materia

Cálculo numérico nunha variable

unidade didáctica 1

Introdución á análise numérica. Erros no cálculo numérico

Carmen Rodríguez Iglesias

Departamento de Matemática Aplicada
Facultade de Matemáticas

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA
VICERREITORÍA DE ESTUDANTES, CULTURA E FORMACIÓN CONTINUA

titulación
Grao en Matemáticas

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

unidade **didáctica 1**

Introdución á análise numérica. Erros no cálculo numérico

Carmen Rodríguez Iglesias

Departamento de Matemática Aplicada
Facultade de Matemáticas

unidadedidáctica
UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Deseño
Unidixital
Servizo de Edición Dixital
da Universidade de Santiago de Compostea

Edita
Vicerreitoría de Estudantes,
Cultura e Formación Continua
da Universidade de Santiago de Compostela
Servizo de Publicacións
da Universidade de Santiago de Compostela

Imprime
Unidixital

Dep. Legal: C 51 - 2013
ISBN 978-84-9887-962-9

MATERIA: Cálculo Numérico nunha Variable
TITULACIÓN: Grao en Matemáticas
PROGRAMA XERAL DO CURSO
Localización da presente unidade didáctica

Unidade I. Introdución á Análise Numérica. Erros no cálculo numérico.
- Marco da materia.
- Erro absoluto e relativo.
- Sistema de numeración decimal.
- Sistema binario. Erros.
- Conceptos básicos. Test de parada.

Unidade II. Aproximación de raíces dunha ecuación numérica.
- Separación de raíces.
- Concepto de método iterativo.
- Orde de converxencia.
- Converxencia local e global.
- Método de dicotomía.
- Método de iteración funcional.
- Método de Newton-Raphson.

Unidade III. Interpolación polinómica de Lagrange.
- Fórmula de Lagrange.
- Fórmula do erro de Cauchy-Peano.

Unidade IV. Introdución á integración numérica.
- Regras do trapecio e Simpson simples.
- Regras do trapecio e Simpson compostas.
- Fórmulas do erro.

Unidade V. Introdución á derivación numérica.

ÍNDICE

PRESENTACIÓN

A **Matemática Aplicada** caracterízase pola interpretación e creación de teoría matemática que posibilite a resolución de problemas.

A abordaxe dende un punto de vista matemático dun problema real de interese esixe: o establecemento dunha formulación matemática do problema, a análise e a resolución de dita formulación e a interpretación da solución obtida. Estas tres etapas (formulación, análise máis resolución e interpretación de resultados) son os piares básicos do procedemento coñecido como modelado matemático. Este proceso, imprescindible na aplicación das matemáticas, constitúe o campo de acción da *Matemática Aplicada*.
Na etapa de resolución ocorre que os modelos reais son en xeral o suficientemente complicados como para que non se poida obter unha fórmula explícita que relacione as incógnitas cos datos do problema. É aquí onde intervén a **Análise Numérica**, disciplina das matemáticas que se ocupa da descrición e análise de métodos, que se van implantar nunha máquina de cálculo, para a obtención da solución aproximada (ou exacta ocasionalmente) dun problema matemático. Estes métodos son camiños que se constrúen botando man de calquera posible ferramenta matemática que poida valer. É por isto que a *Análise Numérica* aliméntase das propias matemáticas nas que vive.

Cálculo numérico nunha variable é unha materia con contidos da disciplina *Análise Numérica*. Neste sentido constitúe, xunto coas materias cronoloxicamente posteriores *Análise numérica matricial*, *Métodos numéricos en optimización e ecuacións diferenciais*, *Modelización matemática* (as tres obrigatorias), *Análise numérica de ecuacións en derivadas parciais* e *Taller de problemas industriais* (ámbalas dúas optativas), un elo básico na formación dun matemático aplicado, formación que o alumnado poderá incrementar e perfeccionar coa realización do *Máster universitario en enxeñaría matemática*.

Calquera **método** dos estudados no campo da *Análise Numérica*, e en particular dos estudados en *Cálculo numérico nunha variable*, está elaborado **para ser aplicado**. A súa aplicación contén unha fase final que constitúe un conxunto de certas operacións aritméticas e lóxicas nunha orde determinada e sen ambigüidade que é preciso executar nunha **máquina de cálculo**, motivo polo que esta ferramenta resulta imprescindible. Para realizar esta tarefa, precísanse ter adquiridos certos coñecementos básicos de informática, entre eles o inicio na aprendizaxe dunha linguaxe de programación (formación proporcionada pola materia *Informática* do primeiro curso do *Grao en Matemáticas*).

OS OBXECTIVOS

Dado que a unidade que estamos a desenvolver é a primeira que se imparte no Grao en Matemáticas correspondente á disciplina *Análise Numérica*, e dada a necesidade que esta disciplina ten dunha máquina de cálculo, os obxectivos que se perseguen nesta unidade son:

— Introducirse no coñecemento e manexo de conceptos e técnicas básicas da *Análise Numérica*.

— Desenvolverse con comodidade no manexo dunha linguaxe de programación sobre o ordenador e coñecer certos problemas que derivan do uso desta ferramenta.

En definitiva, trátase de que o alumno se prepare para poder asimilar as unidades posteriores.

OS CONTIDOS BÁSICOS

Comezaremos enmarcando a materia *Cálculo Numérico nunha variable*, da que forma parte a unidade que estamos a tratar, dentro da disciplina de matemáticas á que pertence, a *Análise Numérica*. Finalmente, desenvolveremos os contidos da unidade.

1. Marco da materia Cálculo Numérico nunha variable

Realizaremos unha breve introdución á *Análise Numérica* motivando a súa existencia, onde atoparemos de modo natural a materia que nos ocupa.

1.1. A disciplina Análise Numérica

A *Análise Numérica* trata o deseño e a análise de métodos, que se van executar nunha máquina de cálculo, para aproximar dunha maneira eficiente a solución dun problema expresado matematicamente. A forma práctica na que un método é introducido nun ordenador é chamada por algúns autores *algoritmo*, aínda que normalmente ámbolos dous termos **método numérico** e **algoritmo** son utilizados indistintamente.

As ideas da *Análise Numérica* remóntanse a moitos séculos atrás. Atribúese a Herón de Alexandría, na segunda metade do primeiro século a. C., o cálculo mediante aproximacións sucesivas da raíz cadrada dun número positivo c, é dicir, se x é unha aproximación, defínese a seguinte como $\frac{1}{2}(x + \frac{c}{x})$, o que constitúe (con notación actual) o algoritmo:

$$\begin{cases} x_0 \text{ número positivo dado,} \\ x_{n+1} = \frac{1}{2}(x_n + \frac{c}{x_n}) \, , n = 0, 1, \ldots \end{cases}$$

Isto significa que tratamos de nos aproximar cada vez máis ao valor $\sqrt{c}$ construíndo unha sucesión de números $\{x_n\}_{n\geq 0}$, que chamaremos **iterantes**, mediante as operacións elementais suma, multiplicación e división. No momento en que decidamos parar esa construción, o último elemento calculado é o que se vai tomar como aproximación do número $\sqrt{c}$.

Non querendo facer un percorrido histórico, citamos por exemplo a Karl Friedrich Gauss (1777-1855), cuxas achegas á resolución de sistemas de ecuacións lineares seguen sendo útiles na actualidade.

Merece mención especial Leonhard Euler (1707-1783), cuxo extenso traballo contén un gran número de algoritmos e fórmulas que aparecen normalmente como desenvolvementos en serie. Con Euler acadouse un punto culminante na utilización e desenvolvemento de algoritmos.

A partir do século XVIII, os coñecementos matemáticos fóronse perfeccionando, pero os razoamentos construtivos eran sempre superados por outros de tipo lóxico cuxo interese era máis determinar se existe solución a un determinado problema que calcula-la de forma efectiva. O motivo deste proceso de abstracción era que os algoritmos para o cálculo das solucións dos problemas resultaban, aínda que finitos, irrealizables debido á cantidade de cálculo que esixían. A aparición das computadoras na segunda metade do século XX supuxo o rexurdimento dos métodos construtivos.

Como verdadeira disciplina, a *Análise numérica* comezou a súa andaina cando en 1947 se creou o Instituto de Análise Numérica na Universidade de California, e desenvolveuse espectacularmente nestas últimas décadas motivado polo desenvolvemento informático.

Poderíase dicir que, se dende a antigüidade ata 1945 a velocidade de cálculo se multiplicara por 10 mediante artefactos rudimentarios (como o ábaco), dende entón ata agora multiplicouse por máis dun millón. Isto non significa que tódolos algoritmos poidan ser tratados por un ordenador, pois algúns non é posible aínda levalos a cabo cos ordenadores actuais. O progreso tecnolóxico é o que permite recuperar e refinar métodos baseados en ideas antigas e deseñar métodos novos.

Desde Bacon e Galileo no século XVII, o desenvolvemento das ciencias naturais enriqueceuse ao combinar as metodoloxías experimentais coas teóricas, estas últimas moi relacionadas coas matemáticas. A partir do desenvolvemento das computadoras, xorde a *simulación numérica*, cuxo obxectivo é a recreación matemática dun proceso. Funciona como un laboratorio cuxa ferramenta principal é a computadora e onde se pode deseñar un experimento que proporciona un mellor entendemento do proceso ou fenómeno que se estuda, o que permite predicir as consecuencias da súa execuxión e, polo tanto, a toma de decisións ao respecto, é dicir, o seu control. Esta metodoloxía pódese considerar parte da *Análise Numérica*, e está estreitamente relacionada con varias áreas da computación e doutras disciplinas. Así fálase de física, química, bioloxía e ata finanzas computacionais que se ocupan do estudo de modelos matemáticos e algoritmos computacionais eficientes para a resolución de problemas que xorden nestas ramas da ciencia.

1.2. Por que esta disciplina

A pesar de todo o dito ata o momento, concretamos a continuación algúns dos principais motivos polos que a *Análise Numérica* se fai unha disciplina imprescindible.

1. Hai problemas que se sabe que teñen solución, pero non dispoñemos dunha fórmula que exprese dita solución. Así, por exemplo, aínda sabendo que un polinomio de grao n ten exactamente n raíces (non necesariamente diferentes), está probado (Galois, 1811-1832) que cando $n \geq 5$ non é posible *calcula-las por radicais* (é dicir, empregando unicamente as catro operacións elementais e os radicais, isto é, raíces cadradas, cúbicas, etc.)
2. Hai problemas con solución exacta coñecida pero esta non é explotable na práctica. Por exemplo, a integral $\int_a^b \mathrm{e}^{x^2}\,dx$ podemos pensar en calculala utilizando a regra de Barrow. Unha primitiva do integrando é $F(x) = \sum_{k=0}^{\infty} \frac{x^{2k+1}}{(2k+1)!k!}$ que, ao ser absolutamante converxente para calquera valor de x, permite reducir o cálculo da integral ao problema *non trivial* dunha suma infinita.
3. Hai problemas para os que a aplicación dos métodos clásicos é irrealizable. Isto ocorre por exemplo co método de Cramer: é ben coñecido que a resolución dun sistema linear de n ecuacións con n incógnitas, con matriz non singular, require da orde de $(n+1)!n$ operacións, o que, incluso cos potentes ordenadores de hoxe en día, é impracticable. Un ordenador rápido coma o IBM Blue Gene, que fai 70.72×10^{12} operacións por segundo (70.72 TFLOPS), necesitaría 4000 horas para resolver un sistema de orde 20. É entón absolutamente necesario deseñar novos métodos.
4. Hai problemas dos que se sabe algo e, por suposto, para os que non se dispón de solución analitica coñecida, algúns deles moi complexos, pero de moito interese social en resolver. Así, por exemplo, o tempo meteorolóxico sobre a superficie terrestre está gobernado por ecuacións matemáticas complicadas que non se poden resolver analiticamente. Dado que interesa predicilo, o que se fai é aproximar ese modelo matemático por un modelo que xa é posible resolver, é dicir, para o que xa é posible aproximar a súa solución.

O desenvolvemento espectacular das computadoras trouxo consigo unha demanda crecente de métodos numéricos máis potentes para resolver problemas cada vez máis complexos que abarcan campos diferentes como física, química, enxeñaría, ciencias da saúde, tecnoloxía espacial, etc. Entre eles están a predición do clima, o estudo das proteínas, o desenvolvemento de tumores cancerosos, o control de voo dunha nave espacial, etc. Todos estes problemas esixen resultados cuantitativos que permitan tomar decisións, algunhas a moi curto prazo, o que implica ter que dispoñer de métodos eficientes que den resultados coa precisión desexada no menor tempo posible. Ten interese especial a simulación numérica de sistemas cuxa simulación experimental non se pode levar a cabo pola perigosidade que encerra, como ocorre cos sistemas de control das centrais nucleares.

A pesar de que a velocidade de cálculo aumentou espectacularmente nestes últimos anos, aínda hai algoritmos que non poden ser abordados cos ordenadores actuais. A elección, entre métodos xa existentes, ou a construción dun método novo apropiado para aproxima-la solución dun problema está intimanente ligada aos cambios tecnolóxicos do computador. A *Análise Numérica*, da man do progreso computacional, constitúe por tanto unha disciplina imprescidible do coñecemento.

1.3. A nosa materia dentro desta disciplina

Ao longo da titulación, e dentro das materias impartidas polo Departamento de Matemática Aplicada, estudaranse algúns dos diferentes métodos que hai para atacar diferentes tipos de problemas.

Nesta materia (*Cálculo numérico nunha variable*) veremos algúns métodos elementais da *Análise Numérica*. Concretamente:

- Métodos para resolver ecuacións numéricas non lineares: trátase de calcula-los ceros dunha función real de variable real.
- Métodos para resolver problemas de interpolación polinómica: trátase de sustituír unha función pouco manexable, ou da que só se coñecen algúns valores, por outra máis sinxela e que cumpre certas condicións de coincidencia coa primeira. Dita función máis sinxela utilizarase en vez da primeira para estimar valores desta.
- Métodos de integración e derivación numérica: trátase de aproximar integrais e derivadas de funcións.

Asemade, o estudo destes métodos permitiranos coñecer e familiarizarnos con certos conceptos que se manexan continuamente nesta disciplina.
En Quarteroni e Saleri (2006) pódense ver as formulacións matemáticas xenéricas dos tipos de problemas que se pretenden resolver, xunto con exemplos físicos de interese que se modelan de tal xeito.

En materias posteriores estúdanse, por exemplo, métodos para resolver sistemas lineares, métodos para resolver sistemas non lineares, métodos para aproxima-los autovalores e autovectores dunha matriz, métodos para aproxima-la solución dunha ecuación diferencial ordinaria, etc.

Recordemos que estes métodos deben ser levados a cabo na práctica, e que calquera deles redúcese finalmente a un conxunto ordenado das diferentes operacións aritméticas e lóxicas, operacións que deberán ser realizadas necesariamente por un ordenador.

No anteriormente exposto, aparecen tres elementos básicos:

- problema matemático que queremos resolver (e, polo tanto, elemento fixo);
- método a utilizar (elemento variable, a escoller entre varios posibles);
- ordenador no que se van face-los cálculos (elemento normalmente fixo),

sendo o obxectivo final obter unha **boa aproximación** (xa veremos máis adiante o significado disto) da solución do problema no menor tempo posible.
Para manexarse ben con estes elementos, é moi importante ter unha boa base

matemática e saber utilizar algunha linguaxe de programación que nos permita comunicarnos co ordenador e coñecer certos aspectos que derivan do uso deste.

2. Erro absoluto e relativo

Definición.- Se un número real x se aproxima por outro número real $\widetilde{x}$, chámase **erro absoluto** á cantidade $|x - \widetilde{x}|$.

Definición.- Se un número real x se aproxima por outro número real $\widetilde{x}$, chámase **erro relativo** á cantidade

$$\left|\frac{x - \widetilde{x}}{x}\right|, \quad \text{se} \quad x \neq 0.$$

Observar que o erro absoluto ten unidades, pero o erro relativo non as ten, é adimensional. Observar tamén que, se $x = 1$, ámbolos dous erros coinciden.

Exemplo 1.-Sexan $x = 2$ kg e $\widetilde{x} = 1.9$ kg. Tense,
$|x - \widetilde{x}| = 0.1$, e $\left|\frac{x-\widetilde{x}}{x}\right| = 0.05$.

Exemplo 2.-Sexan $y = 2000$ g e $\widetilde{y} = 1900$ g. Tense,
$|y - \widetilde{y}| = 100$, e $\left|\frac{y-\widetilde{y}}{y}\right| = 0.05$.

Exemplo 3.-Sexan $z = 8000$ g e $\widetilde{z} = 7900$ g. Tense,
$|z - \widetilde{z}| = 100$, e $\left|\frac{z-\widetilde{z}}{z}\right| = 0.0125$.

Nos exemplos anteriores x e y expresan o mesmo, só que en magnitudes diferentes. O mesmo ocorre con $\widetilde{x}$ e $\widetilde{y}$. Polo tanto, cométese o mesmo erro relativo ao aproximar x por $\widetilde{x}$ que ao aproximar y por $\widetilde{y}$. Non obstante, o erro absoluto que se comete non é o mesmo.
Entre z e $\widetilde{z}$ hai unha diferenza de 100 g, a mesma que entre y e $\widetilde{y}$; de feito, os erros absolutos de ámbalas dúas aproximacións coinciden. Non obstante, todos apreciamos que non é o mesmo equivocarse en 100 g nun peso de 2 kg que nun peso de de 8 kg. Efectivamente, o erro relativo da aproximación de z, 0.0125, é menor que o da aproximación de y, 0.05.

Observar que o erro absoluto non dá unha idea da importancia do erro cometido frente ao valor da magnitud medida.

Pode parecer, segundo o exposto anteriormente, que o erro absoluto nos pode levar a engano e que o erro relativo é fiel informador. Non obstante, hai casos nos que a magnitude dos números xoga un papel determinante e non nos podemos fiar só do erro relativo. Como exemplo ilustrativo, o mencionado en Viaño Rey (1995) do cohete que se debe pousar na lúa despois de ter percorrido os, aproximadamente, 350000 km que a separan da terra. O punto no que debe alunizar o cohete é fixado de antemán, podéndose tolerar un erro absoluto dunhas decenas de metros respecto ao punto prefixado. Poderíanos tranquilizar deixar cometer un erro relativo de

$\frac{2}{350000} \approx 5.71 \times 10^{-6}$; non obstante, en tal caso, teríamos $\left|\frac{x-\widetilde{x}}{x}\right| = \frac{2}{|x|}$, con $x = 350000$ km; polo tanto, $|x - \widetilde{x}| = 2$, é dicir, estaríamos cometendo un erro absoluto de 2 km, que pode resultar fatal para o obxectivo da viaxe por mor da existencia de cráteres, montañas, etc. (Engadir simplemente que, por exemplo, un erro absoluto duns 30 m significa un erro relativo $\left|\frac{x-\widetilde{x}}{x}\right| = \frac{0.030}{|x|} = \frac{0.030}{350000} \approx 8.57 \times 10^{-8}$, é dicir, habería que ser máis esixentes).
Noutras situacións, un erro relativo como o considerado no exemplo co cohete pode non ter importancia: non nos imos sorprender porque nun pedido de 350000 folios nos envíen 2 folios menos.

3. Sistema de numeración en base 10 (decimal)

O sistema de numeración que usamos normalmente é o decimal. Neste sistema a representación dos números está constituída por sucesións dos dez símbolos «$0, 1, 2, 3, 4, 5, 6, 7, 8, 9$». Tales sucesións poden estar precedidas polos símbolos «$+$» ou «$-$», para indicar se é positivo ou negativo, e poden ter o «.» (punto raíz), que separa a parte enteira do número (á esquerda), da súa parte decimal (á dereita).
O sistema decimal é un sistema posicional, é dicir, a posición ocupada por cada díxito ten un significado exacto e determina a contribución de dita cifra ao valor numérico da sucesión. Así, 26 e 62 están construídos polas mesmas cifras pero teñen distintos significados.
Cada cifra da sucesión é multiplicada por unha potencia de 10 cuxo expoñente está determinado pola posición da cifra con respecto ao punto raíz. O valor 10 é a base do noso sistema de numeración, que por esta razón se denomina *sistema posicional en base* 10. Os expoñentes da parte enteira do número son positivos e medran en unidades a partir do cero; os expoñentes da parte fraccionaria son negativos e diminúen en unidades a partir de -1. Así por exemplo, $678.234 = 6 \times 10^2 + 7 \times 10^1 + 8 \times 10^0 + 2 \times 10^{-1} + 3 \times 10^{-2} + 4 \times 10^{-3}$ (678 é a parte enteira e $.234$ é a parte fraccionaria).
Un exemplo de sistema non posicional constitúeo a numeración romana. Nela, ao valor 5 correspóndelle o símbolo V, mentres que ao valor 50 correspóndelle o símbolo L. Para pasar de 5 a 50 non basta cambiar a posición do símbolo do 5 (V), senón que hai que introducir un símbolo novo (L).
A descrición feita do sistema posicional decimal suxire a posibilidade de usar un sistema de numeración nunha base distinta de 10. Co sistema en base 10 úsanse 10 cifras para representar cada número; en xeral, para representar un número nunha base b necesítanse b símbolos. Así, cando se considera como base un enteiro $b > 10$ non bastan as 10 cifras do sistema decimal, e é preciso usar símbolos novos.
As bases máis usadas, ademais da 10, son 2, 8 e 16. O sistema en base 2, chamado *binario*, usa as cifras «$0, 1$». O sistema en base 8, *sistema octal*, usa as cifras «$0, 1, 2, 3, 4, 5, 6, 7$». O sistema en base 16, *sistema hexadecimal*, usa «$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F$».

Cada sistema de numeración ten a súa aritmética e as súas regras de uso.

En computación científica utilízase normalmente a base 2. Cada un dos caracteres «0» e «1», chamadas cifras binarias ou *bits* (do inglés *binary digits*) é representado fisicamente no ordenador por un dos dous posibles estados dos compoñentes[1] de do ordenador.

O ordenador recibe normalmente a información en sistema decimal, que é transformada en binario por un programa interno. Posteriormente, efectúa as operacións pertinentes, pasa o resultado a decimal e informa ao usuario dese resultado. Así, en principio, o usuario non tería por que coñecer necesariamente a representación dos datos na máquina. Non obstante, para a solución de moitos problemas é conveniente que o usuario coñeza o sistema binario e os principios fundamentais da aritmética dun ordenador, aínda que nunca vaia contar en aritmética binaria. Isto será o que trataremos de facer a continuación, pero, para facilitar a súa comprensión, referirémonos inicialmente á representación decimal.

3.1. Representación decimal normalizada (ou representación de números en base 10 en punto flotante)

Todo número real x pódese escribir en forma decimal

$$x = \pm(a_q a_{q-1} \dots a_1 a_0 . b_1 b_2 \dots b_p \dots) = a_q \times 10^q + a_{q-1} \times 10^{q-1} + \dots + a_1 \times 10^1 + a_0 \times 10^0 + b_1 \times 10^{-1} + b_2 \times 10^{-2} + \dots + b_p \times 10^{-p} + \dots$$

Así, por exemplo,
$456.123 = 4 \times 10^2 + 5 \times 10^1 + 6 \times 10^0 + 1 \times 10^{-1} + 2 \times 10^{-2} + 3 \times 10^{-3}$.

A parte do número á esquerda do punto (punto decimal) dise *parte enteira*, e as súas cifras dinse cifras da parte enteira. A parte do número á dereita do punto dise *parte decimal*, e as súas cifras dinse cifras da parte decimal ou cifras decimais. Chámanse **cifras significativas** as cifras que están a partir da primeira distinta de cero; así, por exemplo, no número anterior 456.123 tódalas cifras son significativas; en cambio, as cifras significativas de 0.00567 son só 567.

Multiplicando pola potencia de 10 adecuada, dedúcese que todo número real admite unha expresión da forma

$$x = \pm(0.d_1 d_2 \dots d_p d_{p+1} \dots) \times 10^n = \pm(\textstyle\sum_{k=1}^{\infty} d_k \times 10^{-k}) \times 10^n,$$

onde os díxitos d_k cumpren: $1 \leq d_1 \leq 9$ e $0 \leq d_i \leq 9, \quad i = 2, 3, 4 \dots$

(Basta axustar o expoñente n para que o punto decimal se sitúe antes da primeira cifra decimal non nula.)

[1] Dispositivos de rexistro magnético e electrónico. Cada un destes dispositivos só pode estar magnetizado nun sentido, o que representa o «0», ou o que representa o «1».

Esta forma chámase **expresión decimal normalizada**, **expresión decimal en punto flotante**[2] (ou tamén expresión decimal en coma flotante) ou **notación científica normalizada**.

O número real $R = 0.d_1d_2 \ldots d_k \ldots = d_1 \times 10^{-1} + d_2 \times 10^{-2} + \ldots + d_k \times 10^{-k} + \ldots$ cumpre $\frac{1}{10} \leq R \leq 1$, e chámase **mantisa do número**. Á potencia n chámase **expoñente**.
Por convenio, o cero represéntase da forma $+0.00 \ldots 0 \times 10^0$.

Así, por exemplo, $456.123 = +0.456123 \times 10^3$, e $0.00567 = +0.567 \times 10^{-2}$.

Baseados nos conceptos de erro absoluto e erro relativo, respectivamente, defínense a continuación dous conceptos importantes relacionados coa aproximación de números.

3.2. Aproximación con s cifras decimais exactas

Definición.- Un número real $\widetilde{x}$ dise unha **aproximación** de x **con** s **cifras decimais exactas** (ou que x e $\widetilde{x}$ **coinciden en** s **cifras decimais**) se s é o maior enteiro non negativo tal que

$$|x - \widetilde{x}| \leq \tfrac{1}{2} \times 10^{-s} \quad (= 0.5 \times 10^{-s} = 5 \times 10^{-(s+1)} = 5 \times 10^{-s-1}),$$

ou, de forma equivalente, se $0.5 \times 10^{-(s+1)} \leq |x - \widetilde{x}| \leq 0.5 \times 10^{-s}$.

Exemplo1.- 1.27450 e 1.27431 coinciden en 3 cifras decimais (o que coincide coa idea usual de coincidencia): $|1.27450 - 1.27431| = 0.00019 \leq 0.0005 = 0.5 \times 10^{-3}$. (Ollo!, notar que, efectivamente, $s = 3$ é o maior número enteiro que cumpre o requirido pola definición; de feito, $0.5 \times 10^{-4} = 0.00005 \leq 0.00019$).
Exemplo2.- 1.27431 e 1.27382 coinciden tamén en 3 cifras decimais: $|1.27431 - 1.27382| = 0.00049 \leq 0.0005 = 0.5 \times 10^{-3}$.
Exemplo3.- 127.431 e 127.382 coinciden en 1 cifras decimal: $|127.431 - 127.382| = 0.049 \leq 0.05 = 0.5 \times 10^{-1}$.

Observar como 1.27431 e 1.27382 parecen estar máis próximos que $127.431 = 1.27431 \times 10^2$ e $127.382 = 1.27382 \times 10^2$. Isto é debido ao concepto de erro absoluto no que está baseado o criterio de aproximación de coincidencia en cifras decimais, e no que a magnitude dos números inflúe. O criterio de aproximación que definimos a continuación, baseado no concepto de erro relativo, corrixirá este efecto.

[2] Chámase así porque a posición do punto decimal non é fixa

3.3. Aproximación con s cifras significativas exactas

Definición.- Un número real $\widetilde{x}$ dise unha **aproximación** de x **con** s **cifras significativas exactas** (ou que x e $\widetilde{x}$ **coinciden en** s **cifras significativas**) se s é o maior enteiro non negativo tal que

$\left|\frac{x-\widetilde{x}}{x}\right| \leq \frac{1}{2} \times 10^{-(s-1)} (= 0.5 \times 10^{-(s-1)} = 5 \times 10^{-s})$,

ou, de forma equivalente, se $0.5 \times 10^{-s} \leq \left|\frac{x-\widetilde{x}}{x}\right| \leq 0.5 \times 10^{-(s-1)}$

Exemplo
1.27450 e 1.27431 coinciden en 4 cifras significativas (o que coincide coa idea usual de coincidencia): $\left|\frac{1.27450-1.27431}{1.27450}\right| = 0.000149\ldots \leq 0.0005 = 5 \times 10^{-4}$. (Ollo!, debemos ter en conta que, efectivamente, $s = 4$ é o maior número enteiro que cumpre o requirido pola definición; de feito, $0.00005 = 5 \times 10^{-5} \leq 0.000149\ldots$)
1.27431 e 1.27382 coinciden tamén en 4 cifras significativas: $\left|\frac{1.27431-1.27382}{1.27431}\right| = 0.000384\ldots \leq 5 \times 10^{-4}$.
127.431 e 127.382 coinciden en 4 cifras significativas: $\left|\frac{127.431-127.382}{127.431}\right| = 0.000384\ldots \leq 5 \times 10^{-4}$.
0.00127431 e 0.00127382 coinciden en 4 cifras significativas.

Observar como con este critero, 1.27431 e 1.27382 están igual de próximos que $127.431 = 1.27431 \times 10^2$ e $127.382 = 1.27382 \times 10^2$, e tamén igual de próximos que 0.00127431 e 0.00127382.

Nota.- Dacordo coas definicións anteriores, falarase da **coincidencia en polo menos** s **cifras decimais** ou en **polo menos** s **cifras significativas**, respectivamente, cando s é un número enteiro (non ten por que ser o maior) para o que se cumpre a desigualdade indicada na definición correspondente. Así, por exemplo, os números que coinciden con 124.26 en polo menos 4 cifras significativas son os x que cumpren $\left|\frac{1.27450-x}{1.27450}\right| \leq 5 \times 10^{-4}$. Non obstante, e dacordo coa definición, os números que coinciden con 124.26 en 4 cifras significativas son os x que cumpren $5 \times 10^{-5} \leq \left|\frac{1.27450-x}{1.27450}\right| \leq 5 \times 10^{-4}$.

Dado que para facer cálculos non se poden manexar infinitas ou demasiadas cifras, é absolutamente necesario traballar con aproximacións de números. As dúas maneiras clásicas de aproximación son as definidas a continuación.

3.4. Aproximación por redondeo

Dado $x = \pm(0.d_1 d_2 \ldots d_k d_{k+1} \ldots) \times 10^n$ en forma decimal en punto flotante, o número x^* que **aproxima a** x **por redondeo a** p **cifras decimais** defínese tendo en conta o valor do díxito $p+1$ como segue:

$$\left|\frac{x-x^*}{x}\right| = \begin{cases} \pm(0.d_1 d_2 \ldots d_p) \times 10^n & \text{se} \quad 0 \leq d_{p+1} \leq 4, \\ \pm(0.d_1 d_2 \ldots d_p + 10^{-p}) \times 10^n & \text{se} \quad 5 \leq d_{p+1} \leq 9. \end{cases}$$

Exemplo.- Redondeo a 4 cifras:
$x = -0.432713$ é aproximado por $x^* = -0.4327$.
$x = 0.99995$ é aproximado por $x^* = 0.1 \times 10^1 = 1$. (Observar que $0.99995 + 0.0001 = 1.00005$.)
$\pi = 0.31415926\ldots \times 10^1$ é aproximado por $\pi^* = 0.3142 \times 10^1$. (Observar que $0.31415926\ldots + 0.0001 = 0.31425926\ldots$)

3.5. Aproximación por truncamento

Dado $x = \pm(0.d_1d_2\ldots d_kd_{k+1}\ldots)\times 10^n$ en forma decimal en punto flotante, o número $\widehat{x}$ que **aproxima a x por truncamento a p cifras decimais**, defínese como o número que resulta simplemente de descartar en x tódolos díxitos ocupando as posicións maiores que p, é dicir,

$$\widehat{x} = \pm(0.d_1d_2\ldots d_p) \times 10^n.$$

Exemplo.- Truncamento a 4 cifras:
$x = -0.432713$ é aproximado por $\widehat{x} = -0.4327$.
$x = 0.99995$ é aproximado por $\widehat{x} = 0.9999$.
$\pi = 0.31415926\ldots \times 10^1$ é aproximado por $\widehat{\pi} = 0.3141 \times 10^1$.

3.6. Acotación do erro de redondeo e truncamento

Sendo $\widetilde{x}$ o número que aproxima a $x = \pm(0.d_1d_2\ldots d_kd_{k+1}\ldots) \times 10^n$ por redondeo ou por truncamento a p cifras decimais, pódese demostrar que se cumpren as seguintes acotacións (ver detalles en, por exemplo, Viaño Rey (1995)):

i)

$$|x - \widetilde{x}| \leq \begin{cases} (0.5 \times 10^{-p}) \times 10^n = 0.5 \times 10^{n-p} & \text{se se aproxima por redondeo,} \\ 10^{-p} \times 10^n = 10^{n-p} & \text{se se aproxima por truncamento.} \end{cases}$$

ii)

$$\left|\frac{x - \widetilde{x}}{x}\right| \leq \begin{cases} 5 \times 10^{-p} = 0.5 \times 10^{1-p} & \text{se se aproxima por redondeo,} \\ 10^{1-p} & \text{se se aproxima por truncamento.} \end{cases}$$

4. Sistema de numeración en base 2 (binario)

A representación dos números en base 2 introdúcese comodamente comparándoa coa base 10.

En primeiro lugar, pódese probar que todo número real x se pode escribir en forma binaria:

$x = \pm(a_qa_{q-1}\ldots a_1a_0.b_1b_2\ldots b_p\ldots) = a_q \times 2^q + a_{q-1} \times 2^{q-1} + \ldots + a_1 \times 2^1 + a_0 \times 2^0 + b_1 \times 2^{-1} + b_2 \times 2^{-2} + \ldots + b_p \times 2^{-p} + \ldots$

Así, por exemplo, o número real 14.75 (expresado en base 10), en base 2 ten a seguinte expresión:
$1110.11_{(2)} = 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$.

A proba disto, é dicir, como pasar de base 10 a base 2, pódese ver con detalle, por exemplo, en Viaño Rey (1995).

4.1. Representación binaria normalizada (ou representación de números en base 2 en punto flotante)

Se na expresión en forma binaria dun número x dada anteriormente, se multiplica pola potencia de 2 adecuada, dedúcese que todo número real admite unha **expresión normalizada**

$x = \pm(0.d_1 d_2 \dots d_p d_{p+1} \dots) \times 2^n = \pm(\sum_{k=1}^{\infty} d_k 2^{-k}) 2^n$,

onde os díxitos d_k cumpren: $d_1 = 1$ e $0 \leq d_i \leq 1, \quad i = 2, 3, 4 \dots$

Esta forma chámase tamén **expresión binaria en punto flotante**.

Así, por exemplo, $14.75_{(10)}$ ten a seguinte expresión normalizada en binario: $(0.111011) \times 2^4$. En efecto,
$1110.11_{(2)} = 1 \times 2^3 + 1 \times 2^2 + 6 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} = (1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + 0 \times 2^{-4} + 1 \times 2^{-5} + 1 \times 2^{-6}) \times 2^4 = (0.111011) \times 2^4$.

No sistema decimal dispoñemos de dez díxitos para expresar un número; no binario só de dous. Cantos menos díxitos teña o sistema, máis posicións se necesitan para expresar un mesmo número. Así, $14.75_{(10)}$ ocupa catro posicións (sen contar o signo e o punto), e $1110.11_{(2)}$ ocupa seis. Notar tamén que hai números que no sistema decimal constan dunha cantidade finita de díxitos e en binaro constan de infinitos díxitos; así, por exemplo, $0.1_{(10)} = (0.0\overline{0011})_{(2)}$.

4.2. Representación dos números nun ordenador

O sistema de numeración natural que utilizan os ordenadores é o binario. Nun ordenador, os bits (0 ou 1) agrúpanse en grupos de 8 bits, que se denominan bytes. Á súa vez, os bytes agrúpanse en palabras. Actualmente, os ordenadores utilizan en xeral palabras de 32 ou 64 bits (4 ou 8 bytes). Chámase lonxitude dunha palabra á cantidade de bits que a conforman.
En xeral, un ordenador típico dedica 32 posicións para gardar un número, podendo ser dita cantidade ampliable a 64.

A maneira en que se usan os bits para rexistrar os números enteiros e os fraccionarios varían en función do deseño do ordenador.

Para representar os números enteiros, utilízase normalmente unha palabra de 32 bits, un bit para o signo, e os trinta e un restantes para o valor do número. Isto significa que non podemos gardar os infinitos números enteiros que hai.

Recordar a forma binaria normalizada de expresión dos números reais

$$x = \pm(0.d_1 d_2 \dots d_p d_{p+1} \dots) \times 2^n = \pm(\sum_{k=1}^{\infty} d_k \times 2^{-k}) \times 2^n.$$

É claro que tampouco se van poder gardar todos no ordenador. Hai que limitar a cantidade de díxitos a un número finito p ($p \geq 1$) e tamén hai que limitar os valores que pode tomar o expoñente n. Pois ben, chámase **sistema de punto flotante** $F(2, p, m, M)$ ao conxunto finito de números da forma

$\pm(0.d_1 \dots d_p) \times 2^n, \quad d_1 = 1,\ 0 \leq d_i \leq 1, \quad i = 2, \dots p, \quad m \leq n \leq M,$

máis o número cero (que por convenio considérase $(0.0 \dots 0) \times 2^0$.)

O sistema cambiará ao cambiar calquera dos valores de p, m e M.

Un sistema desta forma é o que está gardado no ordenador, e o seu conxunto finito de números, chamados **números máquina**, serve para aproximar ao conxunto dos números reais.
Denotaremos por **Fl(x)** ao número máquina $F(2, p, m, M)$ que se utiliza para aproximar x. Aínda que os ordenadores utilizan formas moi variadas para elixir tal $\mathrm{Fl}(x)$, describiremos só as clásicas: redondeo e truncamento.
Aproximación por redondeo.- Consiste en tomar como aproximación de $x = \pm(0.d_1 d_2 \dots d_k d_{k+1} \dots) \times 2^n$ o número $\mathrm{Fl}(x)$ dado por:

$$\mathrm{Fl}(x) = \begin{cases} \pm(0.d_1 d_2 \dots d_p) \times 2^n & \text{se} \quad d_{p+1} = 0, \\ \pm(0.d_1 d_2 \dots d_p + 2^{-p}) \times 2^n & \text{se} \quad d_{p+1} = 1. \end{cases}$$

Aproximación por truncamento.- Dado $x = \pm(0.d_1 d_2 \dots d_k d_{k+1} \dots) \times 2^n$, o número máquina $F(2, p, m, M)$ que aproxima a x por truncamento, defínese como o número que resulta simplemente de descartar en x tódolos díxitos ocupando as posicións maiores que p, é dicir,

$$\mathrm{Fl}(x) = \pm(0.d_1 d_2 \dots d_p) \times 2^n.$$

Estas aproximacións pódense ver con máis detalle en Viaño Rey (1995).

O ordenador que utilizaremos nas prácticas permítenos traballar co sistema $\mathbf{F(2, 53, -1022, 1025)}$. A continuación expoñemos unha serie de detalles e conceptos que é preciso coñecer ao respecto.

1. O maior número enteiro representado é o número maior que en binario, xunto co seu signo, ocupa 32 posicións

$$+11111111111111111111111111111111_{(2)},$$

é dicir, é o número[3] $1 \times 2^{30} + 1 \times 2^{29} + \ldots + 1 \times 2^1 + 1 \times 2^0 = 1 + 2 + 2^2 + \ldots + 2^{30} = 2^{31} - 1 = 2147483647$. Na máquina, dado que ao signo positivo se lle asocia o 0, este número gárdase da forma

1 **bit (signo)**	**31 bits**
0	$1111111111111111111111111111111$

(O cero ocupando o bit 32, e logo 31 uns).

O cero enteiro gárdase na máquina en 32 posicións

1 **bit (signo)**	**31 bits**
0	$00000000000000000000000000000000$

O enteiro menor representado é -2^{31} (sería o $-(2^{31}-1)$, pero, grazas a certas técnicas de almacenaxe, o que en principio se tomaría como -0 pasa a ser o -2^{31}).

2. Os números reais do sistema de punto flotante $F(2, 53, -1022, 1025)$ son os números da forma:

   $\pm(0.d_1 \ldots d_{53}) \times 2^n$, $d_1 = 1$, $0 \leq d_i \leq 1, i = 2, \ldots 53$, $-1022 \leq n \leq 1025$, máis o número cero (que por convenio é $(0.0 \ldots 0) \times 2^0$).

   Recordar que esta forma de punto flotante, na que aparecen os díxitos a partir do primeiro distinto de cero, chámase forma normalizada.

   Estes números constitúen o conxunto finito de números que manexa a máquina establecida cando se traballa con 53 díxitos significativos (**dobre precisión**). Cara á súa representación na máquina **segundo a norma estándar IEEE**, exprésanse da forma:

   $\pm(1.t_1 \ldots t_{52}) \times 2^{\exp}, \quad 0 \leq t_i \leq 1, \quad i = 1, \ldots 52$, $\mathbf{-1023} \leq \exp \leq \mathbf{1024}$ máis o número cero.
   ($t_1 \ldots t_{52}$ son os díxitos da parte fraccionaria da mantisa. O bit principal é o 1).

   52 díxitos da mantisa máis o díxito 1 son en total **p=53**, que é a **precisión** da máquina. Recordar que tanto este valor como os expoñentes extremos -1023 e 1024 poden variar dependendo da máquina.

3. Hai un total de $2^{64} + 1 \approx 1.8 \times 10^{19}$ números máquina[4]: $2^{52} \approx 4.5 \times 10^{15}$ mantisas diferentes; $1024 - (-1023) + 1 = 2048 = 2^{11}$ expoñentes diferentes; dous signos e o número cero.

[3] Recordar que a suma dos n primeiros termos $a_1, a_2, \ldots a_n$ dunha serie xeométrica de razón $r \neq 1$ é $\frac{a_n r - a_1}{r-1}$.

[4] Variacións con repetición de m elementos tomados de n en n: $\mathrm{VR}_m^n = m^n$.

4. Cada un destes números almacénase na máquina nun agrupamento de 64 bits. Os bits numéranse do 1 ao 64. O bit 64 utilízase para recoller o signo (0 positivo, 1 negativo). O número 1 da parte enteira non é necesario almacenalo; é de feito o primeiro díxito significativo, que forzosamente vale 1. Os restantes 63 bits repártense entre o expoñente (11 bits) e a parte fraccionaria da mantisa (52 bits).

   $\pm(1.t_1 \dots t_{52}) \times 2^{\exp}$ exprésase da forma $\pm(1.t_1 \dots t_{52}) \times 2^{\mathbf{e}-\mathbf{c}}$ tomando $\mathbf{e} = \exp + c$, e sendo **c** un número de **desprazamento** que se utiliza para evitar empregar un bit para gardar o signo do expoñente. Ao número **e** chámaselle **expoñente desprazado**, **expoñente escalado** ou **expoñente almacenado**, porque é o que realmente se almacena na máquina.
   Con 11 díxitos binarios pódese contar ata 2047 ($(11111111111)_{(2)} = 1\times2^{10}+1\times2^9+\ldots+1\times2^1+1\times2^0 = 1+2+2^2+\ldots+2^{10} = 2^{11}-1 = 2047$), polo que este será o valor máximo do expoñente almacenado. Tomando $c = 1023$, mentres o expoñente $\exp$ varía entre -1023 e 1024, o expoñente almacenado varía entre 0 e 2047.

Signo (1 bit)	**Expoñente escalado (11 bits)**	**Díxitos $d_1 \dots d_{52}$ da mantisa (52 bits)**

   Así, por exemplo, o número $-18.625_{(10)} = 1110110.101_{(2)} = (-1.110110101)\times 2^6$ almacénase como segue (debemos ter en conta que o expoñente escalado neste caso é $6 + 1023 = 1029 = 10000000101_{(2)}$)

Signo (1 bit)	**Expoñente escalado (11 bits)**	**Díxitos $d_1 \dots d_{52}$ da mantisa (52 bits)**
1	10000000101	11011010100000000000000 ... 0

   Os expoñentes escalados extremos **e=0** e **e=2047** (que veñen dos expoñentes extremos -1023 e 1024) utilízanse para representar **valores especiais**, que son:
   - O **valor** 0, que non se pode representar directamente debido á suposición de que o primeiro díxito da mantisa é 1. O cero represéntase con tódolos bits do expoñente escalado e da parte fraccionaria da mantisa nulos (é dicir, **e=0** e $t_1 \dots t_{52}$ todos cero). Hai un $+0$ e un -0, dependendo de que o bit do signo sexa 1 ou 0, aínda que ambos valores son equivalentes.
   - O **infty**, tamén chamado **overflow** e denotado polos símbolos *Infinity*, *infty* ou *inf*. Represéntase con tódolos bits do expoñente escalado a 1 e tódolos da mantisa nulos (é dicir, **e=2047** e $t_1 \dots t_{52}$ todos cero). Dependendo do bit do signo, distínguense entre *Infinity* e *-Infinity*. Estes números especiais serven para definir o resultado dunha operación que é maior en valor absoluto que calquera número representable.
   - O valor **Not a number**, denotado polo símbolo **Nan**. Serve para etiquetar os resultados de operacións ilegais (raíces cadradas de números negativos, arcosenos con argumento superior á unidade, $\frac{\pm0}{\pm0}, +\infty -$

$\infty, \frac{\pm\infty}{\pm\infty}, \ldots$). Represéntase con tódolos bits do expoñente escalado a 1 (**e=2047**) e valores non nulos dos bits da mantisa, que se utilizan para definir varias clases de *NaN* (clasificados nas categorías de *QNaN* para operacións indeterminadas e *SNaN* para operacións non válidas).

— **Números non normalizados**: se tódolos bits do expoñente escalado son nulos (**e=0**) pero os da mantisa non son todos nulos, suponse que o bit principal non é 1 como no caso ordinario, e chámanse a estes números *números non normalizados*. Os números non normalizados representan valores de $\pm(0.t_1 \ldots t_{52}) \times 2^{-1022}$. Debemos ter en conta que tales números non teñen efectivamente a mantisa normalizada. Son polo tanto números máis pequenos en valor absoluto que calquera número normalizado representable, e utilízanse para representar **underflows**. Pódese pensar neles como un tipo de cero.

A táboa seguinte resume o anterior.

Cando o programa produce como resultado dun cálculo un número que non pode representar porque o resultado non está definido ou porque a operación é ilegal, asigna ao resultado un dos valores especiais anteriores, dando mensaxes como NaN, overflow ou underflow, e seguidamente da un erro de execución, que pode rematar, dependendo da gravidade do erro e do compilador, nun aborto da execución do programa. Os resultados de operacións con valores especiais dependen do compilador. Hai compiladores que poñen á disposición do usuario opcións de tratamento destes valores especiais.

exp	exp escalado (e)	exp escalado en binario	**Valor numérico que representa**	
-1023	0	$(00000000000)_{(2)}$	± 0 se $t_i = 0, \forall i = 1 \ldots 52$	$\pm$ Cero
	0	$(00000000000)_{(2)}$	$\pm(0.t_1 \ldots t_{52})_{(2)} \times 2^{-1022}$ se $\exists i, t_i \neq 0$	Números non normalizados
-1022	1	$(00000000001)_{(2)}$	$\pm(1.t_1 \ldots t_{52})_{(2)} \times 2^{-1022}$	
-1021	2	$(00000000010)_{(2)}$	$\pm(1.t_1 \ldots t_{52})_{(2)} \times 2^{-1021}$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	Números normais
0	1023	$(01111111111)_{(2)}$	$\pm(1.t_1 \ldots t_{52})_{(2)} \times 2^{0}$	
1	1024	$(10000000000)_{(2)}$	$\pm(1.t_1 \ldots t_{52})_{(2)} \times 2^{1}$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
1023	2046	$(11111111110)_{(2)}$	$\pm(1.t_1 \ldots t_{52})_{(2)} \times 2^{1023}$	
1024	2047	$(11111111111)_{(2)}$	$\pm\infty$ se $t_i = 0, \forall i = 1 \ldots 52$	$\pm$ *Infinity*
1024	2047	$(11111111111)_{(2)}$	*NaN* se $\exists i, t_i \neq 0$	*Not a number*

5. Os números normais máquina son os números:
$\pm(1.t_1 \ldots t_{52}) \times 2^{\exp}, \quad 0 \leq t_i \leq 1, \quad i = 1, \ldots 52, \quad -1022 \leq \exp \leq 1023,$
ou se se quere, os números:

   $\pm(0.1t_2 \ldots t_{52}) \times 2^{\exp}, 0 \leq t_i \leq 1, \quad i = 1, \ldots 52, \quad \mathbf{-1021} \leq \exp \leq \mathbf{1024}.$

   Hai en total **53 díxitos significativos**.

   O maior número máquina positivo é

$$(1.1 \ldots 1) \times 2^{1023} = (0.1 \ldots 1) \times 2^{1024} \approx 1.79 \times 10^{308}.$$

   O menor número máquina positivo é

$$(1.0 \ldots 0) \times 2^{-1022} = (0.1 \ldots 0) \times 2^{-1021} \approx 2 \cdot 22 \times 10^{-308}.$$

O menor e o maior número máquina negativos son, respectivamente, os negativos dos anteriores maior e menor positivos.

Todos estes datos sinalados en «negrita» relativos aos números normais son, como xa dixemos, característicos de cada ordenador; e é a información que o ordenador proporciona se se lle pregunta acerca do sistema co que traballa. Nas linguaxes de programación existen funcións intrínsecas para isto (ver nos «Anexos» o apartado «Datos do sistema punto flotante do ordenador»).

6. Para facerse unha idea de como son os números máquina do intervalo $[2^j, 2^{j+1})$, $j \in [-1022, 1023]$, tomaremos por exemplo só 3 bits na parte fraccionaria (en vez de 52) e escribirémolos de menor a maior:
$(1.000) \times 2^j = 2^j$,
$(1.001) \times 2^j = (1 + 2^{-3}) \times 2^j$,
$(1.010) \times 2^j = (1 + 2^{-2}) \times 2^j$,
$(1.011) \times 2^j = (1 + 2^{-2} + 2^{-3}) \times 2^j$,
$(1.100) \times 2^j = (1 + 2^{-1}) \times 2^j$,
$(1.101) \times 2^j = (1 + 2^{-1} + 2^{-3}) \times 2^j$,
$(1.110) \times 2^j = (1 + 2^{-1} + 2^{-2}) \times 2^j$,
$(1.111) \times 2^j = (1 + 2^{-1} + 2^{-2} + 2^{-3}) \times 2^j$,
Hai $2^3 = 8$ números, que son as posibilidades que hai de construír con dous díxitos números diferentes que ocupen tres posicións (variacións con repetición de dous elementos tomados de tres en tres).

   Observar que cada un destes números constrúese sumándolle ao inmediatamente anterior $2^{-3} \times 2^j$ (é dicir, 2^{j-3}). Así, por exemplo, $(1.001) \times 2^j + 2^{-3} \times 2^j = (1 + 2^{-3}) \times 2^j + 2^{-3} \times 2^j = (1 + 2^{-3} + 2^{-3}) \times 2^j = (1 + 2 \times 2^{-3}) \times 2^j = (1 + 2^{-2}) \times 2^j = (1.010) \times 2^j$.
   Polo tanto, fixado o expoñente j, os números consecutivos constrúense sumándolle 2^{-3} á mantisa inmediatamente anterior.

7. Hai un total de $2^{52} \approx 4.5 \times 10^{15}$ números máquina en cada un dos intervalos $[2^j, 2^{j+1})$ e $(-2^{j+1}, -2^j]$ para todo enteiro $j \in [-1022, 1023]$.

8. A distancia entre dous números máquina consecutivos depende da diferenza entre os valores consecutivos da mantisa (2^{-52} nesta máquina) e do expoñente. A distancia entre dous números máquina consecutivos do intervalo $[2^j, 2^{j+1}]$ (ou do intervalo $[-2^{j+1}, -2^j]$) é 2^{j-52} (ou o que é o mesmo, $\frac{2^j}{2^{52}}$).
   En xeral, os números máquina non están igualmente espazados, pero si o están en cada un dos intervalos $[2^j, 2^{j+1}]$.
   Notar, por exemplo, que os números máquina do intervalo $[2^0, 2^1] = [1, 2]$ e do intervalo $[-2, -1]$ están igualmente espazados a unha distancia 2^{-52}.
   Dado que a lonxitude do intervalo $[2^j, 2^{j+1}]$ aumenta a medida que j medra, canto máis grandes sexan os números máquina, máis espazados estarán uns dos outros.

9. Dacordo con todo o anterior acerca da distribución dos núneros máquina, podemos dicir, por exemplo:
   a) No intervalo $[0, 2^{-1022})$ hai un só número máquina, o cero; non obstante, no intervalo $[2^{-1022}, 2^{-1021})$, contiguo ao anterior e da mesma amplitude ($2^{-1021} - 2^{-1022} = 2^{-1021} \times (1 - 2^{-1}) = 2^{-1021} \times (1 - \frac{1}{2}) = 2^{-1021} \times (\frac{1}{2}) = 2^{-1022}$), hai $2^{52} \approx 4.5 \times 10^{15}$.
   b) No intervalo $[2^{1024}, \infty)$ non hai ningún número máquina.
   c) Hai tantos números máquina no intervalo $[2^{1023}, 2^{1024})$, que é un intervalo de amplitude moi grande ($2^{1023} \approx 8.99 \times 10^{307}$), coma no intervalo $[2^{-1022}, 2^{-1021})$, de amplitude moi pequena.
   d) A distancia entre dous números máquina consecutivos do intervalo $[2^{1023}, 2^{1024})$ é $2^{1023-52} = 2^{971} \approx 1.99 \times 10^{292}$, mentres que a distancia entre dous números máquina consecutivos do intervalo $[2^{-1022}, 2^{-1021})$ é $2^{-1022-52} = 2^{-1074} \approx 4.94 \times 10^{-324}$.
   e) Hai practicamente tantos números máquina normais no intervalo $(0, 1)$ coma no intervalo $[1, \infty)$: no intervalo $(0, 1)$ están os subintervalos $[2^{-1022}, 2^{-1021}), \ldots, [2^{-1}, 2^{0})$; en total 1022 subintervalos, cada un deles con 2^{52} números. No intervalo $[1, \infty)$ están os subintervalos $[2^{0}, 2^{1}), \ldots, [2^{1023}, 2^{1024})$; en total 1024 subintervalos, cada un deles con 2^{52} números.

   NOTA.- Cando a máquina traballa en simple precisión, manexa os números do sistema punto flotante $F(2, 24, -126, 129)$. Neste caso a máquina garda os números en agrupamentos de 32 bits (8 para o expoñente en exceso, e 23 para os díxitos da parte fraccionaria da mantisa). Neste caso a **precisión** da máquina é 24. Todo é análogo ao visto en dobre precisión tomando agora: a constante de escalado $c = 127$ e $\mathbf{-127} \leq \exp \leq \mathbf{128}$.

   Os únicos números que están representados de maneira exacta na máquina son os que aparecen na táboa do apartado anterior como números normais (máis o cero). Calquera outro número real é aproximado por un dos da máquina.

10. Denomínase **épsilon** da máquina ao número máquina positivo máis pequeno tal que $1 +$ **épsilon** > 1 na máquina. Na máquina establecida, e en dobre precisión, ten o valor **épsilon**$= 2^{1-p} = 2^{1-53} = 2^{-52} \approx 2.22 \times 10^{-16}$.
    Notar entón que a distancia entre o número 1 (que é un núnero máquina) e o seguinte número máquina é xustamente **épsilon**.

    Recapacitemos na distancia relativa entre dous números máquina consecutivos da nosa máquina modelo. Pensemos nos números máquina dun intervalo $[2^j, 2^{j+1}]$, $j \in [-1022, 1023]$ fixado. Dous números consecutivos calquera deste intervalo distan 2^{j-52}, sendo os números máquina deste intervalo ordenados de menor a maior:

    $(1.0\ldots0)_{(2)} \times 2^j = 2^j, \ldots, (1.1\ldots1)_{(2)} \times 2^j = (1 + 2^{-1} + \ldots + 2^{-52}) \times 2^j$, $(1.0\ldots0)_{(2)} \times 2^{j+1} = 2^{j+1}$.

Se calculamos o erro relativo entre cada dous consecutivos resultaría:

$$\frac{2^{j-52}}{2^j} = 2^{-52}, \ldots\ldots\ldots, \frac{2^{j-52}}{(1+2^{-1}+\ldots+2^{-52})\times 2^j}.$$

Observar que no numerador aparece sempre a mesma cantidade (que é a distancia entre ámbolos dous), e o denominador cada vez é un pouco maior pero, en calquera caso nunca supera a 2^{j+1}, caso no que o cociente vale $\frac{2^{j-52}}{2^{j+1}} = 2^{-53}$. Non se pode esixir entón, entre dous números máquina, un erro relativo igual ou inferior a 2^{-53}; o erro relativo máis pequeno que se pode lograr expresado na forma 2^j é 2^{-52}. Pois ben, o valor 2^{-52} é o **épsilon**.

Notar que, efectivamente, o **épsilon** da máquina caracteriza a precisión desta: canto menor sexa o **épsilon**, maior cantidade de números estarán representados na máquina.

Como xa dixemos, cando un número real x non é un número máquina, é aproximado por un dos da máquina, que denotamos como $\mathrm{Fl}(x)$. Demóstrase que:

$$\left|\frac{x-\mathrm{Fl}(x)}{x}\right| \leq \begin{cases} 2^{1-p} & \text{se se aproxima por truncamento,} \\ \frac{2^{1-p}}{2} & \text{se se aproxima por redondeo.} \end{cases}$$

É dicir, o erro relativo máximo que se comete cando se aproxima un número real por un da máquina é **épsilon** se a aproximación se fai por truncamento e $\frac{\textbf{épsilon}}{2}$ se a aproximación se fai por redondeo. Na máquina establecida, $\frac{\textbf{épsilon}}{2} = \frac{2^{-52}}{2} = 2^{-53}$.

Cando se fai calquera operación elemental $\circ$ (símbolo que representará a $+, -, \times, \div$) entre dous números máquina x e y, o resultado $x \circ y$ calcúlase utilizando as $2p$ cifras significativas que poden achegar ámbolos dous números (os ordenadores dispoñen de medios para isto) e, a continuación, normalízase (é dicir, aproxímase por un número máquina $\mathrm{Fl}(x \circ y)$). Pois ben, demóstrase tamén que:

$$\left|\frac{x\circ y-\mathrm{Fl}(x\circ y)}{x\circ y}\right| \leq \begin{cases} 2^{1-p} & \text{se se aproxima por truncamento,} \\ \frac{2^{1-p}}{2} & \text{se se aproxima por redondeo.} \end{cases}$$

É dicir, tamén o erro relativo máximo que se comete cando se fai unha operación elemental en punto flotante é **épsilon** se a aproximación se fai por truncamento e $\frac{\textbf{épsilon}}{2}$ se a aproximación se fai por redondeo.

O valor $\mathbf{u} = \frac{\textbf{épsilon}}{2}$ denomínase **unidade de redondeo**. Aparece en tódalas análises de erros que se producen cando se realizan cálculos en formato de representación flotante con aritmética de redondeo.

Recordar que, en base 10, dise que $\tilde{x}$ aproxima a x en s cifras significativas exactas (ou que x e $\tilde{x}$ coinciden nas s primeiras cifras significativas) se s é o maior enteiro non negativo tal que $\frac{|x-\tilde{x}|}{|x|} \leq 5 \times 10^{-s}$. Dado que o erro relativo nos indica o número de díxitos significativos nos que coinciden os números implicados, o valor de **épsilon** $= 2^{-52} \approx 2.22 \times 10^{-16}$ significa que nesta máquina gárdanse de forma exacta tódolos números con 15 cifras significativas e tamén a maior parte dos números con 16 cifras significativas. **En resumo, non se pode esixir, nesta máquina, que un número en base 10 aproxime a outro en máis de 16 cifras significativas**. É por isto que nos test de parada que faremos na programación dos nosos algoritmos (técnica que veremos máis adiante), non debemos esixir un erro relativo entre dous iterantes consecutivos menor que o **épsilon** da máquina.

4.3. Erros comúns ao traballar con números en punto flotante

Chámase **erro local de redondeo** ao erro que se comete cando se aproxima un número x por un dos da máquina (número punto flotante $\mathrm{Fl}(x)$). Así, por exemplo, o número π debe ser aproximado necesariamente, xa que non é un número máquina.
Nas operacións básicas realizadas no ordenador (operacións no sistema punto flotante) prodúcense tamén normalmente erros.
Tódolos procesadores están deseñados para cometer un erro máximo igual á unidade de redondeo ao aproximar un número por un número punto flotante, e tamén ao realizar calquera das operacións elementais (suma, resta, multiplicación e división), tal como xa indicamos na sección anterior. Isto último significa que o ordenador calcula unha operación elemental de maneira tal que o resultado que se obtén ten correctos o máximo número p de díxitos representable.
En Viaño Rey (1995) pódese ver con detalle a análise dos erros producidos nos cálculos ocasionados polas operacións realizadas en aritmética punto flotante. Aquí mostraremos simplemente algúns exemplos onde se opera con dita aritmética, e onde aparecen erros comúns que se suelen cometer como consecuencia das propiedades dela. Remataremos este apartado con algúns consellos a seguir para tratar de evitar estes erros.

Para a comprensión dos exemplos, requírese saber que: para sumar (restar) números flotantes iguálanse os expoñentes, súmanse (réstanse) as mantisas e logo normalízase o resultado. Para multiplicar (dividir) números flotantes, multiplícanse (divídense) as mantisas, súmanse (réstanse) os expoñentes e logo normalízase o resultado. Nestes exemplos traballaremos en base 10, xa que é a aritmética coa que nós sabemos traballar. Nos «Anexos» mostraremos resultados de exercicios propostos nas actividades prácticas da unidade, realizados no ordenador e, polo tanto, procedentes de cálculos feitos en base 2.

— **Exemplo 1**
Sexa $x = 0.235 = 0.235 \times 10^0$ e $y = 0.00123 = 0.123 \times 10^{-2}$. Tense

$$\mathrm{Fl}(x+y) = \mathrm{Fl}(0.235 \times 10^0 + 0.00123 \times 10^0) = 0.236 \times 10^0 \neq 0.23623 = x+y.$$

Como se pode observar, o resultado operando en aritmética finita con $p = 3$ díxitos significativos é 0.236×10^0; o exacto é 0.23623. Prodúcese así no resultado un erro absoluto de 0.00023, que corresponde a un erro relativo de 9.7×10^{-4}, que indica que tódolos díxitos do resultado son exactos (é dicir, son os que teñen que ser dacordo coa aritmética coa que se traballa).

— **Exemplo 2**
Sexa $x = 1867 = 0.1867 \times 10^4$ e $y = 0.32 = 0.3200 \times 10^0$. Tense

$$\mathrm{Fl}(x+y) = \mathrm{Fl}(0.1867\times10^4+0.000032\times10^4) = 0.1867\times10^4 \neq 1867.32 = x+y$$

Trabállase agora con aritmética finita con $p = 4$ díxitos significativos. Como no caso anterior, tamén hai erro no resultado da suma.

Como ilustran os dous exemplos anteriores, aínda que non haxa erro na representación flotante dos operandos ($x = \mathrm{Fl}(x)$ e $y = \mathrm{Fl}(y)$), case sempre hai erro no resultado da súa suma ($\mathrm{Fl}(x + y) \neq x + y$).

Observar que, no segundo exemplo, o número máis pequeno non se tivo en conta. Non obstante, o erro relativo cometido no resultado é 1.7×10^{-4}, o que indica que tódolos díxitos do resultado son exactos; tan exactos como os operandos.

— **Exemplo 3**
Tomaremos agora dous números próximos entre si e consideraremos a súa resta.
Sexan os números de 6 díxitos $x = 0.467546$ e $y = 0.462301$. Representémolos só con 4 díxitos, é dicir, $\mathrm{Fl}(x) = 0.4675$ e $\mathrm{Fl}(y) = 0.4623$. Ao ser restados, tense $\mathrm{Fl}(x - y) = 0.0052 \neq x - y = 0.005245$.

Observar que o resultado obtido, 0.0052, é o exacto para a resta dos números flotantes de 4 díxitos de precisión, pero non é o exacto para a resta dos números orixinais (0.005245). De feito, o resultado perdeu díxitos, só ten dous díxitos de precisión, cando os operandos tiñan catro. Este fenómeno de perda de díxitos de precisión denomínase **cancelación** ou diferenza cancelativa, e pode resultar perigosa (**cancelación catastrófica** ou cancelación excesiva) se o resultado da resta é utilizado noutras operacións, pois para elas un dos operandos ten só dous díxitos de precisión.

— **Exemplo 4** (Cancelación catastrófica)
Sexan os números de 6 díxitos $x = 0.732112$ e $y = 0.732110$. Representémolos só con 5 díxitos, é dicir, $\mathrm{Fl}(x) = 0.73211$ e $\mathrm{Fl}(y) = 0.73211$. Ao ser restados, tense $\mathrm{Fl}(x - y) = 0 \neq x - y = 0.000002 = 0.2 \times 10^{-5}$.
Observar que neste caso perdéronse tódolos díxitos significativos no resultado; non obstante, este é o valor exacto da resta dos números flotantes. Utilicemos agora este resultado nunha operación posterior, como $\mathrm{Fl}(\mathrm{Fl}(x - y) * \mathrm{Fl}(y)) = 0 \neq (x - y)y = 0.14642\ldots \times 10^{-5}$. Observar que o resultado é completamente inexacto. Tódolos díxitos son incorrectos,

produciuse unha cancelación catastrófica.

— **Exemplo 5**
Sexa $x = 1867 = 0.1867 \times 10^4$ e $y = 0.201 = 0.201 \times 10^0$. Tense

$\mathrm{Fl}(xy) = \mathrm{Fl}((0.1867 \times 10^4) * (0.2010 \times 10^0)) = \mathrm{Fl}(0.0375267 \times 10^4) = 0.3753 \times 10^3 \neq 375.267 = xy$, e

$\mathrm{Fl}(x/y) = \mathrm{Fl}(0.1867 \times 10^4 / 0.2010 \times 10^0) = \mathrm{Fl}(0.928855\ldots \times 10^4) = 0.9289 \times 10^4 \neq 9288.55721393034\ldots = x/y$.

En resumo, case sempre se cometen erros cando se representa un número no ordenador e cando se realizan operacións aritméticas nel (é dicir, $\mathrm{Fl}(x) \neq x$, $\mathrm{Fl}(x \pm y) \neq x \pm y$, $\mathrm{Fl}(xy) \neq xy$, $\mathrm{Fl}(x/y) \neq x/y$). Aínda sendo o resultado exacto na aritmética na que o ordenador traballe, pódense perder díxitos significativos (como é no caso da cancelación), o que constitúe un **erro** a tódolos efectos.

Hai que saber tamén que as operacións de suma e multiplicación flotantes son conmutativas, pero non son asociativas. Así, aínda que $x+y+z = (x+y)+z = x+(y+z)$, ocorre que

$$\mathrm{Fl}(\mathrm{Fl}(x+y)+\mathrm{Fl}(z)) \neq \mathrm{Fl}(\mathrm{Fl}(x)+\mathrm{Fl}(y+z)),$$

e polo tanto a orde coa que se opera é importante, xa que afecta a como se van acumulando os erros. Unha orde inadecuada pódeos ir aumentando.

Para evitar a propagación de erros nos cálculos, na medida do posible, débese:

— Evitar dividir por cantidades moi pequenas.
Supoñamos x aproximado por $\mathrm{Fl}(x)$ con erro de redondeo ε (é dicir, $x = \mathrm{Fl}(x) + \varepsilon$). Ao dividir por δ, se δ é «pequeno, relativamente», entón o novo erro de redondeo ε/δ é «grande» (maior que o inicial); pero se δ é relativamente grande, ε/δ é «pequeno».

— Evitar multiplicar por números grandes.
Multiplicar por números grandes pose levar os resultados rapidamente á zona de overflow. Se, por exemplo, se ten que realizar $\frac{x \cdot y}{p \cdot q}$, con x, y, p, q «grandes» en magnitude, é preferible facer: $(\frac{x}{p}) \cdot (\frac{y}{q})$.

— Evitar restar números relativamente iguais.

Trátase de evitar a perda de cifras significativas. Así, por exemplo, se desexamos avaliar nun sistema de punto flotante a expresión $\sqrt{x^2+1} - 1$, sendo x un valor positivo cercano a cero, procederase da forma:

$$\sqrt{x^2+1} - 1 = \frac{\sqrt{x^2+1}-1}{\sqrt{x^2+1}+1} \cdot (\sqrt{x^2+1}+1) = \frac{(x^2+1)-1}{\sqrt{x^2+1}+1} = \frac{x^2}{\sqrt{x^2+1}+1}.$$

— Evitar o uso de expresións complicadas.
Débense utilizar sempre as expresións máis simplificadas para ter que realizar así menos operacións e, polo tanto, provocar menos erros. Por exemplo, á hora de avaliar un polinomio de grao n nun punto x, en vez de realizar tódalas operacións que nos suxire a aplicación directa da expresión $P(x) = a_0 \times x^n + \ldots + a_{n-1} \times x + a_n$, para o que hai que realizar[5] da orde de $\frac{n^2}{2}$ operacións básicas, é mellor utilizar o método de Horner, xa que só precisa realizar $2n$ operacións.

— Sumar os números en magnitude crecente.
Trátase de que os sumandos máis pequenos non sexan ignorados. Se $a < b < c$, para calcular $a + b + c$, débese poñer $(a + b) + c$, xa que a suma dos pequenos realízase primeiro, contribuíndo así mellor á suma total.

— Evitar repeticións de avaliacións funcionais.
Se, por exemplo, se vai calcular $y = \mathrm{e}^x + \ln(\mathrm{e}^x) + \sin(3\mathrm{e}^x)$, é preferible facer $z = \mathrm{e}^x$ e logo facer $y = z + \ln(z) + \sin(3z)$

Nota.- No ordenador hai unha orde de prioridade establecida nas operacións: exponenciación, multiplicación_división, suma_resta. As operacións da mesma prioridade realízanse de esquerda a dereita. As prioridades pódense modificar co uso dos parénteses.
Outras recomendacións a seguir á hora de realizar os cálculos están relacionadas coa linguaxe de programación, polo que serán indicadas no seu momento nas clases prácticas de ordenador.

5. Algúns conceptos básicos de interese cara ás próximas unidades

Na *Análise Numérica* trabállase sempre con tres elementos fundamentais: **problema** que hai que resolver, **método** para resolvelo, e **máquina** para realizar os cálculos requiridos polo método.

Hai problemas que por si mesmos presentan inconvenientes, por exemplo, os **problemas mal condicionados**, que son aqueles nos que unha 'pequena' modificación nos datos provoca, traballando con aritmética exacta ou con aritmética suficientemente precisa, unha variación «grande» na solución. Como exemplo, o problema de cálculo das raíces dun polinomio debido a Wilkinson e que se pode ver en Young-Gregory (1973): o polinomio $P_{20}(x) = (x-1)(x-2)\cdots(x-20) = x^{20} - 210\,x^{19} + \cdots$ ten como raíces os vinte números naturais $1,\ 2, \ldots,\ 20$; non obstante, modificando lixeiramente o coeficiente que acompaña á potencia 19 da forma $Q_{20}(x) = P_{20}(x) - 2^{-23}x^{19}$, resulta un novo polinomio $Q_{20}(x)$ cuxas vinte raíces son dez reais e dez complexas, estas con parte imaxinaria

[5] A suma dende o primeiro termo a_1 ata o enésimo a_n dunha serie aritmética é $s = \frac{(a_1+a_n)n}{2}$.

dunha magnitude non próxima a cero ($\pm 2.81\ i, \pm 2.51\ i, \ldots$). Esta grande variación das raíces con respecto á pequena variación dos datos non é debida á acumulación dos erros de redondeo (os valores das raíces son os que teñen que ser), é dicir, non está provocada pola máquina de cálculo; é unha característica do propio problema: é un problema mal condicionado. Os problemas mal condicionados precisan dunha análise máis exhaustiva cara á súa resolución.

Hai **métodos que** por si mesmos son teoricamente válidos, pero **non son utilizables na práctica** porque resultan **camiños demasiado longos**, por exemplo o método de Cramer para resolver un sistema linear (precisa dunha cantidade excesiva de operacións, tal como xa vimos).

Da interrelación problema-método-máquina xorde a **inestabilidade** numérica dun método. Recordar que a máquina é finita, de maneira que pode ocorrer o seguinte: fixada a máquina e o problema a resolver, pode haber un método que teoricamente nos garanta unha boa aproximación da solución e, non obstante, ao executalo os erros de redondeo desfiguran a solución. Dise entón que dito **método** é **numericamente inestable** ao aplicalo a dito problema. Este mesmo método, aplicado a outro problema e coa mesma ferramenta de cálculo, pode non presentar este inconveniente. Que debemos facer en tal caso? Pois cambiar o método, xa que cambiar a unha máquina máis potente pode non ser posible. Analizar a propagación dos erros de redondeo producidos nos cálculos realizados por un método sobre un problema e cunha determinada aritmética de traballo é unha parte, non facil de realizar, na análise do método. Un exemplo de inestabilidade numérica móstrase no «Anexo».

O ordenador que se vaia utilizar, por moi potente que sexa, sempre é finito, o que pode ser un obstáculo á hora de manexar unha excesiva cantidade de datos.

Como ocorre sempre ante calquera problema que haxa que resolver, canta máis información se teña acerca del, mellor, xa que permitirá, por unha parte, aplicarlle o método máis adecuado e, por outra parte, asegurarse de que o problema está ben resolto.

Cando un método proporciona a solución dun problema nun número finito de operacións, dise que o método é **directo**. Así, por exemplo, o método de Cramer é un método directo. Tamén son métodos directos os que temos usado normalmente para calcular os ceros dun polinomio de grao non superior a 4. Pero non obstante, como sabemos, para grao superior a 4 xa non é posible usar este tipo de métodos. O mesmo ocorre cando queremos calcular os ceros dunha función transcendente $F(x)$, é dicir, que non sexa polinómica (ou, o que é o mesmo, cando queremos resolvela ecuación numérica $F(x) = 0$). Nestas ocasións, coma en moitas outras, os métodos son de **tipo iterativo**. Caracterízanse estes por **construír unha sucesión de elementos**, chamados **iterantes**, coa intención de que **converxa á solución** que andamos buscando, e así poder aproximala. Neste caso, os iterantes son números. Noutro caso, os iterantes poden ser vectores ou outro tipo de estructuras.

Continuaremos centrados nos métodos para aproxima-las raíces dunha ecuación numérica para unha mellor comprensión de como se suele actuar.
Evidentemente, se houbese un método «marabilloso» para aproxima-los ceros de calquera ecuación, estudaríase ese e ningún outro. Pero a realidade é que cada problema (neste caso cada ecuación) pode presentar casuísticas diferentes, de tal xeito que mesmo hai métodos que non se lle poden aplicar.
Que método é o mellor? Primeiro hai que fixar o problema que debemos resolver, agora podemos contestar: entre tódolos que se lle poden aplicar, o mellor é o que chega antes a unha mesma boa aproximación da solución. Notar que a bondade se mide en tempo, non en cantidade de operacións, xa que estas poden ser moitas pero máis fáciles e polo tanto de cálculo rápido.
No estudo dos métodos existen toda unha serie de resultados abstractos que nos garanten, baixo certas condicións, que un determinado método nos proporciona a solución dun problema. O ideal sería que para o noso problema particular se poidese comprobar de modo relativamente fácil que obedece a un contexto abstracto dos xa estudados. Se isto non é posible, deberemos analiza-lo noso problema dun modo particular e construír un método que o resolva.

Supoñamos que aplicamos un método iterativo converxente para aproximar unha raíz α dunha ecuación dada (é dicir, α cumpre $F(\alpha) = 0$ para unha $F(x)$ dada). Que o método sexa **converxente** significa que se poden construír sucesións $\{x_n\} = \{x_0, x_1, x_2, ...\}$ con dito método que converxen a α. Poñámonos a construír unha destas sucesións. Cando paramos esta construción? (observar que, salvo que algún iterante coincida con α, non acabariamos máis). É preciso por tanto adoptar un **criterio ou test de parada**. Deterémonos nesta cuestión, dada a importancia que ten á hora de implantar este tipo de métodos no ordenador.

5.1. Sobre o test de parada nun proceso iterativo

Supoñamos que podemos construír unha sucesión $\{x_n\}_{n\geq 0}$ converxente á raíz α dunha ecuación dada $F(x) = 0$; isto significa que nos podemos aproximar a α canto queiramos. Salvo que $x_n = \alpha$ para algún n, o proceso é infinito, logo, cando paramos o proceso de construción da sucesión?

1. O ideal sería proceder como segue:
   - — Construímos o iterante x_n.
   - — Calculamos a distancia de x_n a α, $|x_n - \alpha|$, e comprobamos se esa distancia $|x_n - \alpha|$ é «pequena», é dicir, se $|x_n - \alpha| < \epsilon$ (onde ϵ é un valor de tolerancia «pequeno» prefixado de antemán). É dicir, miramos se estamos o suficientemente próximos de α segundo un criterio prefixado.
     - Se si, PARAMOS o proceso, e aproximamos o valor descoñecido α que andamos buscando, polo valor coñecido x_n.
     - Se non, seguimos o proceso.

     (Esta parte do proceso é a que se chama **control de converxencia ou test de parada)**.
   - — Construímos o iterante x_{n+1}.
   - — E así sucesivamente.

Cando paramos o proceso porque o test de parada é satisfactorio, o que facemos é quedarnos co elemento x_n como aproximación de α. Cometemos entón o erro $|x_n - \alpha|$.

2. Notar que o procedemento anterior é absurdo, xa que implica coñecer α, que é precisamente o que queremos calcular. Sería entón interesante dispoñer dunha boa estimación da distancia $|x_n - \alpha|$, e dicir, dispoñer dunha «boa» $cota$ do erro, $|x_n - \alpha| < cota$, e de maneira que esta $cota$ si poidese ser calculada. Deste xeito, procederiamos como segue:
   — Construímos o iterante x_n.
   — Calculamos a $cota$ e comprobamos se esa $cota$ é "pequena", é dicir, se $cota < \epsilon$.
      • Se si, PARAMOS (xa que $|x_n - \alpha| < cota < \epsilon$), e aproximamos o valor decoñecido α que andamos buscando, polo valor coñecido x_n.
      • Se non, seguimos o proceso.
   — Construímos o iterante x_{n+1}.
   — E así sucesivamente.

   Así, por exemplo, no método de dicotomía[6] disponse da seguinte estimación do erro

$$|x_n - \alpha| < \frac{b-a}{2^{n+1}}.$$

   Neste caso, a cota $\frac{b-a}{2^{n+1}}$ é facilmente calculable e permítenos saber a cantidade n de iterantes que debemos construír se queremos ter garantido que o erro que cometemos ao aproximar α por x_n non supera a un ϵ prefixado: basta tomar n tal que $\frac{b-a}{2^{n+1}} \leq \epsilon$, de onde se deduce, aplicando logaritmo, que n debe cumprir $n \geq \frac{\ln(b-a/\epsilon)}{\ln(2)} - 1$. Calculamos entón o número $\frac{\ln(b-a/\epsilon)}{\ln(2)} - 1$; consideramos o número natural inmediatamente superior e chamámolo $niter$. Pois ben, segundo o exposto anteriormente, temos garantido que $|x_{niter} - \alpha| < \epsilon$, cumprindo esta mesma desigualdade tódolos iterantes con índice superior a $niter$ (e que, por suposto xa non se calcularán, dado que xa temos o test prefixado satisfeito e non se debe traballar gratuitamente). Pode que iterantes con índice inferior a $niter$ satisfagan tamén o test prefixado, pero non o podemos garantir.
   Observar que, deste xeito, non é preciso en cada iterante calcular o valor da $cota$ e comparala coa tolerancia, senón que unha vez fixada a tolerancia ϵ, xa podemos saber antes de comezar o proceso cantos elementos da sucesión debemos construír.

3. O problema é que normalmente non se dispón deste tipo de estimacións, ou son estimacións non facilmente calculables. Na práctica procédese entón como segue:

[6] Método que constrúe sucesións converxentes á solución α da ecuación $F(x) = 0$ baixo as condicións de que a e b son extremos dun intervalo que contén a α, no que F é continua e $F(a)F(b) < 0$.

— Construímos o iterante x_n.
— Calculamos $|x_n - x_{n-1}|$ e comprobamos se $|x_n - x_{n-1}| < \epsilon$.
  - Se si, PARAMOS, e aproximamos o valor decoñecido α polo valor coñecido x_n.
  - Se non, seguimos o proceso.
— Construímos o iterante x_{n+1}.
— E así sucesivamente.

Con respecto ó test de parada, sinalamos o seguinte:

a) Pode dar lugar a uhna **falsa converxencia**: $|x_n - x_{n-1}|$ pode ser «pequeno» porque a converxencia da sucesión sexa lenta e, non obstante, estarmos aínda lonxe de α. Outro control que se recomenda é calcular $F(x_n)$ e comparalo co cero, xa que andamos buscando un punto onde F vale cero; o que se implanta averiguando se $|F(x_n)| \leq \delta$ (sendo $\delta > 0$ un valor «pequeno» fixado de antemán), xa que en programación non se debe pedir a igualdade co número cero. É aconsellable pedir o cumprimento dos dous test simultaneamente.

b) A medida de proximidade utilizada entre os dous últimos iterantes construídos pode ser absoluta, $|x_{n+1} - x_n| \leq \epsilon$, ou relativa $\dfrac{|x_{n+1} - x_n|}{|x_n|} \leq \epsilon$, dependendo do nivel de esixencia que queiramos establecer. Observar, con respecto a isto:

b1)
- $|x_n| > 1 \Longrightarrow \dfrac{1}{|x_n|} < 1 \Longrightarrow \dfrac{|x_{n+1} - x_n|}{|x_n|} < |x_{n+1} - x_n|,$

  polo tanto, cando $|x_n| > 1$, tense

  $$|x_{n+1} - x_n| < \epsilon \Longrightarrow \frac{|x_{n+1} - x_n|}{|x_n|} < \epsilon,$$

  é dicir, o test do erro absoluto é máis severo que o test do erro relativo.
- $0 < |x_n| < 1 \Longrightarrow \dfrac{1}{|x_n|} > 1 \Longrightarrow \dfrac{|x_{n+1} - x_n|}{|x_n|} > |x_{n+1} - x_n|,$

  polo tanto, cando $0 < |x_n| < 1$, tense

  $$\frac{|x_{n+1} - x_n|}{|x_n|} < \epsilon \Longrightarrow |x_{n+1} - x_n| < \epsilon,$$

  é dicir, o test do erro relativo é máis severo que o test do erro absoluto.

b2) Seguindo a recomendación de evitar nun computador dividir por números «pequenos» (xa que isto ocasiona erros locais de redondeo grandes), cando $|x_n|$ está próximo a cero, o test do erro relativo

$$\frac{|x_{n+1} - x_n|}{|x_n|} < \epsilon$$

pódese implantar da forma

$$|x_{n+1} - x_n| \leq \epsilon |x_n|.$$

Non obstante, isto tamén pode presentar problemas se o límite da sucesión $\{x_n\}$ é un valor cercano a cero, xa que $|x_n|$ ϵ pode ser moi pequeno: pode que demasiado pequeno para ser representado no ordenador.

b3) Unha estratexia que se pode seguir é traballar coa expresión

$$\frac{|x_{n+1} - x_n|}{|x_n| + 1} < \epsilon,$$

xa que:

- cando os x_n son grandes, da practicamente igual traballar con esta expresión que coa expresión $\frac{|x_{n+1} - x_n|}{|x_n|} < \epsilon$, é dicir, estamos esixindo un test de erro relativo.
- cando con $\{x_n\}$ nos estamos achegando a un número próximo a cero, traballar coa expresión anterior é coma traballar coa expresión $\frac{|x_{n+1} - x_n|}{1} < \epsilon$, é dicir, estamos esixindo un test de erro absoluto.

En calquera caso, hai que ter sempre presente que o test de parada débese adecuar ás características do problema que se estea a resolver.

NOTA.- Acerca de que valores se poden asignar a δ e aos ϵ referidos neste texto, hai que ter presente quen son os números máquina e a súa distribución na recta real, o que nos informará da distancia absoluta e relativa entre ditos números (recordar, por exemplo, que, na nosa máquina, o número positivo máis pequeno é $(1.0\ldots0)\times 2^{-1022} \approx 2{\cdot}22\times 10^{-308}$, que a distancia entre dous números máquina consecutivos do intervalo $[2^j, 2^{j+1}]$ ($j \in [-1022, 1023]$) é 2^{j-52}, e que a máquina non detecta un erro relativo menor que o chamado **épsilon** da máquina, cuxo valor é $2^{-52} \approx 2.22 \times 10^{-16}$). Valores típicos de ϵ e δ son $10^{-6} \leq \epsilon, \delta \leq 10^{-10}$.

OS PRINCIPIOS METODOLÓXICOS

A unidade didáctica desenvolverase nun período de tres semanas, a razón de 4 horas por semana distribuídas da seguinte forma:

— 2 horas de teoría: dirixidas a todo o alumnado que constitúe oficialmente un grupo. Nelas desenvolveranse os contidos da unidade e calquera estudante poderá establecer as dúbidas correspondentes que estime oportunas. Asemade, a profesora ou o profesor encargado da docencia, poderá requiri-la participación do alumnado.

— 1 hora de prácticas de ordenador nun laboratorio informático: dirixida a un subgrupo de aproximadamente 20 estudantes de maneira que cada estudante teña acceso a un posto de ordenador. Cada estudante deberá traballar sobre unha práctica previamente proposta polo profesorado responsable. Estas sesións utilízanse para que o alumnado se familiarice cunha linguaxe de programación e aprenda a utilizala para realizar co ordenador os cálculos que se demandarán nas clases teóricas. Asemade, esta actividade axudaralle ao alumnado a profundar no estudo dos conceptos adquiridos e a afianzarse neles. A profesora ou profesor que corresponda, indicará con detalle as directrices que se deben seguir para a realización das prácticas. Así mesmo, atenderá as cuestións presentadas polo alumnado e levará un seguimento dos traballos de laboratorio que se vaian realizando.

  Nos «Anexo» da UD axuntamos algunha das prácticas propostas para seren executadas no ordenador.

— 1 hora de seminario nun laboratorio informático: dirixida a un subgrupo de aproximadamente 20 estudantes. O docente, ou calquera estudante orientado polo docente, realizará exercicios previamente propostos na clase teórica ou extraídos dun boletín de exercicios. Nestas sesións trátase de que o alumnado manexe e comprenda os conceptos desenvolvidos nas clases teóricas, interactuando entre si e co profesorado encargado.

  NOTA. Disporase dun **curso virtual** como apoio á docencia presencial que permitirá ao alumnado obter material sobre o que traballar (boletíns de exercicios, prácticas de ordenador, etc.) e material de apoio para aclarar contidos e dispoñer de axuda para a realización dos exercicios e das prácticas, conectarse a páxinas web de interese, informarse das súas cualificacións, etc.

AVALIACIÓN

Non se fai unha avaliación específica da unidade didáctica, faise sobre a materia da que forma parte. En dita avaliación, o peso da unidade é cuantitativamente moi pequeno. Isto é porque as ferramentas, conceptos e técnicas que nela se ven son manexados continuamente no resto das unidades da materia. Non obstante, o peso da unidade é cualitativamente fundamental, xa que sen esta «posta a punto», o estudante dificilmente poderá realizar tódalas tarefas que a materia lle vai esixir.

ANEXOS

Esta sección está dedicada a mostrar algunhas das prácticas de ordenador que se deben realizar.

Práctica inicial

1. Crear un directorio co nome CN1V. Deste directorio colgarán tódalas prácticas que se realizarán nesta materia.
2. No directorio CN1V crear un directorio co nome PRACTICA_INICIAL.
3. EXERCICIO PRÁCTICO INICIAL PARA REALIZAR EN FORTRAN 90.
   (Tódolos programas relativos a este exercicio gardaranse no directorio PRACTICA_INICIAL.)

   Para aproximar $\alpha = \sqrt{a}$, $a > 0$ dado, considérase a sucesión:

$$\begin{cases} x_0 \text{ dado} , \\ x_{n+1} = \frac{x_n(x_n^2+3a)}{3x_n^2+a} , n = 0, 1, \ldots \end{cases}$$

   Pódese probar que toda sucesión construída da forma anterior converxe a α.

   a) Construír un subprograma subroutine:
   raizdea(a,x0,nitmax,eps,raiz,iterc),
   que proporcione unha aproximación de α (os nomes *a, x0, nitmax, eps, raiz, iterc*) representan, respectivamente, o número a, o iterante inicial x_0, a cantidade máxima de iteracións a realizar, o $\varepsilon > 0$ «pequeno» que se utilizará no test de parada $|x_{n+1} - x_n| < \varepsilon$, a raíz cadrada de a e a cantidade de iteracións realizadas).
   b) Construír un subprograma de lectura dos datos.
   c) Construír o programa principal correspondente.
   d) Utilizar o programa anterior para calcular $\sqrt{27}$.

Datos do sistema punto flotante do ordenador

Realizar un programa en FORTRAN que utilice as funcións intrínsecas *kind*, *digits*, *huge*, *tiny*, *precision*, *range*, *eps*, *minexponent*, *maxexponent* para saber o sistema de cálculo do ordenador. Comprobar que se obteñen os resultados que se mostran a continuación:

```
Para i enteiro,  kind(i)=              4
Para x real simple precision,  kind(x)=              4
Para y real dobre precision,  kind(y)=              8

Para i ENTEIRO
p= digits(i)=            31
base=radix(i)=             2
i_max= huge(i)=    2147483647


Para x REAL SIMPLE PRECISION
precision decimal= precision(x)=             6
rango_expon_decimal= range(x)=            37

base=radix(x)=             2
p= digits(x)=            24
m= minexponent(x)=          -125
M= maxexponent(x)=           128
x_max= huge(x)=    3.40282347E+38
x_min= tiny(x)=    1.17549435E-38
eps_maquina= epsilon(x)=    1.19209290E-07


Para y REAL DOBRE PRECISION
precision decimal= precision(y)=            15
rango_expon_decimal= range(y)=           307

base=radix(y)=             2
p= digits(y)=            53
m= minexponent(y)=         -1021
M= maxexponent(y)=          1024
y_max= huge(y)=    1.79769313486231571E+308
y_min= tiny(y)=    2.22507385850720138E-308
eps_maquina= epsilon(y)=    2.22044604925031308E-016
1.+eps_maquina= 1.+epsilon(y)=     1.0000000000000002
```

Exemplos sobre erros de redondeo

Construír un programa en FORTRAN 90 que declare sp e dp como constantes «parameter» con valores 4 e 8 respectivamente, que declare as variables hs e hd en simple e dobre precisión, respectivamente, e que realice os cálculos que se indican a continuación. Comprobar se se obteñen os resultados que se mostran.

```
   OLLO á hora de traballar en simple ou en dobre precisión:

1 =           1

1. =   1.0000000

1.e-10 =  1.00000001E-10

1e-10 =  1.00000001E-10

1_sp =           1

1._sp =   1.0000000

1_dp =                   1

1._dp =   1.0000000000000000

1.e-10_dp =  1.0000000000000004E-010


    EXEMPLOS ONDE SE OBSERVAN ERROS DE REDONDEO
    *******************************************

 EXEMPLO 1:  traballando en simple precisión:
 ---------
(1.+10.**(-10))*(1.+10.**(-10))=    1.0000000


              traballando en dobre precisión:

(1._dp+10._dp**(-10))*(1._dp+10._dp**(-10))=    1.0000000002000000


   NOTAR que o valor exacto é  1+2*(10**(-10))+10**(-20),
   e que polo tanto, incluso en dobre precisión, cometese un erro= 10**(-20)
-------------------------------------------------

  OLLO:      traballando en simple precisión:
```

```
 epsilon(hs)=    1.19209290E-07

 1.+ epsilon(hs)=     1.0000001

               traballando en dobre precisión:

 epsilon(hd)=    2.22044604925031308E-016

 1.+ epsilon(hd)=     1.0000000000000002

-------------------------------------------------

  EXEMPLO 2:    en simple precisión:
  ---------
 1.+10.**(-17)=     1.0000000

                 en dobre:

 1._dp+10._dp**(-17)=     1.0000000000000000

  Nótese que 10**(-17) < epsilon(hd)

-----------------------------------------------

 1.+10.**(-6)=     1.0000010

 1._dp+10._dp**(-15)=     1.0000000000000000
```

Exemplo (cancelación catastrófica)

```
  DÉBESE EVITAR RESTAR NÚMEROS CASE IGUAIS,
  ****************************************
  porque isto ocasiona perda de díxitos de precisión,
  e isto pode ser perigoso (Cancelación catastrófica).

 EXEMPLO 1
 ---------

r=   0.1234567890123450E+00

s=   0.1234567890123400E+00

 A maquina calcula:
```

```
 t=r-s=    4.99600361081320443E-015

 NOTAR que o valor exacto de t é  t=5.e-15.

-----------------------------------------

 EXEMPLO 2
 ---------

 Proporciónanse:

  u=12345678901234567.

  u=12345678901234566.

  A máquina calcula:

 w=u-v=     2.0000000000000000

 NOTAR que o valor exacto de w é  w=1
```

Exemplo (un proceso numericamente inestable)

```
  EXEMPLO DE PROCESO NUMERICAMENTE INESTABLE
  ******************************************
         Ver, por exemplo, Kincaid-Cheney (1991)

         Sucesión de números reais construídos da forma:
                  x(0)=1,
                  x(1)=1/3,
                  x(n+1)=(13/3)x(n)-(4/3)x(n-1)  para n>=1

-------------------------------------------------|
    |                      |                     |
 n  | VALOR EXACTO DE x(n) |VALOR APROXIMADO DE x(n)|
    |                      |                     |
----|----------------------|---------------------|
   0|  0.100000000000000E+01|  0.100000000000000E+01|
   1|  0.333333333333333E+00|  0.333333333333333E+00|
   2|  0.111111111111111E+00|  0.111111111111111E+00|
   3|  0.370370370370370E-01|  0.370370370370371E-01|
   4|  0.123456790123457E-01|  0.123456790123458E-01|
   5|  0.411522633744856E-02|  0.411522633744908E-02|
   6|  0.137174211248285E-02|  0.137174211248494E-02|
   7|  0.457247370827618E-03|  0.457247370835984E-03|
```

8|	0.152415790275873E-03| 0.152415790309338E-03
9|	0.508052634252908E-04| 0.508052635591546E-04
10|	0.169350878084303E-04| 0.169350883438852E-04
11|	0.564502926947676E-05| 0.564503141129633E-05
12|	0.188167642315892E-05| 0.188168499043720E-05
13|	0.627225474386307E-06| 0.627259743499413E-06
14|	0.209075158128769E-06| 0.209212234581194E-06
15|	0.696917193762563E-07| 0.702400251859573E-07
16|	0.232305731254188E-07| 0.254237963642228E-07
17|	0.774352437513958E-08| 0.165164173303559E-07
18|	0.258117479171319E-08| 0.376727466125784E-07
19|	0.860391597237732E-09| 0.141226678880699E-06
20|	0.286797199079244E-09| 0.561751946332923E-06
21|	0.955990663597479E-10| 0.224595619560173E-05
22|	0.318663554532493E-10| 0.898347425249695E-05
23|	0.106221184844164E-10| 0.359337801666845E-04
24|	0.354070616147214E-11| 0.143735081718970E-03
25|	0.118023538715738E-11| 0.574940313893291E-03
26|	0.393411795719127E-12| 0.229976125124564E-02
27|	0.131137265239709E-12| 0.919904500354003E-02
28|	0.437124217465697E-13| 0.367961800136793E-01
29|	0.145708072488566E-13| 0.147184720054557E+00
30|	0.485693574961885E-14| 0.588738880218174E+00
31|	0.161897858320628E-14| 0.235495552087268E+01
32|	0.539659527735428E-15| 0.941982208349071E+01
33|	0.179886509245143E-15| 0.376792883339628E+02
34|	0.599621697483809E-16| 0.150717153335851E+03
35|	0.199873899161270E-16| 0.602868613343405E+03
36|	0.666246330537565E-17| 0.241147445337362E+04
37|	0.222082110179188E-17| 0.964589781349449E+04
38|	0.740273700597295E-18| 0.385835912539779E+05
39|	0.246757900199098E-18| 0.154334365015912E+06
40|	0.822526333996994E-19| 0.617337460063647E+06
41|	0.274175444665665E-19| 0.246934984025459E+07
42|	0.913918148885549E-20| 0.987739936101835E+07
43|	0.304639382961850E-20| 0.395095974440734E+08
44|	0.101546460987283E-20| 0.158038389776294E+09
45|	0.338488203290944E-21| 0.632153559105175E+09
46|	0.112829401096981E-21| 0.252861423642070E+10
47|	0.376098003656604E-22| 0.101144569456828E+11
48|	0.125366001218868E-22| 0.404578277827312E+11
49|	0.417886670729560E-23| 0.161831311130925E+12
50|	0.139295556909853E-23| 0.647325244523699E+12

Nota.- Pódese comprobar que a relación recurrente anterior tamén se pode xerar da forma $x_n = (\frac{1}{3})^n$ (ver a bibliografía recomendada).

BIBLIOGRAFÍA

HENRICI, Peter (1972): *Elementos de Análisis Numérico*, Trillas.

METCALF, Michael; REID, John K. e COHEN, Malcolm (2004): *FORTRAN 95/2003 explained*, Oxford: University Press.

VIAño Rey, Juan Manuel (1995): *Lecciones de métodos numéricos 1.- Introducción general y análisis de errores*, Tórculo edicións.

VIAño Rey, Juan Manuel (1997): *Lecciones de métodos numéricos 2.- Métodos de resolución de ecuaciones numéricas no lineales*, Tórculo edicións.

VIAño Rey, Juan Manuel e BURGUERA González, Margarita (2000): *Lecciones de métodos numéricos 3.- Interpolación*, Tórculo edicións.

Complementaria:

BURDEN, Richard L. e FAIRES, J. Douglas (cop. 2001): *Numerical Analysis* (7th edition), Brooks/Cole Thomson Learning.

ISAACSON, Eugene e KELLER, Herbert Bishop (1994): *Analysis of Numerical Methods*, John Wiley.

KINCAID, David e CHENEY, Elliot Ward (1991): *Análisis Numérico: las Matemáticas del Cálculo Científico*, John Wiley.

QUARTERONI, Alfio e SALERI, Fausto (2006): *Cálculo Científico con Matlab y Octave*, Italia, Milano: Springer-Verlag.

YOUNG, David M. e GREGORY, Robert Todd (1973): *A Survey of Numerical Mathematics*, Addison-Wesley.

unidadesdidácticas
UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Unha colección orientada a editar materiais docentes de calidade e pensada para apoiar o traballo do profesorado e do alumnado de todas as materias e titulacións da universidade

Impreso en papel 100% reciclado e libre de cloro

USC
UNIVERSIDADE DE SANTIAGO DE COMPOSTELA
SERVIZO DE NORMALIZACIÓN LINGÜÍSTICA

9 788498 879629