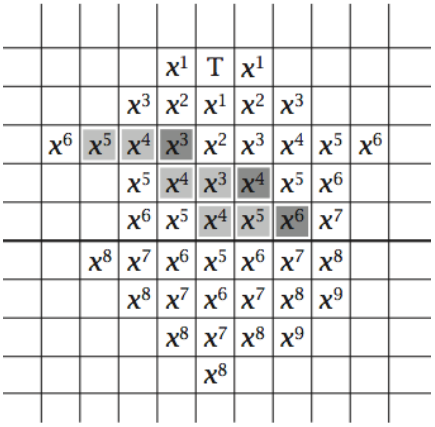
- Cada moneda puede moverse solo horizontalmente -de izquierda a derecha o de derecha a izquierda- o verticalmente -hacia arriba-.
- En ambos casos, una moneda solo se puede mover en determinada dirección si tiene, en dicha dirección, una moneda adyacente. En dicho caso, el movimiento se realiza saltando sobre la moneda compañera y cayendo una casilla más allá de esta, pero solo podrá hacerse si dicha casilla no está ocupada por otra moneda. Además, al realizar el movimiento, la moneda sobre la que se produce el salto, es retirada del tablero.

El nombre del juego se debe a la interpretación de las monedas como ranas que pueden saltar de un casillas a otras de acuerdo con ciertas reglas. Otra interpretación habitual -y de hecho, la original- es que el tablero representa una zona de conflicto entre dos países, la línea gruesa representa la frontera entre ambos países y las monedas son exploradores. La zona que queda por encima de la frontera es un desierto y se trata de conocer qué zonas del desierto pueden explorarse siguiendo las reglas anteriores.
Se supone que uno podría realizar saltos con enorme libertad, pues las reglas son aparentemente poco restrictivas. Sin embargo, algunas cosas no se pueden hacer, como veremos. Con solo dos monedas es fácil colocar una moneda por encima de la linea fronteriza. Ahora bien, para colocar una moneda en la segunda fila, por encima de la linea fronteriza, necesitamos $4 = 2^2$ monedas. Se puede probar con relativa facilidad que, para llegar a la fila 3, hacen falta $8 = 2^3$ monedas, y que esto se puede lograr de varias formas. Sin embargo, en la fila 4 se rompe la armonía, pues es posible demostrar que para llegar a ella son necesarias 20 monedas (en vez de $16 = 2^4$). Bien. El caso de la fila 5 es peor aún y, para su resolución, se requiere utilizar una serie geométrica. Veamos lo que sucede:

Teorema 2.27

En el juego de la rana saltarina es imposible alcanzar la fila 5 por encima de la linea fronteriza, independientemente de la disposición y la cantidad de monedas que se tengan.

Demostración. Consideramos la ecuación $x^2 + x = 1$. Sea $\chi = \frac{-1+\sqrt{5}}{2}$ la única solución positiva de la ecuación. Colocamos, sobre cada casilla del tablero, una potencia de χ del siguiente modo:



Es decir, sobre la fila quinta, por encima de la linea fronteriza, elegimos una casilla en la que colocamos el número $1 = \chi^0$ (o, si se prefiere, para identificar la casilla, la letra T). Dicha casilla representa la posición alcanzada por una moneda cuando avanzamos en el juego, en el caso de que en efecto podamos alcanzar la fila quinta en algún momento, cosa que asumimos momentáneamente con la esperanza de llegar a algún tipo de contradicción. A su derecha y a su izquierda, si nos desplazamos k casillas, colocamos el número χ^k. Finalmente, a partir de cualquiera de las casillas de dicha fila, si en ella se ha colocado el número χ^k, colocamos en las casillas que hay en la misma columna, si nos desplazamos s casillas, el número χ^{k+s}. De este modo tenemos etiquetadas todas las casillas de un semiplano infinito. Otra forma de explicar este etiquetado es la siguiente: una vez elegida la casilla T que ocuparía la moneda, en la fila 5, esta se etiqueta con 1 y dada cualquier otra casilla R del tablero infinito, la etiquetamos con $\chi^{d(T,R)}$, donde $d(T,R)$ representa el número de casillas más pequeño que podemos recorrer en un camino que sigue solo direcciones horizontales y verticales que une ambas casillas, cantidad que llamamos distancia del tablero entre las casillas T, R.

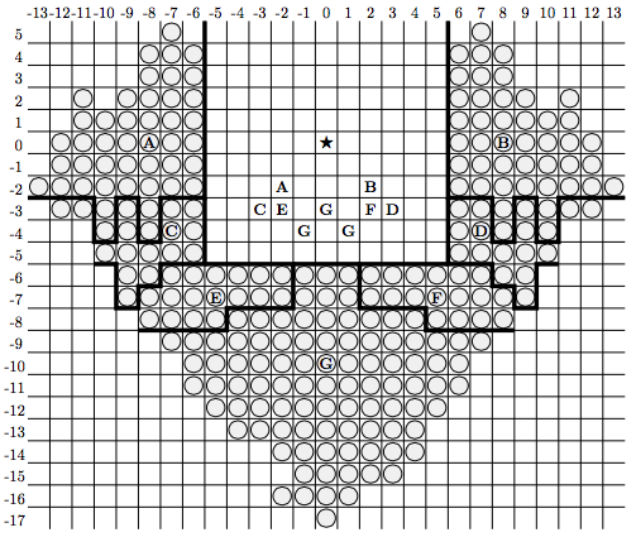
Si consideramos la suma S de todos los valores con los que hemos etiquetado las filas que quedan por debajo de la frontera, observamos que, al ser $1 - \chi = \chi^2$,

$$\begin{aligned} S &= (\chi^5 + \chi^6 + \cdots) + 2(\chi^6 + \chi^7 + \cdots) + 2(\chi^7 + \chi^8 + \cdots) + \cdots \\ &= \chi^5(1 + \chi + \cdots) + 2\chi^6(1 + \chi + \cdots) + 2\chi^7(1 + \chi + \cdots) + \cdots \\ &= \frac{1}{1-\chi}\chi^5 + \frac{2}{1-\chi}(\chi^6 + \chi^7 + \cdots) \\ &= \frac{\chi^5}{1-\chi} + \frac{2\chi^6}{1-\chi}\frac{1}{1-\chi} \\ &= \frac{\chi^5}{\chi^2} + \frac{2\chi^6}{\chi^2}\frac{1}{\chi^2} \\ &= \chi^3 + 2\chi^2 = \chi - \chi^2 + 2\chi^2 = \chi + \chi^2 = 1. \end{aligned}$$

Ahora bien, imaginemos que colocamos monedas libremente en la zona que queda bajo la línea fronteriza. Podemos asociar a nuestra disposición de monedas un peso, consistente en sumar los valores de las etiquetas que ocupan las monedas. Obviamente, dicho valor será siempre estrictamente menor que 1 si disponemos de una cantidad finita de monedas (de hecho, también lo es para infinitas monedas, siempre que no llenemos todo el semiplano inferior determinado por la frontera). Lo interesante es que con cada movimiento del juego el peso asociado a la nueva disposición de monedas coincide con el que tenía en la posición anterior. Se sigue que nunca podremos colocar una moneda en el escaque que tiene etiqueta 1, lo cual contradice nuestra hipótesis de partida (de que el teorema era incorrecto) y demuestra el teorema. □

Como se ve, el uso de la serie geométrica permite demostrar que existen ciertas obstrucciones o que un juego tiene determinados límites. Esto es así porque la idea de asociar pesos a las posiciones de un juego, de modo que dichos pesos sean invariantes conforme evoluciona el juego, permite conocer ciertas propiedades no triviales del juego y, con frecuencia, requiere utilizar series numéricas. Si se piensa, este tipo de ideas podrían ayudar a diseñar nuevos juegos.

Varias observaciones son importantes: El juego de la rana saltarina (también llamado solitario de los exploradores) admite numerosas variantes y su estudio desde el punto de vista matemático está aún vigente. Una de las variantes de mayor interés consiste en averiguar si una situación concreta es o no posible. Y, caso de ser efectivamente posible, cuál es la posición inicial más simple a partir de la cual se puede llegar a la situación descrita. Por ejemplo, durante mucho tiempo se consideró abierto el problema de averiguar si, colocando exploradores en el exterior de un cuadrado de lado 11 es o no posible colocar un explorador en el centro de dicho cuadrado. Este problema fue resuelto recientemente en sentido positivo por los matemáticos italianos Luciano Guala, Stefano Leucci, Emanuele Natale, y Roberto Tauraso, siendo la configuración ganadora la siguiente:



(En la figura se destacan las posiciones de varios exploradores, denotados por A, B, C, D, E, F, G, que están encerrados en un recinto que se ha resaltado, y en el interior del cuadrado de lado 11 se muestra la posición a la que llega cada uno de estos exploradores utilizando solamente las piezas que hay en el recinto que le contiene. Así, en el interior del cuadrado de lado 11 se ven las posiciones alcanzadas por dichos exploradores. A partir de estas posiciones es fácil comprobar que se puede alcanzar el centro del cuadrado).

Además, curiosamente este solitario de los exploradores es un juego bastante antiguo en el que el propio Leibniz, uno de los fundadores del Cálculo, mostró interés, como revela claramente la siguiente cita[4]:

"No hace demasiado tiempo se ha extendido el uso de un juego excelente, llamado Solitario, al que he jugado por mi cuenta (...) Se llena un tablero con piedras colocadas en agujeros, que pueden ser eliminadas. Ahora, ninguna piedra puede ser eliminada a menos que esta pueda ser saltada por otra piedra adyacente que debe caer en un agujero vacío también adyacente. Gana quien sea capaz de quedarse, en la última jugada, con una sola piedra. El juego puede jugarse de forma más elegante en sentido inverso: después de haber colocado a voluntad una piedra en un tablero vacío, colocar el resto de piedras observando, para la adición de piedras, la misma regla que se indicó anteriormente para su eliminación. Por tanto, podríamos llenar el tablero o, lo que es más inteligente, dar forma a una figura predeterminada con las piedras, tal vez un triángulo, un

[4]Gottfried Wilhelm Leibniz, Miscellanea Berolinensia 1 (1710) 24.

cuadrilátero, un octógono o alguna otra forma. si ello es posible. Pero esta tarea no es en absoluto siempre posible, por lo que resultaría valioso el arte de averiguar lo que se puede lograr (...) ”

Una vez se ha definido el concepto de serie numérica, así como su convergencia y/o divergencia, y se han introducido algunos ejemplos no triviales, explicamos algunas propiedades elementales de las series:

- Si $\sum_{k=0}^{\infty} a_k < \infty$, entonces $\lim_{k\to\infty} a_k = 0$. La implicación contraria es falsa porque $\sum_{k=1}^{\infty} \frac{1}{k} = +\infty$. En efecto, si denotamos por $H_n = \sum_{k=1}^{n} \frac{1}{k}$ entonces

$$\begin{aligned} H_{2^{m+1}} &= H_{2^m} + \frac{1}{2^m+1} + \frac{1}{2^m+2} + \cdots + \frac{1}{2^{m+1}} \\ &\geq H_{2^m} + \frac{1}{2^{m+1}} + \frac{1}{2^{m+1}} + \cdots + {}^{(2^m \text{ términos})} \frac{1}{2^{m+1}} \\ &= H_{2^m} + 2^m \frac{1}{2^{m+1}} = H_{2^m} + \frac{1}{2}, \end{aligned}$$

  lo que quiere decir que la serie armónica contiene infinitos bloques de términos que suman $1/2$ y, por tanto, las sumas parciales tienden a $+\infty$. De hecho,

$$H_{2^{m+1}} \geq \frac{1}{2} + H_{2^m} \geq \frac{1}{2} + \frac{1}{2} + H_{2^{m-1}} \geq \cdots \geq \frac{m+1}{2}$$

  y, por tanto, $\lim_{m\to\infty} H_{2^m} = \infty$. Los números H_n se llaman números armónicos y aparecen con frecuencia en el estudio asintótico de algoritmos.

- Si $\sum_{k=0}^{\infty} a_k < \infty$ y $\sum_{k=0}^{\infty} b_k < \infty$, entonces $\sum_{k=0}^{\infty}(\alpha a_k + \beta b_k) = \alpha \sum_{k=0}^{\infty} a_k + \beta \sum_{k=0}^{\infty} b_k < \infty$, sean quienes sean los números α, β.

- Si $\sum_{k=0}^{\infty} |a_k| < \infty$ (se dice que la serie es absolutamente convergente), entonces $\sum_{k=0}^{\infty} a_k < \infty$. Por otra parte, existen series convergentes que no lo son absolutamente, como la serie $\sum_{k=1}^{\infty}(-1)^k/k$.

- Si $\sum_{k=0}^{\infty} a_k < \infty$ y $\sum_{k=0}^{\infty} b_k < \infty$, entonces $(\sum_{k=0}^{\infty} a_k)\,(\sum_{k=0}^{\infty} b_k) = \sum_{n=0}^{\infty} (\sum_{k=0}^{n} a_k b_{n-k}) < \infty$.

Particularmente importantes son las series de términos no negativos, entre otras razones porque la convergencia absoluta de una serie (que es la convergencia de una serie de términos no negativos) implica la convergencia de la misma y, por tanto, a menudo el estudio de la convergencia de una serie pasa por el de una serie de términos no negativos asociada a la misma. Otra razón fundamental es que estas series tienen sumas parciales monótonas y, por tanto, su convergencia es decidida directamente a partir de la acotación uniforme de dichas sumas. Además, para las series de términos positivos es sencillo demostrar varios criterios de convergencia que resultan muy sencillos de aplicar. Finalmente, en numerosas aplicaciones en algorítmica aparecen este tipo de series, siendo estas divergentes, y lo que se estudia es el orden de magnitud de la sucesión de sumas parciales asociadas. Estos son, por tanto, los temas principales a los que se presta atención en esta parte de la asignatura.

Teorema 2.28

Sea $f : [1, \infty) \to \mathbb{R}^+$ continua y decreciente a partir de un punto. Supongamos que $a_k = f(k)$ para todo $k \in \mathbb{N}$. Entonces $\sum_{k=1}^{\infty} a_k < \infty$ si y solo si $\int_1^{\infty} f < \infty$. Además, la sucesión de sumas parciales $S_N = \sum_{k=1}^{N} a_k$ satisface $S_N \sim J_N$, donde $J_N = \int_1^N f$.

Demostración. Como la función decrece en un intervalo del tipo $[a, \infty)$ para cierto $a \geq 0$, podemos considerar solo la suma de los términos a_k con $k \geq [\alpha]$ o, si se quiere, podemos asumir sin pérdida de generalidad que $a = 1$. Entonces el área $A_n = \int_n^{n+1} f$ satisface claramente las desigualdades

$$a_{n+1} = f(n+1) \leq A_n \leq f(n) = a_n$$

y, por tanto,

$$S_N - a_1 = \sum_{k=1}^{N} a_{k+1} \leq \int_1^N f \leq \sum_{k=1}^{N} a_k = S_N, \text{ para todo } N \in \mathbb{N}.$$

De la primera de las desigualdades anteriores se deduce que si $\int_1^\infty f = \lim_{N\to\infty} \int_1^N f < \infty$, entonces S_N permanece acotada superiormente y, por tanto, converge (por ser una sucesión monótona creciente). Análogamente, de la segunda desigualdad se deduce que la convergencia de la sucesión de sumas parciales implica la convergencia de la integral $\int_1^\infty f$. Es más, si denotamos $J_N = \int_1^N f$, las desigualdades anteriores se expresan como

$$S_N - a_1 \leq J_N \leq S_N$$

Por tanto, si $\lim_{N\to\infty} S_N = \infty$, tenemos que

$$\lim_{N\to\infty} \frac{S_N - a_1}{S_N} = 1 \leq \lim_{N\to\infty} \frac{J_N}{S_N} \leq 1$$

Es decir, $\lim_{N\to\infty} \frac{J_N}{S_N} = 1$ y, por tanto, $S_N \sim J_N$.
Por otra parte, si $\lim_{N\to\infty} S_N = \alpha$ entonces J_N es una sucesión monótona creciente y acotada, que tiene límite un cierto $\beta \geq 0$. Si $\beta \neq 0$ ambas sucesiones son equivalentes. Ahora bien, la única forma de que $\beta = 0$ es que f sea idénticamente nula, en cuyo caso $S_N = J_N = 0$ para todo N. Esto concluye la prueba. □
Como consecuencia del resultado anterior podemos estudiar en detalle la serie armónica generalizada:

$$\sum_{k=1}^{\infty} \frac{1}{k^p},$$

cosa que hacemos estudiando separadamente los casos $0 < p < 1$, $p = 1$ y $p > 1$. Concretamente, la función $f(x) = 1/x^p$ decrece en $[1, \infty)$ (pues $f'(x) = (-p)x^{-p-1} < 0$ en dicho intervalo) y, por tanto, podemos aplicar el teorema anterior. Tenemos, pues, que considerar las integrales $J_n = \int_1^n \frac{1}{t^p}$. Ahora bien, si $p = 1$ entonces $J_n = \log(n)$ y por tanto la serie armónica $\sum_{k=1}^{\infty} \frac{1}{k}$ diverge, siendo sus sumas parciales del orden de $\log(n)$. Si $p \neq 1$, entonces $J_n = \frac{n^{-p+1}}{-p+1} - \frac{1}{-p+1} \sim n^{1-p}$, que converge a infinito si $0 \leq p < 1$ (y por tanto en esos casos la serie diverge y el orden de magnitud de las sumas parciales es n^{1-p}) y si $p > 1$ la serie converge a $\frac{1}{p-1}$.

Teorema 2.29

Sea $f : [1, \infty) \to \mathbb{R}^+$ continua y creciente a partir de un punto. Supongamos que $a_k = f(k)$ para todo $k \in \mathbb{N}$. Entonces $\sum_{k=1}^{\infty} a_k = \infty$. Además, si $J_N = \int_1^N f$ satisface $a_k << J_k$, entonces la sucesión de sumas parciales $S_N = \sum_{k=1}^{N} a_k$ satisface $S_N \sim J_N$.

Demostración. Que la serie diverge se sigue de que, para $n > a$, donde hemos asumido que $f_{|[a,\infty)}$ es una función creciente, se tiene que $a_{n+1} = f(n+1) \geq \int_n^{n+1} f \geq f(n) = a_n \geq f(a) > 0$ y, por tanto, $\sum_{k=[a]+1}^{\infty} a_k = +\infty$.
Para probar la segunda afirmación asumimos, sin pérdida de generalidad, que $a = 1$. Entonces el resultado se deduce de las desigualdades

$$S_N - a_N = S_{N-1} = \sum_{k=1}^{N-1} a_k \leq \int_1^N f = J_N \leq \sum_{k=2}^{N} a_k = S_N - a_1, \text{ para todo } N \in \mathbb{N}.$$

En efecto, dividiendo por J_N obtenemos que

$$\frac{S_N - a_N}{J_N} \leq 1 \leq \frac{S_N - a_1}{J_N},$$

y, al ser $\lim_{N\to\infty} \frac{a_N}{J_N} = 0$, tenemos, por la primera de las desigualdades, que

$$\lim_{N\to\infty} \frac{S_N}{J_N} = \lim_{N\to\infty} \frac{S_N - a_N}{J_N} \leq 1$$

y, además, por la segunda de las desigualdades, tenemos que

$$\lim_{N\to\infty} \frac{S_N}{J_N} = \lim_{N\to\infty} \frac{S_N - a_1}{J_N} \geq 1$$

Es decir, $\lim_{N\to\infty} \frac{J_N}{S_N} = 1$ y, por tanto, $S_N \sim J_N$. □

Un ejemplo sencillo en el que se utiliza el teorema anterior es la estimación, para $p > 0$, del orden de magnitud de las sumas parciales de la serie (divergente) $\sum_{k=1}^{\infty} k^p$. Claramente, este ejemplo se obtiene al aplicar el teorema con $f(t) = t^p$, que es creciente en $[1, \infty)$. Así, $S_n = \sum_{k=1}^{n} k^p$ verifica $S_n \sim J_n = \int_1^n t^p = \frac{n^{p+1}}{p+1} - \frac{1}{p+1} \sim n^{p+1}$, pues $n^p << J_n$.

Además de los dos criterios integrales que acabamos de mostrar, se explican otros criterios útiles para la determinación del carácter de una serie de términos no negativos. Concretamente, se introducen el criterio de comparación, el criterio de la raíz y el criterio del cociente:

Teorema 2.30 Criterio de comparación

Sean $(a_n), (b_n)$ sucesiones de términos no negativos y consideremos sus series asociadas: $\sum_{n=1}^{\infty} a_n$, $\sum_{n=1}^{\infty} b_n$, cuyas sumas parciales asociadas son denotadas por S_N, T_N, respectivamente. Entonces:

- Si $a_n \sim b_n$, entonces ambas series tienen forzosamente el mismo carácter. Además, $S_N \sim T_N$.
- Si $a_n << b_n$ y $\sum_{n=1}^{\infty} b_n < \infty$, entonces $\sum_{n=1}^{\infty} a_n < \infty$.
- Si $a_n << b_n$ y $\sum_{n=1}^{\infty} a_n = \infty$, entonces $\sum_{n=1}^{\infty} b_n = \infty$.

Teorema 2.31 Criterio de la raíz

Sea $\sum_{n=1}^{\infty} a_n$ una serie de términos no negativos. Sea $\lim_{n\to\infty}(\sup_{k\geq n} |a_k|^{1/k}) = L$. Entonces:

- Si $L < 1$, la serie converge.
- Si $L > 1$, la serie diverge
- Para $L = 1$ existen ejemplos donde la serie converge y ejemplos donde la serie diverge.

Teorema 2.32 Criterio del cociente

Sea $\sum_{n=1}^{\infty} a_n$ una serie de términos no negativos. Supongamos que $\lim_{n\to\infty} \frac{a_{n+1}}{a_n} = L$. Entonces:

- Si $L < 1$, la serie converge.
- Si $L > 1$, la serie diverge
- Para $L = 1$ existen ejemplos donde la serie converge y ejemplos donde la serie diverge.

N El criterio de comparación es fundamental para poder demostrar los criterios de la raíz y del cociente. Por ejemplo, si $L = \lim_{n\to\infty}(\sup_{k\geq n}|a_k|^{1/k}) < 1$, entonces existe $\rho < 1$ tal que $L < \rho$ y por tanto, a partir de cierto valor $N_0 \in \mathbb{N}$ se tiene que $|a_n|^{\frac{1}{n}} < \rho$, lo que significa que $|a_n| < \rho^n$ y, por tanto, $\sum_{n\geq N_0}|a_n| \leq \sum_{n\geq N_0}\rho^n \leq \frac{1}{1-\rho} < \infty$. Por otra parte, si $L > 1$ entonces podemos buscar $\rho > 1$ con $L > \rho$ y, a partir de cierto valor N_0 se verifica la desigualdad $|a_n|^{\frac{1}{n}} > \rho$, lo que nos lleva a $\sum_{n\geq N_0}|a_n| > \sum_{n\geq N_0}\rho^n = \infty$.
De forma similar, si $L = \lim_{n\to\infty}\frac{a_{n+1}}{a_n} < 1$, entonces

$$|a_{N_0+k}| \leq \rho|a_{N_0+k-1}| \leq \rho^2|a_{N_0+k-2}| \leq \cdots \leq \rho^k|a_{N_0}|$$

para cierto $\rho \in (L, 1)$ y todo $k \geq 0$, lo que conduce a la desigualdad

$$\sum_{n\geq N_0}|a_n| = \sum_{k=0}^{\infty}|a_{N_0+k}| \leq |a_{N_0}|\sum_{k=0}^{\infty}\rho^k < \infty.$$

(El caso $L > 1$ también se deduce de forma similar)

Otro criterio interesante para el estudio de series de términos no negativos es el siguiente:

Teorema 2.33 (Criterio de condensación de Cauchy)

Supongamos que $a_1 \geq a_2 \geq a_3 \geq \cdots \geq 0$. Entonces la serie $\sum_{k=1}^{\infty}a_k$ converge si y solo si la serie

$$\sum_{k=0}^{\infty}2^k a_{2^k}$$

converge.

Demostración, Como las sumas parciales de ambas series son sucesiones crecientes, basta estudiar cuándo están acotadas superiormente. Para ello, consideramos las sumas

$$S_n = \sum_{k=1}^{n}a_k \text{ y } T_k = \sum_{i=0}^{k}2^i a_{2^i}.$$

Si $n \leq 2^k$, entonces

$$\begin{aligned} S_n &= a_1 + \cdots + a_n \\ &\leq a_1 + \cdots + a_{2^{k+1}-1} \\ &= a_1 + (a_2 + a_3) + (a_4 + \cdots + a_7) + \cdots + (a_{2^k} + a_{2^k+1} + \cdots a_{2^{k+1}-1}) \\ &\leq a_1 + 2a_2 + 2^2 a_{2^2} + \cdots 2^k a_{2^k} = T_k \end{aligned}$$

Por otra parte, si $n > 2^k$, entonces

$$\begin{aligned} S_n &= a_1 + a_2 + \cdots + a_n \\ &\geq a_1 + a_2 + \cdots + a_{2^k} \\ &= a_1 + a_2 + (a_3 + a_4) + \cdots + (a_{2^{k-1}+1} + \cdots a_{2^k}) \\ &\geq \frac{1}{2}a_1 + a_2 + 2a_4 + \cdots 2^{k-1}a_{2^k} = \frac{1}{2}T_k. \end{aligned}$$

Es decir, para $n \leq 2^k$ tenemos $S_n < T_k$ y para $n > 2^k$ tenemos que $2S_n > T_k$. Esto significa que ambas series están o bien simultáneamente no acotadas o bien simultáneamente acotadas, que es lo que queríamos probar. □
Cuando una serie no es absolutamente convergente, pero converge, se produce el siguiente fenómeno interesante: el valor de la suma depende del orden con el que se suman los términos (esto no sucede con las series absolutamente convergentes, aunque dicha propiedad no es trivial). De hecho, se verifica el siguiente resultado:

Teorema 2.34 (de Reordenación de Riemann)

Supongamos que $\sum_{k=0}^{\infty} a_k < \infty$ pero $\sum_{k=0}^{\infty} |a_k| = \infty$. Entonces para cada $\alpha \in \mathbb{R}$ existe una permutación $\sigma : \mathbb{N} \to \mathbb{N}$ tal que

$$\sum_{k=0}^{\infty} a_{\sigma(k)} = \alpha.$$

Demostración. Esbozamos aquí la prueba, pues contiene un argumento muy elegante. Si la serie $\sum_{k=0}^{\infty} a_k$ converge pero no lo hace absolutamente, ello se debe a que tanto los términos positivos como los términos negativos que aparecen en la misma son series divergentes. Esto es así porque si ambas convergen, entonces la serie es absolutamente convergente y si una de ellas converge pero la otra diverge, no podemos estar ante una serie convergente. Así pues, descomponemos la sucesión $\{a_k\} = \{p_k\} \cup \{(-1)n_k\}$, donde p_k denota el k-ésimo término de la sucesión (cuando la leemos en su orden original, $a_0, a_1, a_2, \cdots$,) que es > 0 y $(-1)n_k$ denota el k-ésimo término de la sucesión que es < 0. Entonces $\sum_{k=0}^{\infty} p_k = \sum_{k=0}^{\infty} n_k = +\infty$.
Dado $\alpha \in \mathbb{R}$, procedemos del siguiente modo: Comenzamos a sumar los términos positivos $p_1, p_2, \cdots$ hasta que, por primera vez, superemos el valor de α:

$$p_1 + p_1 + \cdots + p_{N_1} > \alpha.$$

Esto es siempre posible porque la serie de término general p_k diverge.
A continuación vamos sumando los términos negativos $-n_1, -n_2, \cdots$ hasta que el nuevo valor quede por debajo de α:

$$p_1 + p_1 + \cdots + p_{N_1} - n_1 - n_2 - \cdots - n_{M_1} < \alpha,$$

cosa que podemos hacer por idéticos motivos (la serie $\sum_{k=1}^{\infty} n_k$ diverge. Pues bien, se trata de repetir este proceso de forma indefinida. Como $\lim_{k\to\infty} p_n = \lim_{k\to\infty} n_k = 0$, la cantidad que rebasamos el valor de α en una u otra dirección, en cada paso, tiende a cero. Por tanto, la serie

$$p_1 + p_1 + \cdots + p_{N_1} - n_1 - n_2 - \cdots - n_{M_1} + p_{N_1+1} + p_{N_1+2} + \cdots + p_{N_1+N_2} - n_{M_1+1} - n_{M_1+2} - \cdots - n_{M_1+M_2} + \cdots,$$

que es una serie obtenida a partir de la original mediante un determinado proceso de reordenación, converge al valor de α. □

N Obsérvese que, de manera indirecta, acabamos de demostrar que hay al menos tantas permutaciones de $\mathbb{N}$ distintas como números reales. De hecho, ambos conjuntos tienen idéntico cardinal.

Para que el teorema anterior adquiera fuerza, represente un hecho sorprendente, es necesario exponer al menos un ejemplo de serie convergente que no lo es absolutamente. Para ello, basta considerar la suma $\sum_{k=1}^{\infty} \frac{(-1)^k}{k}$. Veamos que, en efecto, esta serie converge (que no lo hace absolutamente se debe a que la serie armónica diverge). Consideremos su suma parcial N-ésima:

$$S_N = (-1) + \frac{1}{2} + \frac{(-1)}{3} + \cdots + \frac{(-1)^N}{N}$$

Entonces $|S_{N+1} - S_N| = \frac{1}{N+1}$, $|S_{N+2} - S_N| = |\frac{1}{N+1} - \frac{1}{N+2}| < \frac{1}{N+1}$, ... y, en general,

$$|S_{N+k} - S_N| \leq \frac{1}{N+1} \text{ (para todo } k \geq 1).$$

Esto prueba que la sucesión $\{S_N\}$ es de Cauchy y, por tanto, converge.
Aunque esta demostración es perfectamente válida, no sería apropiada para mostrarla en clase porque no se han explicado las sucesiones de Cauchy. Así, una prueba directa para la convergencia de la serie se hace necesaria. Veamos cómo lograrlo. Para empezar, sabemos que

$$1 - x + \cdots (-1)^N x^N = \frac{1 - (-1)^{N+1} x^{N+1}}{1+x}$$

y, por tanto,

$$\frac{1}{1+x} = 1 - x + \cdots (-1)^N x^N + \frac{(-1)^{N+1} x^{N+1}}{1+x}$$

Integrando a ambos lados de la igualdad en el intervalo $[0, 1]$, obtenemos

$$\log 2 = \sum_{k=0}^{N} \frac{(-1)^k}{k+1} + (-1)^{N+1} \int_0^1 \frac{x^{N+1}}{1+x} dx$$

Por tanto, multiplicando a ambos lados por -1, obtenemos que

$$\begin{aligned} -\log 2 &= \sum_{k=0}^{N} \frac{(-1)^{k+1}}{k+1} + (-1)^N \int_0^1 \frac{x^{N+1}}{1+x} dx \\ &= S_{N+1} + (-1)^N \int_0^1 \frac{x^{N+1}}{1+x} dx \end{aligned}$$

Se sigue que

$$| -\log 2 - S_{N+1}| \leq \left| \int_0^1 \frac{x^{N+1}}{1+x} dx \right| \leq \left| \int_0^1 x^{N+1} dx \right| = \frac{1}{N+2} \to 0$$

y, por tanto,

$$\sum_{k=1}^{\infty} \frac{(-1)^k}{k} = -\log 2.$$

De este modo no sólo hemos probado la convergencia de la serie sino que, además, hemos determinado su valor exacto.
En realidad, este ejemplo es un caso particular de un fenómeno general, que fue estudiado por Leibniz y que exponemos a continuación:

Definición 2.7

Una serie $\sum_{k=1}^{\infty} a_k$ se dice "alternada" si a_n y a_{n+1} tienen signo opuesto para todo $n \in \mathbb{N}$.

Teorema 2.35 (Criterio de Leibniz)

Supongamos que $\sum_{k=1}^{\infty} a_k$ es una serie alternada, que $|a_{n+1}| \leq |a_n|$ para todo n y que $\lim_{n\to\infty} a_n = 0$. Entonces:

- La serie $\sum_{k=1}^{\infty} a_k$ converge a cierto valor $\alpha \in \mathbb{R}$

- Si $S_N = a_1 + a_2 + \cdots + a_N$ es la suma parcial N-ésima de la serie, entonces $|\alpha - S_N| \leq |a_{N+1}|$.
- Con la notación anterior, el signo de $\alpha - S_N$ coincide con el de a_{N+1}.

2.4.2. Las prácticas

Las clases de practicas de este tema se dedican esencialmente a la resolución de ejercicios en pizarra -y, ocasionalmente, con apoyo de SAGE. En los problemas se abordan, principalmente, la convergencia/divergencia de series, el cálculo explícito de la suma cuando la serie es convergente, y la determinación del orden de magnitud de las sumas parciales de una serie cuando esta diverge. Algunos ejemplos sirven, además, para relacionar las series numéricas con temas relacionados con la informática -como el análisis asistólico de algoritmos-. Otros ejemplos tienen motivación geométrica y sirven, sobre todo, para asimilar los conceptos y teoremas fundamentales del tema. SAGE se utiliza para visualizar la relación que existe entre series de términos positivos e integrales, y también se usa para dibujar la sucesión de sumas parciales de una serie dada, antes de proceder a estudiar su posible convergencia/divergencia. Exponemos aquí algunos ejemplos de problemas tipo que se resuelven en las sesiones de prácticas, así como la manera de usar SAGE para visualizar los conceptos que se han mostrado en las clases de teoría.

Ejemplo 2.7 Utilizar el criterio de condensación de Cauchy para estudiar el carácter de la serie

$$\sum_{n=1}^{\infty} \frac{1}{n^p}$$

Si $p < 0$, entonces $1/n^p = n^{|p|}$, que no converge a cero y, por tanto, la serie diverge. Suponemos, pues, que $p > 0$. Entonces la sucesión $a_n = 1/n^p$ es decreciente y positiva, por lo que podemos aplicar el Criterio de Condensación de Cauchy para afirmar que esta converge si y solo si la siguiente serie converge:

$$\sum_{k=0}^{\infty} 2^k a_{2^k} = \sum_{k=0}^{\infty} 2^k \frac{1}{(2^k)^p} = \sum_{k=0}^{\infty} (2^{(1-p)})^k.$$

Ahora bien, esta última serie es una serie geométrica de razón $r = 2^{1-p}$ y, obviamente, $r < 1$ si y solo si $p > 1$. Por tanto, la serie converge solo cuando $p > 1$. En el resto de casos, diverge.

Ejemplo 2.8 Estudiar el carácter de las siguientes series:

(a) $\sum_{n=1}^{\infty} \frac{\sqrt{\log n}}{n^2+\sqrt{n}}$ (b) $\sum_{n=1}^{\infty} \frac{\cos(2n)}{n^2}$ (c) $\sum_{n=1}^{\infty} \frac{(n+1)!}{4^n+5^n}$

Consideramos en primer término la serie dada en (a). Se trata de una serie de términos positivos cuyo término general satisface que

$$\frac{\sqrt{\log n}}{n^2+\sqrt{n}} << \frac{n^{1/2}}{n^2} = \frac{1}{n^{3/2}}$$

y, como

$$\sum_{n=1}^{\infty} \frac{1}{n^{3/2}} < \infty$$

se trata de una serie convergente.

Tomemos ahora la serie definida en (b). Obviamente, $\left|\frac{\cos(2n)}{n^2}\right| \leq \frac{1}{n^2}$ y como la serie $\sum_{n=1}^{\infty} \frac{1}{n^2} < \infty$, se tiene que la serie $\sum_{n=1}^{\infty} \frac{\cos(2n)}{n^2}$ es absolutamente convergente y, por tanto, convergente.

Finalmente, la serie definida por (c) diverge por la sencilla razón de que $(n+1)! >> 4^n + 5^n$ y, por tanto, el término general de la serie, $\frac{(n+1)!}{4^n+5^n}$ no converge a 0, que es una condición necesaria para la convergencia de la serie.

Ejemplo 2.9 Utilizar SAGE para visualizar la sucesión de sumas parciales de la serie $\sum_{k=1}^{\infty} \frac{1}{k^p}$ para $p = 1/2$ y $p = 2$. Haced lo mismo con la serie alternada $\sum_{k=1}^{\infty} (-1)^k \frac{1}{k^p}$ para los mismos valores del parámetro p.

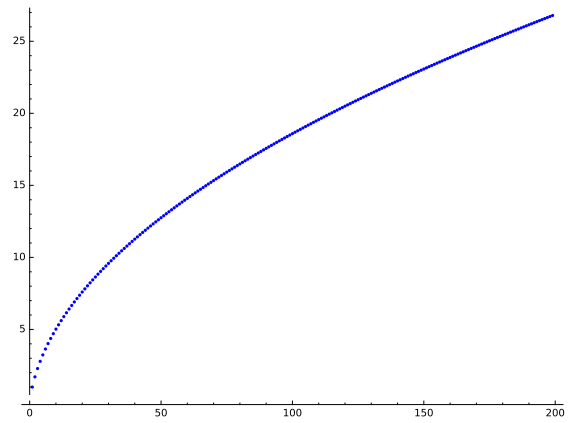
Comenzamos visualizando las sumas parciales de estas series utilizando SAGE. Primero tomamos $p = 1/2$. Para ello, utilizamos el siguiente programa de SAGE que define las sumas parciales de las series $\sum_{k=1}^{\infty} \frac{1}{k^p}$ y $\sum_{k=1}^{\infty} (-1)^k \frac{1}{k^p}$ con $p = 1/2$:

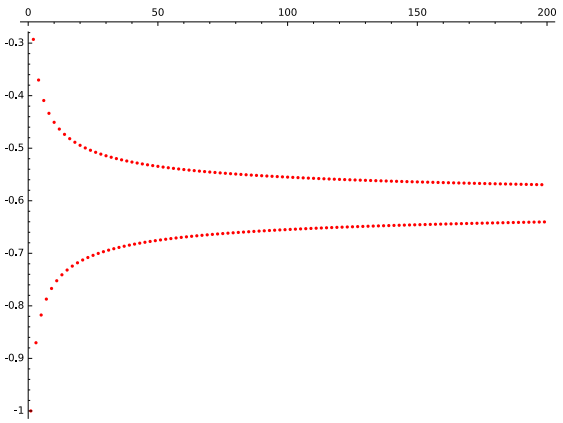
```
k=var('k');p=0.5;
def a(n):
    if n==0:
        return 0
    else:
        return float(sum(1/(k^p),k,1,n))

def b(n):
    if n==0:
        return 0
    else:
        return float(sum((-1)^k/(k^p),k,1,n))
```

y, a continuación, pintamos ambas sucesiones (en azul a la términos positivos, en rojo la serie alternada)

```
sage: list_plot([(n,a(n)) for n in range(1,200)],color='blue')
Graphics object consisting of 1 graphics primitive
sage: list_plot([(n,b(n)) for n in range(1,200)],color='red')
Graphics object consisting of 1 graphics primitive
```





Obviamente, la serie de términos positivos diverge y la alternada converge, que es lo que queríamos visualizar. Repetimos ahora los cálculos con $p = 2$:

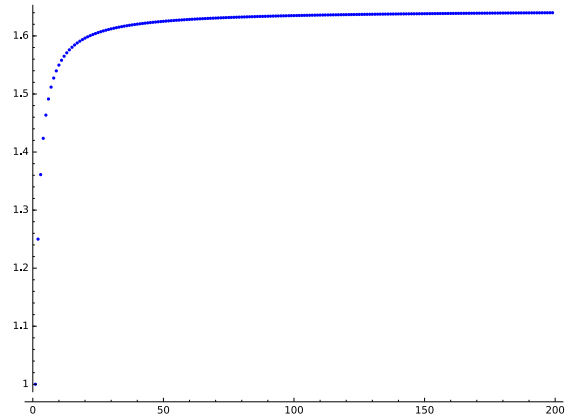
```
k=var('k');p=2;
def h(n):
    if n==0:
        return 0
    else:
```

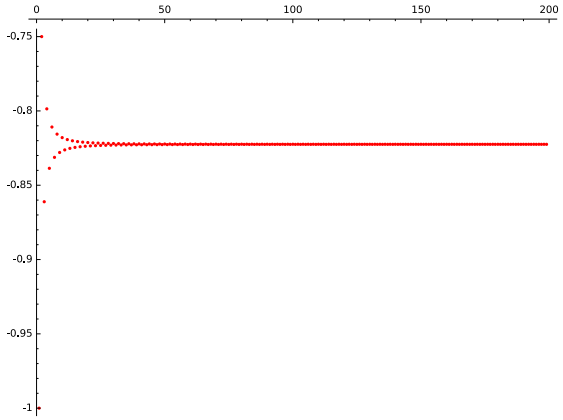
```
            return float(sum(1/(k^p),k,1,n))

    def t(n):
        if n==0:
            return 0
        else:
            return float(sum((-1)^k/(k^p),k,1,n))
```

```
sage: list_plot([(n,h(n)) for n in range(1,200)],color='blue')
Graphics object consisting of 1 graphics primitive
sage: list_plot([(n,t(n)) for n in range(1,200)],color='red')
Graphics object consisting of 1 graphics primitive
```

Lo cual proporciona los siguientes gráficos:





Esta vez ambas series convergen, como era de esperar.

Ejemplo 2.10 La información introducida cada año en el disco duro de un ordenador es la mitad que la introducida en el año anterior. Si el primer año se almacenan 20 GB, ¿cuál es el tamaño mínimo que debe tener el disco para garantizar que no se va a saturar?

Este problema es un ejemplo sencillo del uso de las series geométricas. El primer año se utilizan 20 GB y el k-ésimo año se usan $\frac{1}{2^k}20$ GB. Por tanto, el tope de memoria a usar, transcurrida toda la eternidad, es

$$20\sum_{k=0}^{\infty}\left(\frac{1}{2}\right)^k = 20\cdot 2 = 40 \text{ GB}.$$

Se necesita un disco duro de al menos 40 GB.

Ejemplo 2.11 Estudiar el carácter de las siguientes series:

$$\text{(a) } \sum_{n=1}^{\infty}\frac{k^n n!}{n^n}, \text{ donde } 0<k<e \qquad \text{(b) } \sum_{n=1}^{\infty} n^p p^n, \text{ donde } p>0 \qquad \text{(c) } \sum_{n=2}^{\infty}\frac{(1}{(n\log n)^2}$$

En la primera de las series a considerar el término general es $a_n = \frac{k^n n!}{n^n}$. Por tanto, para aplicar el criterio del cociente hay que estimar el cociente

$$\frac{a_{n+1}}{a_n} = \frac{k^{n+1}(n+1)!}{n^n}\cdot\frac{n^n}{k^n n!} = k\left(\frac{n}{n+1}\right)^n \to \frac{k}{e} < 1$$

Por tanto, se trata de una serie convergente.
Consideramos ahora la segunda serie. Si $p = 1$ el término general es n y la serie diverge. Aplicamos el criterio de la raíz al resto de casos. Como $a_n = n^p p^n$, es claro que

$$\lim_{n\to\infty} |a_n|^{\frac{1}{n}} = \lim_{n\to\infty} p(n^{1/n})^p = p.$$

Por tanto, la serie converge para $p < 1$ y diverge para $p \geq 1$.
Finalmente, para la serie definida en el apartado (c) usamos el criterio integral: $a_n = f(n)$ donde $f(x) = 1/(x \log x)^2$ es una función positiva decreciente en el intervalo $[2, \infty)$. Por tanto, la convergencia de la serie está completamente ligada a la de la integral impropia $\int_2^\infty \frac{1}{x^2(\log x)^2} dx$. Ahora bien, como $0 < \frac{1}{x^2(\log x)^2} << \frac{1}{x^2}$ en dicho intervalo, y como $\int_2^\infty \frac{1}{x^2} dx < \infty$, podemos concluir que estamos ante una serie convergente.

2.5 Cronograma

Se propone la siguiente tabla, en la que se han distribuido por horas todas las actividades a realizar -incluyendo exámenes, etc.,- en un curso académico completo:

Actividad Formativa	Metodología	Horas Presenciales	Trabajo Autónomo	Volumen de trabajo
Teoría Temas 1 y 2	Lección magistral	4	6	10
Ejercicios Temas 1 y 2	Resolución de ejercicios teórico-prácticos	10	20	30
Práctica 1	Examen práctico	1.5	0	1.5
Práctica 2	Examen práctico	1.5	0	1.5
Evaluación Teoría Temas 1 y 2	Examen escrito	2	0	2
Teoría Tema 3	Lección magistral	4	6	10
Ejercicios Tema 3	Resolución de ejercicios teórico-prácticos	10	20	30
Práctica 3	Examen práctico	1.5	0	1.5
Práctica 4	Examen práctico	1.5	0	1.5
Teoría Tema 4	Lección magistral	4	6	10
Ejercicios Tema 4	Resolución de ejercicios teórico-prácticos	10	20	30
Evaluación Teoría Temas 3 y 4	Examen escrito	2	0	2
Evaluación Final	Examen teórico-práctico	2	0	2
Tutoría	Tutorías individualizadas	6	12	18
	Totales	60	90	150

Cada curso consta de 14 semanas completas y en la FIUM se ha decidido que siempre se imparte la docencia al completo, de modo que si en una semana hay algún día festivo, las clases de dicho día se recuperan, manteniendo el horario de dicho día, el viernes más cercano -podría ser con antelación o el viernes de esa misma semana-, razón por la cual los horarios de la FIUM se concentran -excepto en dichos viernes excepcionales, en los que también se pueden programar exámenes- de lunes a jueves. Así, podemos pensar que siempre disponemos de 14 semanas completas para distribuir la docencia de nuestra asignatura. A la vista de este hecho, y de la tabla mostrada más arriba, se propone dedicar un total de:

- 4 semanas completas a los Temas 1 y 2.
- 5 semanas completas al Tema 3.

- 5 semanas completas al Tema 4.

Se debe tener en cuenta que cada alumno recibe cada semana 2 horas de teoría y 1 hora y 40 minutos de prácticas. Los problemas se explican tanto en las clases de teoría -como ejemplos de los resultados que se introducen- como en las prácticas, donde se usa pizarra y se dispone, además, del uso de ordenadores. El software que utilizamos es SAGE

Esta página fue dejada intencionalmente blanca

Bibliografía

[1] H. AIKEN ET. AL. Perspectivas de la revolución de los computadores, Madrid, Alianza Ed., 1975.

[2] J. M. ALMIRA, Norbert Wiener. Un matemático entre ingenieros, Madrid, Nivola, 2009.

[3] J. M. ALMIRA, Fourier. Un matemático al servicio de la física, Barcelona, RBA, 2017.

[4] J. M. ALMIRA, M. AGUILAR-DOMINGO, Neuromatemáticas. El lenguaje eléctrico del cerebro., Madrid, CSIC y Libros de la Catarata, 2016.

[5] T. M. APOSTOL, Análisis matemático, Barcelona, Ed. Reverté, 1977.

[6] W. ASPRAY, John von Neumann y los orígenes de la computación moderna, Barcelona, Gedisa, 1993.

[7] G. V. BARD, Sage for undergraduates. American Mathematical Society, 2014.

[8] N.L. BIGGS, Matemática discreta, Madrid, Vicens-Vives, 1994.

[9] PH. BRETON, Historia y crítica de la informática, Madrid, Cátedra, 1989.

[10] SH. BREUER, G. ZWAS, Numerical mathematics. A laboratory approach, New York, Cambridge University Press, 1993.

[11] R.L. BURDEN, J.D. FAIRES, Numerical Analysis, PWS-Kent, Boston, 1989.

[12] A CASAMAYOU ET. AL. Calcul mathématique avec SAGE, CreateSpace, 2013.

[13] C.A. COELLO COELLO Breve historia de la computación y sus pioneros, México, Fondo de Cultura Económica, 2003.

[14] J. COPELAND, Inteligencia artificial, Alianza Universidad, 1996.

[15] R. COURANT, H. ROBBINS, ¿Qué son las matemáticas? Conceptos y métodos fundamentales, México, Fondo de Cultura Económica, 2002.

[16] G. DAHLQUIST, A. BJORCK, Numerical Methods, Prentice-Hall, 1974.

[17] M. DAVIS, La computadora universal. De Leibniz a Turing, Madrid, Debate, 2002

[18] PH. J. DAVIS, Interpolation and Approximation, Dover, 1975.

[19] PH. J. DAVIS, P. RABINOWITZ, Methods of Numerical Integration, Computer Science and Applied Mathematics, Academic-Press, 1984.

[20] G. C. DONOVAN, A. R. MILLER, T. J. MORELAND, Phatological Functions for Newton's Method, Amer. Math. Monthly, January (1993) 53–58.

[21] G. DYSON, La catedral de Turing. Los orígenes del universo digital, Madrid, Debate, 2015.

[22] D. FERNÁNDEZ FERREYRA, E. A. PILOTTA, El método de Newton y estrategias de globalización para la resolución numérica de ecuaciones no lineales, Revista de Educación Matemática 19 (1) (2004) 3–19.

[23] R. W. FLOYD, Assigning Meanings to Programs, En J. T. SCHWARTZ, Mathematical Aspects of Computer Science, Proceedings of Symposium on Applied Mathematics. 19. American Mathematical Society. pp. 19–32, 1967.

[24] J. GLENN BROOKSHEAR, Teoría de la computación, Wilmington, Addison-Wesley Iberoamericana, 1993.

[25] J. GLEICK, La Información: Historia y realidad, Barcelona, Ed. Crítica, 2012.

[26] L. GOLDSCHLAGER, A. LISTER, Introducción a la ciencia de la computación, México, Prentice-Hall, 1986.

[27] R.L. GRAHAM, D. E. KNUTH, O. PATASHNIK, Concrete mathematics (2th Edition), New York, Addison-Wesley Pu. Co., 1994.

[28] L. GUALÀ, S. LEUCCI, E. NATALE, R. TAURASO, Large Peg-Army Maneuvers, Proc. of the 8th International Conference on Fun with Algorithms (FUN 2016) Vol. 49 of Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 18:1-18:15, 2016.

[29] M. DE GUZMÁN, El rincón de la pizarra. Ensayos de visualización en Análisis Matemático, Pirámide, 1996.

[30] E. HAIRER, G. WANNER, Analysis by its history, Undergraduate Texts in Mathematics, Springer, 2000.

[31] J. HAVIL, Nonplussed! Mathematical proofs of implausible ideas, Princeton University Press, 2007.

[32] R. HONSBERGER, Mathematical Gems II, Dolciani Math. Expositions 2 Mathematical Association of America, 1976.

[33] E. ISAACSON, H. B. KELLER, Analysis of Numerical Methods, Wiley, 1966.

[34] O. KNILL, Some fundamental theorems in mathematics, arXiv:1807.08416, 2018.

[35] D. E. KNUTH, El arte de programar ordenadores. Algoritmos Fundamentales, Barcelona, Reverté, 2002.

[36] D. E. KNUTH, El arte de programar ordenadores. Clasificación y búsqueda, Barcelona, Reverte, 2002.

[37] V. I. KRILOV, Aproximate Calculation of Integrals, Mac Millan, 1962.

[38] E. LEHMAN, T. LEIGHTON, A. R. MEYER Mathematics for computer science, 12th Media Services, 2017.

[39] E. LINÉS ESCARDÓ, Principios de Análisis Matemático, Barcelona, Ed. Reverte, 2009.

[40] N. MARTÍ OLIET, C. SEGURA, A. VERDEJO, Algoritmos correctos y eficientes. Diseño razonado ilustrado con ejercicios, Garceta, 2012.

[41] A. MARTÍNEZ-FINKELSHTEIN, J. J. MORENO-BALCÁZAR, Métodos Numéricos: Aproximación en $\mathbb{R}$, Servicio de Publicaciones de la Universidad de Almería, 1999.

[42] X. MOLERO PRIETO (COORDINADOR), Un viaje a la historia de la informática, Valencia, Ed. Universidad Politécnica de Valencia, 2016.

[43] M. MURESAN, A concrete approach to Classical Analysis, New York, Springer, 2009.

[44] I. P. NATANSON, Constructive function theory, vols: 1,2,3., Ungar, New York, 1964-65.

[45] R. PEÑA MARI, De Euclides a Java. Historia de los algoritmos y de los lenguajes de programación, Madrid, Nivola, 2006.

[46] J. R. PIERCE, A. M. NOLL, Señales. La ciencia de las telecomunicaciones. Barcelona, Reverte, 1995.

[47] W.H. PRESS, S.A. TAUKOLSKY, W.T. VETTERLING, B.P. FLANNERY, Numerical Recipes in C. The art of Scientific Computing, Cambridge University Press, 1995.

[48] M. SPIVAK, Calculus 3th Edición, Barcelona, Ed. Reverte, 2012.

[49] W. SQUIRE, Integration for Engineers and Scientists, American Elsevier, 1970.

[50] J. SZABADOS, P. VÉRTESI, Interpolation of functions, World Scientific, 1990.

[51] V. TORRA, Del ábaco a la revolución digital. Algoritmos y computación, Barcelona, RBA, 2010.