UNIVERSITAT DE BARCELONA
Facultat de Matemàtiques i Informàtica

GRAU DE MATEMÀTIQUES

Treball final de grau

INTERPOLACIÓ I QUADRATURA EN DUES VARIABLES

Autor: Vidal Melero Iglesias

Director: Jaume Timoneda
Realitzat a: Departament Matemàtiques i Informàtica

Barcelona, 20 de juny de 2019

Abstract

Numerical integration, also known as quadrature, belongs to the group of problems of numerical analysis. The quadrature study has been extensively developed for functions of one dimension. Although it has many applications in various areas of science, the numerical calculation of integrals in higher dimensions has not been so much worked, mainly because of the difficulties involved in the fact that there are many more integration regions than in the 1-dimensional case.

In this study it is developed the quadrature in two variables, explaining with detail some of the methods that can be used, and presenting practical results to support theoretical concepts.

Specifically, the quadrature product formulas are developed, using the results of dimension 1 to arrive at valid 2-dimensional methods. Mainly, Simpson's quadrature method and the 1-dimensional Trapezoidal rule are used to extend it to integrals on rectangular regions in the plane.

Finally, the concept of adaptive quadrature is introduced and worked, which aims to reduce the cost and improve the computational efficiency of the product formulas, taking into account the nature of the function in the different zones of the integration region.

Resum

La integració numèrica, també coneguda com a quadratura, pertany al grup de problemes de l'anàlisi numèric. L'estudi de la quadratura ha estat àmpliament desenvolupat per a funcions d'una dimensió. Tot i que presenta moltes aplicacions en diverses àrees de la ciència, el càlcul numèric d'integrals en dimensions superiors no ha estat tan treballat, a causa, principalment, de les dificultats que comporta el fet d'haver-hi moltes més regions d'integració que en el cas 1-dimensional.

En aquest estudi es desenvolupa la quadratura en dues variables, explicant amb detall alguns dels mètodes que es poden utilitzar i presentant resultats pràctics per a recolzar els conceptes teòrics.

Concretament, es desenvolupen les fórmules producte de quadratura, utilitzant els resultats de dimensió 1 per a arribar a mètodes vàlids 2-dimensionals. Principalment, s'usen el mètode de quadratura de Simpson i la regla dels Trapezis 1-dimensionals per a estendre'ls a integrals sobre regions rectangulars del pla.

Finalment, s'introdueix i es treballa amb el concepte de quadratura adaptativa, que té com a objectiu disminuir el cost i augmentar l'eficiència computacional de les fórmules producte, tenint en compte la naturalesa de la funció a les diferents zones de la regió d'integració.

Agraïments

La presentació d'aquest treball no és un fet més en el meu projecte acàdemic: presentar aquest treball té un significat especial, en tant que suposa tenir a l'abast l'assoliment d'un objectiu personal com és la finalització del meu grau. Arribar fins aquí ha estat possible gràcies al recolzament de moltes persones; a totes elles, el meu més sincer agraïment.

En primer lloc, el meu agraïment al meu tutor, en Jaume Timoneda. Sense el seu acompanyament des del primer moment, aquest treball no hauria estat possible: gràcies per la seva orientació en el gran món de les Matemàtiques, gràcies per plantejar-me reptes a cada pas, gràcies per la seva paciència durant les nostres trobades i xerrades, gràcies per les seves correccions i, especialment, gràcies pel seu testimoni educatiu més enllà de les Matemàtiques.

En segon lloc, també les gràcies a tots els meus professors i professores de Matemàtiques que, al llarg de la meva formació, de manera més evident o implícitament, han fet possible el despertar en mi d'una curiositat matemàtica que m'ha portat fins aquí.

Finalment, gràcies també a totes aquelles persones que, des de la proximitat més personal, m'han recolzat en tot moment.

A tots vosaltres, el meu agraïment.

Índex

1 Introducció

El projecte

Motivació

L'àmbit de l'anàlisi numèric i la programació m'han resultat interessants al llarg del grau. És per això que volia un tema que relacionés aquestes dues àrees. Per tant, agraeixo que el meu tutor em proposés la quadratura en dues variables, ja que he pogut treballar i ampliar conceptes treballats al grau de molt interès per a mi.

Objectius

L'objectiu d'aquest estudi és desenvolupar alguns dels mètodes de quadratura en dues variables, ampliant així la teoria treballada al grau sobre la quadratura en una variable. Especial i concretament, és interessant veure el concepte dels algorismes adaptatius, per tal d'entendre que, tot i que un programa sigui correcte, hi poden haver altres formes de resoldre'l, de manera que alguna d'aquestes pot ser més eficient.

A banda dels objectius matemàtics, el treball és una bona oportunitat per a fer recerca i saber quines fonts bibliogràfiques poden ser interessants per assolir els objectius. A més, és interessant per a aprendre a escriure un estudi matemàtic amb correcció.

Estructura de la Memòria

El treball està estructurat en 7 seccions; sent a partir de la segona on comença el cos de l'estudi.

En un primer moment, a les seccions 2 i 3 es presenta un resum dels conceptes claus de la interpolació i la quadratura en una dimensió vists al llarg del grau a Mètodes Numèrics. Aquests resultats són necessaris per a l'estudi 2-dimensional, que comença en la secció 4.

A continuació trobem les seccions 4 i 5, que aquí presentem conjuntament per la interrelació del seus continguts. L'estructura és similar a l'anterior: a la secció 4 es presenten resultats sobre la interpolació en dues variables, treballant sobre regions rectangulars, i es mostren el polinomi interpolador de Newton i de Lagrange; a la secció 5, s'introdueix la quadratura en dues variables, l'estudi de les fórmules producte i la quadratura en altres regions del plà.

Finalment, a la secció 6 es desenvolupen els algorismes adaptatius. Cal destacar l'apartat de resultats, que és una comparació de l'eficiència de les fórmules producte amb els algorismes adaptatius.

A la secció 7 es presenten les conclusions de l'estudi.

A la secció 8, l'annex, es pot consultar el codi dels algorismes adaptatius, així com algun altre programa per recolzar i presentar resultats relacionats amb la teoria.

2 Interpolació en una dimensió

Per tal de poder començar a treballar amb la interpolació en dues dimensions, és important recordar conceptes claus de la interpolació vists al llarg del grau, ja que més endavant veurem que farem ús de diversos resultats per a extendre el concepte de interpolació a dues dimensions. Per aquest motiu no s'inclouen demostracions.

Enunciat

Donades unes dades (x_1, y_1),..., (x_n, y_n), on $x_i \neq x_j$ si $i \neq j$. Els valors y_j poden ser les imatges de x_j per una funció coneguda f: [a,b] $\subset \mathbb{R} \longrightarrow \mathbb{R}$. En qualsevol cas, volem trobar una funció g complint $g(x_i) = y_i$.

2.1 Interpolació mitjançant polinomis

Una manera habitual de tractar les dades (x_1, y_1),..., (x_n, y_n) esmentades és mitjançant funcions g d'espais de funcions amb propietats conegudes, que siguin còmodes de treballar. Un exemple d'aquests espais de funcions són els polinomis. Per a dimensió 1 tenim existència i unicitat, i anomenarem el polinomi únic que interpola les dades polinomi interpolador.

És a dir, donats $x_1, ..., x_{n+1}$ punts diferents dos a dos. Per a valors arbitraris $y_1, ..., y_{n+1}$, $\exists!$ polinomi de grau $\leq$ n, p, que s'anomena polinomi interpolador, tal que $p(x_i) = y_i$ $\forall i = 1, ..., n+1$.

2.2 Error en la interpolació polinomial

Siguin $x_1, ..., x_{n+1} \in \mathbb{R}$.

Sigui f una funció amb $n+1$ derivades contínues en l'interval tancat més petit que conté $x, x_1, ..., x_{n+1}$, que anomenarem I. Si p és el polinomi interpolador, aleshores:

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_1) \cdots (x - x_{n+1}), \text{ on } \xi = \xi(x) \in I$$

En el cas de tenir punts equiespaiats a interpolar, que és un cas comú en la interpolació, l'error es pot estimar. Siguin $x_0, ..., x_n$ punts en l'interval $[a, b] \in \mathbb{R}$. Sigui $h = \dfrac{b-a}{n}$, de manera que $x_i = a + ih$.

Teorema: Si $f \in C^{n+1}$, per $x \in [a, b]$, l'error de truncament es pot estimar:

$$\left|f(x) - p(x)\right| \leq \frac{h^{n+1}}{4(n+1)} \max_{y \in [a,b]} \left|f^{(n+1)}(y)\right|$$

2.3 Mètodes per calcular el polinomi interpolador

Per tal de calcular el polinomi interpolador que resol el problema de la interpolació, s'han desenvolupat diversos mètodes. En aquesta secció recordarem els mètodes de Newton i de Lagrange. Aquesta secció és molt rellevant, doncs aquests mètodes seran utilitzats en la interpolació en dues variables, tal i com es veu en els apartats 4.2 i 4.6.

2.3.1 Mètode de Newton

Donades unes dades (x_1, y_1),..., (x_{n+1}, y_{n+1}).

Observació:

Sigui $p(x)$ el polinomi interpolador d'aquests punts (x_i, y_i). Si escrivim:

$p(x) = c_0 + c_1(x - x_1) + ... + c_n(x - x_1) \cdots (x - x_n) \Rightarrow c_k = f[x_1, ..., x_{k+1}]$

Definició

$$f[x_1, ..., x_{k+1}] = \frac{f[x_2, ..., x_{k+1}] - f[x_1, ..., x_k]}{x_{k+1} - x_1}$$

Teorema:

Si f és n vegades derivable amb derivades contínues en l'interval I tancat més petit que conté $x_1, ..., x_{n+1}$, aleshores $\exists \xi \in$ I tal que:

$$f[x_1, ..., x_{n+1}] = \frac{f^n(\xi)}{n!}$$

2.3.2 Mètode de Lagrange

Donades unes dades (x_1, y_1),..., (x_{n+1}, y_{n+1}).

Definició:

Es defineix el polinomi interpolador de Lagrange com: $p_n(x) =: \sum_{j=1}^{n+1} y_j l_j(x)$, on:

$$l_j(x) = \prod_{i \neq j} \frac{(x - x_i)}{(x_j - x_i)}$$

Observem que:

$$l_j(x_i) = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}$$

3 Quadratura en una dimensió

3.1 Enunciat

El concepte de quadratura fa referència a la integració numèrica. Veiem-n'he l'enunciat del problema:

Donada f: [a,b] $\subset \mathbb{R} \longrightarrow \mathbb{R}$, volem calcular $\int_a^b f(x)dx$. Aproximem f mitjançant el polinomi interpolador p en punts $x_0, ..., x_n$, i.e:

$$x_i = a + ih, \text{ on } h = \frac{b-a}{n}$$

Per tant, es pot esperar que:

$$\int_a^b f(x)dx \approx \int_a^b p(x)dx$$

3.2 Error de truncament

Definim l'error de truncament com:

$$R_T =: \int_a^b f(x)dx - \int_a^b p(x)dx = \int_a^b (f(x) - p(x))dx$$

Utilitzant el resultat de l'error de la interpolació vist a la secció 2.2, resulta que:

$$R_T = \int_a^b \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0) \cdots (x - x_n)dx$$

on $\xi(x) \in (a, b)$.

3.3 Fórmules de Newton-Cotes

Les fórmules de Newton-Cotes són una família de fórmules per a la quadratura que es basen en avaluar l'integrand a $n + 1$ punts equiespaiats.

N'hi ha de dos tipus: tancades, que fan servir el valor de la funció a tots els punts, i obertes, que no fan servir el valor de la funció als extrems. L'expressió general de les fórmules tancades és:

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} w_i f(x_i),$$

on:

$$x_i = ih + x_0,\ h = \frac{(b-a)}{n}$$

En les següents seccions veurem exemples de fórmules tancades.

3.3.1 Fórmules simples

Regla del trapezi

Considerem el cas simple en què interpolem en x_0 i x_1. És a dir, $x_0 = a, x_1 = b$ i, per tant, $h = b - a$.

Interpolant, $p_1(x) = f_0 + \dfrac{f_1 - f_0}{h}(x - x_0)$. Aleshores:

$$\boxed{\int_a^b f(x)dx = \frac{h}{2}(f_0 + f_1) + R_T}$$

Usant l'apartat 3.2, s'obté l'error en la integració i, de fet, es pot demostrar que:

$$\boxed{R_T = -\frac{1}{12}f''(\varsigma)h^3, \text{ on } \varsigma \in (a, b).}$$

Regla de Simpson

Suposem que volem solucionar el problema d'integració numèrica amb els nodes inicials x_0, x_1, x_2 i els seus respectius valors per f, f_0, f_1, f_2. Observem que, per tant, $h = \dfrac{b-a}{2}$. Aleshores:

$$\boxed{\int_a^b f(x)dx = \frac{h}{3}(f_0 + 4f_1 + f_2) + R_T}$$

on:

$$\boxed{R_T = -\frac{h^5}{90}f^{(4)}(\varsigma) \text{ , on } \varsigma \in (x_0, x_2).}$$

3.3.2 Fórmules compostes

Idea

Es tracta de realitzar una partició de l'interval inicial $[a, b]$ en $n+1$ punts $a, x_1, ..., x_{n-1}, b$. Per tant, $h = \dfrac{(b-a)}{n}$. De manera que:

$$\int_a^b f(x)dx = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} f(x)dx, \text{ on } x_0 = a, x_n = b \text{ i } x_{i+1} - x_i = h.$$

Ara doncs, ja podem aplicar les fórmules simples a cada un dels termes del sumatori, per tal d'obtenir l'aproximació a la integral.

Regla composta dels trapezis

Apliquem la fórmula dels trapezis a cadascun dels termes del sumatori. Es pot demostrar que:

$$\boxed{\int_a^b f(x)dx = T(h) - \frac{b-a}{12}h^2 f"(\varsigma), \text{ on } \varsigma \in [a,b].}$$

on $T(h) = h(\frac{1}{2}f_0 + f_1 + ... + f_{n-1} + \frac{1}{2}f_n)$.

Regla de Simpson composta

Sigui $n = 2m$ el nombre de subintervals en l'interval [a,b]. Apliquem la regla de Simpson simple a les integrals $\int_{x_{2i}}^{x_{2i+2}} f(x)dx$, de manera que s'obté:

$$\boxed{\int_a^b f(x)dx = S(h) - \frac{b-a}{180}h^4 f^{(4)}(\varsigma), \text{ on } \varsigma \in [a,b].}$$

on $S(h) = \frac{h}{3}(f_0 + 4\sum_{i=0}^{m-1} f_{2i+1} + 2\sum_{i=1}^{m-1} f_{2i} + f_{2m})$.

3.4 Algorismes adaptatius en una dimensió

Les regles compostes de quadratura divideixen l'interval inicial $I = [a,b]$ en n subintervals i, en cada un d'ells, s'aproxima la integral de la funció utilitzant una regla de quadratura. Aquest mètode no té en compte la naturalesa de la funció, ja que sol passar que certs intervals generen un error més petit on la funció és més plana i, per tant, no calen subdivisions tan acurades. És per aquest motiu que s'introdueix el concepte de quadratura adaptaiva, que precisament té com a objectiu solucionar aquest problema. Aquests algorismes segueixen el següent procediment:

1) Càlcul numèric de $S = \int_a^b f(x)dx$ mitjançant una regla de quadratura.

2) Divisió de I en dos subintervals $I_1 = [a, \frac{a+b}{2}]$, $I_2 = [\frac{a+b}{2}, b]$ i càlcul de $S_i = \int_{I_i} f(x)dx$ mitjançant una regla de quadratura.

3) Comparació de $S_1 + S_2$ amb S. Si el resultat és prou acurat, el donem per vàlid. Si no ho és, tornem al pas 1) per a cadascun dels intervals I_i.

Tot i que en una dimensió no veurem amb detall l'algorisme, veiem el resultat d'utilitzar Trapezis adaptatius en una dimensió per a la funció $f(x) = e^x \sin(x)$ per tal de mostrar gràficament l'objectiu de l'algorisme. Els resultats s'han pres per $I = [-1,4]$ amb un error demanat de $\epsilon = 10^{-3}$.

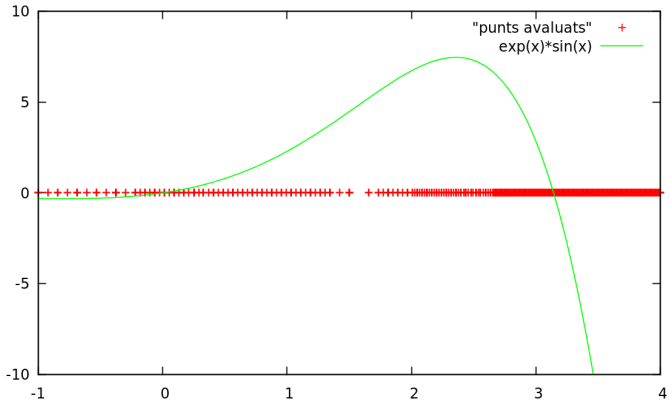

Figura 1: Punts avaluats

En aquest gràfic es pot observar la funció $f(x) = e^x \sin(x)$ en verd, i els punts on s'ha avaluat la funció durant l'execució del programa en color vermell.

Notem que, en les zones on la funció és més plana, s'han requerit menys avaluacions que en les zones on la funció té pendent elevat.

4 Interpolació en dues variables

4.1 Introducció

Un cop vists els conceptes i alguns dels mètodes utilitzats en la interpolació en una dimensió, podem considerar el cas 2-dimensional. Un dels principals problemes i motius pel qual la teoria sobre la interpolació en dimensions superiors no està tan desenvolupada com la interpolació 1-dimensional és la falta d'unicitat. És a dir, a diferència del cas de dimensió 1, on hi ha la unicitat del polinomi interpolador, en dimensions superiors, per norma general, no hi ha un sol polinomi interpolador.

Per exemple, suposem que tenim un conjunt de punts p_{ij}:(x_i, y_j) en $\mathbb{R}^2$. Tenim una funció $f : \mathbb{R}^2 \longrightarrow \mathbb{R}$, i volem interpolar f. Imaginem que tots els punts $(p_{ij}, f(p_{ij})) \in \mathbb{R}^3$ es troben en una recta a $\mathbb{R}^3$ (veure figura 2), aleshores existeixen infinits plans que contenen la recta (veure figura 3) i, per tant, no tenim unicitat.

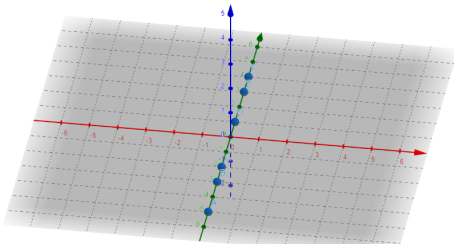

Figura 2: Punts en l'eix y

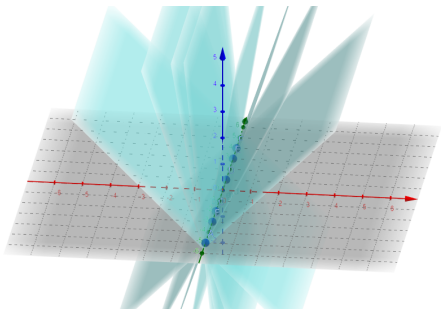

Figura 3: Plans z= $\lambda x, \lambda \in \mathbb{R}$

A la figura 2 veiem l'existència d'infinits plans {z= $\lambda x, \lambda \in \mathbb{R}$}, és a dir, infinits polinomis P(x,y)=λx que interpolen f en els punts de la figura 2.

Ens restringirem a interpolar en rectangles, i.e, en regions senzilles, de manera que podrem demostrar la unicitat i veure resultats concrets.

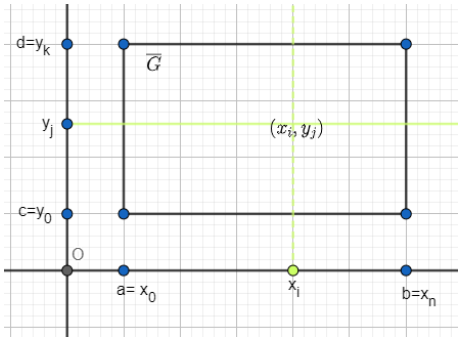

Figura 4: Regió G

$G = \{(x, y) \in \mathbb{R}^2 |\ a \leq x \leq b,\ c \leq y \leq d\}$

Agafarem $n+1$ puns en $[a, b]$ i $k+1$ punts en $[c, d]$. Això dóna un total de $(n+1)(k+1)$ punts. L'objectiu és, donats uns valors f_{ij}, trobar un polinomi $p \in P_{nk}$ tal que:

$p(x_i, y_j) = f_{ij}\ \forall\ i = 0, ..., n, j = 0, ..., k.$

Cal remarcar que, similarment al cas de dimensió 1, els valors f_{ij} poden correspondre a les imatges de (x_i, y_j) per una funció coneguda $f : [a, b] \times [c, d] \longrightarrow \mathbb{R}$.

Usarem molts resultats explicats en la secció 2.

4.2 Existència del polinomi interpolador

Usarem el polinomi interpolador de Lagrange per a una dimensió introduït a 2.3.2.

Notació 1. $l_{ns}(x) = \prod_{\nu=0,\nu\neq s}^{n} \frac{x - x_\nu}{x_s - x_\nu}$

Primer de tot calculem els polinomis de grau n $l_{ni}(x)\ \forall$ i $\in \{0, ..., n\}$. Calculem també els polinomis $l_{kj}(y)$ de grau $k\ \forall$ j $\in \{0, ..., k\}$.

Definim:

$l_{ij}^{nk}(x, y) = l_{ni}(x) l_{kj}(y)$

Finalment, definim el polinomi:

$$\boxed{p(x, y) = \sum_{\substack{0\leq i\leq n \\ 0\leq j\leq k}} f_{ij} l_{ij}^{nk}(x, y)}$$

Veiem que, efectivament, p compleix la condició d'interpolació.

Cal comprovar que, donat x_m i y_u qualsevol, amb $m \in \{0, ..., n\}$ i $u \in \{0, ..., k\}$, es compleix que $p(x_m, y_u) = f_{mu}$

$$p(x_m, y_u) = \sum_{\substack{0 \leq i \leq n \\ 0 \leq j \leq k}} f_{ij} l_{ij}^{nk}(x_m, y_u)$$

Per casos:

- Si $i \neq m$

$$l_{ij}^{nk}(x_m, y_u) = l_{ni}(x_m) l_{kj}(y_u) = \prod_{\nu=0, \nu \neq i}^{n} \frac{x_m - x_\nu}{x_i - x_\nu} \cdot l_{kj}(y_u) = 0$$

- Si $j \neq u$

$$l_{ij}^{nk}(x_m, y_u) = l_{ni}(x_m) l_{kj}(y_u) = l_{ni}(x_m) \cdot \prod_{\nu=0, \nu \neq j}^{k} \frac{y_u - y_\nu}{y_j - y_\nu} = 0$$

- Si $i = m, j = u$

$$l_{mu}^{nk}(x_m, y_u) = l_{nm}(x_m) l_{ku}(y_u) = \prod_{\nu=0, \nu \neq m}^{n} \frac{x_m - x_\nu}{x_m - x_\nu} \cdot \prod_{\nu=0, \nu \neq u}^{k} \frac{y_u - y_\nu}{y_u - y_\nu} = 1$$

Per tant, $p(x_m, y_u) = f_{mu}$.

A més, notem que $p(x, y)$ és un polinomi de grau com a molt n en x i com a molt k en y.

4.3 Unicitat del polinomi interpolador

4.3.1 Matriu de VanderMonde

Per tal de demostrar la unicitat del polinomi interpolador, cal conèixer la matriu de VanderMonde, així com alguna de les seves propietats.

Definició. Diem que V matriu $m \times n$ és una matriu de VanderMonde si és de la forma:

$$\begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & . & . & . & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & . & . & . & \alpha_2^{n-1} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 1 & \alpha_m & \alpha_m^2 & . & . & . & \alpha_m^{n-1} \end{pmatrix}$$

Proposició. Si V_n és matriu de VanderMonde $n \times n$ i $\alpha_i \neq \alpha_j \ \forall i \neq j \Rightarrow \det(V_n) \neq 0$.

Demostració.

$$\begin{vmatrix} 1 & \alpha_1 & \alpha_1^2 & . & . & . & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & . & . & . & \alpha_2^{n-1} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 1 & \alpha_n & \alpha_n^2 & . & . & . & \alpha_n^{n-1} \end{vmatrix}$$

Restant la fila 1 de la fila j, el determinant és equivalent a:

$$\begin{vmatrix} 1 & \alpha_1 & \alpha_1^2 & . & . & . & \alpha_1^{n-1} \\ 0 & \alpha_2 - \alpha_1 & \alpha_2^2 - \alpha_1^2 & . & . & . & \alpha_2^{n-1} - \alpha_1^{n-1} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & \alpha_n - \alpha_1 & \alpha_n^2 - \alpha_1^2 & . & . & . & \alpha_n^{n-1} - \alpha_1^{n-1} \end{vmatrix}$$

Ara, sense canviar el valor del determinant, fem, en ordre:

- Columna n= (columna n) - (columna n-1) α_1 .
- Columna n-1= (columna n-1) - (columna n-2) α_1.
  El primer pas:

$$\begin{vmatrix} 1 & \alpha_1 & \alpha_1^2 & . & . & . & \alpha_1^{n-1} - \alpha_1^{n-2}\alpha_1 \\ 0 & \alpha_2 - \alpha_1 & \alpha_2^2 - \alpha_1^2 & . & . & . & (\alpha_2^{n-1} - \alpha_1^{n-1}) - (\alpha_2^{n-2} - \alpha_1^{n-2})\alpha_1 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & \alpha_n - \alpha_1 & \alpha_n^2 - \alpha_1^2 & . & . & . & (\alpha_n^{n-1} - \alpha_1^{n-1}) - (\alpha_n^{n-2} - \alpha_1^{n-2})\alpha_1 \end{vmatrix}$$

$$= \begin{vmatrix} 1 & \alpha_1 & \alpha_1^2 & . & . & . & 0 \\ 0 & \alpha_2 - \alpha_1 & \alpha_2^2 - \alpha_1^2 & . & . & . & (\alpha_2 - \alpha_1)\alpha_2^{n-2} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & \alpha_n - \alpha_1 & \alpha_n^2 - \alpha_1^2 & . & . & . & (\alpha_n - \alpha_1)\alpha_n^{n-2} \end{vmatrix}$$

- Columna j= (columna j) - (columna $j-1$) α_1.
- Finalment, columna 2= (columna 2) - (columna 1) α_1.

El resultat és:

$$\begin{vmatrix} 1 & 0 & 0 & . & . & . & 0 & 0 \\ 0 & \alpha_2-\alpha_1 & (\alpha_2-\alpha_1)\alpha_2 & . & . & . & (\alpha_2-\alpha_1)\alpha_2^{n-3} & (\alpha_2-\alpha_1)\alpha_2^{n-2} \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ 0 & \alpha_n-\alpha_1 & (\alpha_n-\alpha_1)\alpha_n & . & . & . & (\alpha_n-\alpha_1)\alpha_n^{n-3} & (\alpha_n-\alpha_1)\alpha_n^{n-2} \end{vmatrix}$$

Observem que, excepte la primera fila, la k-èssima fila té $\alpha_k - \alpha_1$ com a factor comú. Per tant, el determinant és equivalent a:

$$\det(V_n) = \prod_{k=2}^{n}(\alpha_k-\alpha_1)\begin{vmatrix} 1 & 0 & . & . & . & . & 0 \\ 0 & 1 & \alpha_2 & . & . & . & \alpha_2^{n-2} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & 1 & \alpha_n & . & . & . & \alpha_n^{n-2} \end{vmatrix} = \prod_{k=2}^{n}(\alpha_k-\alpha_1)\begin{vmatrix} 1 & \alpha_2 & . & . & . & \alpha_2^{n-2} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 1 & \alpha_n & . & . & . & \alpha_n^{n-2} \end{vmatrix}$$

Observem, doncs, que el resultat ha estat: $\det(V_n) = \prod_{k=2}^{n}(\alpha_k-\alpha_1)\det V_{n-1}$. Seguint el procés es demostra que:

$$\boxed{\det(V_n) = \prod_{1\le i<j\le n}(\alpha_j-\alpha_i)}$$

Es dedueix que, aleshores, el determinant és diferent de zero si els elements α_i són diferents dos a dos.

4.3.2 Unicitat del polinomi interpolador

Teorema. *Existeix un únic polinomi $\hat{p} \in P_{n,k}$ de grau com a màxim k en y i de grau com a màxim n en x que satisfà les $(n+1)(k+1)$ condicions d'interpolació.*

La demostració deriva de la unicitat de la interpolació en una dimensió.

Sigui el polinomi interpolador $q(x,y) = \sum_{\nu=0}^{n}\sum_{j=0}^{k} a_{\nu j}x^{\nu}y^{j}$

Per a tot $i = 0, ..., n$: $q_i(y) := q(x_i, y) = \sum_{j=0}^{k}(\sum_{\nu=0}^{n} a_{\nu j}x_i^{\nu})y^j = \sum_{j=0}^{k} b_{ij}y^j$

Per tant, q_i són polinomis d'una variable. A més, $\forall i \in \{0, ...n\}$, q_i interpola els valors f_{ij}, $j = 0, ..., k$. Podem utilitar el resultat d'unicitat del polinomi interpolador de dimensió 1, de manera que els polinomis $q_i(y)$ són únics. Finalment, d'aquí es conclou que els coeficients $b_{ij} = \sum_{\nu=0}^{n} a_{\nu j}x_i^{\nu}$ són únics.

Falta demostrar que els coeficients $a_{\nu j}$ són únics.

Per a cada $0 \leq j \leq k$: $\sum_{\nu=0}^{n} a_{\nu j} x_i^{\nu} = b_{ij}$, $\forall\ 0 \leq i \leq n$

Per exemple, per $j = 0$ tenim:

$$\begin{bmatrix} b_{00} \\ b_{10} \\ . \\ . \\ . \\ b_{n0} \end{bmatrix} = \begin{bmatrix} x_0^0 & . & . & . & . & x_0^n \\ x_1^0 & . & . & . & . & x_1^n \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ x_n^0 & . & . & . & . & x_n^n \end{bmatrix} \times \begin{bmatrix} a_{00} \\ a_{10} \\ . \\ . \\ . \\ a_{n0} \end{bmatrix}$$

Per tant tenim un sistema amb una matriu quadrada de VanderMonde, on els coeficients x_i són diferents dos a dos. Utilitzant el resultat vist a l'apartat 4.3.1 tenim que la matriu té determinant diferent de zero. Aleshores el sistema té solució única, i.e, els coeficients $a_{\nu j}$ són únics. Finalment, se'n dedueix que el polinomi interpolador $q(x, y)$ és únic.

4.4 Exemple

Imaginem que rebem una sèrie de punts (x, y, z). Utilitzem el polinomi interpolador de Lagrange vist a la secció 4.2. En aquest exemple he usat la funció $f(x, y) = \sin(x + y)$ per a obtenir els valors z.

x_i \ y_j	2	3	6	8	-1	-3	-2
2	-0.756802495	-0.958924275	0.989358247	-0.544021111	0.841470985	-0.841470985	0
3	-0.958924275	-0.279415498	0.412118485	-0.999990207	0.909297427	0	0.841470985
4	-0.279415498	0.656986599	-0.544021111	-0.536572918	0.141120008	0.841470985	0.909297427
5	0.656986599	0.989358247	-0.999990207	0.420167037	-0.756802495	0.909297427	0.141120008
6	0.989358247	0.412118485	-0.536572918	0.990607356	-0.958924275	0.141120008	-0.756802495
7	0.412118485	-0.544021111	0.420167037	0.65028784	-0.279415498	-0.756802495	-0.958924275
8	-0.544021111	-0.999990207	0.990607356	-0.287903317	0.656986599	-0.958924275	-0.279415498

A l'annex es pot consultar el codi que evalua el polinomi interpolador de Lagrange en un punt (x, y) després de llegir aquesta taula. Observem ara una taula de resultats, donant punts (x, y) arbitraris i comparant el valor de $\sin(x + y)$ amb el valor interpolat al programa.

Punt (x, y)	Valor interpolat	$\sin(x + y)$	error
$(2.5, 3)$	-0.703332	-0.70554	$2.208298 \cdot 10^{-3}$
$(4.5, 6)$	-0.87981	-0.8797	$1.214028 \cdot 10^{-4}$
$(2.1, 3.1)$	-0.893607	-0.883	$1.015272 \cdot 10^{-2}$
$(-1, 2)$	-23.07	0.84147	23.9159

Taula 1: Valors de $f(x, y) = \sin(x + y)$

Observem que, en l'últim cas, com que hem donat una cordenada x prou distant de la

resta de coordenades x_i, el valor de sortida té un error gran respecte del valor $\sin(1)$. De fet, el valor interpolat no pertany a $[-1,1]$.

4.5 Error en la interpolació

Obtindrem l'error a partir de l'error en la interpolació d'una variable explicat a la secció 2.2.

Definim:

- $(P_y f)(x,y) = \sum_{j=0}^{k} f(x,y_j) l_{kj}(y)$
- $(P_x f)(x,y) = \sum_{\nu=0}^{n} f(x_\nu, y) l_{n\nu}(x)$

Fixat $x \in [a,b]$, $P_y f$ és el polinomi que interpola $f(x,y)$ en els punts $y_0, ..., y_k$. Succeeix el mateix a $P_x f$ per a $y \in [c,d]$ fixat.

Recordem que el polinomi que interpola al conjunt de punts (x_i, y_j) de G és $\hat{p}$, que es pot expressar com la composició dels dos polinomis que acabem de definir. És a dir:

(Pf)(x,y):=$P_x(P_y f)(x,y) = \hat{p}(x,y)$.

Si $f \in C^{n+k+2}(G)$, aleshores:

Sigui $\|\cdot\|$ la norma subinfinit. Recordem que $\|f\|$=sup$\big|f(x)\big|$.

$$\|f - Pf\| \leq \|f - P_y f\| + \|P_y f - Pf\|$$

Acotem els dos termes de l'expressió. Donat que $P_y f$ és el polinomi interpolador de f en els punts $y_0, ..., y_k$, aleshores podem aplicar la fórmula de l'error en una variable, de manera que obtenim:

$$\|f - P_y\| \leq \frac{1}{4(k+1)} \left\| \frac{\partial^{k+1} f}{\partial y^{k+1}} \right\| h_y^{k+1}, \text{ on } h_y := \max_{0 \leq j \leq k-1} |y_{j+1} - y_j|.$$

Falta acotar $\|P_y f - Pf\|$.

$\|P_y f - Pf\| = \|P_y f - P_x(P_y f)\|$, i, tenim l'expressió de l'error del polinomi interpolador P_x interpolant $P_y f$. Per tant:

$$\|P_y f - Pf\| \leq \frac{1}{4(n+1)} \left\| \frac{\partial^{n+1}(P_y f)}{\partial x^{n+1}} \right\| h_x^{n+1}, \text{ on } h_x := \max_{0 \leq \nu \leq n-1} |x_{\nu+1} - x_\nu|.$$

Observem que:

$$\frac{\partial^{n+1}(P_y f)}{\partial x^{n+1}} = \frac{\partial^{n+1}\Big(\sum_{j=0}^{k} f(x,y_j) l_{kj}(y)\Big)}{\partial x^{n+1}} = P_y\Big(\frac{\partial^{n+1}(f)}{\partial x^{n+1}}\Big)$$

Aquest últim terme és el polinomi que interpola $\dfrac{\partial^{n+1}(f)}{\partial x^{n+1}}$ com a funció de y, i aleshores:

$$\left\| \frac{\partial^{n+1}(f)}{\partial x^{n+1}} - P_y\Big(\frac{\partial^{n+1}(f)}{\partial x^{n+1}}\Big) \right\| \leq \frac{1}{4(k+1)} \left\| \frac{\partial^{n+k+2}(f)}{\partial x^{n+1}\partial y^{k+1}} \right\| h_y^{k+1}$$

Finalment, doncs:

$$\left\| \frac{\partial^{n+1}(P_y f)}{\partial x^{n+1}} \right\| \leq \left\| \frac{\partial^{n+1}(f)}{\partial x^{n+1}} \right\| + \frac{1}{4(k+1)} \left\| \frac{\partial^{n+k+2}(f)}{\partial x^{n+1}\partial y^{k+1}} \right\| h_y^{k+1}$$

L'expressió final de la cota de l'error és:

$$\|f - Pf\| \leq \frac{1}{4(k+1)} \left\| \frac{\partial^{k+1} f}{\partial y^{k+1}} \right\| h_y^{k+1} + \frac{1}{4(n+1)} \left\| \frac{\partial^{n+1}(f)}{\partial x^{n+1}} \right\| h_x^{n+1} +$$

$$+ \frac{1}{16(k+1)(n+1)} \left\| \frac{\partial^{n+k+2}(f)}{\partial x^{n+1}\partial y^{k+1}} \right\| h_x^{n+1} h_y^{k+1}$$

4.6 Polinomi interpolador mitjançant el polinomi de Newton

Fins ara hem vist el polinomi interpolador per a demostrar la seva l'existència fent ús del polinomi interpolador de Lagrange d'una dimensió. Veiem ara un altre mètode. Es tracta d'utilitzar el polinomi interpolador de Newton en una variable. Recordem que seguim treballant en la regió tancada G.

Primer de tot considerem que y és fixa, de manera que podem interpolar pels $n+1$ punts x_i.

El polinomi resultant seria:

(1) $p(x,y) = \sum_{i=0}^{n} w_{i-1}(x) f[x_0, x_1, ..., x_i; y]$, on $w_1(x) = 1$, $w_i(x) = w_{i-1}(x)(x - x_i)$, $i = 0, 1, ...;$

Ara falta determinar les funcions $f[x_0, x_1, ..., x_i; y]$. Ens les mirem com a funcions de y, de manera que podem interpolar en els punts $y_0, ..., y_k$ mitjançant les diferències dividides de Newton per a una variable.

(2) $f[x_0, x_1, ..., x_i; y] = \sum_{j=0}^{k} w_{j-1}(y) f[x_0, x_1, ..., x_i; y_0, y_1, ..., y_j]$.

Finalment, substituïnt (2) en (1), podem trobar l'expressió final del polinomi interpolador.

Veiem el mètode per al cas en que tenim quatre punts (x_0, y_0), (x_0, y_1), (x_1, y_0), (x_1, y_1).

$$\text{p(x,y)} = f[x_0; y] + f[x_0, x_1; y](x - x_0) = f(x_0, y) + \frac{f(x_1, y) - f(x_0, y)}{x_1 - x_0}(x - x_0)$$

$$f(x_0, y) \simeq f(x_0, y_0) + f[x_0; y_0, y_1](y - y_0) = f(x_0, y_0) + \frac{f(x_0, y_1) - f(x_0, y_0)}{y_1 - y_0}(y - y_0)$$

D'altra banda:

$$f[x_0, x_1; y] \simeq \frac{f(x_1, y_0) - f(x_0, y_0)}{x_1 - x_0} + \frac{\frac{f(x_1,y_1)-f(x_0,y_1)}{x_1-x_0} - \frac{f(x_1,y_0)-f(x_0,y_0)}{x_1-x_0}}{y_1 - y_0}(y - y_0)$$

Finalment:

$$p(x, y) = f(x_0, y_0) + \frac{f(x_0, y_1) - f(x_0, y_0)}{y_1 - y_0}(y - y_0) + \frac{f(x_1, y_0) - f(x_0, y_0)}{x_1 - x_0}(x - x_0) +$$

$$+ \frac{\frac{f(x_1, y_1) - f(x_0, y_1)}{x_1 - x_0} - \frac{f(x_1, y_0) - f(x_0, y_0)}{x_1 - x_0}}{y_1 - y_0}(y - y_0)(x - x_0)$$

Es pot comprovar que, efectivament, es compleixen les condicions d'interpolació, i.e:

$p(x_0, y_0) = f(x_0, y_0)$, $p(x_0, y_1) = f(x_0, y_1)$, $p(x_1, y_0) = f(x_1, y_0)$ i $p(x_1, y_1) = f(x_1, y_1)$.

Trobem ara una expressió per a l'error:

Recordem que l'expressió del polinomi interpolador de Newton en una variable en els punts $x_0, ..., x_n$ és:

$p(x) = c_0 + c_1(x - x_0) + ... + c_n(x - x_1) \cdots (x - x_{n-1})$, on $c_k = f[x_0, ..., x_k]$.

Observem que, si interpolem en un punt més x_{n+1}, només caldria afegir un terme més a $p(x)$. De manera que:

$$\hat{p}(x) = f(x_0) + f[x_0, x_1](x - x_0) + ... + f[x_0, ..., x_{n+1}](x - x_0) \cdots (x - x_n)$$

Però, si $x_{n+1} = x$, aleshores podem trobar una nova expressió per a l'error:

$$\boxed{f(x) - p(x) = [\prod_{j=0}^{n}(x - x_j)]f[x_0, x_1, ..., x_n, x]}$$

Per tant, tornant al cas bidimensional, només hauríem d'afegir aquest terme corresponent a l'error per a cada pas que interpolem, concretament en (1) i (2). D'aquesta manera trobaríem l'expressió exacta per a $f(x, y)$.

Concretament, en (1) ens surt un terme d'error corresponent a fixar la y iinterpolar respecte els $n + 1$ punts x_i. En (2) ens surt un terme d'error per a cada $f[x_0, x_1, ..., x_i; y]$ corresponent a fixar x i interpolar respecte els $k + 1$ punts y_j.

5 Quadratura en dues variables

5.1 Introducció

Donada $f : [a,b] \times [c,d] \subset \mathbb{R}^2 \longrightarrow \mathbb{R}$, volem calcular $\int_a^b \int_c^d f(x,y)dxdy$. Similarment al cas de dimensió 1 explicat en el l'apartat 3, si trobem el polinomi interpolador $p(x,y)$, aleshores es pot esperar que:

$$\int_a^b \int_c^d f(x,y)dxdy \simeq \int_a^b \int_c^d p(x,y)dxdy$$

5.1.1 Principals inconvenients

Tal i com succeeix amb la interpolació en dimensions superiors, la integració numèrica no ha estat tan desenvolupada en dimensions superiors a 1. Un dels principals problemes és, a diferència del cas 1-dimensional, la varietat de dominis d'integració. És a dir, en el cas 1-dimensional existeixen tres casos pel que respecta als intervals d'integració: intervals finits, (a,b), intervals $(-\infty,a)$ o (a,∞) i $(-\infty,\infty)$. D'altra banda, en el cas multidimensional tenim moltes més regions d'integració. Per tant, seguint amb l'esquema i donada la complexitat esmentada, integrarem en rectangles de $\mathbb{R}^2$. És a dir, continuem treballant en la regió $G = \{(x,y) \in \mathbb{R}^2 |\ a \leq x \leq b,\ c \leq y \leq d\}$.

5.1.2 Fórmules producte

Cas general Suposem que tenim una regió B en l'espai euclidià r-dimensional amb coordenades $(x_1,...,x_r)$ i una regió U en l'espai euclidià s-dimensional amb coordenades $(y_1,...,y_s)$. Suposem també que tenim una regla de quadratura R sobre B i una regla de quadratura S sobre U.

$$R(f) = \sum_{j=1}^{m} w_j f(x_j) \approx \int_B f(x)dx, \text{ on } x_j \in B$$
$$S(f) = \sum_{k=1}^{n} v_k f(y_k) \approx \int_U g(y)dy, \text{ on } y_k \in G$$

Aleshores podem construir una regla d'integració a $B \times U$:

$$(R \times S)(f) = \sum_{j=1}^{m}\sum_{k=1}^{n} w_j v_k h(x_j,y_k) \approx \int_{B\times U} f(x,y)dxdy$$

De fet:

Teorema. Si R integra $f(x)$ exactament sobre B i S integra $g(y)$ exactament sobre U. Llavors $R \times S$ integra exactament $h(x,y) = f(x)g(y)$ sobre $B \times U$.

Dem. $\int_{B\times U} h(x,y)dxdy = \int_{B\times U} f(x)g(y)dxdy = \int_B f(x)dx \int_U g(y)dy =$

$$= \sum_{j=1}^{m} w_j f(x_j) \sum_{k=1}^{n} v_k g(y_k) = \sum_{j=1}^{m} \sum_{k=1}^{n} w_j v_k h(x_j, y_k) = (R \times S)(h)$$

Cas bidimensional Tal i com acabem de veure, una regla producte s'escriu com el producte de dues regles de cuadratura. Sigui $G = [a, b] \times [c, d]$. És a dir, estem en el cas particular del cas general en que $B = [a, b] \subset \mathbb{R}$, $U = [a, b] \subset \mathbb{R}$. Siguin R i S dues regles de cuadratura a $[a, b]$ i $[c, d]$, respectivament.

Per tant, donada la funció h:$[a, b] \times [c, d] \subset \mathbb{R}^2 \longrightarrow \mathbb{R}$ integrable, aplicant el resultat del cas general obtenim un regla d'integració a G:

$$(R \times S)(h) := \sum_{i=0}^{n} \sum_{j=0}^{k} w_i v_j h(x_i, y_j)$$

Aquest resultat es pot interpretar també com, per a cada $x \in [a, b]$ fixada, definim $\phi_x : [c, d] \longrightarrow \mathbb{R}$ com $\phi_x(y) := h(x, y)$. Observem que es tracta d'una funció integrable.

Definim $\psi : [a, b] \longrightarrow \mathbb{R}$ com $\psi(x) = S(\phi_x) = \sum_{j=0}^{k} v_j h(x, y_j)$.

Finalment:

$$R(\psi) = \sum_{i=0}^{n} w_i (\sum_{j=0}^{k} v_j h(x_i, y_j)) = \sum_{i=0}^{n} \sum_{j=0}^{k} w_i v_j h(x_i, y_j) = (R \times S)(h)$$

Trapezis producte compost Sigui $G = [a, b] \times [c, d]$ i sigui $\{x_0, ..., x_n\}$ una partició de l'interval $[a, b]$, de tal manera que $x_i = a + ih_n \ \forall i \in \{0, ..., n\}$, on $h_n = \dfrac{b-a}{n}$. Sigui $\{y_0, ..., y_k\}$ una partició de l'interval $[c, d]$, de tal manera que $y_j = c + jh'_n \ \forall j \in \{0, ..., k\}$, on $h'_k = \dfrac{d-c}{k}$.

Donades f: $[a, b] \subset \mathbb{R} \longrightarrow \mathbb{R}$ i g: $[c, d] \subset \mathbb{R} \longrightarrow \mathbb{R}$ funcions integrables, podem aplicar la regla de trapezis compost vist a la secció 3.3.2. Per tant:

$$T_n(f) = \frac{h_n}{2} \sum_{i=1}^{n} \big(f(x_{i-1}) + f(x_i)\big)$$
$$T'_k(g) = \frac{h'_k}{2} \sum_{j=1}^{k} \big(g(y_{j-1}) + g(y_j)\big)$$

Per tant, donada la funció q:$[a, b] \times [c, d] \subset \mathbb{R}^2 \longrightarrow \mathbb{R}$ integrable, podem aplicar el resultat anterior.

Observem que, per a la regla de Trapezis, $w_i = \dfrac{h_n}{2}$ per a $x_i = a, b$, i $w_i = h_n$ per a $x_i \neq a, b$. Similarment, $v_j = \dfrac{h'_k}{2}$ per a $y_j = c, d$, i $v_j = h'_k$ per a $y_j \neq c, d$. De tal manera que:

$$(T_n \times T'_k)(q) = \frac{h_n h_{k'}}{4} \Big(q(a, c) + q(a, d) + q(b, c) + q(b, d) + 2 \sum_{i=1}^{n-1} (q(x_i, c) + q(x_i, d)) +$$

$$+2\sum_{j=0}^{k-1}(q(a,y_j)+q(b,y_j))+4\sum_{i=1}^{n-1}\sum_{j=1}^{k-1}q(x_i,y_j)\Big)$$

Una expressió equivalent és:

$$(T_n\times T'_k)(q)=\frac{h_nh_{k'}}{4}\sum_{i=1}^{n}\sum_{j=1}^{k}\Big(q(x_{i-1},y_{j-1})+q(x_{i-1},y_j)+q(x_i,y_{j-1})+q(x_i,y_j)\Big)$$

Observem que el valor de la funció en els punts (x_i,y_j) situats als vèrtexs del rectangle $[a,b]\times[c,d]$ es multipliquen per 1, els punts situats a les arestes es multipliquen per 2 i, els punts que es troben a l'interior del rectangle, es multipliquen per 4.

Simpson producte compost Sigui $G=[a,b]\times[c,d]$ i sigui $\{x_0,...,x_n\}$ (amb $n=2m$) una partició de l'interval $[a,b]$, de tal manera que $x_i=a+h_ni\ \forall i\in\{0,...,n\}$, on $h_n=\dfrac{(b-a)}{n}$. Sigui $\{y_0,...,y_k\}$ (amb $k=2t$) una partició de l'interval $[c,d]$, de tal manera que $y_j=c+h'_nj\ \forall j\in\{0,...,k\}$, on $h'_k=\dfrac{(d-c)}{k}$.

Donades $f\colon[a,b]\subset\mathbb{R}\longrightarrow\mathbb{R}$ i $g\colon[c,d]\subset\mathbb{R}\longrightarrow\mathbb{R}$ funcions integrables, podem aplicar la regla de trapezis compost vist a la secció 3.3.2. Per tant:

Notació: $f_i=f(x_i)$, $g_j=g(y_j)$.

$$S_n(f)=\frac{h_n}{3}(f_0+4f_1+2f_2+4f_3+2f_4+...+4f_{n-1}+f_n)$$

$$S_k(g)=\frac{h'_k}{3}(g_0+4g_1+2g_2+4g_3+2g_4+...+4g_{k-1}+f_k)$$

Per tant, donada la funció $q\colon[a,b]\times[c,d]\subset\mathbb{R}^2\longrightarrow\mathbb{R}$ integrable, aplicant el resultat anterior:

Observem que, per a la regla de Simpson, $w_i=\dfrac{h_n}{3}$ per a $x_i=a,b$, $w_i=\dfrac{2h_n}{3}$ per a $x_i\neq a,b$ i i parell, i $w_i=\dfrac{4h_n}{3}$ per a $x_i\neq a,b$ i i senar. Similarment, $v_j=\dfrac{h'_k}{2}$ per a $y_j=c,d$, $v_j=\dfrac{2h'_k}{3}$ per a $y_j\neq c,d$ i j parell i $v_j=\dfrac{4h'_k}{3}$ per a $y_j\neq c,d$ i j senar . De tal manera que:

$$(S_n\times S'_k)(q)=\frac{h_nh'_k}{9}\Big(q(a,c)+q(a,d)+q(b,c)+q(b,d)+2\sum_{i=1}^{n/2-1}(q(x_{2i},c)+q(x_{2i},d))+$$
$$+4\sum_{i=0}^{n/2-1}(q(x_{2i+1},c)+q(x_{2i+1},d))+2\sum_{j=1}^{k/2-1}(q(a,y_{2j})+q(b,y_{2j}))+$$
$$+4\sum_{j=0}^{k/2-1}(q(a,y_{2j+1})+q(b,y_{2j+1}))+16\sum_{i=0}^{n/2-1}\sum_{j=0}^{k/2-1}q(x_{2i+1},y_{2j+1})+4\sum_{i=1}^{n/2-1}\sum_{j=1}^{k/2-1}q(x_{2i},y_{2j})+$$
$$+8\sum_{i=0}^{n/2-1}\sum_{j=1}^{k/2-1}q(x_{2i+1},y_{2j})+8\sum_{i=1}^{n/2-1}\sum_{j=0}^{k/2-1}q(x_{2i},y_{2j+1})\Big)$$

Taula de resultats

Observem ara una taula que mostra resultats d'aproximar $\int_1^2 \int_1^2 \sin(x+y)dydx$ utilitzant Simpson producte i Trapezis producte, fent variar el nombre n de nodes de la partició de $[a,b]$. D'aquesta manera podem comparar els dos mètodes. Consultar el codi a l'apartat 8.3 de l'annex.

Tenim en compte que:

$$\int_1^2 \int_1^2 \sin(x+y)dxdy = -\sin(4) + 2\sin(3) - \sin(2) \approx 129745 \cdot 10^{-1}.$$

Observació: els resultats estan presos per $n = k$.

n	Trapezis	ϵ_T	Simpson	ϵ_S
10	$1.295288 \cdot 10^{-1}$	$2.1618 \cdot 10^{-4}$	$1.297452 \cdot 10^{-1}$	$1.443 \cdot 10^{-7}$
100	$1.29742922 \cdot 10^{-1}$	$2.162413 \cdot 10^{-6}$	$1.297451 \cdot 10^{-1}$	$1.441 \cdot 10^{-11}$
1000	$1.297450846 \cdot 10^{-1}$	$7.585 \cdot 10^{-14}$	$1.297451 \cdot 10^{-1}$	$7.58559 \cdot 10^{-14}$
5000	$1.29745083 \cdot 10^{-1}$	$8.66 \cdot 10^{-10}$	$1.297451 \cdot 10^{-1}$	$2.8782 \cdot 10^{-12}$

Taula 2: Aproximacions de $\int_1^2 \int_1^2 \sin(x+y)dxdy$

Observació Observem un resultat inesperat en els valors de la taula. Quan n= 1000, ϵ_S té un error d'ordre 10^{-14}. Caldria esperar, doncs, que, quan n= 5000, ϵ_S tingués un ordre més petit que 10^{-14}, però observem que l'error obtingut té ordre 10^{-12}, és a dir, un error més gran. Això és degut al fet que, en augmentar el nombre de subdivisions n, l'error que correspon a la quadratura mitjançant el mètode de Simpson compost disminueix, però l'error computacional en les operacions augmenta. Això ens indica que en el càlcul numèric cal tenir en compte quina precisió demanem, ja que pot ser que demanant-ne menys, obtinguem un error més petit.

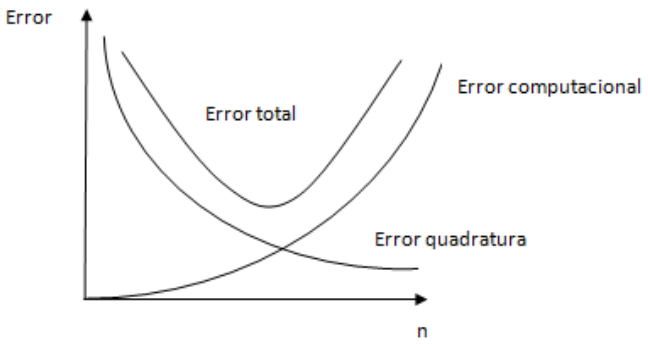

Figura 5: Gràfic de l'error total

Com podem observar, arriba un punt en que, malgrat augmentar el valor de n, l'error total augmenta.

Per tal de entendre-ho millor, i tenint en compte que ens serà de gran utilitat per a la programació dels algorismes adaptatius explicats en la següent secció, vegem les fórmules

producte en el cas $n = 2$ i $k = 2$, utilitzant la fórmula de Simpson i, en el cas $n = 1$, $k = 1$, utilitzant la fórmula dels Trapezis.

Trapezis producte simple Es tracta d'aplicar dues vegades la fórmula dels trapezis simples vist a 3.3.1. És a dir:

$$\int_a^b \int_c^d f(x,y)dxdy \approx \int_a^b \frac{h_y}{2}(f(x,c)+f(x,d))dx = \frac{h_y}{2}\int_a^b (f(x,c)+f(x,d))dx=$$

$$=\frac{h_x h_y}{4}\Big(f(a,c)+f(a,d)+f(b,c)+f(b,d)\Big)$$

,on: $h_x = (b-a)$ i $h_y = (d-c)$.

Es comprova que coincideix amb la fórmula de trapezis compostos per $n = 1, k = 1$.

Tractament de l'error Desenvolupem ara també el terme de l'error, que ens ajudarà a afinar l'algorisme Trapezis adaptatiu. Recordem que l'error en el cas 1-dimensional és:

$R_T = -\frac{1}{12}f''(\varsigma)h^3$, on $\varsigma \in (a,b)$ i $h = (b-a)$.

Per tant:

$$I = \int_a^b \int_c^d f(x,y)dydx = \int_a^b \left(\frac{f(x,c)+f(x,d)}{2}(d-c) - \frac{(d-c)^3}{12}\frac{\partial^2 f}{\partial y^2}(\psi)\right)dx =$$

$$= \int_a^b \frac{f(x,c)}{2}(d-c)dx + \int_a^b \frac{f(x,d)}{2}(d-c)dx - \int_a^b \frac{(d-c)^3}{12}\frac{\partial^2 f}{\partial y^2}(\psi)dx =$$

$$= \frac{f(a,c)+f(b,c)}{4}(d-c)(b-a)+\frac{(d-c)}{2}\frac{(b-a)^3}{12}\frac{\partial^2 f}{\partial x^2}(\omega_1)+\frac{f(a,d)+f(b,d)}{4}(d-c)(b-a)+$$

$$+\frac{(d-c)}{2}\frac{(b-a)^3}{12}\frac{\partial^2 f}{\partial x^2}(\omega_2) - \int_a^b \frac{(d-c)^3}{12}\frac{\partial^2 f}{\partial y^2}(\psi)dx$$

on,

$\psi = \psi(y), \omega_1 = \omega_1(x)$ i $\omega_2 = \omega_2(x) \in [a,b]\times[c,d]$

De fet, si escrivim: $\alpha_0 = \sup_{\omega\in[a,b]\times[c,d]}\left|\frac{\partial^2 f}{\partial x^2}(\omega)\right|$ i $\beta_0 = \sup_{\psi\in[a,b]\times[c,d]}\left|\frac{\partial^2 f}{\partial y^2}(\psi)\right|$, aleshores:

$$|I - T(f)| \leq \frac{(b-a)(d-c)}{12}\Big((b-a)^2\alpha_0 + (d-c)^2\beta_0\Big)$$

Simpson producte simple Es tracta d'aplicar dues vegades la fórmula de Simpson vist a 3.3.1. És a dir:

$$\int_a^b \int_c^d f(x,y)dydx \approx \int_a^b \frac{h_y}{3}\Big(f(x,y_0)+4f(x,y_1)+f(x,y_2)\Big)dx = \frac{h_x h_y}{9}\Big(f(x_0,y_0)+$$
$$4f(x_1,y_0)+f(x_2,y_0)+4f(x_0,y_1)+16f(x_1,y_1)+4f(x_2,y_1)+f(x_0,y_2)+4f(x_1,y_2)+$$
$$f(x_2,y_2)\Big)$$

Es pot comprovar que coincideix amb la fórmula de Simpson compost per $n = 2, k = 2$.

Tractament de l'error Desenvolupem ara també el terme de l'error, que ens ajudarà a afinar l'algorisme Simpson adaptatiu. Recordem que en el cas 1-dimensional és:

$$\int_a^b f(x)dx = \frac{h}{3}(f_a + 4f_c + f_b) + R_T, \text{ on } R_T = \frac{-h^5}{90}f^{(4)}(\sigma), \text{ on } \sigma = \sigma(x) \in (a,b)$$

Escrivim $h = \frac{b-a}{2}$ i $h' = \frac{d-c}{2}$. A més, $e = a + h$ i $s = c + h'$. Per tant:

$$I = \int_a^b \int_c^d f(x,y)dydx = \int_a^b \left(\frac{h'}{3}\Big(f(x,c) + f(x,s) + f(x,d)\Big) - \frac{h'^5}{90}\frac{\partial^4 f}{\partial y^4}(\sigma)\right)dx =$$

$$= \int_a^b \frac{h'}{3}f(x,c)dx + \int_a^b \frac{h'}{3}4f(x,s)dx + \int_a^b \frac{h'}{3}f(x,d)dx + \int_a^b \frac{-h'^5}{90}\frac{\partial^4 f}{\partial y^4}(\sigma)dx =$$

$$= I_1 + I_2 + I_3 + I_4$$

, on:

$$I_1 = \frac{h'}{3}\frac{h}{3}\Big(f(a,c) + 4f(e,c) + f(b,c)\Big) + \frac{h'}{3}\Big(-\frac{h^5}{90}\frac{\partial^4 f}{\partial x^4}(\epsilon_1)\Big), \text{ on } \epsilon_1 = \epsilon_1(x) \in [a,b] \times [c,d]$$

$$I_2 = \frac{4h'}{3}\frac{h}{3}\Big(f(a,s) + 4f(e,s) + f(b,s)\Big) + \frac{4h'}{3}\Big(-\frac{h^5}{90}\frac{\partial^4 f}{\partial x^4}(\epsilon_2)\Big), \text{ on } \epsilon_2 = \epsilon_2(x) \in [a,b] \times [c,d]$$

$$I_3 = \frac{h'}{3}\frac{h}{3}\Big(f(a,d) + 4f(e,d) + f(b,d)\Big) + \frac{h'}{3}\Big(-\frac{h^5}{90}\frac{\partial^4 f}{\partial x^4}(\epsilon_3)\Big), \text{ on } \epsilon_3 = \epsilon_3(x) \in [a,b] \times [c,d]$$

$$I_4 = \int_a^b \frac{-h'^5}{90}\frac{\partial^4 f}{\partial y^4}(\sigma)dx, \text{ on } \sigma = \sigma(y) \in [a,b] \times [c,d]$$

Si anomenem $\alpha_0 = \sup_{\epsilon \in [a,b]\times[c,d]} \left|\frac{\partial^4 f}{\partial x^4}(\epsilon)\right|$ i $\beta_0 = \sup_{\sigma \in [a,b]\times[c,d]} \left|\frac{\partial^4 f}{\partial y^4}(\sigma)\right|$, aleshores:

$$|S(f) - I| \leq \frac{2h'(h^5)}{90}\alpha_0 + \frac{h'^5}{90}2h\beta_0$$

5.2 Canvi de variable

Imaginem que tenim una regió $B \in \mathbb{R}^2$ en coordenades x, y i una regió estàndard $B' \in \mathbb{R}^2$ en coordenades u, v; és a dir, tenim regles d'integració per a la regió $B^{'}$. Suposem també que existeix un canvi de variable de B' a B. Escrivim $x = \phi(u,v), y = \psi(u,v)$.

Suposem també que ϕ i ψ tenen derivades parcials contínues i el Jacobià:

$$J(u,v) = \begin{vmatrix} \frac{\partial \phi}{\partial u} & \frac{\partial \phi}{\partial v} \\ \frac{\partial \psi}{\partial u} & \frac{\partial \psi}{\partial v} \end{vmatrix} \neq 0, \forall (u,v) \in B'$$

Sigui la regla de quadratura a B':

$$\int\int_{B'} h(u,v)dudv = \sum_{k=1}^{n} w_k h(u_k, v_k), \text{ on } (u_k, v_k) \in B'$$

Aleshores:

$$\int\int_B f(x,y)dxdy = \int\int_{B'} f(\phi(u,v), \psi(u,v))|J(u,v)|dudv \approx$$

$$\approx \sum_{k=1}^{n} w_k f(\phi(u_k, v_k), \psi(u_k, v_k))|J(u_k, v_k)| = \sum_{k=1}^{n} W_k f(x_k, y_k), \text{ on:}$$

$x_k = \phi(u_k, v_k)$, $y_k = \psi(u_k, v_k)$ i $W_k = w_k|J(u_k, v_k)|$

Exemple: Cas disc unitat En el cas del disc unitat, que es pot escriure com $\{(x,y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}$, es tracta de calcular:

$$I = \int_{-1}^{1} (\int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}} f(x,y)dy)dx$$

Si ho escrivim en coordenades polars, i tenint en compte que:

$$J(r,\phi) = \begin{vmatrix} \cos\phi & -r\sin\phi \\ \sin\phi & r\cos\phi \end{vmatrix} = r$$

Llavors, usant els resultats generals, com $r = 1$:

$$\int_{-1}^{1} (\int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}} f(x,y)dy)dx = \int_0^{2\pi} \int_0^1 f(\cos\phi, \sin\phi)drd\phi,$$

que és una integral sobre la regió rectangular $[0, 2\pi] \times [0, 1]$, on podem usar els mètodes vists per tal d'obtenir una aproximació.

5.3 Quadratura en regions elementals

Definició:

Diem que una regió $D \in \mathbb{R}^2$ és elemental si és y-simple o x-simple. Una regió és y-simple (anàleg per x-simple) si està acotada per dues funcions $\phi_1(x)$, $\phi_2(x)$. Podem veure la gràfica de la situació:

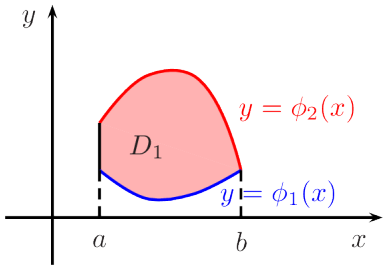

Figura 6: Regió elemental

Siguin R i S dues regles de cuadratura a $[a,b]$ i $[c,d]$, respectivament. És a dir, donades f: $[a,b] \subset \mathbb{R} \longrightarrow \mathbb{R}$ i g: $[c,d] \subset \mathbb{R} \longrightarrow \mathbb{R}$ funcions integrables, aleshores podem escriure R i S com:

$$R(f) := \sum_{i=0}^{n} w_i f(x_i), \text{ on } w_i \in \mathbb{R} \forall i \in \{0, ..., n\}$$

$$S(g) := \sum_{j=0}^{k} v_j g(y_j), \text{ on } v_j \in \mathbb{R} \forall j \in \{0, ..., k\}$$

Per tant, donada la funció h:$[a,b] \times [c,d] \subset \mathbb{R}^2 \longrightarrow \mathbb{R}$ integrable, volem calcular:

$$I = \int_a^b (\int_{\phi_1(x)}^{\phi_2(x)} h(x,y)dy)dx$$

La idea és, fixada $x \in [a,b]$, transformar la integral $\int_{\phi_1(x)}^{\phi_2(x)} h(x,y)dy$ en $\int_c^d h_2(x,y)dy$.

Això ho podem realitzar mitjançant un camí, és a dir, mitjançant l'aplicació:

$$\omega_x(t) := \frac{\phi_2(x) - \phi_1(x)}{(d-c)}(t-c) + \phi_1(x), \text{ on } t \in [c,d]$$

Es pot comprovar que $\omega_x([c,d]) = [\phi_1(x), \phi_2(x)]$.

A més, $\omega_x'(t) = \dfrac{\phi_2(x) - \phi_1(x)}{(d-c)} \neq 0 \ \forall t \in [c,d]$.

Per tant:

$$\int_{\phi_1(x)}^{\phi_2(x)} h(x,y)dy = \int_c^d \frac{[\phi_2(x) - \phi_1(x)]}{(d-c)} h(x, \omega_x(t))dt$$

Observació: A priori podria semblar necessaria la condició $\phi_2(x) > \phi_1(x) \forall x \in [a,b]$, per tal que $\omega_x'(t) \neq 0 \ \forall t \in [c,d]$ i poder fer el canvi en la integral. Això, però, es pot solucionar separant les integrals en els zeros de $\phi_2(x) > \phi_1(x)$, de tal manera que sempre es compleix aquesta o condició, o bé $\phi_1(x) > \phi_2(x)$. Finalment:

$$\int_a^b \Big(\int_{\phi_1(x)}^{\phi_2(x)} h(x,y)dy \Big) dx. = \int_a^b \Big(\int_c^d \frac{[\phi_2(x) - \phi_1(x)]}{(d-c)} h(x, \omega_x(t))dt \Big) dx$$

És a dir, es tracta de la integral de $h_2(x,y)$ sobre la regió rectangular $[a,d] \times [c,d]$. Per tant, ja podem aplicar la regla producte $R \times S$, tal i com s'explica en la secció 5.1.2.

Exemple Veiem l'exemple del disc unitat, que es pot expressar com $C = \{(x,y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\}$. Volem calcular:

$$I = \int_{-1}^{1} (\int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}} h(x,y)dy)dx$$

Observem que es compleix $\phi_1(x) = -\sqrt{1-x^2} < \phi_2(x) = \sqrt{1-x^2} \ \forall x \in [a,b]$.

Sigui $[c,d]=[0,2\pi]$. Definim $\omega_x(t) := \dfrac{\sqrt{1-x^2}(t)}{\pi} - \sqrt{1-x^2} = (\sqrt{1-x^2})(2t-1)$

Com que $\omega_x'(t) = \dfrac{\sqrt{1-x^2}}{\pi}$:

$$\int_{-1}^{1}\Big(\int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}} h(x,y)dy\Big)dx = \int_{-1}^{1}\Big(\int_{0}^{2\pi} \sqrt{1-x^2}h(x,\omega_x(t))dt\Big)dx$$

De fet, si $h(x,y)=1$:

$$\int_{-1}^{1}\Big(\int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}} h(x,y)dy\Big)dx = \int_{-1}^{1}\Big(\int_{0}^{2\pi} \frac{\sqrt{1-x^2}}{\pi}dt\Big)dx = 2\pi$$

5.4 Quadratura en altres regions

Un cop hem vist les fórmules producte, podem desenvolupar un mètode per tal d'aproximar $\int_U f(x,y)dydx$ a qualsevol regió U. Es tracta d'omplir la regió de subregions rectangulars i, en cadascuna d'elles, aplicar les fórmules producte i finalment fer la suma total. Observem una imatge del procediment.

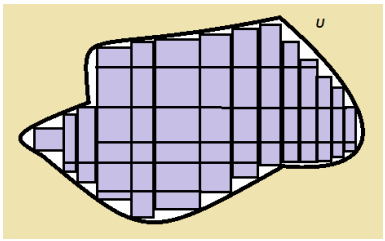

Figura 7: Regió U

Dificultats i error Observem que en aquesta ocasió tenim tres errors: en primer lloc, l'error generat a l'omplir amb rectangles la figura; en segon lloc, l'error generat per cada un dels rectangles en fer la quadratura utilitzant les fórmules producte; i per últim, cal sumar l'error computacional en les operacions aritmètiques durant el procediment.

D'altra banda, la dificultat d'aquest mètode resideix en el pas d'omplir una regió amb rectangles, ja que U pot tenir zones molt irregulars on la tasca sigui complicada i no sigui possible fer una aproximació de $\int_U f(x,y)dydx$ prou acurada.

Exemple A l'apartat 8.4 de l'annex es pot consultar el codi del programa que aplica aquest procediment a qualsevol triangle rectangle. L'algorisme va omplint el triangle amb quadrats de costat ϵ (introduït per l'usuari) i calculant, mitjançant Trapezis producte, l'aproximació de la integral.

Observem aquí els punts equidistants on s'ha avaluat la funció a la meitat de l'execució i en acabar. S'ha avaluat la funció $f(x,y)=\sin(x+y)$ en el triangle de vèrtexs $(0,0),(1,0),(0,1)$ amb $\epsilon = 0.01$.

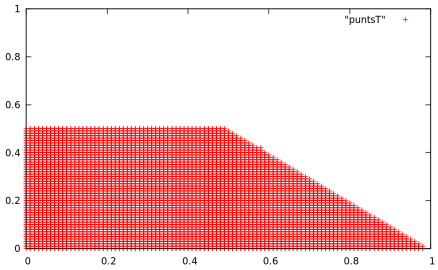

Figura 8: Punts avaluats a mitja execució

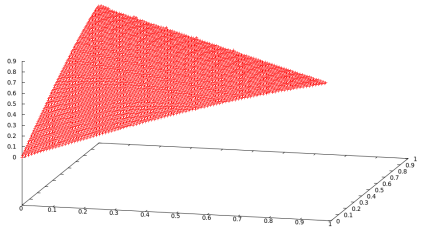

Figura 9: Punts avaluats en finalitzar

Taula de resultats

Observem els resultats de $\int_0^1 \int_0^{x+1} \sin(x+y)dydx$.

Tenim en compte que $\int_0^1 \int_0^{x+1} \sin(x+y)dydx = \sin(1) - \cos(1) \approx 0.3011686$.

ϵ	Resultat	Total Subdivisions	Error
10^{-1}	0.1892736	36	$1.11895 \cdot 10^{-1}$
10^{-2}	0.2894414	4860	$1.172731 \cdot 10^{-2}$
10^{-3}	0.2999794	498586	$1.189306 \cdot 10^{-3}$

Taula 3: Aproximacions de $\int_0^1 \int_0^{x+1} \sin(x+y)dydx$

6 Algorismes adaptatius

Recordem que voliem aproximar la integració d'una funció $f : [a,b] \times [c,d] \subset \mathbb{R}^2 \longrightarrow \mathbb{R}$ mitjançant la integració del polinomi interpolador. Ara bé, com ja hem vist, aquesta integral ens donarà un error respecte la integral de f. Els algorismes adaptatius tenen com a objectiu aproximar la integral de forma que es disminueix el nombre d'avaluacions de la funció necessàries.

De fet, la idea és equivalent a la del cas 1-dimensional explicat a 3.4. En els següents apartats farem el tractament de l'error per a implementar l'algorisme.

6.1 Tractament de l'error

6.1.1 Trapezis adaptatius

Sigui $f : [a,b] \times [c,d] \subset \mathbb{R}^2 \longrightarrow \mathbb{R}$. Anomenem $h_1 = b - a$, $h_2 = d - c$.

Tal i com s'explica en pas dos de l'algorisme a la secció 6.2, es divideix la regió inicial en 4 subregions, que anomenem G_i:

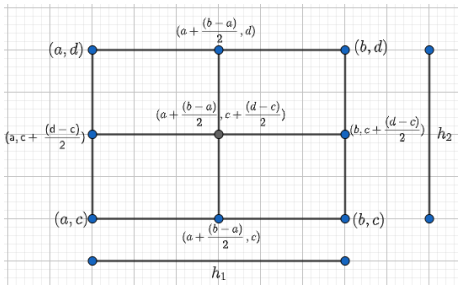

Figura 10: Subdivisió inicial

Anomenem $I = \int_a^b \int_c^d f(x,y)dxdy$. De la secció 5.1.2 podem extreure que, si S és la primera aproximació de I mitjançant Trapezis, aleshores:

$$I - S = \frac{h_2 h_1^3}{12}\alpha_0 + \frac{h_1 h_2^3}{12}\beta_0 \text{ , on } \alpha_0 = \frac{\partial^2 f}{\partial x^2}(\omega) \text{ i } \beta_0 = \frac{\partial^2 f}{\partial y^2}(\psi), \text{ on } \omega, \psi \in [a,b] \times [c,d]$$

Sigui S_2 la suma de les quatre aproximacions $I_i, i = \{1, ..., 4\}$ mitjançant Trapezis de cada una de les subregions de fig.10. De cadascuna d'aquestes regions G_i tindrem un error ϵ_i respecte de I_i. De fet:

$$\epsilon_i = \frac{\frac{h_2}{2}\left(\frac{h_1}{2}\right)^3}{12}\alpha_0^{(i)} + \frac{\frac{h_1}{2}\left(\frac{h_2}{2}\right)^3}{12}\beta_0^{(i)}, \text{ on } \alpha_0^{(i)} = \frac{\partial^2 f}{\partial x^2}(\omega_i) \text{ i } \beta_0^{(i)} = \frac{\partial^2 f}{\partial y^2}(\psi_i), \text{ on } \omega_i, \psi_i \in G_i$$

Ara, suposem que es compleix: $\alpha_0^i \approx \alpha_0$ i $\beta_0^i \approx \beta_0$ $\forall i \in \{1,2,3,4\}$:

$$I - S_2 \approx 4\left(\frac{\frac{h_2}{2}\left(\frac{h_1}{2}\right)^3}{12}\alpha_0\right) + 4\left(\frac{\frac{h_1}{2}\left(\frac{h_2}{2}\right)^3}{12}\beta_0\right) = \frac{1}{4}\left(\frac{h_2 h_1^3}{12}\alpha_0\right) + \frac{1}{4}\left(\frac{h_1 h_2^3}{12}\beta_0\right)$$

De manera que, $I - S_2 - (I - S) = S - S_2 \approx \frac{-3}{4}\frac{h_2 h_1^3}{12}\alpha_0 + \frac{-3}{4}\frac{h_1 h_2^3}{12}\beta_0$

Si escrivim, $S - S_2 \approx \frac{-3}{4}k$, aleshores $k \approx \frac{-4}{3}(S - S_2)$.

Finalment: $|I - S_2| \approx |\frac{1}{4}\frac{-4}{3}(S - S_2)| = \frac{1}{3}|S - S_2|$

Si volem $|I - S_2| \leq \epsilon \Rightarrow |S - S_2| < 3\epsilon$.

A més, $I - S_2 \approx \frac{1}{3}(S_2 - S)$.

Per tant, ja tenim una fita per al pas 3 de l'algorisme.

6.1.2 Simpson adaptatiu

Sigui $f : [a,b] \times [c,d] \subset \mathbb{R}^2 \longrightarrow \mathbb{R}$. Anomenem $h_1 = b - a$, $h_2 = d - c$.

Tal i com s'explica en pas dos de l'algorisme de la secció 6.2, es divideix la regió inicial en 4 subregions, que anomenem G_i (veure fig.10).

Anomenem $I = \int_a^b \int_c^d f(x,y)dxdy$. De la secció podem extreure que, si S és la primera aproximació de I mitjançant Simpson, aleshores:

$$I - S = \frac{2h'(h^5)}{90}\alpha_0 + \frac{h'^5}{90}2h\beta_0, \text{ on } \alpha_0 = \frac{\partial^4 f}{\partial x^4}(\epsilon) \text{ i } \beta_0 = \frac{\partial^4 f}{\partial y^4}(\sigma), \text{ amb } \epsilon, \sigma \in [a,b] \times [c,d]$$

Sigui S_2 la suma de les quatre aproximacions $I_i, i = \{1, ..., 4\}$ mitjançant Simpson de cada una de les 4 subregions. De cadascuna d'aquestes regions G_i tindrem un error ϵ_i respecte de I_i. De fet:

$$\epsilon_i = \frac{2h'(h^5)}{90}\alpha_0^i + \frac{h'^5}{90}2h\beta_0^i, \text{ on } \alpha_0^i = \frac{\partial^4 f}{\partial x^4}(\epsilon_i) \text{ i } \beta_0^i = \frac{\partial^4 f}{\partial y^4}(\sigma_i), \text{ amb } \epsilon_i, \sigma_i \in G_i$$

Ara, si suposem que $\alpha_0^i \approx \alpha_0$ i$\beta_0^i \approx \beta_0 \ \forall i \in \{1,2,3,4\}$:

$$I - S_2 \approx 4\left(\frac{2h'}{2}\frac{\left(\frac{h}{2}\right)^5}{90}\alpha_0\right) + 4\left(\frac{\left(\frac{h'}{2}\right)^5}{90}2\frac{h}{2}\beta_0\right) = \frac{1}{16}\frac{2h'(h)^5}{90}\alpha_0 + \frac{1}{16}\frac{2h(h')^5}{90}\beta_0$$

De manera que, $I - S_2 - (I - S) = S - S_2 \approx \frac{15}{16}\left(\frac{2h'(h)^5}{90}\alpha_0 + \frac{(h')^5 2h}{90}\beta_0\right)$

Si escrivim, $S - S_2 \approx \frac{15}{16}k$, aleshores $k \approx \frac{16}{15}(S - S_2)$.

Per tant, $|I - S_2| = |\frac{1}{15}|S - S_2|$

Si volem $|I - S_2| \leq \epsilon \Rightarrow |S - S_2| < 15\epsilon$.

A més, $I - S_2 \approx \frac{1}{15}(S_2 - S)$.

Per tant, ja tenim una fita per al pas 3 de l'algorisme.

6.2 Algorisme

Els passos de l'algorisme són:

1) Calcular la integral en el rectangle $G = \{(x,y) \in \mathbb{R}^2 |\ a \leq x \leq b,\ c \leq y \leq d\}$.

2) Dividir aquesta regió en quatre subregions, i calcular la integral en cadascuna d'elles.

3) Comparar la suma S de les quatre integrals amb el valor I de la integral obtinguda a G. En aquest punt ens trobem amb dues possibles situacions:

Sigui ϵ l'error demanat:

Usant l'algorisme de trapezis adaptatius:

3.1) La diferència de S i I és més petita que 3ϵ ; en aquest cas, donem per vàlid el valor de la integral. Retornem S+ (I-S).

3.2) La diferència de S i I és més gran que 3ϵ; aleshores dividim la regió en 4 subregions i tornem al pas 1 per a cadascuna d'elles.

Observacions:

i) Si usem l'algorisme Simpson adaptatiu, aleshores només cal canviar 15ϵ per 3ϵ en el pas 3.

ii) Cada cop que subdividim una regió en 4 subregions, l'error que es demana en cada una d'elles és $\frac{\epsilon}{4^n}$, on n és el nombre de subdivisions fetes.

iii) Es tracta d'un algorisme recursiu.

Notem que, d'aquesta forma, obtenim una aproximació vàlida de la integral optimitzant el nombre de crides.

6.3 Resultats

6.3.1 Taula de resultats

A continuació es mostra una taula amb aproximacions de $\int_a^b \int_c^d \sin(x+y)dx$, usant Trapezis adaptatius i Simpson adaptatiu, i fent variar l'error ϵ per tal de veure com varia l'aproximació. Els resultats es mostren a la regió $G = [1,2] \times [1,2]$. Cal tenir en compte que:

$$\int_a^b \int_c^d \sin(x+y)dx = -\sin(b+d) + \sin(a+d) + \sin(b+c) - \sin(a+c),$$

per tant $\int_1^2 \int_1^2 \sin(x+y)dx \approx 0.1297450846$

ϵ	Trapezis	$error_T$	Simpson	$error_S$
10^{-3}	0.12974489	$1.956176 \cdot 10^{-7}$	0.12974494	$1.396405 \cdot 10^{-7}$
10^{-6}	0.1297450846	$2.247 \cdot 10^{-13}$	0.1297450848	$2.366518 \cdot 10^{-10}$
10^{-7}	0.1297450846	$1.831 \cdot 10^{-15}$	0.1297450845	$2.2416 \cdot 10^{-12}$

Taula 4: Aproximacions de $\int_1^2 \int_1^2 \sin(x+y)dx$

6.3.2 Representació dels punts

En aquesta secció treballarem amb les funcions $\sin(x+y)$ i $e^{-(x^2+y^2)}\sin((x^2+y^2)\pi)$. En el cas de la primera funció treballarem en la regió $G_1 = [1,5] \times [1,4]$, i en el de la segona $G_2 = [-0.5,2] \times [-0.5,2]$. Observem el gràfic d'ambdues funcions en les respectives regions.

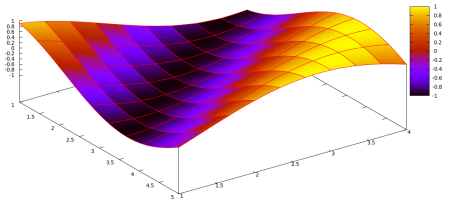

Figura 11: Gràfica de $sin(x+y)$ a G_1

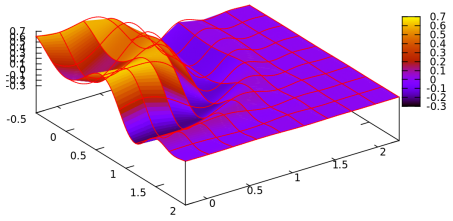

Figura 12: Gràfica de $e^{-(x^2+y^2)} \sin\left((x^2+y^2)\pi\right)$ a G_2

Mostrem ara una imatge que representa els punts en què s'ha avaluat cada una de les funcions en l'execució del programa Trapezis adaptatius. D'aquesta manera es vol emfatitzar l'objectiu de l'algorisme, i veure com, depenent de la naturalesa de la funció a cada regió, han estat necessàries més avaluacions.

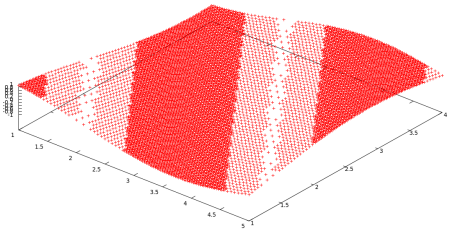

Figura 13: Representació dels punts avaluats

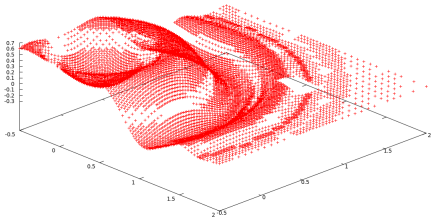

Figura 14: Representació dels punts avaluats

Mostrem ara una imatge que representa els punts en què s'ha avaluat cada una de les funcions en l'execucio del programa Simpson adaptatiu.

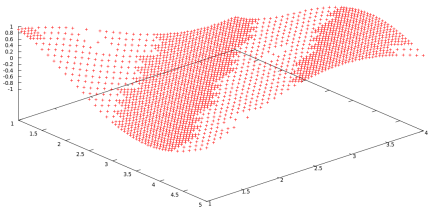

Figura 15: Representació dels punts avaluats

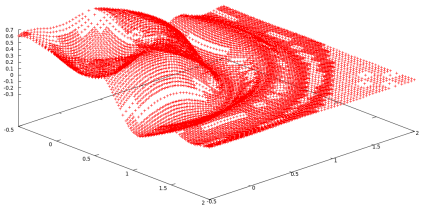

Figura 16: Representació dels punts avaluats

6.3.3 Comparació

En aquest apartat es vol comparar els programes de Trapezis i Simpson composts amb Trapezis i Simpson adaptatius. D'aquesta manera es vol emfatitzar la utilitat dels algorismes adaptatius per la seva eficiència. La taula compara el nombre d'avaluacions totals de la funció f que han calgut per arribar a l'aproximació de $\int_1^2 \int_1^2 \sin(x+y) dy dx$, amb un error ϵ similar.

ϵ_T	Trapezis	Trapezis adaptatius	ϵ_S	Simpson	Simpson adaptatiu
10^{-4}	400	20	10^{-7}	121	45
10^{-6}	40000	148	10^{-11}	10201	765
10^{-8}	4000000	1108	10^{-14}	1002001	11277

Taula 5: Comparació mètodes compostos i adaptatius

Es pot observar clarament l'eficiència dels algorismes adaptatius implementats.

6.4 Interfície gràfica

Donat que el codi en C només serveix per aproximar la funció que està implementada en elmoment de l'execució, és a dir, de forma no interactiva, la següent interfície gràfica proporciona a l'usuari l'opció de introduir la funció $f(x, y)$ que desitja, així com la regió $[a, b] \times [c, d]$ i l'error ϵ, de tal manera que es calcula l'aproximació de $\int_a^b \int_c^d sin(x + y)dx$ mitjançant Trapezis adaptatius i es mostra per pantalla. A l'annex es pot consultar el codi en Java. Mostrem ara una imatge de la interfície, i continuem amb l'exemple de $sin(x + y)$ per tal de comparar el resultat. Cal esmentar que, quan l'usuari introdueix dades errònies, el programa mostra per pantalla un missatge d'error indicant-ne el tipus.

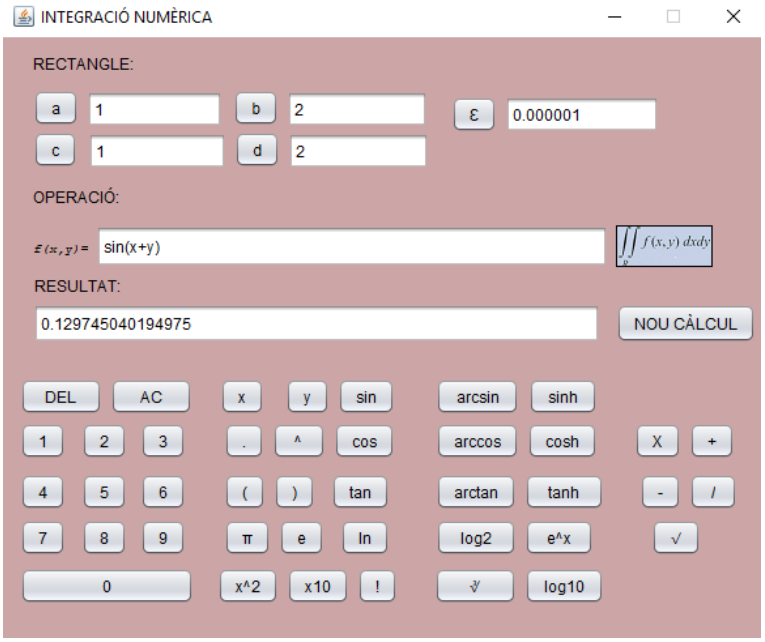

Figura 17: Interfície gràfica

Cal destacar que la gran dificultat era transformar la funció introduida per l'usuari (String) en una funció. Per tal de fer això he utilitzat un parser. A la bibliografia es pot consultar l'enllaç a la pàgina web per a més informació del seu ús.

7 Conclusions

En aquest estudi hem desenvolupat la interpolació i quadratura en dues dimensions. Degut que la quadratura és un problema amb moltes aplicacions a diferents àrees de la ciència, he pogut obtenir informació sobre el tema a partir de diverses fonts.

Concretament, hem vist alguns dels mètodes d'interpolació i quadratura en dues dimensions, fent ús de resultats més coneguts en una dimensió. Al llarg de l'estudi, s'adjunten resultats de caire pràctic, que tenen com a objectiu recolzar els mètodes presentats.

Un cop finalitzat el treball, he pogut comprovar la rellevància de fer un ús òptim de la programació en les matemàtiques. En aquest cas, es pot observar l'optimització de la quadratura mitjançant els algorismes adaptatius, emfatitzant el fet que s'aconsegueixen resultats igual de precisos amb més eficiència.

Finalment, no m'agradaria tancar aquest treball sense dir que fer-lo m'ha demanat i m'ha ofert una bona oportunitat per a aprendre. En primer lloc, el desenvolupament del treball m'ha exigit desenvolupar el rigor propi d'un treball universitari: rigor en la recerca bibliogràfica, en la comprensió de les idees, en el tractament de la informació i en la seva exposició; en segon lloc, aquest treball m'ha ofert una gran oportunitat per ampliar els meus coneixements sobre les matemàtiques, especialment en l'àmbit dels mètodes numèrics.

8 Annex

Programes en C:

En els programes següents treballem amb $f(x, y)$ qualsevol. De fet, gran part dels resultats al llarg de la memòria han estat presos per $f(x, y) = sin(x + y)$.

```
double f ( double x, double y) {
   double z;
   z = sin(x+y);
   return ( z );
}
```

8.1 Trapezis adaptatius

```
int it=0;
FILE* fitxer;

double f ( double x, double y );

double trapezi ( double ( *f ) ( double, double ), double a, double b, double
    c, double d);
double adapTrapezi ( double ( *f ) ( double, double ), double a, double
    b,double c, double d, double eps, double sum);

int main ( void ) {
   double a, b,c, d, eps, trap, adap,error;

   fitxer= fopen("punts", "wt");

   printf ( "doneu interval (a, b) = \n" );
   scanf ( "%lf %lf", &a, &b );

   printf ( "doneu interval (c, d) = \n" );
   scanf ( "%lf %lf", &c, &d );

   printf ( "doneu tolerancia eps = \n" );
   scanf ( "%lf", &eps );

   trap = trapezi ( f, a, b, c, d );
   adap = adapTrapezi ( f, a, b, c, d, eps, trap);
   error= adap-(-sin(4)+2*sin(3)-sin(2));

   printf ( "Al aproximar el valor de la integral de sin(x+y) al rectangle (a,
       b) = (%23.15e, %23.15e)x (c, d) = (%23.15e, %23.15e) \n",   a, b, c,d );
   printf ( "usant Trapezis Adaptatius amb tol=%23.15e obtenim el valor =
       %23.15e \n", eps, adap );
   printf("s'ha fet un total de %d avaluacions de la funcio\n", it);
   printf("L'error es de: %le\n", error);

   fclose(fitxer);
```

```
   return 0;
}

double adapTrapezi ( double ( *f ) ( double, double), double a, double b,double
    c, double d, double eps, double sum) {

   double esqa, esqb, dreta, dretb, z, t, adap, suma;


   fprintf(fitxer, "%le %le %le\n", a, c, ( *f ) ( a, c ));
   fprintf(fitxer, "%le %le %le\n", a, d, ( *f ) ( a, d ));
   fprintf(fitxer, "%le %le %le\n", b, c, ( *f ) ( b, c ));
   fprintf(fitxer, "%le %le %le\n", b, d, ( *f ) ( b, d ));

   z = ( a + b ) / 2.;
   t = ( c + d ) / 2.;
   esqb = trapezi ( f, a,z,c,t );
   esqa = trapezi ( f, a, z,t, d );
   dreta= trapezi (f, z, b, t, d);
   dretb= trapezi(f, z, b, c, t);
   suma= esqb+ dreta+ esqa+dretb;

   if ( fabs ( suma - sum ) <= 3*eps ) {
      adap = suma+((suma-sum)/3.);
      return ( adap );
   }
   adap = adapTrapezi ( f, a, z, c, t, eps / 4., esqb) + adapTrapezi ( f, a, z,
       t, d, eps / 4., esqa) + adapTrapezi ( f, z, b, t, d,   eps / 4., dreta) +
       adapTrapezi ( f, z, b, c, t, eps / 4., dretb);
   return ( adap );
}

double trapezi ( double ( *f ) ( double, double ), double a, double b, double
    c, double d ) {
   double trap;
   trap = ( ( *f ) ( a, c ) + ( *f ) ( a,d ) + ( *f ) ( b,c ) + ( *f ) ( b,d ) )
       * (b-a)/ 2. *(d-c)/2.;
   it+=4;
   return ( trap );
}
```

8.2 Simpson adaptatiu

```
int it=0;
FILE* fitxer;
double f1 ( double x, double y );
double simpson ( double ( *f1 ) ( double, double ), double a, double c, double
    d, double f);
double adapSimpson ( double ( *f1 ) ( double, double ), double a,double c,
    double d, double f, double eps, double sum);

int main ( void ) {
```

```
   double a,c, d, f, eps, simp, adap, error;


   fitxer= fopen("puntss", "wt");

   printf ( "doneu els valors a, c = \n" );
   scanf ( "%lf %lf", &a, &c);
   if(a>=c ){
      printf("valors incorrectes, es requereix a<c \n");
      return 0;
   }

   printf ( "doneu els valors d, f = \n" );
   scanf ( "%lf %lf", &d, &f );
   if(d>=f){
      printf("valors incorrectes, es requereix d<f \n");
      return 0;
   }

   printf ( "doneu tolerancia eps = \n" );
   scanf ( "%lf", &eps );

   simp = simpson ( f1, a, c, d, f );
   adap = adapSimpson ( f1, a, c, d, f, eps, simp);

   printf ( "Al aproximar el valor de la integral de sin(x+y) al rectangle (a,
       c) = (%23.15e, %23.15e)x (d, f) = (%23.15e, %23.15e) \n",   a, c, d,f );
   printf ( "usant Simpson Adaptatiu amb tol=%23.15e obtenim el valor = %23.15e
       \n", eps, adap );

   fclose(fitxer);


   error= adap-(-sin(4)+2*sin(3)-sin(2));
   printf("Amb un error de: %le\n", error);
   printf("la funcio s'ha avaluat %d vegades\n", it);

   return 0;
}

double adapSimpson ( double ( *f1 ) ( double, double), double a,double c,
    double d, double f, double eps, double sum) {
   double esqa, esqb, dreta, dretb, adap, suma, b, e;

   b= (a+c)/2.;
   e=(d+f)/2.;

   fprintf(fitxer, "%le %le %le\n", a, d, ( *f1 ) ( a, d ));
   fprintf(fitxer, "%le %le %le\n", a, e, ( *f1 ) ( a, e ));
   fprintf(fitxer, "%le %le %le\n", a, f, ( *f1 ) ( a, f ));
   fprintf(fitxer, "%le %le %le\n", b, d, ( *f1 ) ( b, d ));
   fprintf(fitxer, "%le %le %le\n", b, e, ( *f1 ) ( b, e ));
   fprintf(fitxer, "%le %le %le\n", b, f, ( *f1 ) ( b, f ));
   fprintf(fitxer, "%le %le %le\n", c, d, ( *f1 ) ( c, d ));
   fprintf(fitxer, "%le %le %le\n", c, e, ( *f1 ) ( c, e ));
   fprintf(fitxer, "%le %le %le\n", c, f, ( *f1 ) ( c, f ));
```

```
   esqb = simpson ( f1, a, b, d, e );
   esqa = simpson ( f1, a, b, e, f );
   dreta= simpson ( f1, b, c, e, f );
   dretb= simpson ( f1, b, c, d, e);

   suma= esqb + esqa + dreta + dretb;

   if ( fabs ( suma - sum ) <= 15*eps ) {
      adap = suma+ (suma-sum)/15.;
      return ( adap );
   }
   adap = adapSimpson ( f1, a, b, d, e, eps/4. , esqb) + adapSimpson ( f1, a, b,
       e, f, eps/4., esqa) + adapSimpson ( f1, b, c, e, f, eps/4., dreta) +
       adapSimpson ( f1, b, c, d, e, eps/4., dretb);
   return ( adap );
}

double simpson ( double ( *f1 ) ( double, double ), double a, double c, double
    d, double f ) {
   double sim, b, e;

   b= (a+c)/2.;
   e=(d+f)/2.;

   sim = ( ( *f1 ) ( a, d )+ 4*(( *f1 ) ( b,d ))+ ( *f1 ) ( c,d ) + 4*(( *f1 ) (
       a,e )) + 16*(( *f1 ) ( b,e )) + 4*(( *f1 ) ( c,e )) +   ( *f1 ) ( a,f )+
       4*(( *f1 ) ( b,f )) + ( *f1 ) ( c,f ) ) * (c-a)/ 18. *(f-d)/2.;

   it+=9;
   return ( sim );
}
```

8.3 Trapezis i Simpson compostos

```
int ittrap=0, itsim=0;
double f ( double x, double y );
double trapezi ( double ( *f ) ( double, double ), double x, double y, double
    z, double t, int n, int k);
double simpson(double(*f)(double, double), double a, double b, double c, double
    d, int n, int k);

int main ( void ) {
   double a, b,c, d, trap, sim;
   int n, k;

   double valor, errort, errors;
   valor= -sin(4)+2*sin(3)-sin(2);

   printf ( "doneu interval (a, b) = \n" );
   scanf ( "%lf %lf", &a, &b );
```

```
   printf ( "doneu interval (c, d) = \n" );
   scanf ( "%lf %lf", &c, &d );

   printf ( "doneu el nombre de nodes n en (a,b)= \n" );
   scanf ( "%d", &n );

   printf ( "doneu el nombre de nodes k en (c,d)= \n" );
   scanf ( "%d", &k );

   trap = trapezi ( f, a, b, c, d, n, k );
   sim= simpson(f, a, b, c, d, n, k);
   errort= fabs(valor-trap);
   errors= fabs(valor- sim);

   printf ( "Al aproximar el valor de la integral de sin(x+y) al rectangle (a,
       b) = (%23.15e, %23.15e)x(c, d) = (%23.15e, %23.15e) \n",   a, b, c,d );
   printf ( "usant Trapezis Compost obtenim el valor = %23.15e\n", trap);
   printf("amb un error de %le\n", errort);
   printf("amb un total de %d avaluacions\n",ittrap);
   printf("usant Trapezis Compost obtenim el valor = %23.15e \n", sim);
   printf("amb un error de %le\n", errors);
   printf("amb un total de %d avaluacions\n",itsim);

   return 0;
}



double trapezi ( double ( *f ) ( double, double ), double a, double b, double
    c, double d, int n, int k ) {
   double trap=0, h1, h2;
   int i, j;

   h1= (b-a)/n;
   h2= (d-c)/n;

   for(i=1; i<n+1; i++){
      for(j=1; j<n+1; j++){
         trap+= (( *f ) ( a+(i-1)*h1, c+(j-1)*h2 )+( *f ) ( a+(i-1)*h1, c+(j)*h2
             )+( *f ) ( a+(i)*h1, c+(j-1)*h2 )+( *f ) ( a+(i)*h1, c+(j)*h2 ));
         ittrap+=4;

      }
   }
   trap*=(h1*h2)/4.;


   return ( trap );
}

double simpson(double(*f)(double, double), double a, double b, double c, double
    d, int n, int k){
   double sim=0, h1, h2;
   int i, j;
```

```
h1=(b-a)/n;
h2= (d-c)/n;

for(i=0; i<n+1; i++){
   for(j=0; j<n+1; j++){
      itsim+=1;
      if(i==0 && j==0){
         sim+= ((*f)(a,c));
      }
      if(i==0 && j!=0){
         if(j==k){
            sim+= (*f)(a,d);
         }else{
            if(j%2==0){
               sim+= 2*((*f)(a, c+j*h2));
            }else{
               sim+= 4*((*f)(a, c+j*h2));
            }
         }

      }
      if(i!=0 && j==0){
         if(i==n){
            sim+= (*f)(b,c);
         }else{
            if(i%2==0){
               sim+= 2*((*f)(a+i*h1, c));
            }else{
               sim+= 4*((*f)(a+i*h1, c));
            }
         }
      }
      if(i!=0 && j!=0){
         if(i==n && j==k){
            sim+= (*f)(b,d);
         }
         if(i==n && j!=k){
            if(j%2==0){
               sim+= 2*((*f)(b, c+j*h2));
            }else{
               sim+= 4*((*f)(b, c+j*h2));
            }
         }
         if(i!=n && j==k){
            if(i%2==0){
               sim+= 2*((*f)(a+i*h1, d));
            }else{
               sim+= 4*((*f)(a+i*h1, d));
            }
         }
         if(i!=n && j!=k){
            if(i%2!=0 && j%2!=0){
               sim+= 16*((*f)(a+i*h1, c+j*h2));
            }
            if(i%2==0 && j%2!=0){
               sim+= 8*((*f)(a+i*h1, c+j*h2));
```

```
            }
            if(i%2!=0 && j%2==0){
               sim+= 8*((*f)(a+i*h1, c+j*h2));
            }
            if(i%2==0 && j%2==0){
               sim+= 4*((*f)(a+i*h1, c+j*h2));
            }
         }
      }
   }
   sim*= (h2*h1)/9.;
   return sim;

}
```

\subsection{Polinomi interpolador de Lagrange}

\begin{lstlisting}

```
double func(int i, int j, double **mat, double x, double y, int n);

int main ( void ) {

   int n, i, j;
   double **mat;
   FILE* file;
   double x, y, res=0, error;

   printf("Dona'm la dimensio n de la matriu\n");
   scanf("%d", &n);

   printf("Dona'm el punt x,y per avaluar\n");
   scanf("%le %le", &x, &y);

   mat=(double**)malloc(n*sizeof(double*));
   for(i=0; i<n; i++){
      mat[i]=(double*)malloc(n*sizeof(double));
   }

   file= fopen("input", "r");

   for(i=0; i<n; i++){

      for(j=0; j<n; j++){
         if(!fscanf(file, "%le", &mat[i][j])){
            printf("input incorrecte\n");
            return 0;
         }

      }
   }

   for(i=1; i<n; i++){
```

```
      for(j=1; j<n; j++){
         res+= mat[i][j]*func(i,j, mat, x, y, n);
      }
   }

   error= fabs(res-sin(x+y));
   printf("El valor en interpolar en el punt (%le, %le) es %le \n", x, y, res);
   printf("amb un error de %le \n", error);

   fclose(file);
   return 0;
}

double func(int i, int j, double **mat, double x, double y, int n){

   double res=1;
   double lx, ly;
   int z;

   for(z=1; z<n; z++){
      if(z!=i){
         lx= (x-mat[z][0])/(mat[i][0]-mat[z][0]);
         res=res*lx;
      }
      if(z!=j){
         ly= (y-mat[0][z])/(mat[0][j]-mat[0][z]);
         res= res*ly;
      }
   }

   return res;

}
```

8.4 Interpolació triangle rectangle

```
FILE *fitxer;
double f ( double x, double y );
double trapezi ( double ( *f ) ( double, double ), double a, double b, double
    c, double d);

int main ( void ) {
   double a, b,c, d, eps, trap=0, error;
   int it=0;
   int cont=0;
   int it2=0;

   fitxer= fopen("punts avaluats", "wt");

   printf ( "doneu el vertex inferior esquerra (a, c) = \n" );
   scanf ( "%lf %lf", &a, &c );
```

```
   printf ( "doneu el vertex inferior dret(b, c) = \n" );
   scanf ( "%lf %lf", &b, &c );

   printf ( "doneu el vertex superior(a, d) = \n" );
   scanf ( "%lf %lf", &a, &d );

   printf ( "doneu tolerancia eps = \n" );
   scanf ( "%lf", &eps );

   while(c+(it+1)*eps< d){
      while((a+(it2+1)*eps)<(a+(c+(it+1)*eps-d)*(b-a)/(c-d))){
         trap+= trapezi(f, (a+it2*eps), a+(it2+1)*eps, c+it*eps, c+(it+1)*eps);
         it2+=1;
         cont+=1;
      }
      it2=0;
      it+=1;

   }

   error= trap-(sin(1)-cos(1));

   printf ("L'aproximacio de la integral de la funcio sobre el triangles es %le
       i s'han fet un total de %d subdivisions\n", trap, cont);
   printf("L'error es: %le\n", error);

   fclose(fitxer);
   return 0;
}



double trapezi ( double ( *f ) ( double, double ), double a, double b, double
    c, double d) {

   double trap;

   if(c<0.5){
      fprintf(fitxer, "%le %le\n", a, c);
      fprintf(fitxer, "%le %le\n", a, d);
      fprintf(fitxer, "%le %le\n", b, c);
      fprintf(fitxer, "%le %le\n", b, d);
   }
   trap = ( ( *f ) ( a, c ) + ( *f ) ( a,d ) + ( *f ) ( b,c ) + ( *f ) ( b,d ) )
       * (b-a)/ 2. *(d-c)/2.;
   return ( trap );
}
```

8.5 Interfície gràfica (Java)

Observació 1 En el següent codi java, a la classe IntegracióNumèrica per tema d'espai, s'ha obviat la implementació dels botons, que són de l'estil:

```
   private void jButtonXActionPerformed(java.awt.event.ActionEvent evt) {
        String str= this.jOperacio.getText();
        str+="x";
        this.jOperacio.setText(str);
    }
```

Observació 2 El codi està organitzat en tres classes. La classe Error, que s'encarrega de mostrar un missatge d'error quan l'usuari introdueix un input incorrecte. La classe integració, que conté el codi de Trapezis adaptatius. I per últim, la classe IntegracióNumèrica, que conté el codi de la interfície gràfica.

```
package integracio;

/**
 *
 * @author Vidal
 */
public class Error extends javax.swing.JDialog {

    String error;
    /**
     * Creates new form Error
     */
    public Error(java.awt.Frame parent, boolean modal, String error) {
        super(parent, modal);
        this.error= error;

        initComponents();
        this.jTextField1.setText(this.error);
    }


    private void jButtonACCEPTActionPerformed(java.awt.event.ActionEvent evt) {
        this.dispose();
    }

    /**
     * @param args the command line arguments
     */


    // Variables declaration - do not modify
    private javax.swing.JButton jButtonACCEPT;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JTextField jTextField1;
    // End of variables declaration
}

package integracio;
```

```
import org.mariuszgromada.math.mxparser.*;

/**
 *
 * @author Vidal
 */
public class Integracio {

    private String s;
    private Function f;
    private double a, b, c, d, eps;

    public Integracio(String s, double a, double b, double c, double d, double
        eps){
        this.s=s;
        this.a=a;
        this.b=b;
        this.c=c;
        this.d=d;
        this.eps=eps;
        this.f= new Function("f(x,y)="+s);

    }


    public double main(){

        double trap, adap;
   trap = trapezi ( a, b, c, d );
   adap = adapTrapezi ( a, b, c, d, eps, trap );

        return adap;
    }

    double adapTrapezi ( double a, double b,double c, double d, double eps,
        double sum ) {
   double esqa, esqb, dreta, dretb, z, t, adap;
   z = ( a + b ) / 2.;
   t = ( c + d ) / 2.;
   esqb = trapezi ( a,z,c,t );
   esqa = trapezi ( a, z,t, d );
   dreta= trapezi (z, b, t, d);
   dretb= trapezi( z, b, c, t);
   if ( Math.abs ( esqb + dreta + esqa + dretb - sum ) <= eps ) {
      adap = esqb + dreta + esqa + dretb;
      return ( adap );
   }
   adap = adapTrapezi ( a, z, c, t, eps / 4., esqb ) + adapTrapezi ( a, z, t, d,
       eps / 4., esqa ) + adapTrapezi ( z, b, t, d,   eps / 4., dreta ) +
       adapTrapezi ( z, b, c, t, eps / 4., dretb );
   return ( adap );
    }

    public double trapezi ( double a, double b, double c, double d ) {
   double trap;
```

```
   trap=
       (f.calculate(a,c)+f.calculate(a,d)+f.calculate(b,c)+f.calculate(b,d))*(b-a)/
       2. *(d-c)/2.;
        return ( trap );
    }


}

package integracio;

import java.awt.Color;
import java.awt.Container;
import java.awt.Font;
import java.awt.Graphics;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JLabel;

/**
 *
 * @author Vidal
 */
public class IntegracioNumerica extends javax.swing.JFrame {

    /**
     * Creates new form IntegracioNumerica
     */
    public IntegracioNumerica() {
        initComponents();
        this.setTitle("INTEGRACIO NUMERICA");


        Container c = this.getContentPane();
        c.setBackground(new java.awt.Color(204, 166, 166));

        this.jButtonNC.setEnabled(false);

    }


    private void jButtonDELActionPerformed(java.awt.event.ActionEvent evt) {
        this.jOperacio.setText("");
    }

    private void jButtonXActionPerformed(java.awt.event.ActionEvent evt) {
        String str= this.jOperacio.getText();
        str+="x";
        this.jOperacio.setText(str);
    }

    private void jButtonACActionPerformed(java.awt.event.ActionEvent evt) {
        String str= this.jOperacio.getText();
```

```
        if (str != null && str.length() > 0 ) {
            str = str.substring(0, str.length() - 1);
        }
        this.jOperacio.setText(str);
    }


    private void jButtonOkActionPerformed(java.awt.event.ActionEvent evt) {
        String s= this.jOperacio.getText();
        if (s.equals("")){
            Error e = new Error(this,true, "Introdueix una operacio");
            e.setTitle("Excepcio");
            e.pack();
            e.setVisible(true);
            return;
        }
        String a = this.jTextFielda.getText();
        String b = this.jTextFieldb.getText();
        String c = this.jTextFieldc.getText();
        String d = this.jTextFieldd.getText();
        String eps = this.jTextFieldeps.getText();
        try{
            double valuea = Double.parseDouble(a);
            double valueb = Double.parseDouble(b);
            double valuec = Double.parseDouble(c);
            double valued = Double.parseDouble(d);
            double valueeps = Double.parseDouble(eps);
            Integracio integracio= new Integracio(s, valuea, valueb, valuec,
                valued, valueeps);
            double res= integracio.main();
            String resultat= String.valueOf(res);
            this.jTextFieldRES.setText(resultat);
            this.jButtonNC.setEnabled(true);
        }catch(Exception e){
            String ex= e.getMessage();
            Error error = new Error(this,true, ex);
            error.setTitle("Excepcio");
            error.pack();
            error.setVisible(true);
        }

    }

    private void jButtonNCActionPerformed(java.awt.event.ActionEvent evt) {
        this.jButtonNC.setEnabled(false);
        this.jTextFieldRES.setText("");
        this.jTextFielda.setText("");
        this.jTextFieldb.setText("");
        this.jTextFieldc.setText("");
        this.jTextFieldd.setText("");
        this.jTextFieldeps.setText("");
        this.jOperacio.setText("");
    }

}
```

Referències

[1] Philip J. Davis, Philip Rabinowitz: *Methods of Numerical Integration*, 2nd Ed. Academic Press, 1984.

[2] Eugene Isaacson, Herbert Bishop Keller: *Analysis of Numerical Methods*, John Wiley & Sons, New York, 1966.

[3] Günther Hämmerlin, Karl-Heinz Hoffmann: *Numerical Mathematics*, Springer-Verlag, 1991.

[4] Sudhir R.Ghorpade, Balmohan V.Limaye: *A Course in Multivariable Calculus and Analysis*, Editorial Board, 2010.

Parser http://mathparser.org/