

Universitat de Lleida
Escola Politècnica Superior
Enginyeria Tècnica en Informàtica de Sistemes

Treball Final de Carrera

**Disseny, Implementació i Optimització
d'un Algorisme per al Càlcul de les
Constants d'Isocronia**

Autor: Albert Solé i Bosch

Directors de projecte: Jaume Giné i Josep M. Ribó

Bellpuig, 4 de Juliol del 2.009

PRÒLEG

Aquest treball consta, en la seva part dedicada als continguts, dels següents capítols:

El primer capítol, de caràcter introductori, enuncia el problema tractat de manera detallada; a partir d'aquest enunciat, enumera i justifica els objectius establerts per a la seva resolució i, finalment, descriu les eines utilitzades –tant pel que fa al programari com al maquinari- durant tot el procés.

En el segon capítol, es planteja matemàticament el problema principal d'aquest treball –el del càlcul de les constants d'isocronia- i es presenta un mètode –de naturalesa matemàtica- per a la seva resolució; així mateix, s'hi inclou un exemple de càlcul de les constants d'isocronia.

El tercer capítol es dedica a fer una descripció dels requeriments de disseny, tant dels de dades com dels d'algorismes, extrets a partir de l'estudi matemàtic del problema present en el capítol anterior.

El capítol quart inclou el disseny de dades i d'algorismes que es deriva dels requeriments enumerats en el capítol anterior.

En el capítol cinquè, es presenten els resultats obtinguts mitjançant l'eina de programari obtinguda a partir del disseny decidit en el capítol anterior. Es fa referència als fluxos d'entrada i sortida requerits per aquesta eina i es tracten qüestions de rendiment i temporització derivades de la implementació.

Finalment, el sisè capítol reuneix les conclusions extretes d'aquest treball i, alhora, exposa tot un seguit de possibles millores proposades per a futures ampliacions del mateix.

Abans d'acabar, els annexos contenen informació suplementària i de referència, com pot ser la localització del codi en llenguatge de programació *C++* de la implementació, exemples representatius i significatius dels diversos algorismes implementats i una introducció a la llibreria per al treball amb grans nombres *NTL* utilitzada en aquest treball.

Tanmateix, el treball també conté el present pròleg, índexs de referència –de contingut general, de taules, de figures, etc.- i una part dedicada al material bibliogràfic de referència utilitzat.

ÍNDIX

ÍNDIX GENERAL

1. Introducció	pàg. 1
1.1. Enunciat del problema	pàg. 1
1.2. Objectius proposats	pàg. 1
1.3. Eines utilitzades	pàg. 2
2. Aproximació matemàtica	pàg. 3
2.1. Introducció al problema matemàtic	pàg. 3
2.2. Consideracions prèvies	pàg. 4
2.3. Condicions algebraiques d'un centre. Constants de Poincaré-Liapunov	pàg. 8
2.4. El problema del centre-focus	pàg. 10
2.5. Resolució del problema del centre-focus	pàg. 14
2.6. El problema del centre isocró	pàg. 17
2.7. Resolució del problema del centre isocró	pàg. 22
3. Requeriments de disseny	pàg. 28
3.1. Algorisme de càlcul de les constants d'isocronia	pàg. 28
3.1.1. Algorisme de càlcul de les expressions simbòliques	pàg. 29
3.1.2. Algorisme de càlcul de les expressions trigonomètriques	pàg. 30
3.1.3 Algorisme de càlcul iteratiu/recursiu	pàg. 32
3.2. Requeriments de dades	pàg. 33
3.2.1. Requeriments de dades per a les expressions simbòliques	pàg. 34
3.2.2. Requeriments de dades de les expressions trigonomètriques	pàg. 35

3.3. Requeriments d'algorismes	pàg. 36
3.3.1. Requeriments d'algorismes per a les expressions simbòliques	pàg. 36
3.3.2. Requeriments d'algorismes per a les expressions trigonomètriques	pàg. 38
3.4. Resum de requeriments i consideracions prèvies al disseny	pàg. 38
4. Disseny i implementacions	pàg. 40
4.1. Visió general de les famílies d'estructures de dades implementades	pàg. 40
4.2. Família d'estructures de dades contenidores	pàg. 41
4.2.1. Definició de la família d'estructures de dades contenidores	pàg. 42
4.2.2. Interfície de la família d'estructures de dades contenidores	pàg. 43
4.2.3. Altres aspectes rellevants en el funcionament de la família d'estructures de dades contenidores	pàg. 48
4.3. Família d'estructures de dades simbòliques	pàg. 48
4.3.1. Definició de la família d'estructures de dades simbòliques	pàg. 48
4.3.2. Interfície de la família d'estructures de dades simbòliques	pàg. 50
4.4. Família d'estructures de dades trigonomètriques	pàg. 53
4.4.1. Definició de la família d'estructures de dades trigonomètriques	pàg. 54
4.4.2. Interfície de la família d'estructures de dades trigonomètriques	pàg. 55
4.5. Família d'estructures de dades numèriques	pàg. 58
4.5.1. Definició de la família d'estructures de dades numèriques	pàg. 59
4.5.2. Interfície de la família d'estructures de dades numèriques	pàg. 61
5. Resultats i rendiment	pàg. 65
5.1. Paràmetres d'entrada	pàg. 65
5.2. Dades de sortida	pàg. 65
5.3. Comparació del temps de càlcul	pàg. 66
6. Conclusions i millores	pàg. 68

6.1. Conclusions	pàg. 68
6.2. Possibles millores	pàg. 69
7. Annex	pàg. 70
7.1. Localització del codi en llenguatge de programació C++ de la implementació	pàg. 70
7.2. Exemples representatius i significatius dels diversos algorismes implementats	pàg. 71
7.2.1. Algorisme de cerca sobre llistes ordenades	pàg. 72
7.2.2. Algorismes que implementen operacions aritmètiques	pàg. 75
7.2.3. Integració de termes trigonomètrics	pàg. 79
7.3. Introducció a la llibreria NTL	pàg. 85
Bibliografia	pàg. 87

ÍNDIX DE QUADRES

2.1. Definició de sistema d'equacions diferencials autònom i lineal en el pla	pàg. 4
2.2. Definició de punt crític d'un sistema d'equacions diferencials autònom i lineal en el pla	pàg. 5
2.3. Definició de centre	pàg. 5
2.4. Classificació topològica dels punts crítics en sistemes pertorbats	pàg. 8
2.5. Definició d'integral primera d'un sistema d'equacions diferencials autònom en el pla	pàg. 9
2.6. Teorema 1	pàg. 9
2.7. Teorema 2 –de Poincaré-	pàg. 10
2.8. Definició de centre isocró	pàg. 17
2.9. Resolució de (2.17)	pàg. 23
3.1. Expressions calculades pel procés de càlcul de les expressions simbòliques	pàg. 29

3.2. Expressions calculades pel procés de càlcul de les expressions trigonomètriques	pàg. 31
3.3. Expressions calculades pel procés de càlcul iteratiu/recursiu	pàg. 32
4.1. Definició de la família d'estructures de dades contenidores	pàg. 42
4.2. Interfície de la família d'estructures de dades contenidores	pàg. 43
4.3. Definició de la família d'estructures de dades simbòliques	pàg. 49
4.4. Interfície de la família d'estructures de dades simbòliques	pàg. 51
4.5. Definició de la família d'estructures de dades trigonomètriques	pàg. 54
4.6. Interfície de la família d'estructures de dades trigonomètriques	pàg. 56
4.7. Definició de la família d'estructures de dades numèriques	pàg. 59
4.8. Interfície de la família d'estructures de dades numèriques	pàg. 61
7.1. Codi de l'operació de cerca	pàg. 72
7.2. Codi de les operacions aritmètiques més importants	pàg. 76
7.3. Codi de l'algorisme d'integració	pàg. 83

ÍNDIX DE TAULES

2.1. Termes de cada una de les potències de la variable r de (2.11) –cas quadràtic-	pàg. 15
2.2. Termes de cada una de les potències de la variable r de l'expressió anterior –menys els dos primers; cas quadràtic-	pàg. 22
5.1. Temps de càlcul de les constants d'isocronia per al cas quadràtic $-p = 2 -$	pàg. 66
5.2. Temps de càlcul de les constants d'isocronia per al cas cúbic $-p = 3 -$	pàg. 66
5.3. Temps de càlcul de les constants d'isocronia per al cas quàrtic $-p = 4 -$	pàg. 66
7.1. Casuística dels exponents p i q en l'algorisme del càlcul integral	pàg. 81

ÍNDIX DE FIGURES

2.1. Topologia de les solucions d'un sistema que defineix un centre i un focus en l'origen –espai de fases o pla x - y de les solucions del sistema-	pàg. 7
3.1. Diagrama de blocs de l'algorisme de càlcul de les constants d'isocronia	pàg. 28
3.2. Diagrama de blocs de l'algorisme de càlcul de les expressions simbòliques	pàg. 30
3.3. Diagrama de blocs de l'algorisme de càlcul de les expressions trigonomètriques	pàg. 31
3.4. Diagrama de blocs de l'algorisme de càlcul iteratiu/recursiu	pàg. 33
4.1. Esquema general de les famílies d'estructures de dades implementades	pàg. 40
4.2. Comportament dels iteradors i índexs	pàg. 47

1. INTRODUCCIÓ

En aquest capítol, s'enuncia el problema tractat de manera detallada; a partir d'aquest enunciat, s'enumeren i justifiquen els objectius establerts per a la seva resolució i, finalment, es descriuen les eines utilitzades –tant pel que fa al programari com al maquinari- durant tot el procés.

1.1. ENUNCIAT DEL PROBLEMA

Dissenyar, implementar i optimitzar un algorisme per a calcular les constants d'isocronia¹. Pel que fa al disseny i la implementació, les eines utilitzades seran el llenguatge de programació C++ i la llibreria per a treball amb grans nombres *NTL*; l'optimització, per la seva banda, s'efectuarà sobre els recursos disponibles: espai de memòria utilitzat i temps emprat per a l'execució de l'esmentat algorisme.

1.2. OBJECTIUS PROPOSATS

Vist l'anterior enunciat del problema, es proposen els següents objectius generals:

Desenvolupar una eina de programari capaç de calcular les constants d'isocronia a partir de les següents dades proporcionades per l'usuari: nombre màxim de constants d'isocronia a calcular –en concret, es proporcionarà l'ordre de l'última constant desitjada- i grau del sistema polinomial² requerit per l'usuari.

Les dades proporcionades per aquesta eina de programari en acabar la seva tasca seran, per una banda, les constants d'isocronia demanades per l'usuari i, per l'altra, dades referents a la temporització del mateix procés de càlcul. Així mateix, les constants d'isocronia esmentades hauran de ser guardades automàticament, de manera que puguin ser recuperades immediatament -un cop finalitzats els càlculs- per l'eina de programari *Mathematica*.

¹ El càlcul de les constants d'isocronia s'introdueix en el capítol 2.

² Cas quadràtic, cas cúbic, etc.

Els manipuladors algebraics³ són eines de programari dissenyades específicament per a treballar amb objectes simbòlics de càlcul. No obstant això, el seu rendiment en el càlcul de les constants d'isocronia no és òptim ja que no estan optimitzats per a treballar amb valors no numèrics, fet que és necessari en el càlcul de les constants d'isocronia degut a la utilització de coeficients arbitraris. És per això que un altre dels objectius del treball serà que l'eina de programari implementada executi el càlcul de les constants d'isocronia més ràpidament que els manipuladors algebraics disponibles.

Finalment, aquests càlculs hauran de ser òptims pel que fa a la utilització de recursos, tant d'espai de memòria com de temps d'execució; així, les dades referents a la temporització del procés de càlcul demanades anteriorment haurien de servir per a treballar en aquesta direcció.

La redacció d'aquesta memòria de treball i l'exposició del mateix també formen part d'aquests objectius.

1.3. EINES UTILITZADES

Les eines de programari utilitzades en la realització d'aquest treball han estat:

L'editor de text inclòs en *The Norton Commander 4.5* i el paquet integrat *Microsoft Office 2007*.

La llibreria per a treball amb grans nombres *NTL 5.4.1*.

El compilador de llenguatge C++ *djgpp 2.03*.

El manipulador algebraic *Mathematica 4*.

Els sistemes operatius *Microsoft Windows 98* i *Microsoft Windows XP*.

L'equip informàtic utilitzat per a fer la implementació ha estat un *Intel Pentium-100 Mhz.*; els tests de funcionament i de rendiment s'han realitzat en un *Intel Core 2 Duo-1,83 GHz*.

³ L'eina de programari *Mathematica* és un manipulador algebraic.

2. APROXIMACIÓ MATEMÀTICA

En aquest capítol, es planteja matemàticament el problema principal d'aquest treball –el càlcul de les constants d'isocronia- i es presenta un mètode –de naturalesa matemàtica, i on s'hi inclou, també, un exemple de càlcul- per a la seva resolució.

També es presentaran dos problemes matemàtics estretament relacionats entre si: el problema del centre-focus i el problema del centre isocró. El desenvolupament matemàtic d'aquests dos problemes permetrà entendre com s'ha de plantejar, en termes informàtics, el càlcul de les constants d'isocronia.

2.1. INTRODUCCIÓ AL PROBLEMA MATEMÀTIC

Les equacions diferencials que defineixen sistemes dinàmics continus són una de les eines més importants per tal de construir models que descriguin certs comportaments que es donen en la naturalesa. Per exemple, les equacions diferencials de *Lotka-Volterra* modelen l'evolució de les poblacions de les espècies i les equacions diferencials de *Lorenz* modelen sistemes meteorològics.

La teoria qualitativa intenta obtenir la màxima informació d'aquests sistemes sense conèixer les seves solucions ja que, en general, aquestes no es poden trobar. Alguns dels problemes no resolts completament –introduïts per *H. Poincaré*- i fortament relacionats entre si són:

Saber si un punt crític monodròmic definit per un sistema analític té el comportament d'un centre o d'un focus.

Determinar l'estabilitat d'aquests punts crítics.

Conèixer l'existència i el nombre de cicles límit locals al voltant d'aquests punts crítics.

Determinar les integrals primeres locals al voltant d'aquests punts crítics.

Determinar si un centre és isocró.

La resolució d'aquests problemes passa pel càlcul de les anomenades constants de Poincaré-Liapunov i d'isocronia mitjançant les tècniques matemàtiques desenvolupades pel mateix *H. Poincaré*. Tot això es veurà amb més detall tot seguit.

2.2. CONSIDERACIONS PRÈVIES

Sigui el següent sistema d'equacions diferencials autònom i lineal en el pla –veure el quadre 2.1- de la forma:

$$\begin{cases} \dot{x} = -y \\ \dot{y} = x \end{cases} \quad (2.1)$$

Quadre 2.1. Definició de sistema d'equacions diferencials autònom i lineal en el pla

S'anomena sistema d'equacions diferencials autònom en el pla a:

$$\begin{cases} \dot{x} = f(x, y) \\ \dot{y} = g(x, y) \end{cases} \quad (2.2)$$

on: $x, y \in \mathbb{R}$; $\dot{\cdot} = d/dt$; $f, g: \mathbb{R}^2 \rightarrow \mathbb{R} \in C^1(\mathbb{R}^2)$

En el cas que f i g siguin lineals, (2.2) s'anomena sistema lineal i es pot escriure de la forma:

$$\begin{cases} \dot{x} = ax + by \\ \dot{y} = cx + dy \end{cases} \quad (2.3)$$

on: $a, b, c, d \in \mathbb{R}$

També es pot escriure (2.3) de forma matricial:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = A \begin{pmatrix} x \\ y \end{pmatrix}$$

on: $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$

La matriu de coeficients de (2.1) és:

$$A = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

Els valors propis de A són les arrels del seu polinomi característic:

$$\begin{aligned} \bullet \quad P_A(\gamma) = 0 &\Rightarrow \det(A - \gamma I) = 0 \Rightarrow \begin{vmatrix} 0 - \gamma & -1 \\ 1 & 0 - \gamma \end{vmatrix} = 0 \Rightarrow \gamma^2 + 1 = 0 \\ &\Rightarrow \left. \begin{aligned} \gamma_1 &= i \\ \gamma_2 &= -i \end{aligned} \right\} \end{aligned}$$

D'aquesta manera, es pot veure que (2.1) posseeix un centre en el punt crític (0,0) -veure els quadres 2.2 i 2.3 i la figura 2.1-.

Quadre 2.2. Definició de punt crític d'un sistema d'equacions diferencials autònom i lineal en pla

S'anomena punt crític d'un sistema d'equacions diferencials autònom i lineal en el pla a:

$$(x_0, y_0) \in \mathbb{R}^2$$

on: (x_0, y_0) verifica que $\left. \begin{aligned} f(x_0, y_0) &= 0 \\ g(x_0, y_0) &= 0 \end{aligned} \right\}$

Quadre 2.3. Definició de centre

H. Poincaré va definir el concepte de centre per a un sistema d'equacions diferencials autònom en el pla $-i$, en particular, també per als lineals- com una singularitat isolada i envoltada per una banda contínua d'òrbites tancades; és a dir, un punt singular P que posseeix un entorn U de manera que qualsevol punt $P' \in U$ i diferent de P és no singular i la seva corba integral $-òrbita$ que passa per P' - és tancada i envolta P .

Per a classificar topològicament el punt crític d'un sistema d'equacions diferencials autònom i lineal en el pla es realitza el següent canvi de variables lineal:

$$\begin{pmatrix} u \\ v \end{pmatrix} = P \begin{pmatrix} x \\ y \end{pmatrix}$$

on: $u, v \in R$; $P = \begin{pmatrix} p & q \\ r & s \end{pmatrix}$ és invertible; $p, q, r, s \in R$

Aleshores:

$$\bullet \quad \begin{pmatrix} u \\ v \end{pmatrix} = P \begin{pmatrix} x \\ y \end{pmatrix} \Rightarrow \begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = P \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} \Rightarrow \begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = PA \begin{pmatrix} x \\ y \end{pmatrix} \Rightarrow \begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = PAP^{-1} \begin{pmatrix} u \\ v \end{pmatrix}$$

on: $PAP^{-1} = J$

Si J diagonalitza, la seva forma serà la següent:

$$J = \begin{pmatrix} \gamma_1 & 0 \\ 0 & \gamma_2 \end{pmatrix}$$

on: γ_1, γ_2 són els valors propis de A

Aleshores, aquest punt crític serà un:

Node estable per a $\gamma_1, \gamma_2 > 0$ i reals.

Node inestable per a $\gamma_1, \gamma_2 < 0$ i reals.

Punt de sella per a $\gamma_1 < 0 < \gamma_2$ i reals.

Punt d'estrella per a $\gamma_1 = \gamma_2$ i reals.

Focus per a $\gamma_1 = \alpha + i\beta, \gamma_2 = \alpha - i\beta$ i $\alpha \neq 0$.

Centre per a $\gamma_1 = \alpha + i\beta, \gamma_2 = \alpha - i\beta$ i $\alpha = 0$.

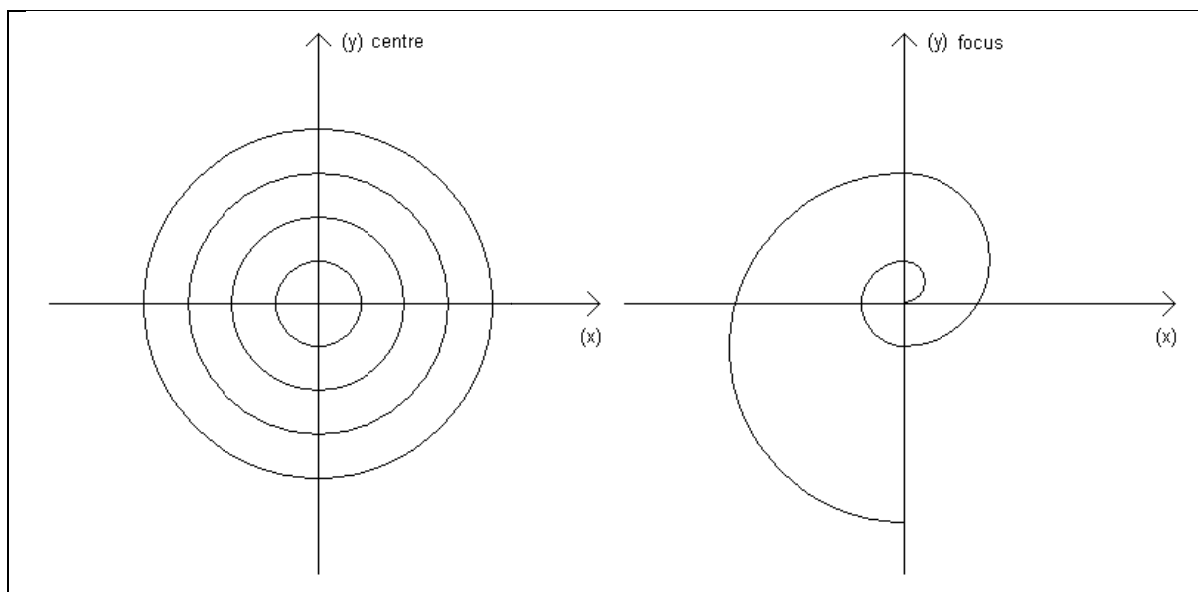
Si J no diagonalitza, la seva forma serà la següent:

$$J = \begin{pmatrix} \gamma & 1 \\ 0 & \gamma \end{pmatrix}$$

on: $\gamma \in R$

En aquest cas, el punt crític és un node degenerat.

Figura 2.1. Topologia de les solucions d'un sistema que defineix un centre i un focus en l'origen –espai de fases o pla x-y de les solucions del sistema-



Sigui ara el següent sistema d'equacions diferencials autònom i no lineal en el pla de la forma:

$$(2.4)$$

on: ; ; ;

Es pot entendre (2.4) com (2.1) pertorbat pels termes no lineals i . El determinant de la matriu de coeficients de la part lineal és diferent de zero i els valors propis d'aquesta són imaginaris purs conjugats. Aleshores, (2.4) posseeix un centre o un focus en el punt crític -per tot això, es poden fer consideracions anàlogues a les del cas del sistema no pertorbat; igualment, veure el quadre 2.4-

Quadre 2.4. *Classificació topològica dels punts crítics en sistemes pertorbats*

Donat el punt crític (x_0, y_0) d'un sistema d'equacions diferencials autònom i no lineal en el pla amb matriu de coeficients A i valors propis d'aquesta γ_1 i γ_2 , aquell es classifica com:

Node degenerat no elemental per a $\det(A) = 0$ i $\operatorname{tr}(A) = 0$.

Node degenerat elemental per a $\det(A) = 0$ i $\operatorname{tr}(A) \neq 0$.

Node no degenerat elemental per a $\det(A) \neq 0$; $\gamma_1, \gamma_2 > 0$ i reals.

Punt de sella no degenerat elemental per a $\det(A) \neq 0$; $\gamma_1, \gamma_2 < 0$ i reals.

Focus no degenerat elemental per a $\det(A) \neq 0$, $\gamma_1 = \alpha + i\beta$, $\gamma_2 = \alpha - i\beta$ i $\alpha \neq 0$.

Centre o focus no degenerats elementals per a $\det(A) \neq 0$, $\gamma_1 = \alpha + i\beta$,
 $\gamma_2 = \alpha - i\beta$ i $\alpha = 0$.

En aquest punt, hom es troba amb el problema plantejat per *H. Poincaré*: saber si un punt crític posseït per un sistema té el comportament d'un centre o d'un focus i saber quan aquest centre és isocró.

2.3. CONDICIONS ALGEBRAIQUES D'UN CENTRE. CONSTANTS DE POINCARÉ-LIAPUNOV

Segons el concepte d'integral primera –veure el quadre 2.5- i donat el *Teorema 1* –veure el quadre 2.6-, es pot deduir que $H(x, y)$ serà integral primera formal de (2.4) si, i només si, $\dot{H}(x, y) \equiv 0$ –o, el que és el mateix, si, i només si, s'anul·len les *constants de Poincaré-Liapunov*-. I, per altra banda, donat el *Teorema 2* –veure el quadre 2.7-, la condició anterior és necessària i suficient per tal que (2.4) tingui un centre en el seu punt crític $(0,0)$. Per acabar, pel *Teorema de la Base de Hilbert*, l'ideal I de polinomis format per totes les *constants de Poincaré-Liapunov* –és a dir: $I = \langle V_1, V_2, V_3, \dots \rangle$ – és finitament generat. Això vol dir que existeix $m \in \mathbb{N}$ de manera que $I = \langle V_1, V_2, V_3, \dots, V_m \rangle$. Per tant, només cal l'anul·lació d'aquestes *constants de Poincaré-Liapunov* per tal de complir les condicions anteriors.

Quadre 2.5. Definició d'integral primera d'un sistema d'equacions diferencials autònom en el pla

S'anomena integral primera d'un sistema d'equacions diferencials autònom en el pla de la forma (2.1) –amb variable auxiliar $t \in \mathbb{R}$ – en un subconjunt $U \subset \mathbb{R}^2$ a una funció $H: U \subset \mathbb{R}^2 \rightarrow \mathbb{R}$ de classe C^1 i no constant en U si la funció H és constant sobre qualsevol solució $(x(t), y(t))$ amb valor inicial $(x(0), y(0)) = (x_0, y_0)$; és a dir, $H(x(t), y(t)) = H(x_0, y_0)$ per a tot t per al qual la solució estigui definida. A més, $H(x, y)$ compleix que:

$$\dot{H}(x, y) = \frac{dH(x, y)}{dx} \dot{x} + \frac{dH(x, y)}{dy} \dot{y} = 0$$

Quadre 2.6. Teorema 1

Sigui un sistema polinomial de grau n de la forma:

$$\left. \begin{aligned} \dot{x} &= -y + \sum_{k=2}^n X_k(x, y) \\ \dot{y} &= x + \sum_{k=2}^n Y_k(x, y) \end{aligned} \right\} (2.5)$$

on: $X_k(x, y), Y_k(x, y)$ són polinomis homogenis de grau k

Aleshores, existeix una sèrie formal de potències de la forma:

$$H(x, y) = \sum_{k=2}^{\infty} H_k(x, y)$$

on: $H_k(x, y)$ són polinomis homogenis de grau k ; $H_2(x, y) = \frac{x^2 + y^2}{2}$

Finalment, es verifica que:

$$\dot{H}(x, y) = \frac{dH(x, y)}{dx} \dot{x} + \frac{dH(x, y)}{dy} \dot{y} = \sum_{k=1}^{\infty} V_{2k} (x^2 + y^2)^k$$

Els coeficients V_{2k} són les anomenades constants de Poincaré-Liapunov i, per a (2.5), són polinomials en els coeficients del sistema.

Quadre 2.7. Teorema 2 –de Poincaré–

L'origen de (2.5) és un centre si, i només si, en un entorn obert de l'origen, (2.5) admet una integral primera no constant i analítica.

Així doncs, si s'imposa l'anul·lació d'aquestes constants, es trobaran les condicions que facin que (2.4) tingui un centre en el seu punt crític (0,0).

2.4. EL PROBLEMA DEL CENTRE-FOCUS⁴

L'expressió de (2.4) –en el cas quadràtic- en coordenades cartesianes és:

$$\left. \begin{aligned} \dot{x} &= -y + X_2(x, y) \\ \dot{y} &= x + Y_2(x, y) \end{aligned} \right\} (2.6)$$

on: $X_2(x, y) = \sum_{i=0}^2 a_{2-i,i} x^{2-i} y^i$; $Y_2(x, y) = \sum_{i=0}^2 b_{2-i,i} x^{2-i} y^i$; $a_{2-i,i}, b_{2-i,i} \in R$; $n \in N$

El canvi de variables per a passar una expressió de coordenades cartesianes a polars és:

$$\left. \begin{aligned} x &= r \cos(\varphi) \\ y &= r \sin(\varphi) \end{aligned} \right\} (2.7)$$

on: $r, \varphi \in R$

⁴ Per tal de facilitar els càlculs, es considera únicament el cas quadràtic $-n = 2$ - i es treballa en coordenades polars.

L'expressió de $X_2(x, y)$ i $Y_2(x, y)$ en coordenades polars és:

- $X_2(r, \varphi) = a_{2,0}r^2\cos^2(\varphi)r^0\sin^0(\varphi) + a_{1,1}r^1\cos^1(\varphi)r^1\sin^1(\varphi) + a_{0,2}r^0\cos^0(\varphi)r^2\sin^2(\varphi) \Rightarrow X_2(r, \varphi) = r^2 \left(a_{2,0}\cos^2(\varphi)\sin^0(\varphi) + a_{1,1}\cos^1(\varphi)\sin^1(\varphi) + a_{0,2}\cos^0(\varphi)\sin^2(\varphi) \right) \Rightarrow X_2(r, \varphi) = r^2 X_2(\varphi)$
- $Y_2(r, \varphi) = b_{2,0}r^2\cos^2(\varphi)r^0\sin^0(\varphi) + b_{1,1}r^1\cos^1(\varphi)r^1\sin^1(\varphi) + b_{0,2}r^0\cos^0(\varphi)r^2\sin^2(\varphi) \Rightarrow Y_2(r, \varphi) = r^2 \left(b_{2,0}\cos^2(\varphi)\sin^0(\varphi) + b_{1,1}\cos^1(\varphi)\sin^1(\varphi) + b_{0,2}\cos^0(\varphi)\sin^2(\varphi) \right) \Rightarrow Y_2(r, \varphi) = r^2 Y_2(\varphi)$

on: $X_2(\varphi) = a_{2,0}\cos^2(\varphi)\sin^0(\varphi) + a_{1,1}\cos^1(\varphi)\sin^1(\varphi) + a_{0,2}\cos^0(\varphi)\sin^2(\varphi)$;
 $Y_2(\varphi) = b_{2,0}\cos^2(\varphi)\sin^0(\varphi) + b_{1,1}\cos^1(\varphi)\sin^1(\varphi) + b_{0,2}\cos^0(\varphi)\sin^2(\varphi)$

L'expressió de \dot{r} i $\dot{\varphi}$ és:

- $r^2 = x^2 + y^2 \Rightarrow 2r\dot{r} = 2x\dot{x} + 2y\dot{y} \Rightarrow \dot{r} = \frac{x\dot{x} + y\dot{y}}{r} \Rightarrow$
 $\Rightarrow \dot{r} = \frac{x(-y + X_2(x, y)) + y(x + Y_2(x, y))}{r} \Rightarrow$
 $\Rightarrow \dot{r} = \frac{r\cos(\varphi)(-r\sin(\varphi) + r^2X_2(\varphi)) + r\sin(\varphi)(r\cos(\varphi) + r^2Y_2(\varphi))}{r} \Rightarrow$
 $\Rightarrow \dot{r} = r^2(\cos(\varphi)X_2(\varphi) + \sin(\varphi)Y_2(\varphi)) \Rightarrow \dot{r} = r^2P_2(\varphi)$

$$\begin{aligned}
\bullet \quad \varphi = \arctg\left(\frac{y}{x}\right) &\Rightarrow \dot{\varphi} = \frac{1}{1+\left(\frac{y}{x}\right)^2} \frac{y_x - \dot{x}y}{x^2} \Rightarrow \dot{\varphi} = \frac{y_x - \dot{x}y}{x^2 + y^2} \Rightarrow \\
&\Rightarrow \dot{\varphi} = \frac{(x + Y_2(x, y))x - (-y + X_2(x, y))y}{x^2 + y^2} \Rightarrow \\
&\Rightarrow \dot{\varphi} = \frac{(r\cos(\varphi) + r^2 Y_2(\varphi))r\cos(\varphi) + (r\sin(\varphi) - r^2 X_2(\varphi))r\sin(\varphi)}{r^2} \Rightarrow \\
&\Rightarrow \dot{\varphi} = 1 + r(\cos(\varphi) Y_2(\varphi) - \sin(\varphi) X_2(\varphi)) \Rightarrow \dot{\varphi} = 1 + rQ_2(\varphi)
\end{aligned}$$

on: $P_2(\varphi) = \cos(\varphi) X_2(\varphi) + \sin(\varphi) Y_2(\varphi)$; $Q_2(\varphi) = \cos(\varphi) Y_2(\varphi) - \sin(\varphi) X_2(\varphi)$

Així, l'expressió de (2.6) en coordenades polars és:

$$\left. \begin{aligned} \dot{r} &= r^2 P_2(\varphi) \\ \dot{\varphi} &= 1 + rQ_2(\varphi) \end{aligned} \right\} (2.8)$$

L'expressió de la integral primera de (2.8) proposada per *H. Poincaré* –veure l'apartat 2.3– en coordenades cartesianes és:

$$H(x, y) = \sum_{k=2}^{\infty} H_k(x, y) \quad (2.9)$$

on: $H_k(x, y) = \sum_{i=0}^k c_{k-i,i} x^{k-i} y^i$; $c_{k-i,i} \in R$; $k \in N$; $c_{2,0} = \frac{1}{2}$; $c_{1,1} = 0$; $c_{0,2} = \frac{1}{2}$

S'utilitza el mateix canvi de variables (2.7). L'expressió de $H(x, y)$ en coordenades polars és:

$$\begin{aligned}
 & \bullet H(r, \varphi) = \\
 & = c_{2,0}r^2 \cos^2(\varphi) \sin^0(\varphi) + \\
 & + c_{1,1}r^2 \cos^1(\varphi) \sin^1(\varphi) + \\
 & + c_{0,2}r^2 \cos^0(\varphi) \sin^2(\varphi) + \\
 & + c_{3,0}r^3 \cos^3(\varphi) \sin^0(\varphi) + \\
 & + c_{2,1}r^3 \cos^2(\varphi) \sin^1(\varphi) + \\
 & + c_{1,2}r^3 \cos^1(\varphi) \sin^2(\varphi) + \\
 & + c_{0,3}r^3 \cos^0(\varphi) \sin^3(\varphi) + \\
 & + c_{4,0}r^4 \cos^4(\varphi) \sin^0(\varphi) + \\
 & + c_{3,1}r^4 \cos^3(\varphi) \sin^1(\varphi) + \\
 & + c_{2,2}r^4 \cos^2(\varphi) \sin^2(\varphi) + \\
 & + c_{1,3}r^4 \cos^1(\varphi) \sin^3(\varphi) + \\
 & + c_{0,4}r^4 \cos^0(\varphi) \sin^4(\varphi) + \dots \Rightarrow \\
 & \Rightarrow H(r, \varphi) = \sum_{k=2}^{\infty} r^k H_k(\varphi) = r^2 \sum_{m=0}^{\infty} r^m H_m(\varphi)
 \end{aligned}$$

on: $H_k(\varphi) = \sum_{i=0}^k c_{k-i,i} \cos^{k-i}(\varphi) \sin^i(\varphi)$; $H_m(\varphi) = H_k(\varphi)$ per a $m = k - 2$;
 $c_{k-i,i} \in R$; $k, m \in N$; $c_{2,0} = \frac{1}{2}$; $c_{1,1} = 0$; $c_{0,2} = \frac{1}{2}$

Així, l'expressió de (2.9) en coordenades polars és:

$$H(r, \varphi) = r^2 \sum_{m=0}^{\infty} r^m H_m(\varphi) \quad (2.10)$$

La derivada de (2.10) és:

$$\begin{aligned}
 \bullet \quad \dot{H}(r, \varphi) &= \frac{dH(r, \varphi)}{dr} \dot{r} + \frac{dH(r, \varphi)}{d\varphi} \dot{\varphi} \Rightarrow \\
 &\Rightarrow \dot{H}(r, \varphi) = \\
 &= \sum_{m=0}^{\infty} (m+2)r^{m+1}H_m(\varphi)r^2P_2(\varphi) + r^{m+2}\dot{H}_m(\varphi)(1+rQ_2(\varphi)) \Rightarrow \\
 &\Rightarrow \dot{H}(r, \varphi) = \\
 &= \sum_{m=0}^{\infty} (m+2)r^{m+3}P_2(\varphi)H_m(\varphi) + r^{m+2}\dot{H}_m(\varphi) + r^{m+3}Q_2(\varphi)\dot{H}_m(\varphi) \Rightarrow \\
 &\Rightarrow \dot{H}(r, \varphi) = \\
 &= r^2 \sum_{m=0}^{\infty} r^m \dot{H}_m(\varphi) + r^{m+1}((m+2)P_2(\varphi)H_m(\varphi) + Q_2(\varphi)\dot{H}_m(\varphi))
 \end{aligned}$$

Així, la derivada de (2.10) és:

$$\dot{H}(r, \varphi) = r^2 \sum_{m=0}^{\infty} r^m \dot{H}_m(\varphi) + r^{m+1}((m+2)P_2(\varphi)H_m(\varphi) + Q_2(\varphi)\dot{H}_m(\varphi)) \quad (2.11)$$

Si s'igualava (2.11) a zero, s'obté l'equació amb les condicions enunciades en l'apartat anterior.

2.5. RESOLUCIÓ DEL PROBLEMA DEL CENTRE-FOCUS

La taula 2.1 conté la llista de tots els termes de cada una de les potències de la variable r de (2.11) que s'han d'igualar a zero.

Taula 2.1. Termes de cada una de les potències de la variable r de (2.11) –cas quadràtic–

potència	termes
r^2	$\dot{H}_0(\varphi)$
r^3	$\dot{H}_1(\varphi) + 2P_2(\varphi)H_0(\varphi) + Q_2(\varphi)\dot{H}_0(\varphi)$
r^4	$\dot{H}_2(\varphi) + 3P_2(\varphi)H_1(\varphi) + Q_2(\varphi)\dot{H}_1(\varphi)$
\vdots	\vdots

De la taula anterior, se n'extreu la relació de recurrència següent, que és la que es computa per a realitzar el càlcul:

$$\left. \begin{aligned} \dot{H}_m(\varphi) &= -((m+1)P_2(\varphi)H_{m-1}(\varphi) + Q_2(\varphi)\dot{H}_{m-1}(\varphi)) \\ H_0(\varphi) &= \frac{1}{2} \\ \dot{H}_0(\varphi) &= 0 \end{aligned} \right\} (2.12)$$

El procediment per a calcular la constant de Poincaré-Liapunov V_m -en el cas quadràtic i per a una m donada) és:

Es calcula recursivament $\dot{H}_m(\varphi)$.

Per a qualsevol $H_j(\varphi)$, $0 \leq j \leq m$ calculada, es comprova que tots els seus termes responen a la seva forma general; és a dir, $H_j(\varphi)$, $0 \leq j \leq m$ és un polinomi trigonomètric –veure l'apartat 2.4–.

Si, per a qualsevol $H_j(\varphi)$, $0 \leq j \leq m$ calculada, existeixen termes que no responen a la seva forma general –la variable φ apareix fora d'una funció trigonomètrica– aquests termes formen la constant de Poincaré-Liapunov V_j -en concret, la forma el coeficient que acompanya la variable φ –.

Es continuen els càlculs ignorant aquests termes. Així, es veu com l'anul·lació de les constants de Poincaré-Liapunov respon a les condicions enunciades anteriorment –veure l'apartat 2.3–.

A tall d'exemple, es desenvolupa el càlcul de les constants de Poincaré-Liapunov fins a la primera constant. D'aquesta manera, l'algorisme recursiu condueix a calcular $H_j(\varphi)$ i $\dot{H}_j(\varphi)$ -des de $j = 1$ - a partir del cas base -per a $j = 0$ -⁵:

- $\dot{H}_1(\varphi) = -2P_2(\varphi)H_0(\varphi) - Q_2(\varphi)\dot{H}_0(\varphi) \Rightarrow \dot{H}_1(\varphi) = -P_2(\varphi)$
- $\dot{H}_1(\varphi) = -P_2(\varphi) \Rightarrow H_1(\varphi) = -\int P_2(\varphi)d\varphi \Rightarrow$
 $\Rightarrow H_1(\varphi) = -a_{2,0}\cos^2(\varphi)\sin(\varphi) + b_{0,2}\cos(\varphi)\sin^2(\varphi) +$
 $+ \left(\frac{a_{1,1} + b_{2,0} + 2b_{0,2}}{3}\right)\cos^3(\varphi) -$
 $- \left(\frac{2a_{2,0} + a_{0,2} + b_{1,1}}{3}\right)\sin^3(\varphi)$
- $\dot{H}_2(\varphi) = -3P_2(\varphi)H_1(\varphi) - Q_2(\varphi)\dot{H}_1(\varphi) \Rightarrow$
 $\Rightarrow \dot{H}_2(\varphi) = -3P_2(\varphi)H_1(\varphi) + Q_2(\varphi)P_2(\varphi)$
- $\dot{H}_2(\varphi) = -3P_2(\varphi)H_1(\varphi) + Q_2(\varphi)P_2(\varphi) \Rightarrow$
 $\Rightarrow H_2(\varphi) = -3\int P_2(\varphi)H_1(\varphi)d\varphi + \int Q_2(\varphi)P_2(\varphi)d\varphi \Rightarrow$
 $\Rightarrow H_2(\varphi) = V_2\varphi + \dots$ ⁶

$$\text{on: } V_2 = \frac{-a_{2,0}a_{1,1} - a_{1,1}a_{0,2} + 2a_{2,0}b_{2,0} - 2a_{0,2}b_{0,2} + b_{2,0}b_{1,1} + b_{1,1}b_{0,2}}{8}$$

El primer terme de $H_2(\varphi)$ no respon a la seva forma general -no pertany a un polinomi trigonomètric-. El coeficient de la variable φ forma la primera constant de Poincaré-Liapunov per al cas quadràtic - V_2 -. Es continuen els càlculs de la mateixa manera, ignorant els termes estranys - $H_j(\varphi) = H_j(\varphi) - V_j\varphi$ - i deduïnt-ne les constants de Poincaré-Liapunov que corresponguin.

⁵ Càlculs realitzats amb el programa *Mathematica 4* -veure l'arxiu *CalculCentreFocus.nb*-.

⁶ S'han omès els termes que responen a la forma general de $H_2(\varphi)$.

2.6. EL PROBLEMA DEL CENTRE ISOCRÓ

Sigui el sistema d'equacions diferencials autònom i lineal en el pla (2.1). La seva expressió en coordenades polars s'obté de la següent manera:

- $r^2 = x^2 + y^2 \Rightarrow 2r\dot{r} = 2x\dot{x} + 2y\dot{y} \Rightarrow \dot{r} = \frac{x\dot{x} + y\dot{y}}{r} \Rightarrow \dot{r} = \frac{-xy + xy}{r} \Rightarrow \dot{r} = 0$
- $\varphi = \operatorname{arctg}\left(\frac{y}{x}\right) \Rightarrow \dot{\varphi} = \frac{1}{1 + \left(\frac{y}{x}\right)^2} \frac{\dot{y}x - x\dot{y}}{x^2} \Rightarrow \dot{\varphi} = \frac{\dot{y}x - x\dot{y}}{x^2 + y^2} \Rightarrow \dot{\varphi} = \frac{xx + yy}{x^2 + y^2} \Rightarrow \dot{\varphi} = 1$

Així, l'expressió de (2.1) en coordenades polars és:

$$\left. \begin{array}{l} \dot{r} = 0 \\ \dot{\varphi} = 1 \end{array} \right\} (2.13)$$

L'expressió (2.13) indica que el centre que posseïx aquest sistema és isocró –veure el quadre 2.8-:

- $\dot{\varphi} = 1 \Rightarrow \frac{d\varphi}{dt} = 1 \Rightarrow dt = d\varphi \Rightarrow T = \int_0^{2\pi} d\varphi \Rightarrow T = 2\pi$

Quadre 2.8. Definició de centre isocró

Sigui el punt (x_0, y_0) centre d'un sistema d'equacions diferencials autònom en el pla; (x_0, y_0) serà isocró si, i només si, el temps de recorregut per a qualsevol òrbita tancada al voltant d'aquest punt és constant i el mateix –el període T és constant per a qualsevol d'aquestes òrbites-.

Sigui ara el sistema d'equacions diferencials autònom i no lineal en el pla (2.4); si aquest defineix un centre isocrò, es pot demostrar que el següent canvi de variables el transforma en un sistema de la forma (2.1):

$$\left. \begin{aligned} X &= x + \sum_{k=2}^{\infty} \tilde{X}_k(x, y) \\ Y &= y + \sum_{k=2}^{\infty} \tilde{Y}_k(x, y) \end{aligned} \right\}$$

on: $\tilde{X}_k(x, y) = \sum_{i=0}^k \tilde{a}_{k-i,i} x^{k-i} y^i$; $\tilde{Y}_k(x, y) = \sum_{i=0}^k \tilde{b}_{k-i,i} x^{k-i} y^i$; $\tilde{a}_{k-i,i}, \tilde{b}_{k-i,i} \in \mathbb{R}$;
 $k \in \mathbb{N}$

Si s'operen els termes de (2.1) –expressats en les noves variables- de la següent manera, s'arriba a una nova condició que ha de complir el mateix sistema (2.1):

$$\bullet \left. \begin{aligned} \dot{X} &= -Y \\ \dot{Y} &= X \end{aligned} \right\} \Rightarrow \left. \begin{aligned} \ddot{X} &= -\dot{Y} \\ \ddot{Y} &= \dot{X} \end{aligned} \right\} \Rightarrow \left. \begin{aligned} \ddot{X} &= -X \\ \ddot{Y} &= -Y \end{aligned} \right\} \Rightarrow \left. \begin{aligned} \ddot{X} + X &= 0 \\ \ddot{Y} + Y &= 0 \end{aligned} \right\}$$

I, en general:

$$\dot{H} + H = 0 \quad (2.14)$$

on: H representa una de les dues variables, X o Y ⁷

⁷ A partir d'ara es prendrà $H \equiv X$.

Així doncs, donada l'expressió de H i imposant la condició de (2.14), s'obtenen les condicions d'isocronia –condicions que permeten a (2.4), expressat en les noves variables i transformat en un sistema de la forma (2.1), definir un centre isocró en el seu punt crític $(0,0)$. D'aquests càlculs⁸, se n'extreuen les constants d'isocronia tal i com es veurà més endavant.

Primerament, l'expressió de H és:

$$\bullet \quad H = X \Rightarrow H = x + \sum_{k=2}^{\infty} \tilde{X}_k(x, y)$$

$$\text{on: } \tilde{X}_k(x, y) = \sum_{i=0}^k \tilde{a}_{k-i,i} x^{k-i} y^i; \tilde{a}_{k-i,i} \in R; k \in N$$

Així, l'expressió de H és:

$$H = x + \sum_{k=2}^{\infty} \tilde{X}_k(x, y) \quad (2.15)$$

Per a calcular l'expressió de (2.15) en coordenades polars, s'utilitza el mateix canvi de variables (2.7). Primerament, s'ha de calcular l'expressió de $\tilde{X}_k(x, y)$ en coordenades polars:

$$\begin{aligned} \bullet \quad \tilde{X}_k(r, \varphi) &= \sum_{i=0}^k \tilde{a}_{k-i,i} r^{k-i} \cos^{k-i}(\varphi) r^i \sin^i(\varphi) \Rightarrow \\ &\Rightarrow \tilde{X}_k(r, \varphi) = r^k \sum_{i=0}^k \tilde{a}_{k-i,i} \cos^{k-i}(\varphi) \sin^i(\varphi) \end{aligned}$$

$$\text{on: } \tilde{a}_{k-i,i} \in R; k \in N$$

⁸ Per tal de facilitar els càlculs, es considera únicament el cas quadràtic $-n = 2$ i es treballa en coordenades polars.

Aleshores, l'expressió de H en coordenades polars és:

$$\bullet H = r \cos(\varphi) + \sum_{k=2}^{\infty} \left(r^k \sum_{i=0}^k \tilde{a}_{k-i,i} \cos^{k-i}(\varphi) \sin^i(\varphi) \right)$$

on: $\tilde{a}_{k-i,i} \in \mathbb{R}; k \in \mathbb{N}$

Així, generalitzant, l'expressió de H en coordenades polars és:

$$H = \sum_{k=0}^{\infty} r^k H_k(\varphi)$$

on: $H_0(\varphi) = 0; H_1(\varphi) = \cos(\varphi); H_{k \geq 2}(\varphi) = \sum_{i=0}^k \tilde{a}_{k-i,i} \cos^{k-i}(\varphi) \sin^i(\varphi)$

Seguidament, i tenint en compte (2.8), l'expressió de \dot{H} és⁹:

$$\begin{aligned} \bullet H &= \sum_{k=0}^{\infty} r^k H_k \Rightarrow H = r^0 H_0 + r^1 H_1 + r^2 H_2 + \dots \\ \bullet \dot{H} &= \frac{dH}{dr} \dot{r} + \frac{dH}{d\varphi} \dot{\varphi} \Rightarrow \\ &\Rightarrow \dot{H} = (1r^0 H_1 + 2r^1 H_2 + \dots) \dot{r} + (r^0 \dot{H}_0 + r^1 \dot{H}_1 + r^2 \dot{H}_2 + \dots) \dot{\varphi} \Rightarrow \\ &\Rightarrow \dot{H} = \\ &= r^0 (\dot{H}_0) + \\ &+ r^1 (\dot{H}_1 + Q \dot{H}_0) + \\ &+ r^2 (\dot{H}_2 + Q \dot{H}_1 + 1P H_1) + \\ &+ r^3 (\dot{H}_3 + Q \dot{H}_2 + 2P H_2) + \dots \end{aligned}$$

⁹ Per tal de facilitar la notació matemàtica, a partir d'ara s'utilitzarà la següent convenció: $H_k = H_k(\varphi)$, $P = P_2(\varphi)$ i $Q = Q_2(\varphi)$.

- $$\begin{aligned} \ddot{H} &= \frac{d\dot{H}}{dr} \dot{r} + \frac{d\dot{H}}{d\varphi} \dot{\varphi} \Rightarrow \\ &\Rightarrow \ddot{H} = (0r^{-1}(\dot{H}_0) + 1r^0(\dot{H}_1 + Q\dot{H}_0) + 2r^1(\dot{H}_2 + Q\dot{H}_1 + 1PH_1) + \dots) \dot{r} \\ &\quad + (r^0(\ddot{H}_0) + \\ &\quad + r^1(\ddot{H}_1 + \dot{Q}\dot{H}_0 + Q\ddot{H}_0) + \\ &\quad + r^2(\ddot{H}_2 + \dot{Q}\dot{H}_1 + Q\ddot{H}_1 + 1\dot{P}H_1 + 1P\dot{H}_1) + \dots) \dot{\varphi} \Rightarrow \\ &\Rightarrow \ddot{H} = \\ &= r^0(\ddot{H}_0) + \\ &\quad + r^1(\ddot{H}_1 + 2Q\ddot{H}_0 + \dot{Q}\dot{H}_0) + \\ &\quad + r^2(\ddot{H}_2 + 2Q\ddot{H}_1 + (2P + \dot{Q})\dot{H}_1 + 1\dot{P}H_1 + Q^2\ddot{H}_0 + (1PQ + Q\dot{Q})\dot{H}_0) + \\ &\quad + r^3(\ddot{H}_3 + 2Q\ddot{H}_2 + (4P + \dot{Q})\dot{H}_2 + 2\dot{P}H_2 + Q^2\ddot{H}_1 + (3PQ + Q\dot{Q})\dot{H}_1) + \\ &\quad + r^4(\ddot{H}_4 + 2Q\ddot{H}_3 + (6P + \dot{Q})\dot{H}_3 + 3\dot{P}H_3 + Q^2\ddot{H}_2 + (5PQ + Q\dot{Q})\dot{H}_2) + \\ &\quad + r^5(\ddot{H}_5 + 2Q\ddot{H}_4 + (8P + \dot{Q})\dot{H}_4 + 4\dot{P}H_4 + Q^2\ddot{H}_3 + (7PQ + Q\dot{Q})\dot{H}_3) + \\ &\quad + r^3((2P^2 + 1\dot{P}Q)H_1) + \\ &\quad + r^4((6P^2 + 2\dot{P}Q)H_2) + \\ &\quad + r^5((12P^2 + 3\dot{P}Q)H_3) + \dots \end{aligned}$$

Finalment, l'expressió de $\ddot{H} + H$ és:

- $$\begin{aligned} \ddot{H} + H &= \\ &= r^0(\ddot{H}_0 + H_0) + \\ &\quad + r^1(\ddot{H}_1 + H_1 + 2Q\ddot{H}_0 + \dot{Q}\dot{H}_0) + \\ &\quad + r^2(\ddot{H}_2 + H_2 + 2Q\ddot{H}_1 + (2P + \dot{Q})\dot{H}_1 + 1\dot{P}H_1 + Q^2\ddot{H}_0 + (1PQ + Q\dot{Q})\dot{H}_0) \\ &\quad + r^3(\ddot{H}_3 + H_3 + 2Q\ddot{H}_2 + (4P + \dot{Q})\dot{H}_2 + 2\dot{P}H_2 + Q^2\ddot{H}_1 + (3PQ + Q\dot{Q})\dot{H}_1) \\ &\quad + r^4(\ddot{H}_4 + H_4 + 2Q\ddot{H}_3 + (6P + \dot{Q})\dot{H}_3 + 3\dot{P}H_3 + Q^2\ddot{H}_2 + (5PQ + Q\dot{Q})\dot{H}_2) \\ &\quad + r^5(\ddot{H}_5 + H_5 + 2Q\ddot{H}_4 + (8P + \dot{Q})\dot{H}_4 + 4\dot{P}H_4 + Q^2\ddot{H}_3 + (7PQ + Q\dot{Q})\dot{H}_3) \\ &\quad + r^3((2P^2 + 1\dot{P}Q)H_1) + \\ &\quad + r^4((6P^2 + 2\dot{P}Q)H_2) + \\ &\quad + r^5((12P^2 + 3\dot{P}Q)H_3) + \dots \end{aligned}$$

Si s'igualava cada una de les potències de la variable r de l'expressió anterior a zero, s'obtenen les condicions d'isocronia $-\dot{H} + H = 0$.

2.7. RESOLUCIÓ DEL PROBLEMA DEL CENTRE ISOCRÓ

La taula 2.2 conté la llista de tots els termes de cada una de les potències de la variable r de l'expressió anterior que s'han d'igualar a zero –s'ometen els dos primers–:

Taula 2.2. Termes de cada una de les potències de la variable r de l'expressió anterior –menys els dos primers; cas quadràtic–

potència	termes
r^0	0
r^1	$2Q\ddot{H}_0 + \dot{Q}\dot{H}_0$
r^2	$2Q\ddot{H}_1 + (2P + \dot{Q})\dot{H}_1 + 1\dot{P}H_1 + Q^2\ddot{H}_0 + (1PQ + Q\dot{Q})\dot{H}_0$
r^3	$2Q\ddot{H}_2 + (4P + \dot{Q})\dot{H}_2 + 2\dot{P}H_2 + Q^2\ddot{H}_1 + (3PQ + Q\dot{Q})\dot{H}_1 + (2P^2 + 1\dot{P}Q)H_1$
r^4	$2Q\ddot{H}_3 + (6P + \dot{Q})\dot{H}_3 + 3\dot{P}H_3 + Q^2\ddot{H}_2 + (5PQ + Q\dot{Q})\dot{H}_2 + (6P^2 + 2\dot{P}Q)H_2$
\vdots	\vdots

De la taula anterior, se n'extreu la relació de recurrència següent, que és la que es computa per a realitzar el càlcul:

$$\left. \begin{array}{l} \ddot{H}_m + H_m = F_m \\ H_0 = 0 \\ \dot{H}_0 = 0 \\ \ddot{H}_0 = 0 \\ H_1 = \cos(\varphi) \\ \dot{H}_1 = -\sin(\varphi) \\ \ddot{H}_1 = -\cos(\varphi) \end{array} \right\} (2.16)$$

$$\text{on: } F_m = -(2Q\ddot{H}_{m-1} + (2(m-1)P + \dot{Q})\dot{H}_{m-1} + (m-1)\dot{P}H_{m-1} + Q^2\ddot{H}_{m-2} + \\ + ((2(m-1) - 1)PQ + Q\dot{Q})\dot{H}_{m-2} + ((m-1)(m-2)P^2 + (m-2)\dot{P}Q)H_{m-2})$$

Abans de donar el procediment per a calcular les constants d'isocronia –en el cas quadràtic-, es fa evident que, per a avaluar (2.16), s'ha de trobar la solució de la següent equació diferencial lineal no homogènia de segon grau:

$$y'' + y = f(x) \quad (2.17)$$

Per a resoldre (2.17), es resol la seva equació diferencial lineal homogènia de segon grau associada i s'aplica el mètode de variació de les constants. Així, es pot veure que la solució particular de (2.17) és –veure el quadre 2.9-:

$$y_p = -\left(\int \sin(x) f(x) dx\right) \cos(x) + \left(\int \cos(x) f(x) dx\right) \sin(x)$$

Quadre 2.9. Resolució de (2.17)

Primerament, es calcula la solució general $-y_h-$ de la següent equació diferencial lineal homogènia de segon grau associada a (2.17):

$$y_h'' + y_h = 0 \quad (2.18)$$

Seguidament, s'aplica el mètode de variació de les constants a la solució calculada anteriorment per a trobar una solució particular $-y_p-$ de (2.17). Es considera que la solució general de (2.18) serà de la forma $y_h = e^{mx}$, ja que les derivades successives d'aquesta funció –i ella mateixa- tenen la mateixa forma; així mateix, una combinació lineal d'aquestes derivades –i també de la mateixa funció- és susceptible de ser nul·la.

Es comprova quina forma ha de tenir m :

- $y_h = e^{mx} \Rightarrow y_h' = m e^{mx}$
- $y_h' = m e^{mx} \Rightarrow y_h'' = m^2 e^{mx}$
- $y_h'' + y_h = 0 \Rightarrow m^2 e^{mx} + e^{mx} = 0 \Rightarrow e^{mx}(m^2 + 1) = 0$

on: $e^{mx} \neq 0$, per a qualsevol m i x

Aleshores:

$$\bullet \quad m^2 + 1 = 0 \Rightarrow \left. \begin{array}{l} m_1 = i \\ m_2 = -i \end{array} \right\} \Rightarrow \left. \begin{array}{l} y_{h1} = e^{ix} \\ y_{h2} = e^{-ix} \end{array} \right\} \Rightarrow \left. \begin{array}{l} y_{h1} = \cos(x) + i\sin(x) \\ y_{h2} = \cos(x) - i\sin(x) \end{array} \right\}$$

Per altra banda, es pot veure que una funció $w(x) = u(x) + iv(x)$ és solució de (2.18) si, i només si, $u(x)$ i $v(x)$ són solució de (2.18) –tenint en compte que $w''(x) = u''(x) + iv''(x)$ –:

$$\bullet \quad \left. \begin{array}{l} u''(x) + u(x) = 0 \\ v''(x) + v(x) = 0 \end{array} \right\} \Rightarrow \left. \begin{array}{l} u''(x) + u(x) = 0 \\ i(v''(x) + v(x)) = i0 \end{array} \right\} \Rightarrow \\ \Rightarrow u''(x) + iv''(x) + u(x) + iv(x) = 0 \Rightarrow w''(x) + w(x) = 0$$

Així, es tenen dues noves solucions –linealment independents- de (2.18)¹⁰:

$$\left. \begin{array}{l} y_{h3} = \cos(x) \\ y_{h4} = \sin(x) \end{array} \right\}$$

Finalment, també es pot veure que, si dues solucions de (2.18) $-y_i$ i y_j - són linealment independents, aleshores $y_h = c_1 y_i + c_2 y_j$ és la solució general de (2.18) –amb c_1 i c_2 paràmetres constants-. Així, la solució general de (2.18) és:

$$y_h = c_1 \cos(x) + c_2 \sin(x) \quad (2.19)$$

Per a aplicar el mètode de variació de les constants a (2.19), es consideren les constants introduïdes en (2.19) com funcions que depenen de x i es deriva –amb $y_1 = \cos(x)$ i $y_2 = \sin(x)$ –:

$$\bullet \quad y_h = c_1 y_1 + c_2 y_2 \Rightarrow y'_h = c'_1 y_1 + c_1 y'_1 + c'_2 y_2 + c_2 y'_2 \Rightarrow \\ \Rightarrow y'_h = c_1 y'_1 + c_2 y'_2 + c'_1 y_1 + c'_2 y_2$$

$$\text{on: } c'_1 y_1 + c'_2 y_2 = 0 \quad (2.20)$$

Aleshores:

$$\bullet \quad y'_h = c_1 y'_1 + c_2 y'_2 \Rightarrow y''_h = c'_1 y'_1 + c_1 y''_1 + c'_2 y'_2 + c_2 y''_2$$

¹⁰ Es pren el signe positiu de la segona solució per conveni.

Se substitueixen les expressions calculades anteriorment en (2.17):

$$\begin{aligned} \bullet \quad y'' + y = f(x) &\Rightarrow c_1' y_1'' + c_1 y_1' + c_2' y_2'' + c_2 y_2' + c_1 y_1 + c_2 y_2 = f(x) \Rightarrow \\ &\Rightarrow c_1 (y_1'' + y_1) + c_2 (y_2'' + y_2) + c_1' y_1' + c_2' y_2' = f(x) \end{aligned}$$

$$\text{on: } y_1'' + y_1 = 0; y_2'' + y_2 = 0$$

Aleshores:

$$\bullet \quad c_1' y_1' + c_2' y_2' = f(x) \quad (2.21)$$

Es resol el sistema format per (2.20) i (2.21) –on ara les incògnites són c_1' i c_2' –:

$$\left. \begin{aligned} y_1 c_1' + y_2 c_2' &= 0 \\ y_1' c_1' + y_2' c_2' &= f(x) \end{aligned} \right\}$$

Segons el mètode de *Cramer*:

$$\begin{aligned} \bullet \quad c_1' &= \frac{\begin{vmatrix} 0 & y_2 \\ f(x) & y_2' \end{vmatrix}}{\begin{vmatrix} y_1 & y_2 \\ y_1' & y_2' \end{vmatrix}} \Rightarrow c_1' = \frac{-y_2 f(x)}{y_1 y_2' - y_1' y_2} \Rightarrow c_1' = \frac{-\sin(x) f(x)}{\cos^2(x) + \sin^2(x)} \Rightarrow \\ &\Rightarrow c_1' = -\sin(x) f(x) \\ \bullet \quad c_2' &= \frac{\begin{vmatrix} y_1 & 0 \\ y_1' & f(x) \end{vmatrix}}{\begin{vmatrix} y_1 & y_2 \\ y_1' & y_2' \end{vmatrix}} \Rightarrow c_2' = \frac{y_1 f(x)}{y_1 y_2' - y_1' y_2} \Rightarrow c_2' = \frac{\cos(x) f(x)}{\cos^2(x) + \sin^2(x)} \Rightarrow \\ &\Rightarrow c_2' = \cos(x) f(x) \end{aligned}$$

Finalment, doncs, la solució particular buscada és:

$$y_p = -\left(\int \sin(x) f(x) dx\right) \cos(x) + \left(\int \cos(x) f(x) dx\right) \sin(x)$$

Finalment, el procediment per a calcular les constants d'isocronia –en el cas quadràtic– és el següent:

$$\text{Es resol recursivament } \ddot{H}_m + H_m = F_m.$$

Per a qualsevol H_j , $0 \leq j \leq m$ calculada, es comprova que tots els seus termes responen a la seva forma general; és a dir, H_j és un polinomi trigonomètric –veure l'apartat 2.6–.

Si, per a qualsevol H_j , $0 \leq j \leq m$ calculada, existeixen termes que no responen a la seva forma general –la variable φ apareix fora d'una funció trigonomètrica- aquests termes formen la constant de Poincaré-Liapunov V_j i la constant de període W_j -en concret, si existeixen termes de la forma $\varphi \cos(\varphi)$, els seus coeficients formen la constant de Poincaré-Liapunov –mòdul les anteriors zero-; i si existeixen termes de la forma $\varphi \sin(\varphi)$, els seus coeficients formen la constant de període-.

Es continuen els càlculs ignorant aquests termes –mòdul els anteriors zero-.

A tall d'exemple, es desenvolupa el càlcul de les constants de Poincaré-Liapunov i les constants de període fins a la primera constant. Així doncs, l'algorisme recursiu condueix a calcular H_j i $\ddot{H}_j + H_j$ -des de $j = 2$ - a partir del cas base –per a $j = 0$ i $j = 1$ -¹¹:

- $\ddot{H}_2 + H_2 =$
 $= -(2Q\ddot{H}_1 + (2P + \dot{Q})\dot{H}_1 + \dot{P}H_1 + Q^2\ddot{H}_0 + (PQ + Q\dot{Q})\dot{H}_0) = F_2 \Rightarrow$
 $\Rightarrow H_2 = -\left(\int \sin(\varphi)F_2 d\varphi\right) \cos(\varphi) + \left(\int \cos(\varphi)F_2 d\varphi\right) \sin(\varphi) \Rightarrow$
 $\Rightarrow H_2 = \left(\frac{b+d+2f}{3}\right) \cos^2(\varphi) + \left(\frac{-2a+2c-e}{3}\right) \cos(\varphi) \sin(\varphi) +$
 $+ \left(\frac{-b+2d+f}{3}\right) \sin^2(\varphi)$
- $\ddot{H}_3 + H_3 =$
 $= -(2Q\ddot{H}_2 + (4P + \dot{Q})\dot{H}_2 + 2\dot{P}H_2 + Q^2\ddot{H}_1 + (3PQ + Q\dot{Q})\dot{H}_1 +$
 $+ (2P^2 + \dot{P}Q)H_1) = F_3 \Rightarrow$
 $\Rightarrow H_3 = -\left(\int \sin(\varphi)F_3 d\varphi\right) \cos(\varphi) + \left(\int \cos(\varphi)F_3 d\varphi\right) \sin(\varphi) \Rightarrow$
 $\Rightarrow H_3 = V_3 \varphi \cos(\varphi) + P_3 \varphi \sin(\varphi) + \dots$ ¹²

$$\text{on: } V_3 = \left(\frac{-a_{2,0}a_{1,1} - a_{1,1}a_{0,2} + 2a_{2,0}b_{2,0} - 2a_{0,2}b_{0,2} + b_{2,0}b_{1,1} + b_{1,1}b_{0,2}}{8}\right);$$

$$W_3 = \left(\frac{-4a_{2,0}^2 - a_{1,1}^2 - 10a_{0,2}^2 - 10b_{2,0}^2 - b_{1,1}^2 - 4b_{0,2}^2 - 10a_{2,0}a_{0,2} + 5a_{2,0}b_{1,1}}{24}\right) +$$

$$+ \left(\frac{a_{1,1}b_{2,0} + a_{0,2}b_{1,1} + 5a_{1,1}b_{0,2} - 10b_{2,0}b_{0,2}}{24}\right)$$

¹¹ Càlculs realitzats amb el programa Mathematica 4 –veure l'arxiu CalculCentrelsocro.nb-.

¹² S'han omès els termes que responen a la forma general de H_3 .

Els dos primers termes de H_3 no responen a la seva forma general –no pertanyen a un polinomi trigonomètric-. El coeficient del terme $\varphi \cos(\varphi)$ forma la primera constant de Poincaré-Liapunov¹³ $-V_3$ - i el coeficient del terme $\varphi \sin(\varphi)$ forma la primera constant de període $-W_3$ -, ambdues per al cas quadràtic. Es continuen els càlculs de la mateixa manera, ignorant els termes estranys $-H_j = H_j - V_j - W_j$ - i deduint-ne les constants de Poincaré-Liapunov i les constants de període que corresponguin.

¹³ Es pot veure que aquesta constant coincideix amb la constant calculada mitjançant el mètode de la resolució del problema del centre-focus.

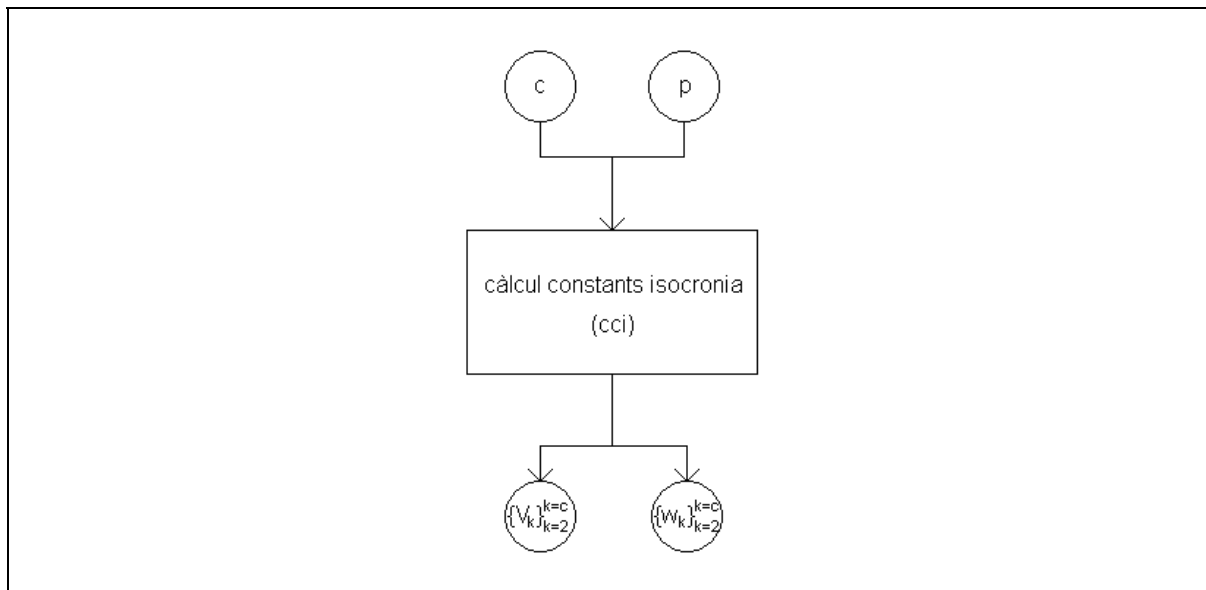
3. REQUERIMENTS DE DISSENY

En aquest capítol, es fa una descripció dels requeriments de disseny, tant dels de dades com dels d'algorismes, extrets a partir de l'estudi matemàtic del problema presentat en el capítol anterior.

3.1. ALGORISME DE CÀLCUL DE LES CONSTANTS D'ISOCRONIA

La figura 3.1 mostra el diagrama de blocs de l'algorisme de càlcul de les constants d'isocronia des del punt de vista més general.

Figura 3.1. Diagrama de blocs de l'algorisme de càlcul de les constants d'isocronia



Tot seguit, s'especifica la naturalesa de les dades d'entrada $-c$ i $p-$ i de sortida $-\{V_k\}_{k=2}^{k=c}$ i $\{W_k\}_{k=2}^{k=c}-$ de l'algorisme de càlcul de les constants d'isocronia i es detalla el disseny descendent aplicat al seu procés principal $-càlcul constants isocronia-$, on aquest es divideix en tres processos secundaris: *càlcul expressions simbòliques*, *càlcul expressions trigonomètriques* i *càlcul iteratiu/recursiu*.

3.1.1. ALGORISME DE CàLCUL DE LES EXPRESSIONS SIMBÒLIQUES

El quadre 3.1 mostra les expressions calculades pel procés de càlcul de les expressions simbòliques.

Quadre 3.1. Expressions calculades pel procés de càlcul de les expressions simbòliques

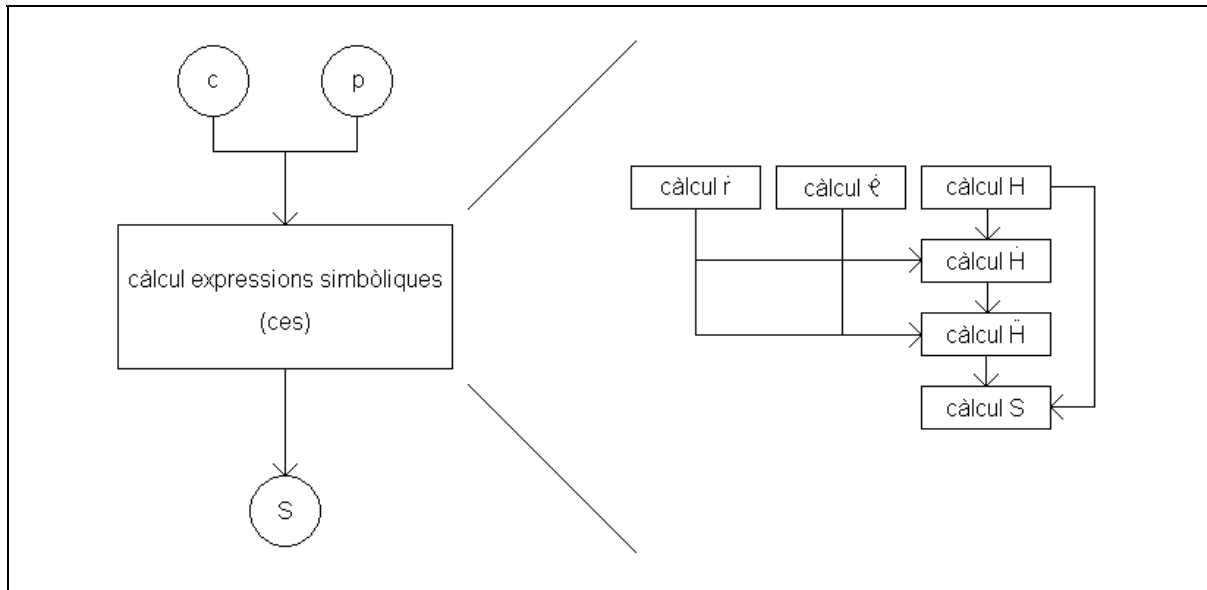
Donats els nombres c i p , on $c, p \in \mathbb{N}$ i $c, p \geq 2$, es calculen les expressions següents:

- $\dot{r} = \sum_{k=2}^p r^k P_k(\varphi)$
- $\dot{\varphi} = \sum_{k=1}^p r^{k-1} Q_k(\varphi)$
- $H = \sum_{k=0}^c r^k H_k(\varphi)$
- $\dot{H} = \frac{dH}{dr} \dot{r} + \frac{dH}{d\varphi} \dot{\varphi}$
- $\ddot{H} = \frac{d\dot{H}}{dr} \dot{r} + \frac{d\dot{H}}{d\varphi} \dot{\varphi}$
- $S = \dot{H} + H$

on: $r, \varphi \in \mathbb{R}$ són variables; $\{P_k(\varphi)\}_{k=2}^{k=p}$, $\{Q_k(\varphi)\}_{k=1}^{k=p}$ i $\{H_k(\varphi)\}_{k=0}^{k=c}$ són funcions simbòliques arbitràries que depenen de φ

Aleshores, les expressions calculades anteriorment $-\dot{r}$, $\dot{\varphi}$, H , \dot{H} , \ddot{H} i S - poden ser qualificades de simbòliques -depenen de r i de funcions simbòliques que, al seu torn, depenen de φ -.

La figura 3.2 mostra el diagrama de blocs de l'algorisme de càlcul de les expressions simbòliques.

Figura 3.2. Diagrama de blocs de l'algorisme de càlcul de les expressions simbòliques

3.1.2. ALGORISME DE CàLCUL DE LES EXPRESSIONS TRIGONOMÈTRIQUES

El quadre 3.2 mostra les expressions calculades pel procés de càlcul de les expressions trigonomètriques.

Quadre 3.2. Expressions calculades pel procés de càlcul de les expressions trigonomètriques

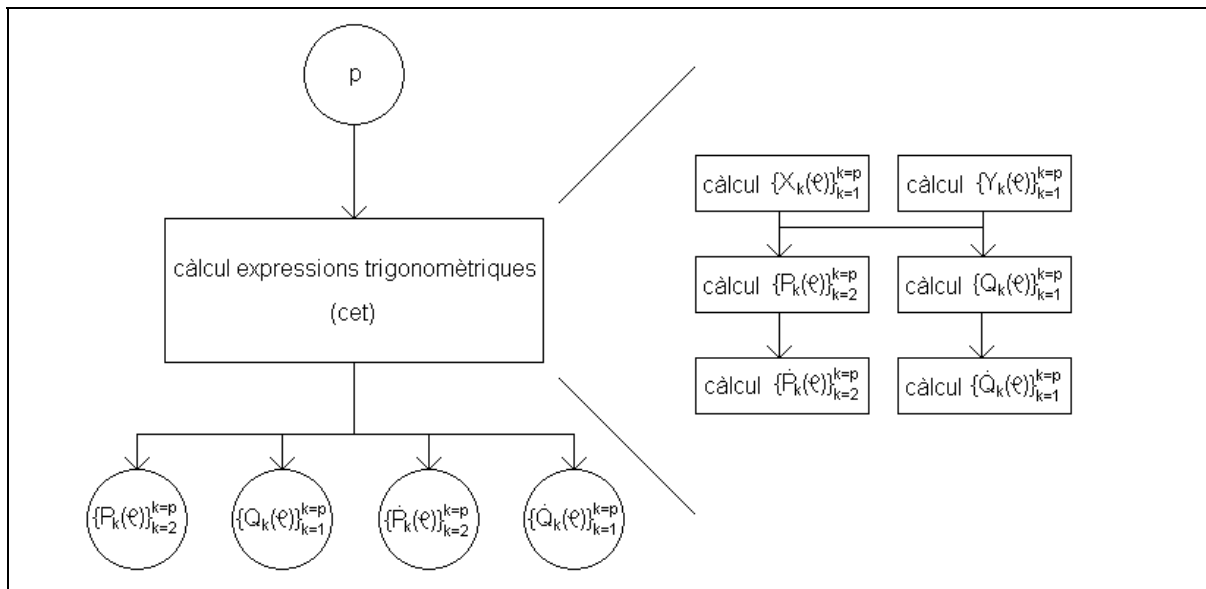
Donat el nombre p , on $p \in \mathbb{N}$ i $p \geq 2$, es calculen les expressions següents:

- $\{X_k(\varphi)\}_{k=1}^{k=p} = \sum_{i=0}^k a_{k-i,i} \cos^{k-i}(\varphi) \sin^i(\varphi)$
- $\{Y_k(\varphi)\}_{k=1}^{k=p} = \sum_{i=0}^k b_{k-i,i} \cos^{k-i}(\varphi) \sin^i(\varphi)$
- $\{P_k(\varphi)\}_{k=2}^{k=p} = \cos(\varphi) X_k(\varphi) + \sin(\varphi) Y_k(\varphi)$
- $\{Q_k(\varphi)\}_{k=1}^{k=p} = \cos(\varphi) Y_k(\varphi) - \sin(\varphi) X_k(\varphi)$
- $\{\dot{P}_k(\varphi)\}_{k=2}^{k=p} = \frac{dP_k(\varphi)}{d\varphi}$
- $\{\dot{Q}_k(\varphi)\}_{k=1}^{k=p} = \frac{dQ_k(\varphi)}{d\varphi}$

on: $\varphi \in \mathbb{R}$ és variable; $\{a_{k-i,i}\}_{k=1,i=0}^{k=p,i=k}, \{b_{k-i,i}\}_{k=1,i=0}^{k=p,i=k} \in \mathbb{R}$ són constants arbitràries¹⁴, $\{X_k(\varphi)\}_{k=1}^{k=p}, \{Y_k(\varphi)\}_{k=1}^{k=p}, \{P_k(\varphi)\}_{k=2}^{k=p}, \{Q_k(\varphi)\}_{k=1}^{k=p}, \{\dot{P}_k(\varphi)\}_{k=2}^{k=p}$ i $\{\dot{Q}_k(\varphi)\}_{k=1}^{k=p}$ són funcions trigonomètriques que depenen de φ

La figura 3.3 mostra el diagrama de blocs de l'algorisme de càlcul de les expressions trigonomètriques.

Figura 3.3. Diagrama de blocs de l'algorisme de càlcul de les expressions trigonomètriques



¹⁴ Com que $Q_1(\varphi) = 1$, aleshores les següents constants no són arbitràries sinó que tenen un valor concret: $a_{0,1} = -1, b_{1,0} = 1$ i $a_{1,0} = b_{0,1}$.

3.1.3. ALGORISME DE CÀLCUL ITERATIU/RECURSIU

El quadre 3.3 mostra les expressions calculades pel procés de càlcul iteratiu/recursiu.

Quadre 3.3. Expressions calculades pel procés de càlcul iteratiu/recursiu

Abans de res, es pot trobar fàcilment que S és de la forma següent –veure el quadre 3.1 i el capítol 2-:

$$S = \sum_{k=2}^{k=c} r^k S_k(\varphi) \quad 15$$

on: $S_k(\varphi) = \dot{H}_k(\varphi) + H_k(\varphi) - F_k(\varphi)$; $F_k(\varphi)$ és una funció trigonomètrica completament coneguda gràcies al càlcul de les expressions $H_k(\varphi)$, $\dot{H}_k(\varphi)$ i $\ddot{H}_k(\varphi)$ d'ordre inferior a k

Donat el nombre k , on $k \in [2, c]$, i de manera iterativa -des de $k = 2$ fins a $k = c$ -, s'obté l'expressió $F_k(\varphi)$ i es calculen les expressions següents:

- $H_k(\varphi) = -(\int \sin(\varphi) F_k(\varphi) d\varphi) \cos(\varphi) + (\int \cos(\varphi) F_k(\varphi) d\varphi) \sin(\varphi)$
- $\dot{H}_k(\varphi) = \frac{dH_k(\varphi)}{d\varphi}$
- $\ddot{H}_k(\varphi) = \frac{d\dot{H}_k(\varphi)}{d\varphi}$

on: $H_0(\varphi) = 0$; $H_1(\varphi) = \cos(\varphi)$

Així, s'obtindrà l'expressió $\{H_k(\varphi)\}_{k=2}^{k=c}$, que serà de la forma:

$$\{H_k(\varphi)\}_{k=2}^{k=c} = \sum_{i=0}^k c_{k-i,i} \cos^{k-i}(\varphi) \sin^i(\varphi) + V_k \varphi \cos(\varphi) + W_k \varphi \sin(\varphi)$$

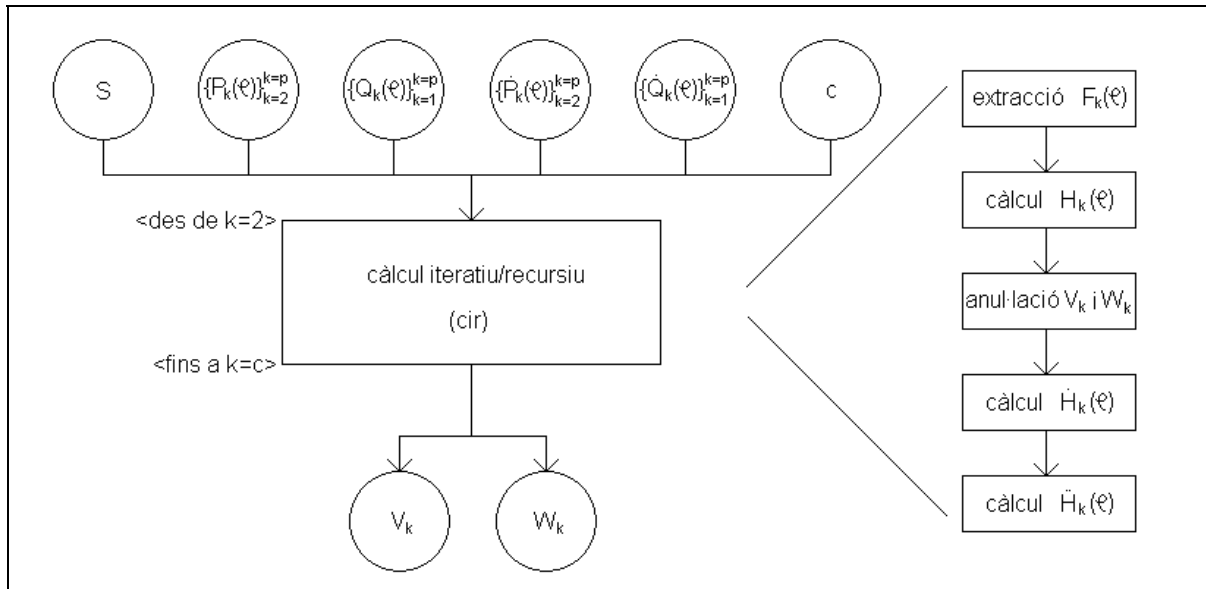
on: $\{c_{k-i,i}\}_{k=2,i=0}^{k=c,i=k} \in R$ són constants; $\{V_k\}_{k=2}^{k=c}$, $\{W_k\}_{k=2}^{k=c}$ són, respectivament, la constant de Poincaré-Liapunov i la constant de període d'ordre k

Finalment, es segueixen els càlculs anul·lant, del valor de $H_k(\varphi)$, les possibles constants d'isocronia existents.

¹⁵ Els límits del sumatori són més amplis, però el càlcul de les constants d'isocronia es limita a ordres en el rang $[2, c]$.

La figura 3.4 mostra el diagrama de blocs de l'algorisme de càlcul iteratiu/recursiu.

Figura 3.4. Diagrama de blocs de l'algorisme de càlcul iteratiu/recursiu



3.2. REQUERIMENTS DE DADES

Vistes les expressions requerides en els algorismes presentats anteriorment, s'especifiquen, tot seguit, els tipus de dades que formen aquestes expressions –tant de les expressions simbòliques com de les expressions trigonomètriques–.

3.2.1. REQUERIMENTS DE DADES PER A LES EXPRESSIONS SIMBÒLIQUES

Una expressió simbòlica té la següent forma:

$$\sum (coe \cdot r^{exp} \prod (sim_{sub}))$$

on: $coe, exp, sub \in N$; $sim \in \{P(\varphi), \dot{P}(\varphi), Q(\varphi), \dot{Q}(\varphi), H(\varphi), \dot{H}(\varphi), \ddot{H}(\varphi)\}$

Les operacions aplicades a una expressió simbòlica són la derivació respecte r , la derivació respecte φ , el producte i la suma. Seguidament, es detalla com l'estructura d'aquestes expressions simbòliques pot ser afectada per les operacions abans esmentades:

- $\frac{d \sum (coe \cdot r^{exp} \prod (sim_{sub}))}{dr} = \sum (coe \cdot exp \cdot r^{exp-1} \prod (sim_{sub}))$
- $\frac{d \sum (coe \cdot r^{exp} (sim_{1sub_1})(sim_{2sub_2}) \cdots (sim_{nsub_n}))}{d\varphi} =$
 $= \sum (coe \cdot r^{exp} (\dot{sim}_{1sub_1})(sim_{2sub_2}) \cdots (sim_{nsub_n})) +$
 $+ \sum (coe \cdot r^{exp} (sim_{1sub_1})(\dot{sim}_{2sub_2}) \cdots (sim_{nsub_n})) + \cdots +$
 $+ \sum (coe \cdot r^{exp} (sim_{1sub_1})(sim_{2sub_2}) \cdots (\dot{sim}_{nsub_n}))^{16}$
- $\sum (coe_1 \cdot r^{exp_1} \prod (sim_{1sub_1})) \cdot \sum (coe_2 \cdot r^{exp_2} \prod (sim_{2sub_2})) =$
 $= \sum (coe_1 \cdot coe_2 \cdot r^{exp_1+exp_2} \prod (sim_{1sub_1} \cdot sim_{2sub_2}))$
- $\sum (coe_1 \cdot r^{exp_1} \prod (sim_{1sub_1})) + \sum (coe_2 \cdot r^{exp_2} \prod (sim_{2sub_2})) =$
 $= \sum (coe_1 \cdot r^{exp_1} \prod (sim_{1sub_1}) + coe_2 \cdot r^{exp_2} \prod (sim_{2sub_2}))$

¹⁶ En aquest cas, \dot{sim}_{xsub_x} representa la derivació respecte φ del símbol sim_{xsub_x} .

Es pot comprovar que les operacions anteriors no modifiquen l'estructura ni els tipus de dades que formen les expressions simbòliques. En referència als exponents, només es tindran en compte els termes sumatoris amb exponent $exp \in [2, c]$; així, tant els coeficients com els exponents es mantenen dins del rang dels nombres naturals. I pel que fa als símbols, es pot veure que l'algorisme de càlcul –veure el quadre 3.1- només permet la presència dels símbols enumerats en aquest apartat.

3.2.2. REQUERIMENTS DE DADES PER A LES EXPRESSIONS TRIGONOMÈTRIQUES

Una expressió trigonomètrica té la següent forma:

$$\left\{ \sum_{sub} coe \cdot \cos^{\exp_c}(\varphi) \sin^{\exp_s}(\varphi) \right\}_{sub}^{17}$$

on: $\exp_c, \exp_s, sub \in N$; $coe \in R$

Les operacions aplicades a una expressió trigonomètrica són la integració respecte φ , la derivació respecte φ , la suma, la resta i el producte per $\cos(\varphi)$ i $\sin(\varphi)$. De la mateixa manera que per al cas de les expressions simbòliques, es detalla, tot seguit, com l'estructura d'aquestes expressions trigonomètriques pot ser afectada per les operacions abans esmentades¹⁸:

$$\begin{aligned} \bullet \quad & \frac{d\{\sum_{sub} coe \cdot \cos^{\exp_c}(\varphi) \sin^{\exp_s}(\varphi)\}_{sub}}{d\varphi} = \\ & = \left\{ \sum_{sub} coe \cdot \exp_s \cdot \cos^{\exp_c+1}(\varphi) \sin^{\exp_s-1}(\varphi) \right\}_{sub} - \\ & - \left\{ \sum_{sub} coe \cdot \exp_c \cdot \cos^{\exp_c-1}(\varphi) \sin^{\exp_s+1}(\varphi) \right\}_{sub} \end{aligned}$$

¹⁷ En aquest cas, la notació $\{\}_{sub}$ denota una família de sumatoris etiquetada amb el subíndex sub com a única característica comuna.

¹⁸ La integració respecte φ no afecta l'estructura de les expressions trigonomètriques per la naturalesa recursiva del seu càlcul; el seu cas es tractarà més endavant amb profunditat. Pel que fa a la suma i la resta, s'omet el seu desenvolupament ja que aquest és obvi; aquestes operacions tampoc no afecten l'estructura de les expressions trigonomètriques.

- $\{\sum coe \cdot \cos^{\exp_c}(\varphi) \sin^{\exp_s}(\varphi)\}_{sub} \cdot \cos(\varphi) =$
 $= \left\{ \sum coe \cdot \cos^{\exp_c+1}(\varphi) \sin^{\exp_s}(\varphi) \right\}_{sub}$
- $\{\sum coe \cdot \cos^{\exp_c}(\varphi) \sin^{\exp_s}(\varphi)\}_{sub} \cdot \sin(\varphi) =$
 $= \left\{ \sum coe \cdot \cos^{\exp_c}(\varphi) \sin^{\exp_s+1}(\varphi) \right\}_{sub}$

3.3. REQUERIMENTS D'ALGORISMES

Seguidament, a partir de les operacions aplicades als dos tipus d'expressions utilitzades, es dedueixen les característiques que hauria de reunir el disseny dels tipus i estructures de dades que implementen aquestes expressions –simbòliques i trigonomètriques- per tal d'optimitzar el rendiment dels algorismes de les operacions esmentades.

3.3.1. REQUERIMENTS D'ALGORISMES PER A LES EXPRESSIONS SIMBÒLIQUES

Les següents són les característiques de disseny proposades per a les expressions simbòliques:

Primerament, es proposa concebre una expressió simbòlica com un conjunt de termes simbòlics –cada un compartirà exponent en r -; al seu torn, un terme simbòlic serà un conjunt de sumands simbòlics –cada un dels quals tindrà com a característica posseir una combinació de símbols única-. D'aquesta manera, les operacions aplicades a les expressions simbòliques –en aquest cas, la suma i el producte- no treballaran sobre possibles redundàncies en les dades –l'exponent i els factors de símbols seran els índexs dels termes simbòlics i dels sumands simbòlics, respectivament-. Així, la notació matemàtica d'una expressió simbòlica pot veure's, ara, de la següent manera, evidenciant l'estructura ara proposada:

$$\sum (r^{\exp} \sum (coe \prod (sim_{sub})))$$

L'operació de suma sobre expressions, termes i sumands simbòlics implicarà realitzar un recorregut seqüencial sobre un dels operands, per tal d'obtenir cada un dels seus sumands simbòlics, i una cerca per exponent i combinació de símbols sobre l'altre operand, per tal de realitzar l'operació sobre el sumand simbòlic corresponent; en el cas que la cerca no trobi el sumand necessari, s'haurà de fer una inserció ordenada en el primer operand per tal de mantenir la possibilitat de realitzar aquestes cerques¹⁹. Pel que fa a la quantitat de dades sobre la que s'han de realitzar les cerques, el marge d'exponents és reduït degut a la pròpia especificació del problema –l'exponent més gran a tractar serà igual al paràmetre c , introduït per l'usuari i restringit a una xifra, com a màxim-; i pel que fa a les combinacions de símbols, aquestes es restringeixen a tres símbols com a màxim, amb totes les combinacions possibles que se'n poden derivar –veure el quadre 3.1-.

L'operació de producte sobre expressions, termes i sumands simbòlics implicarà realitzar recorreguts seqüencials sobre els dos operands, per tal d'obtenir cada un dels seus sumands simbòlics, i insercions ordenades –i per tant, també, cerques- en un dels dos operands, per tal d'introduir els sumands simbòlics resultants.

L'operació de derivació respecte r sobre expressions, termes i sumands simbòlics implicarà, només, realitzar un recorregut seqüencial sobre l'operand en qüestió, per tal d'obtenir cada un dels seus sumands simbòlics i aplicar-hi l'operació de derivació respecte r .

L'operació de derivació respecte φ sobre expressions, termes i sumands simbòlics implicarà realitzar un recorregut seqüencial sobre l'operand en qüestió, per tal d'obtenir cada un dels seus sumands simbòlics, i insercions ordenades –i per tant, també, cerques- en aquest mateix operand, per tal d'introduir els sumands simbòlics resultants.

¹⁹ De moment, no es fa referència a un ordre en el sentit estricte de la paraula, ja que tampoc no s'ha proposat, encara, una estructura de dades concreta per a dur a terme les cerques; simplement, s'evidencia la necessitat de fer insercions que mantinguin la lògica en l'organització de les dades, i que permetin dur a terme aquestes cerques amb les condicions necessàries.

3.3.2. REQUERIMENTS D'ALGORISMES PER A LES EXPRESSIONS TRIGONOMÈTRIQUES

Les següents són les característiques de disseny proposades per a les expressions trigonomètriques:

En aquest cas, es pot veure com l'estructura de dades de les expressions trigonomètriques –veure l'apartat 3.2.2- pot equiparar-se, pel que fa a la seva jerarquia, a l'estructura de dades de les expressions simbòliques proposada en l'apartat anterior: una expressió trigonomètrica és un conjunt de termes trigonomètrics –en aquest cas, no sumats entre ells, i on cada un comparteix subíndex-; al seu torn, cada un d'aquests termes trigonomètrics és un conjunt de sumands trigonomètrics –on cada un té com a característica definitiva el fet de posseir una combinació d'exponents del sinus i del cosinus pròpia-.

Les operacions de suma, resta, producte per $\cos(\varphi)$ i $\sin(\varphi)$ i derivació respecte φ sobre expressions, termes i sumands trigonomètrics implicaran que les estructures de dades d'aquests últims reuneixin les mateixes característiques que les estructures de dades de les expressions, termes i sumands simbòlics –realitzen el mateix tipus d'algorismes-. Per tant, es poden fer les mateixes consideracions que les ja fetes en l'apartat anterior.

Pel que fa a l'operació d'integració respecte φ sobre expressions, termes i sumands trigonomètrics, aquesta serà estudiada més endavant amb més profunditat; serà en aquest estudi més detallat on es faran les consideracions necessàries en relació a l'impacte de l'operació d'integració respecte φ en el disseny de l'estructura de dades de les expressions, termes i sumands trigonomètrics.

3.4. RESUM DE REQUERIMENTS I CONSIDERACIONS PRÈVIES AL DISSENY

A grans trets, doncs, les característiques generals que haurien de complir les estructures de dades utilitzades per a implementar les expressions, termes i sumands –tant simbòlics com trigonomètrics- són:

Possibilitat de realitzar recorreguts seqüencials sobre termes i sumands.

Possibilitat de realitzar cerques –evidentment òptimes- sobre sumands –pel que fa a cerques sobre termes, el seu nombre extremadament reduït converteix una cerca seqüencial en un accés gairebé directe²⁰–.

Possibilitat de realitzar insercions ordenades –en el cas simbòlic, l'índex dels termes és l'exponent de r i el dels sumands és la combinació de símbols; i en el cas trigonomètric, l'índex dels termes és el subíndex i el dels sumands és la combinació dels exponents del $\sin(\varphi)$ i del $\cos(\varphi)$ –. És per això que s'hauran de definir ordres coherents pels índexs esmentats.

Per tot això exposat més amunt, i degut al fet que l'espai de cerca és relativament petit –veure l'apartat 3.3.1–, es proposa l'ús de llistes ordenades, en detriment de taules *hash*, per a la implementació de les estructures de dades de les expressions, termes i sumands simbòlics i trigonomètrics –un altre motiu per a la utilització de llistes ordenades és el fet que no es pot assegurar un bon índex d'ocupació de posicions en una taula *hash*–.

Així mateix, es proposa també una implementació pròpia d'aquesta llista ordenada, per tal d'adaptar-la al màxim a les característiques i necessitats del problema –encara que, d'aquesta manera, es perdí l'avantatge que suposaria la reutilització de codi si s'emprés la implementació de les llistes ordenades existent en la llibreria estàndard de C++: *STL*.

²⁰ Es recorda que el nombre màxim de termes d'una expressió és c , nombre d'una sola xifra.

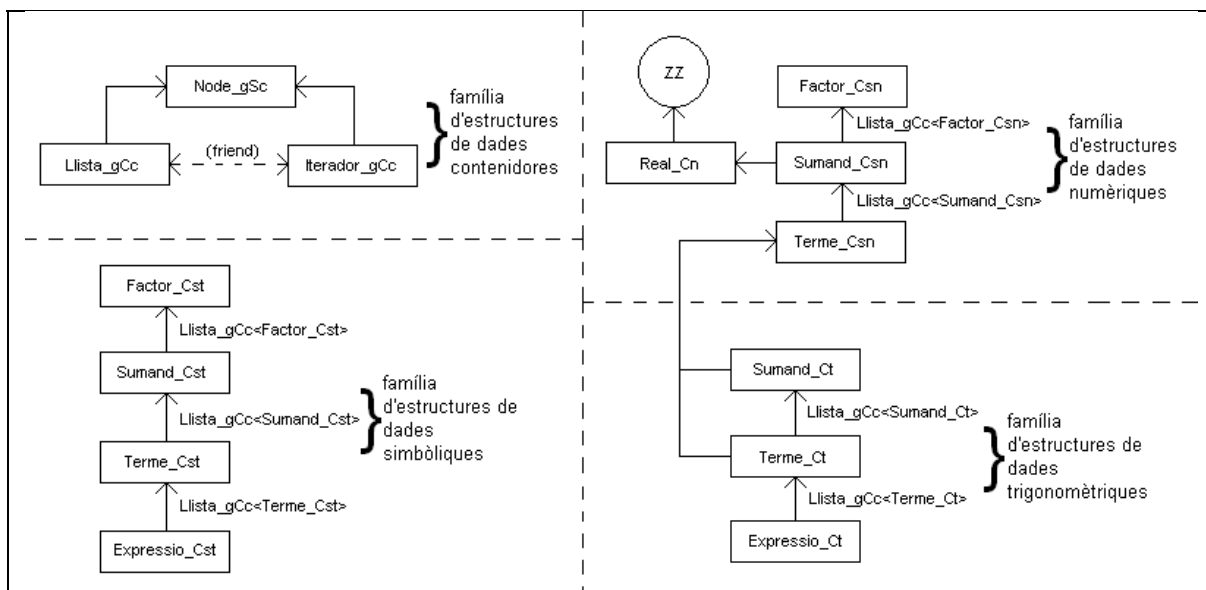
4. DISSENY I IMPLEMENTACIONS

El capítol quart inclou el disseny de dades i d'algorismes que es deriva dels requeriments enumerats en el capítol anterior, així com també les decisions de disseny preses, de manera raonada, durant tot el procés de codificació.

4.1. VISIÓ GENERAL DE LES FAMÍLIES D'ESTRUCTURES DE DADES IMPLEMENTADES

La figura 4.1 inclou un esquema general de les diverses famílies d'estructures de dades implementades, així com les relacions establertes entre elles.

Figura 4.1. Esquema general de les famílies d'estructures de dades implementades



La família d'estructures de dades contenidores conté la classe *Llista_gCc*, que implementa una llista doblement enllaçada, els nodes de la qual són creats de manera dinàmica; així mateix, aquesta família també conté la classe *Iterador_gCc*, que implementa –en una classe separada– un iterador sobre la classe *Llista_gCc*. Degut a la mútua dependència de les dues classes, aquestes són classes amigues.

La família d'estructures de dades numèriques conté la classe *Real_Cn*, que implementa un nombre real representat com un quocient de dos nombres enters de mida arbitrària –aquests nombres enters són suportats per la classe externa *ZZ*, inclosa en la llibreria *NTL*²¹-. Per altra banda, aquesta família també conté les classes *Factor_Csn*, *Sumand_Csn* i *Terme_Csn* –*Terme_Csn* és implementada com una llista d'objectes de la classe *Sumand_Csn* i, igualment, *Sumand_Csn* és implementada com una llista d'objectes de la classe *Factor_Csn*²²-.

La família d'estructures de dades simbòliques conté les classes *Factor_Cst*, *Sumand_Cst*, *Terme_Cst* i *Expressio_Cst* –*Expressio_Cst* és implementada com una llista d'objectes de la classe *Terme_Cst*, *Terme_Cst* és implementada com una llista d'objectes de la classe *Sumand_Cst* i, finalment, *Sumand_Cst* és implementada com una llista d'objectes de la classe *Factor_Cst*-.

Finalment, la família d'estructures de dades trigonomètriques conté les classes *Sumand_Ct*, *Terme_Ct* i *Expressio_Ct* –*Expressio_Ct* és implementada com una llista d'objectes de la classe *Terme_Ct* i, igualment, *Terme_Ct* és implementada com una llista d'objectes de la classe *Sumand_Ct*-. A la vegada, les classes *Sumand_Ct* i *Terme_Ct* també tenen, com a classe agregada, la classe *Terme_Csn*, pertanyent a la família d'estructures de dades numèriques.

4.2. FAMÍLIA D'ESTRUCTURES DE DADES CONTENIDORES

Al final del capítol anterior, s'ha fet evident la necessitat de disposar d'una estructura de dades contenidora i genèrica –vàlida per a fer referència tant a tipus de dades simbòlics com a tipus de dades trigonomètrics, entre d'altres- amb unes característiques ben definides –veure l'apartat 3.4-: aquesta estructura de dades hauria d'implementar una llista d'accés seqüencial amb la possibilitat de poder ordenar els seus elements -per tal de dur-hi a terme cerques-.

²¹ Per a trobar més informació sobre la llibreria *NTL* i, en particular, sobre la classe *ZZ*, veure el capítol annex.

²² El sentit i funcionament d'aquestes classes, així com el de les classes de les dues famílies d'estructures de dades anomenades a continuació, es tractarà amb més detall tot seguit.

4.2.1. DEFINICIÓ DE LA FAMÍLIA D'ESTRUCTURES DE DADES CONTENIDORES

El quadre 4.1 conté la definició de la família d'estructures de dades contenidores –no s'hi inclou la part pública de les classes–.

Quadre 4.1. Definició de la família d'estructures de dades contenidores

```
template <class Tipus_gC> struct Node_gSc {  
  
    Node_gSc * Anterior_aaNc;  
    Node_gSc * Posterior_aaNc;  
    Tipus_gC * Element_aaT;  
  
};  
  
template <class Tipus_gC> class Iterador_gCc {  
  
    private:  
  
        Node_gSc <Tipus_gC> * Referent_aaNc;  
        Node_gSc <Tipus_gC> * Referencia_aaNc;  
  
    public:  
  
        //interfície d'usuari  
  
};  
  
template <class Tipus_gC> class Llista_gCc {  
  
    private:  
  
        Natural_iTn          Mida_aNn;                               //typedef int Natural_iTn  
        Node_gSc <Tipus_gC> * Contingut_aaNc;  
  
    public:  
  
        //interfície d'usuari  
  
};
```

Les decisions de disseny preses en la definició d'aquestes estructures de dades han estat les següents:

- Utilització d'una llista doblement enllaçada amb la possibilitat de definir iteradors sobre ella.
- Implementació de dos atributs per a aquesta llista: un d'ells fa les funcions d'element fantasma –utilitzat per a simplificar el funcionament dels algorismes definits en les seves operacions- i l'altre manté el compte dels elements referenciats.
- Utilització d'un iterador definit en una classe separada de la definició de la classe que implementa la llista; aquests iteradors, definits sobre una llista associada, tenen un comportament circular.
- Implementació de dos atributs per a aquest iterador: el primer es manté com a referència a la llista associada per a possibles comprovacions de coherència en la utilització del mateix iterador i el segon fa les funcions d'apuntador a l'element distingit de la llista associada.

4.2.2. INTERFÍCIE DE LA FAMÍLIA D'ESTRUCTURES DE DADES CONTENIDORES

El quadre 4.2 conté la interfície de la família d'estructures de dades contenidores.

Quadre 4.2. Interfície de la família d'estructures de dades contenidores

```
//interfície d'usuari de la classe Iterador_gCc
Iterador_gCc (const Llista_gCc <Tipus_gC> &);
~Iterador_gCc ( );

Buit iTm Finalitzar_fmIc (); //typedef void Buit iTm

Boolea iTn Cercar_fmIc (const Tipus_gC &); //typedef bool Boolea iTn

Boolea iTn EsFinal_fmIc () const;
```

```

//interfície d'usuari de la classe Llista_gCc

Llista_gCc (
Llista_gCc (const Llista_gCc &);
~Llista_gCc (
);

Buit iTm Inserir_fmLc (const Iterador_gCc <Tipus_gC> &,
Tipus_gC
*);
Buit iTm Inserir_fmLc (const Natural iTn
&,
Tipus_gC
*);
Buit iTm Eliminar_fmLc (const Iterador_gCc <Tipus_gC> &);
Buit iTm Eliminar_fmLc (const Natural iTn
&);

Tipus_gC * Consultar_fcLc ( Iterador_gCc <Tipus_gC> &) const;
Tipus_gC * Consultar_fcLc (const Natural iTn
&) const;
Natural iTn Mida_fcLc (
) const;

```

El funcionament de les operacions de la classe *Iterador_gCc* és el següent:

- El constructor de la classe *Iterador_gCc* crea un lligam entre l'objecte iterador i la llista passada com a referència, de cara a assegurar la coherència en l'ús d'aquest mateix iterador en futures crides. Així mateix, l'element referenciat per l'iterador serà el següent a l'element fantasma -el primer de la llista-; en el cas d'una llista buida, l'iterador farà referència al mateix element fantasma. El seu cost és $\vartheta(1)$.
- El destructor de la classe *Iterador_gCc* destrueix l'objecte iterador. El seu cost és $\vartheta(1)$.
- L'operació *Finalitzar_fmLc* fa que l'element referenciat per l'iterador sigui l'element fantasma. El seu cost és $\vartheta(1)$.
- L'operació *EsFinal_fcLc* retorna 1 si l'element referenciat per l'iterador és l'element fantasma; altrament retorna 0. El seu cost és $\vartheta(1)$.

Degut a la importància i complexitat de la funció *Cercar_fmIc*, aquesta es tracta amb més detall tot seguit²³. La seva especificació és la següent:

- **Def:** $\{Llista_gCc <Tipus_gC> l;$
 $Iterador_gCc <Tipus_gC> i (l);$
 $Tipus_gC o;$
 $Boolea_iTn r\}$
- **Pre:** $\{l$ és una llista ordenada o buida;
 $Tipus_gC$ té definides les operacions $Tipus_gC < Tipus_gC,$
 $Tipus_gC > Tipus_gC$ i $Tipus_gC \neq Tipus_gC\}$
- $r = i . Cercar_fmIc (o)$
- **Post:** $\{r = 0$ si l no conté o , i llavors i fa referència a l'element més petit de l que és més gran que o -si l és buida, llavors i fa referència a l'element fantasma-;
 $r = 1$ si l conté o , i llavors i fa referència a l'element o dins de $l\}$

El funcionament de les operacions de la classe *Llista_gCc* és el següent:

- El constructor de la classe *Llista_gCc* crea una llista buida -amb un sol element, l'element fantasma, necessari per a l'òptim funcionament de les operacions de la classe, però que no es tracta de la mateixa manera que la resta d'elements-. El seu cost és $\vartheta(1)$.
- El constructor copiadore de la classe *Llista_gCc* crea una llista que conté una còpia dels elements -en el mateix ordre- de la llista original. Per al seu correcte funcionament, aquesta operació necessita que *Tipus_gC* defineixi també un constructor copiadore. El seu cost és $\vartheta(1)$.
- El destructor de la classe *Llista_gCc* destrueix la llista. Per al seu correcte funcionament, aquesta operació necessita que *Tipus_gC* defineixi també un destructor. El seu cost és $\vartheta(1)$.
- L'operació *Mida_fcLc* retorna el nombre d'elements que conté la llista -sense comptar l'element fantasma-. Si aquesta és buida, retorna 0. El seu cost és $\vartheta(1)$.

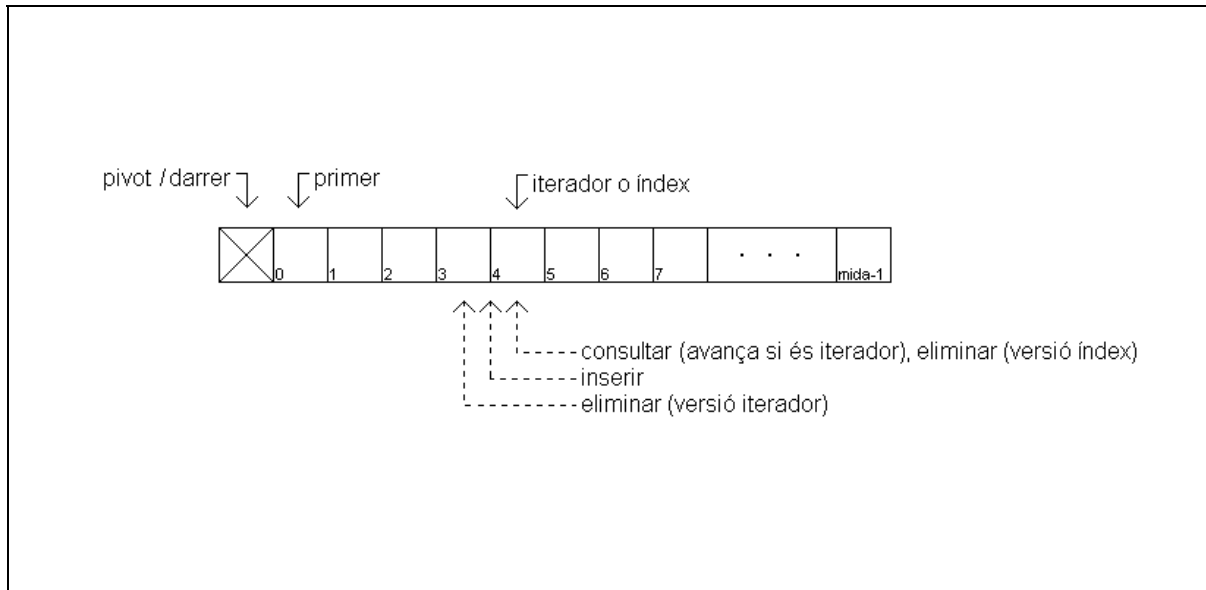
²³ El codi i l'explicació detallada d'aquesta operació es troben en el capítol annex.

Pel que fa a les operacions d'inserció, eliminació i consulta, les consideracions fetes són les següents:

- S'ha dotat la classe *Llista_gCc* de dos versions de cada una de les operacions següents: *Inserir_fmLc*, *Eliminar_fmLc* i *Consultar_fcLc*; aquestes operacions són, respectivament, les d'inserció, eliminació i consulta d'un element en una llista. En la primera versió, les referències als elements de la llista són proporcionades per iteradors –passats com a paràmetre en les operacions d'inserció, eliminació i consulta-; en la segona versió, aquestes referències són proporcionades per nombres naturals –també passats com a paràmetre en les operacions d'inserció, eliminació i consulta- que representen índexs sobre vectors –en aquest cas, la llista es comporta com un vector²⁴-.
- El comportament dels iteradors i índexs en relació amb el funcionament de les operacions d'inserció, eliminació i consulta s'il·lustra en la figura 4.2: la inserció es realitza just en la posició anterior a la de l'iterador o l'índex, l'eliminació afecta l'element anterior a l'element referenciat –per l'iterador- o el mateix element referenciat –per l'índex- i la consulta s'aplica sobre el mateix element indicat per l'iterador o l'índex –en el cas d'una consulta amb iterador, aquest avança una posició després de realitzar l'operació-. D'altra banda, les operacions d'inserció, eliminació i consulta realitzen una comprovació de la coherència de l'iterador o l'índex que prenen com a paràmetre; si aquesta coherència no es dona, es llença una excepció. En aquest sentit, l'operació d'inserció es pot realitzar per a qualsevol iterador, però l'índex ha d'existir dins del rang d'elements presents en la llista –es permet, però, donar un índex que indiqui justament la posició immediatament següent a la de l'últim element de la llista, per tal de poder-la fer créixer utilitzant insercions amb índex-. L'operació d'eliminació no es permet si l'iterador fa referència al primer element –intent d'eliminació de l'element fantasma-; pel que fa a l'eliminació amb índex, aquest ha d'existir dins del rang d'elements presents en la llista. L'operació de consulta no es permet si l'iterador fa referència a l'element fantasma –intent de consulta de l'element fantasma-; pel que fa a la consulta amb índex, aquest ha d'existir dins del rang d'elements presents en la llista.

²⁴ Aquesta dualitat proporcionada per les operacions d'inserció, eliminació i consulta és deguda a la necessitat d'utilitzar l'estructura de dades contenidora implementada tant per a accessos directes als seus elements –vectors- com per a accessos seqüencials als mateixos –l·listes-. Realment, la versió de les operacions que utilitza índexos –i que simula accessos directes- s'implementa amb un funcionament d'accés seqüencial, però només s'utilitza en estructures de dades amb un nombre molt baix d'elements, fet que equipara el seu temps d'accés als elements al d'un accés directe real.

Figura 4.2. Comportament dels iteradors i índexs



- Les operacions d'inserció comproven que l'apuntador a l'element inserit –passat com a paràmetre- no sigui buit –en cas que sigui buit, es llença una excepció-. Per altra banda, les dues operacions d'eliminació necessiten que *Tipus_gC* defineixi un destructor.
- El cost de les operacions d'inserció, eliminació i consulta –en les seves dues versions- és $\vartheta(1)$ excepte en la inserció i l'eliminació mitjançant índexos, en les que el seu cost és $\vartheta(n)$ –ja s'ha dit, però, que les versions que utilitzen índexos s'empren en casos en què el nombre d'elements de l'estructura de dades és molt baix; aleshores, el cost d'aquestes funcions es pot equiparar a $\vartheta(1)$ -.
- Pel que fa a la política de gestió de la memòria dels elements referenciats per la llista, s'ha optat per recórrer el mínim possible a la còpia d'aquests; així, les operacions d'inserció i consulta insereixen i retornen, respectivament, un apuntador als elements referenciats –no fan una còpia dels mateixos, sinó que aquests han d'haver estat creats prèviament-. En canvi, l'operació d'eliminació sí que destrueix els elements, deixant, d'aquesta manera, que la tasca de supressió de deixalles la dugui a terme la pròpia classe -encara que la creació dels elements s'hagi de fer fora d'aquesta-.

4.2.3. ALTRES ASPECTES RELLEVANTS EN EL FUNCIONAMENT DE LA FAMÍLIA D'ESTRUCTURES DE DADES CONTENIDORES

Vista la definició i la interfície de la família d'estructures de dades contenidores, la política més adient a l'hora d'utilitzar els iteradors seria la següent:

- Les insercions sobre una llista buida ompliran aquesta agregant els elements per la cua –degut a la naturalesa circular de la llista-. De totes maneres, per tal d'aconseguir una llista ordenada, abans d'inserir un element, aquest s'ha de cercar a la llista i utilitzar l'iterador modificat per aquesta cerca per a realitzar la inserció.
- Per a buidar una llista només cal situar l'iterador sobre el darrer element de la mateixa –l'element fantasma- i eliminar elements fins que la llista sigui buida. El fet que la consulta d'un element faci avançar l'iterador endavant una posició facilita operacions com la d'eliminar l'element consultat prèviament –l'eliminació afecta l'element anterior a l'indicat per l'iterador-.
- Els recorreguts seqüencials de les llistes també són afavorits per la característica descrita anteriorment.

Pel que fa als índexs, la inserció d'un element en un índex donat fa que aquest element passi a ocupar la posició indicada per l'índex –els elements posteriors a l'índex avançaran una posició-. L'eliminació actua de manera contrària: els elements posteriors a l'element esborrat retrocediran una posició.

4.3. FAMÍLIA D'ESTRUCTURES DE DADES SIMBÒLIQUES

Tot seguit, es detallen les característiques de les estructures de dades que implementen les expressions, termes, sumands i factors simbòlics requerits.

4.3.1. DEFINICIÓ DE LA FAMÍLIA D'ESTRUCTURES DE DADES SIMBÒLIQUES

El quadre 4.3 conté la definició de la família d'estructures de dades simbòliques –no s'hi inclou la part pública de les classes-.

Quadre 4.3. Definició de la família d'estructures de dades simbòliques

```
enum Simbol_Est {
    _P_eSst = 0,
    _Q_eSst = 1,
    _H_eSst = 2,
    _Pp_eSst = 3,
    _Qp_eSst = 4,
    _Hp_eSst = 5,
    _Hpp_eSst = 6
};

class Factor_Cst {
private:
    Simbol_Est Valor_aSst;
    Natural_iTn Subindex_aNn;
public:
    //interfície d'usuari
};

class Sumand_Cst {
private:
    Natural_iTn          Coeficient_aNn;
    Llista_gCc <Factor_Cst> F_aLc;
public:
    //interfície d'usuari
};

class Terme_Cst {
private:
    Natural_iTn          Exponent_aNn;
    Llista_gCc <Sumand_Cst> S_aLc;
public:
    //interfície d'usuari
};
```

```
class Expressio_Cst {  
  
    private:  
  
        Llista_gCc <Terme_Cst> T_aLc;  
  
    public:  
  
        //interfície d'usuari  
  
};
```

Les decisions de disseny preses en la implementació d'aquestes estructures de dades han estat les següents:

- Una expressió simbòlica és una llista de termes simbòlics. Aquí, aquesta llista s'utilitzarà a manera de vector –l'algorisme general necessita accedir als termes simbòlics de manera directa-, fet que s'aconsegueix degut al nombre reduït de termes simbòlics de què disposarà una expressió simbòlica.
- Un terme simbòlic és una llista de sumands simbòlics –a més, el terme simbòlic té un atribut que defineix l'exponent de r del mateix-. En aquest cas, la llista s'utilitzarà per a accessos seqüencials –les operacions aplicades a un terme simbòlic requereixen d'aquest tipus d'accés-.
- Un sumand simbòlic és una llista de factors simbòlics –a més, el sumand simbòlic té un atribut que defineix el coeficient del mateix-. En aquest cas, la llista també s'utilitzarà per a accessos seqüencials –les operacions aplicades a un sumand simbòlic requereixen d'aquest tipus d'accés-.
- Un factor simbòlic està format per dos atributs: el símbol que representa –tipus enumerat- i el seu subíndex.

4.3.2. INTERFÍCIE DE LA FAMÍLIA D'ESTRUCTURES DE DADES SIMBÒLIQUES

El quadre 4.4 conté la interfície de la família d'estructures de dades simbòliques.


```

//interfície d'usuari de la classe Terme_Cst

Terme_Cst (const Natural iTn &);
Terme_Cst (const Natural iTn &,
           const Boolea iTn &,
           const Simbol_Est &);
Terme_Cst (const Terme_Cst &);
~Terme_Cst (
           );

Buit iTm operator = (const Terme_Cst &);
Buit iTm operator += (const Sumand_Cst &);
Buit iTm operator += (const Terme_Cst &);
Buit iTm operator *= (const Sumand_Cst &);
Buit iTm operator *= (const Terme_Cst &);

Terme_Cst * DerivarR_fmcTst (
                       ) const;
Terme_Cst * DerivarF_fmcTst (
                       ) const;
Terme_Cst * Calcular_fmcTst (const Expressio_Ct &,
                             const Expressio_Ct &,
                             const Expressio_Ct &,
                             const Expressio_Ct &,
                             const Expressio_Ct &,
                             const Expressio_Ct &,
                             const Expressio_Ct &) const;

Natural iTn Exponent_fcTst () const;

//interfície d'usuari de la classe Expressio_Cst

Expressio_Cst (
              );
Expressio_Cst (const Natural iTn &,
              const Natural iTn &,
              const Boolea iTn &,
              const Simbol_Est &);
Expressio_Cst (const Expressio_Cst &);
~Expressio_Cst (
              );

Buit iTm operator = (const Expressio_Cst &);
Buit iTm operator += (const Terme_Cst &);
Buit iTm operator += (const Expressio_Cst &);
Buit iTm operator *= (const Terme_Cst &);
Buit iTm operator *= (const Expressio_Cst &);

Terme_Cst * operator [] (const Natural iTn &) const;

Expressio_Cst * DerivarR_fmcEst () const;
Expressio_Cst * DerivarF_fmcEst () const;

```

Pel que fa a les operacions definides en les interfícies d'usuari de les classes simbòliques, les característiques més importants són les següents:

- Ús de la sobrecàrrega dels operadors $+=$ i $*=$ per a les classes *Sumand_Cst*, *Terme_Cst* i *Expressio_Cst*²⁵.
- Ús de la sobrecàrrega dels operadors $<$, $>$ i $!=$ -com a funcions *friend*- per a les classes *Factor_Cst* i *Sumand_Cst*, necessaris per tal de poder dur a terme cerques en llistes d'objectes d'aquestes dues classes.
- Ús de la sobrecàrrega de l'operador $[]$ per a la classe *Expressio_Cst*, necessària per tal de dotar aquesta d'una operació que permeti l'accés directe als elements d'una expressió simbòlica -termes simbòlics-.
- Les operacions *DerivarR* i *DerivarF*, definides per a cada una de les classes simbòliques, implementen la derivació respecte la variable r i respecte la variable φ -respectivament- de factors, sumands, termes i expressions simbòlics. Aquestes operacions retornen apuntadors, ja que els objectes resultants són grans -fet que estalvia moviments importants de dades en memòria, augmentant així l'eficiència de l'algorisme; com a contrapartida, cal vigilar amb la gestió que es fa d'aquests apuntadors-.
- L'operació *Calcular*, definida per a les classes *Sumand_Cst* i *Terme_Cst*, serà l'encarregada de transformar els sumands i termes simbòlics en termes trigonomètrics -pas necessari en la tercera fase de l'algorisme general, segons s'ha vist en el capítol anterior-. En aquest cas, l'operació també retorna un apuntador, per la mateixa raó exposada anteriorment.

4.4. FAMÍLIA D'ESTRUCTURES DE DADES TRIGONOMÈTRIQUES

Tot seguit, es detallen les característiques de les estructures de dades que implementen les expressions, termes i sumands trigonomètrics requerits.

²⁵ En el capítol annex s'inclou el codi de l'operació *Terme_Cst::operator += (const Sumand_Cst &)* -així com també la seva explicació detallada- ja que és un exemple representatiu de la manera com s'han implementat aquestes operacions.

4.4.1. DEFINICIÓ DE LA FAMÍLIA D'ESTRUCTURES DE DADES TRIGONOMÈTRIQUES

El quadre 4.5 conté la definició de la família d'estructures de dades trigonomètriques –no s'hi inclou la part pública de les classes–.

Quadre 4.5. Definició de la família d'estructures de dades trigonomètriques

```
class Sumand_Ct {  
  
    private:  
  
        Terme_Csn Coeficient_aTsn;  
        Natural iTn Cosinus_aNn;  
        Natural iTn Sinus_aNn;  
  
    public:  
  
        //interfície d'usuari  
  
};  
  
class Terme_Ct {  
  
    private:  
  
        Natural iTn          Subindex_aNn;  
        Terme_Csn          Constant_aTsn;  
        Llista_gCc <Sumand_Ct> S_aLc;  
  
    public:  
  
        //interfície d'usuari  
  
};  
  
class Expressio_Ct {  
  
    private:  
  
        Llista_gCc <Terme_Ct> T_aLc;  
  
    public:  
  
        //interfície d'usuari  
  
};
```

Les decisions de disseny preses en la implementació d'aquestes estructures de dades han estat les següents:

- Una expressió trigonomètrica és una llista de termes trigonomètrics. Aquí, aquesta llista s'utilitzarà a manera de vector –l'algorisme general necessita accedir als termes trigonomètrics de manera directa-, fet que s'aconsegueix degut al nombre reduït de termes trigonomètrics de què disposarà una expressió trigonomètrica.
- Un terme trigonomètric és una llista de sumands trigonomètrics –a més, el terme trigonomètric té un atribut que defineix el subíndex que identifica el mateix terme dins de l'expressió trigonomètrica global; així mateix, el terme trigonomètric també té un atribut anomenat *Constant_aTsn* que allotjarà la possible constant d'isocronia associada al mateix terme-. En aquest cas, la llista s'utilitzarà per a accessos seqüencials –les operacions aplicades a un terme trigonomètric requereixen d'aquest tipus d'accés-.
- Un sumand trigonomètric està format per tres atributs: els exponents del sinus i del cosinus i el coeficient del mateix sumand²⁶.

4.4.2. INTERFÍCIE DE LA FAMÍLIA D'ESTRUCTURES DE DADES TRIGONOMÈTRIQUES

El quadre 4.6 conté la interfície de la família d'estructures de dades trigonomètriques.

²⁶ La classe *Terme_Csn* s'explicarà en l'apartat 4.5.

Quadre 4.6. Interfície de la família d'estructures de dades trigonomètriques

```
//interfície d'usuari de la classe Sumand_Ct

Sumand_Ct (const Natural iTn &,
           const Natural iTn &);
Sumand_Ct (const Natural iTn &,
           const Natural iTn &,
           const Simbol_Esn &);
Sumand_Ct (const Terme_Csn &,
           const Natural iTn &,
           const Natural iTn &);
Sumand_Ct (const Sumand_Ct &);
~Sumand_Ct (
           );

Buit iTm operator += (const Sumand_Ct &);
Buit iTm operator -= (const Sumand_Ct &);
Buit iTm operator *= (const Natural iTn &);
Buit iTm operator *= (const Sumand_Ct &);

Buit iTm Negatiu_fmSt ();

Terme_Ct * Integrar_fmSt (const Natural iTn &) const;
Terme_Ct * Derivar_fmSt (const Natural iTn &) const;

Boolea iTn EsBuit_fcSt () const;

friend Boolea iTn operator < (const Sumand_Ct &,
                              const Sumand_Ct &);
friend Boolea iTn operator > (const Sumand_Ct &,
                              const Sumand_Ct &);
friend Boolea iTn operator != (const Sumand_Ct &,
                               const Sumand_Ct &);
```

```

//interfície d'usuari de la classe Terme_Ct

Terme_Ct (const Natural iTn &);
Terme_Ct (const Natural iTn &,
          const Simbol_Esn &);
Terme_Ct (const Terme_Ct &);
~Terme_Ct (
);

Buit iTm operator = (const Terme_Ct &);
Buit iTm operator += (const Terme_Csn &);
Buit iTm operator += (const Sumand_Ct &);
Buit iTm operator += (const Terme_Ct &);
Buit iTm operator -= (const Sumand_Ct &);
Buit iTm operator -= (const Terme_Ct &);
Buit iTm operator *= (const Natural iTn &);
Buit iTm operator *= (const Sumand_Ct &);
Buit iTm operator *= (const Terme_Ct &);

Buit iTm Negatiu_fmTt ();

Terme_Ct * Integrar_fmCtTt () const;
Terme_Ct * Derivar_fmCtTt () const;

Natural iTn Subindex_fcTt () const;
Terme_Csn * Extreure_fcTt () const;

//interfície d'usuari de la classe Expressio_Ct

Expressio_Ct (
);
Expressio_Ct (const Natural iTn &);
Expressio_Ct (const Natural iTn &,
             const Natural iTn &,
             const Simbol_Esn &);
Expressio_Ct (const Expressio_Ct &);
~Expressio_Ct (
);

Buit iTm operator += (const Terme_Ct &);
Buit iTm operator += (const Expressio_Ct &);
Buit iTm operator -= (const Expressio_Ct &);
Buit iTm operator *= (const Sumand_Ct &);

Terme_Ct * operator [] (const Natural iTn &) const;

Expressio_Ct * Derivar_fmCtEt () const;

```

Pel que fa a les operacions definides en les interfícies d'usuari de les classes trigonomètriques, les característiques més importants són les següents:

- Ús de la sobrecàrrega dels operadors $+=$, $-=$ i $*=$ per a les classes *Sumand_Ct*, *Terme_Ct* i *Expressio_Ct*²⁷.
- Ús de la sobrecàrrega dels operadors $<$, $>$ i $!=$ -com a funcions *friend*- per a la classe *Sumand_Ct*, necessaris per tal de poder dur a terme cerques en llistes d'objectes d'aquesta classe.
- Ús de la sobrecàrrega de l'operador $[]$ per a la classe *Expressio_Ct*, necessària per tal de dotar aquesta d'una operació que permeti l'accés directe als elements d'una expressió trigonomètrica –termes trigonomètrics-.
- Les operacions *Integrar* i *Derivar*, definides per a cada una de les classes trigonomètriques, implementen la integració respecte la variable φ i la derivació respecte la variable φ -respectivament- de sumands, termes i expressions trigonomètrics. Aquestes operacions retornen apuntadors, ja que els objectes resultants són grans –fet que estalvia moviments importants de dades en memòria, augmentant així l'eficiència de l'algorisme; com a contrapartida, cal vigilar amb la gestió que es fa d'aquests apuntadors²⁸-.

4.5. FAMÍLIA D'ESTRUCTURES DE DADES NUMÈRIQUES

En el quadre 4.5, s'ha vist la necessitat d'implementar una estructura de dades que contingui el valor dels coeficients dels sumands trigonomètrics –que tindran la mateixa forma que el valor de les constants d'isocronia dels termes trigonomètrics-. Aquests valors seran combinacions lineals de les constants $a_{k-i,i}$ i $b_{k-i,i}$, representades com a valors simbòlics –veure el quadre 3.2-.

Tot seguit, es detallen les característiques de les estructures de dades que implementen els termes, sumands i factors numèrics requerits.

²⁷ En el capítol annex s'inclou el codi de l'operació *Terme_Ct:: operator += (const Sumand_Ct &)* –així com també la seva explicació detallada- ja que és un exemple representatiu de la manera com s'han implementat aquestes operacions.

²⁸ En el capítol annex s'inclou el codi de l'operació *Sumand_Ct:: Integrar_fmCt (const Natural_iTn &)* –així com també la seva explicació detallada- ja que és un exemple també representatiu de la manera com s'han implementat aquestes operacions.

4.5.1. DEFINICIÓ DE LA FAMÍLIA D'ESTRUCTURES DE DADES NUMÈRIQUES

El quadre 4.7 conté la definició de la família d'estructures de dades numèriques –no s'hi inclou la part pública de les classes-.

Quadre 4.7. Definició de la família d'estructures de dades numèriques

```
typedef bool Boolea iTn;
typedef int  Natural iTn;
typedef ZZ  Enter_eTn;                                     ///include <NTL/ZZ.h>

class Real_Cn {
private:
    Enter_eTn Numerador_aEn;
    Enter_eTn Denominador_aEn;

public:
    //interfície d'usuari
};

enum Simbol_Esn {
    _A_eSsn = 0,
    _B_eSsn = 1
};

class Factor_Csn {
private:
    Simbol_Esn Valor_aSsn;
    Natural iTn Subindex1_aNn;
    Natural iTn Subindex2_aNn;

public:
    //interfície d'usuari
};
```

```
class Sumand_Csn {  
  
    private:  
  
        Real_Cn          Coeficient_aRn;  
        Llista_gCc <Factor_Csn> F_aLc;  
  
    public:  
  
        //interfície d'usuari  
  
};  
  
class Terme_Csn {  
  
    private:  
  
        Llista_gCc <Sumand_Csn> S_aLc;  
  
    public:  
  
        //interfície d'usuari  
  
};
```

Les decisions de disseny preses en la implementació d'aquestes estructures de dades han estat les següents:

- Un terme numèric és una llista de sumands numèrics. En aquest cas, la llista s'utilitzarà per a accessos seqüencials –les operacions aplicades a un terme numèric requereixen d'aquest tipus d'accés-.
- Un sumand numèric és una llista de factors numèrics –a més, el sumand numèric té un atribut que defineix el coeficient del mateix-. En aquest cas, la llista també s'utilitzarà per a accessos seqüencials –les operacions aplicades a un sumand numèric requereixen d'aquest tipus d'accés-.
- Un factor numèric està format per tres atributs: el símbol que representa –tipus enumerat- i els seus dos subíndexs.

- Un nombre real és format per dos nombres enters a manera de quocient. Per a aquesta implementació, s'ha fet servir la classe ZZ^{29} –classe que implementa la representació dels nombres enters- de la llibreria per al treball amb grans nombres *NTL*, tal com demana una de les condicions de l'enunciat d'aquest treball. El fet de triar una representació dels nombres reals com el quocient de dos nombres enters, i no utilitzar directament la implementació dels nombres reals –classe *RR* de la mateixa llibreria *NTL*-, és degut a què la classe *RR* fa un arrodoniment dels nombres reals en les seves operacions; en canvi, la classe *ZZ* implementa una representació il·limitada dels nombres enters.

4.5.2. INTERFÍCIE DE LA FAMÍLIA D'ESTRUCTURES DE DADES NUMÈRIQUES

El quadre 4.8 conté la interfície de la família d'estructures de dades numèriques.

Quadre 4.8. *Interfície de la família d'estructures de dades numèriques*

```
//interfície d'usuari de la classe Real_Cn

Real_Cn (const Natural_iTn &);
Real_Cn (const Real_Cn &);
~Real_Cn (          );

Buit_iTm operator += (const Real_Cn &);
Buit_iTm operator -= (const Real_Cn &);
Buit_iTm operator *= (const Natural_iTn &);
Buit_iTm operator *= (const Real_Cn &);
Buit_iTm operator /= (const Natural_iTn &);

Buit_iTm Negatiu_fmRn ();
Buit_iTm Simplificar_fmRn ();

Boolea_iTn EsBuit_fcRn () const;
```

²⁹ En el capítol annex s'inclou més informació sobre la llibreria *NTL* i la classe utilitzada *ZZ*.


```
//interfície d'usuari de la classe Terme_Csn
```

```
Terme_Csn (
);
Terme_Csn (const Natural_iTn &);
Terme_Csn (const Simbol_Esn &,
           const Natural_iTn &,
           const Natural_iTn &);
Terme_Csn (const Terme_Csn &);
~Terme_Csn (
);

Buit_iTm operator = (const Terme_Csn &);
Buit_iTm operator += (const Sumand_Csn &);
Buit_iTm operator += (const Terme_Csn &);
Buit_iTm operator -= (const Sumand_Csn &);
Buit_iTm operator -= (const Terme_Csn &);
Buit_iTm operator *= (const Natural_iTn &);
Buit_iTm operator *= (const Sumand_Csn &);
Buit_iTm operator *= (const Terme_Csn &);
Buit_iTm operator /= (const Natural_iTn &);

Buit_iTm Negatiu_fmTsn ();

Boolea_iTn EsBuit_fcTsn () const;
```

Pel que fa a les operacions definides en les interfícies d'usuari de les classes numèriques, les característiques més importants són les següents:

- Ús de la sobrecàrrega dels operadors $+=$, $-=$, $*=$ i $/=$ per a les classes *Real_Cn*, *Sumand_Csn* i *Terme_Csn*³⁰.
- Ús de la sobrecàrrega dels operadors $<$, $>$ i $!=$ -com a funcions *friend*- per a les classes *Factor_Csn* i *Sumand_Csn*, necessaris per tal de poder dur a terme cerques en llistes d'objectes d'aquests elements.

³⁰ En el capítol annex s'inclou el codi de l'operació *Sumand_Csn::operator *= (const Sumand_Csn &)* i *Terme_Csn::operator += (const Sumand_Csn &)* -així com també les seves explicacions detallades- ja que són uns exemples representatius de la manera com s'han implementat aquestes operacions.

- L'operació *Simplificar_fmRn*, definida per a la classe *Real_Cn*, implementa un algorisme per a la simplificació de nombres reals. Aquesta operació és necessària degut al continu creixement dels coeficients reals de les classes numèriques; és per això que, en cada una de les operacions +=, -=, *= i /= aplicades a la classe *Real_Cn*, es realitza una simplificació dels seus coeficients. Aquesta operació de simplificació fa ús de la funció *GCD*, inclosa en la llibreria *NTL*, que dona com a resultat el màxim comú divisor de dos nombres enters; aplicada aquesta funció sobre el numerador i denominador d'un nombre real, el seu resultat s'utilitza per a simplificar aquests dos últims, deixant així reduïda al mínim la mida dels nombres reals –aquesta operació és de vital importància ja que els nombres reals utilitzats es poden fer molt grans, i són els elements que tenen més pes en memòria de tots els que ha de manipular l'algorisme general-.

5. RESULTATS I RENDIMENT

En el capítol cinquè, es presenten els resultats obtinguts mitjançant l'eina de programari obtinguda a partir del disseny decidit en el capítol anterior: es fa referència als fluxos d'entrada i sortida requerits per aquesta eina i es tracten qüestions de rendiment i temporització derivades de la implementació.

5.1. PARÀMETRES D'ENTRADA

Per a iniciar per línia de comandes l'eina de programari dissenyada, s'executa l'ordre següent:

$$e_cci\ c:x\ p:y$$

on: x és l'ordre màxim de les constants d'isocronia a calcular $-x \in [2,9]$ -; y és l'ordre màxim de les pertorbacions del sistema $-y \in [2,9]$ -

Si no s'introdueixen els paràmetres c o p o els seus valors no són correctes, l'eina de programari genera un missatge d'ajuda en pantalla.

5.2. DADES DE SORTIDA

L'eina de programari dissenyada genera dades de sortida per dues vies diferents: per una banda, es genera l'arxiu $n_cci.nb$, que conté les constants d'isocronia calculades –aquest arxiu és directament interpretable pel manipulador algebraic *Mathematica*-; per l'altra, es generen per pantalla dades referents a la temporització de les diferents etapes de càlcul de l'algorisme general.

5.3. COMPARACIÓ DE TEMPS DE CàLCUL

Les taules 5.1, 5.2 i 5.3 contenen, respectivament, els temps de càlcul de les constants d'isocronia per al cas quadràtic $-p = 2$ -, cúbic $-p = 3$ - i quàrtic $-p = 4$ -. La primera columna indica l'ordre $-c$ - de la constant calculada; la segona columna indica el temps total de càlcul que utilitza el manipulador algebraic *Mathematica*; finalment, la tercera columna indica el temps total de càlcul que utilitza l'eina de programari desenvolupada en aquest treball.

Taula 5.1. Temps de càlcul de les constants d'isocronia per al cas quadràtic $-p = 2$ -

constant d'isocronia -ordre c -	temps de càlcul - <i>Mathematica</i> -	temps de càlcul - <i>e_cci</i> -
$c = 2$	0,110 s.	0,000 s.
$c = 3$	0,593 s.	0,055 s.
$c = 4$	22,813 s.	0,769 s.
$c = 5$	7.224,311 s.	5,989 s.
$c = 6$...	28,846 s.
$c = 7$...	129,835 s.

Taula 5.2. Temps de càlcul de les constants d'isocronia per al cas cúbic $-p = 3$ -

constant d'isocronia -ordre c -	temps de càlcul - <i>Mathematica</i> -	temps de càlcul - <i>e_cci</i> -
$c = 2$	0,109 s.	0,000 s.
$c = 3$	1,422 s.	0,055 s.
$c = 4$	7,765 s.	1,044 s.
$c = 5$	65,437 s.	10,495 s.
$c = 6$	1.533,235 s.	76,648 s.
$c = 7$...	599,341 s.

Taula 5.3. Temps de càlcul de les constants d'isocronia per al cas quàrtic $-p = 4$ -

constant d'isocronia -ordre c -	temps de càlcul - <i>Mathematica</i> -	temps de càlcul - <i>e_cci</i> -
$c = 2$	0,125 s.	0,000 s.
$c = 3$	1,422 s.	0,055 s.
$c = 4$	20,297 s.	1,044 s.
$c = 5$	148,203 s.	11,044 s.
$c = 6$	1.805,793 s.	87,692 s.
$c = 7$...	805,824 s.

Aquests tests de rendiment s'han realitzat en un *Intel Core 2 Duo*-1,83 GHz., sota el sistema operatiu *Microsoft Windows XP*.

En les taules anteriors, es pot veure com el temps de càlcul creix de manera exponencial a mesura que es calculen constants d'isocronia d'ordre superior. Així mateix, els temps de càlcul són notablement més òptims en l'eina desenvolupada en aquest treball que en el manipulador algebraic *Mathematica* –això és degut al fet que l'eina desenvolupada en aquest treball ha estat dissenyada específicament per a resoldre aquest problema; en canvi, el manipulador algebraic *Mathematica* està desenvolupat per a realitzar càlculs de molts i diversos àmbits de la matemàtica. A més, el manipulador algebraic *Mathematica* té un baix rendiment en els càlculs que requereixen la utilització i manipulació de constants arbitràries –com és el cas del càlcul de les constants d'isocronia-.

6. CONCLUSIONS I MILLORES

El sisè capítol reuneix les conclusions extretes d'aquest treball i, alhora, exposa tot un seguit de possibles millores proposades per a futures ampliacions del mateix.

6.1. CONCLUSIONS

De la realització d'aquest treball se'n poden extreure les següents conclusions:

Molt sovint, per tal de resoldre problemes específics –en aquest cas, un problema matemàtic, tot i que aquest comentari es podria aplicar a molts altres àmbits de la computació-, per tal d'optimitzar els resultats, hom es veu amb la necessitat de desenvolupar eines específicament dissenyades per a aquest propòsit.

S'ha obtingut una eina capaç de calcular les constants d'isocronia –en qualsevol dels seus casos- fins a les constants d'ordre nou –s'ha cregut convenient introduir aquesta limitació degut a la pràctica impossibilitat de calcular constants d'ordre superior pel seu cost temporal; de totes maneres, aquesta limitació és fàcilment suprimible, i es pot deixar que els càlculs continuïn per a constants d'ordre superior amb l'única limitació dels recursos, d'espai de memòria i de cost temporal, disponibles en el sistema-. Els resultats obtinguts són directament interpretables pel manipulador algebraic *Mathematica*, d'ús generalitzat dins del camp de la computació matemàtica. També, per la importància que té el cost temporal dels càlculs realitzats per l'algorisme general, l'eina de programari dissenyada extreu, per pantalla, informació referent a aquests temps de càlcul, mostrant quines són les parts de l'algorisme general que tenen un cost temporal més elevat.

Un dels punts crítics en tot el procés ha estat el del disseny de les estructures de dades contenidores –en aquest cas, una llista d'accés seqüencial i directe i un iterador associat a la llista-. L'òptim funcionament d'aquestes classes influeix, de manera decisiva, en el rendiment general de l'aplicació. El fet que l'algorisme de càlcul requereixi constantment de recorreguts exhaustius de grans quantitats d'objectes posa de relleu la gran importància d'un bon disseny d'aquestes estructures de dades contenidores, dels algorismes de cerca i d'una bona gestió de la còpia d'objectes a memòria –minimitzar aquestes còpies mitjançant l'ús exhaustiu de referències als objectes-.

Un altre dels punts crítics és el d'escollir un algorisme òptim per al càlcul integral. Aquest càlcul representa una bona part del cost temporal de l'algorisme general. El càlcul integral aplicat a funcions trigonomètriques té un caràcter intrínsecament recursiu; tot i això, hi ha diversos procediments per arribar a la resolució d'aquests càlculs. L'adopció d'una o altra variant pot influir en el rendiment general. S'ha escollit un procediment que té una implementació fàcilment traduïble a una forma de càlcul iterativa; aquesta variant sembla ser, també, de les més òptimes.

Anteriorment, s'ha fet referència a l'ús de grans quantitats d'objectes en memòria per part de l'algorisme de càlcul, allotjats, aquests, per una estructura de dades contenidora –una llista-. La gran part del pes en memòria d'aquests objectes cal atribuir-lo a la classe *ZZ* agregada a aquests objectes –que forma part de la llibreria per al treball amb grans nombres *NTL*-. Aquesta classe allotja els coeficients numèrics necessaris –pertanyents al conjunt dels nombres reals-. S'ha optat per utilitzar dos objectes del tipus *ZZ* per a construir els coeficients reals en forma de quocient, ja que la classe *RR* –classe dels nombres reals, també pertanyent a la llibreria *NTL*- ofereix un cost temporal molt elevat pel que fa a la seva manipulació.

6.2. POSSIBLES MILLORES

Primerament, una de les possibilitats de millora del present treball pot ser la paral·lelització de l'algorisme de càlcul. Així, els algorismes presentats en el capítol 3 poden adaptar-se per tal que les expressions calculades es puguin obtenir de forma simultània, segons les seves dependències de dades. Després, de la mateixa manera, el càlcul integral aplicat a cada un dels sumands trigonomètrics pot dur-se a terme de manera distribuïda, i beneficiar-se, així, de la millora, en el rendiment temporal, que això suposaria.

Una altra millora podria ser la modificació de l'eina desenvolupada per tal que aquesta pugui carregar els resultats de càlculs realitzats en sessions anteriors i continuar els càlculs per a constants d'ordre superior a partir d'aquestes dades.

Finalment, es podria aprofundir en l'estudi de la optimització dels dos punts de disseny crítics pel que fa al rendiment temporal de l'algorisme general: el disseny de les classes contenidores i l'optimització de l'algorisme d'integració trigonomètrica. També es podria re-implementar la classe contenidora com un AVL –enlloc d'utilitzar llistes dinàmiques- per tal d'aconseguir un cost en les cerques $\vartheta(\log(n))$.

7. ANNEX

Aquest capítol annex conté informació suplementària i de referència, com pot ser la localització del codi en llenguatge de programació C++ de la implementació, exemples representatius i significatius dels diversos algorismes implementats i una introducció a la llibreria per al treball amb grans nombres *NTL* utilitzada en aquest treball.

7.1. LOCALITZACIÓ DEL CODI EN LLENGUATGE DE PROGRAMACIÓ C++ DE LA IMPLEMENTACIÓ

El contingut dels diferents arxius amb el codi en llenguatge C++ de la implementació és el següent:

<i>c_cci.cpp</i>	→ arxiu de codi C++ amb el procediment principal de l'algorisme general.
<i>c_n.cpp</i>	→ arxiu de codi C++ amb les operacions membres de les classes numèriques.
<i>c_sn.cpp</i>	→ arxiu de codi C++ amb les operacions membres de les classes simbòlico-numèriques.
<i>c_st.cpp</i>	→ arxiu de codi C++ amb les operacions membres de les classes simbòliques.
<i>c_t.cpp</i>	→ arxiu de codi C++ amb les operacions membres de les classes trigonomètriques.
<i>e_cci.exe</i>	→ arxiu executable –executa l'eina desenvolupada en aquest treball–.
<i>h_a.h</i>	→ arxiu de codi C++ amb la definició de tipus de dades i classes alfanumèriques.
<i>h_c.h</i>	→ arxiu de codi C++ amb la definició de tipus de dades i classes contenidores.

<i>h_m.h</i>	→ arxiu de codi C++ amb la definició de tipus de dades i classes de memòria.
<i>h_n.h</i>	→ arxiu de codi C++ amb la definició de tipus de dades i classes numèriques.
<i>h_sn.h</i>	→ arxiu de codi C++ amb la definició de tipus de dades i classes simbòlico-numèriques.
<i>h_st.h</i>	→ arxiu de codi C++ amb la definició de tipus de dades i classes simbòliques.
<i>h_t.h</i>	→ arxiu de codi C++ amb la definició de tipus de dades i classes trigonomètriques.

També, l'arxiu *b_fer.bat* és el conjunt d'ordres de sistema operatiu per a compilar els arxius de codi anteriors.

Finalment, els arxius *CalculCentreFocus.nb* i *CalculCentreIsocro.nb* són interpretables directament pel manipulador algebraic *Mathematica* i calculen les constants d'isocronia segons els mètodes del *Centre-Fòcus* i del *Centre Isocró*, respectivament.

7.2. EXEMPLES REPRESENTATIUS I SIGNIFICATIUS DELS DIVERSOS ALGORISMES IMPLEMENTATS

En aquest apartat, es dona detall de diversos algorismes que, per la seva importància o per la seva representativitat en relació amb la manera com s'han implementat les diverses operacions, poden resultar força explicatius. Aquests algorismes són, també, els que afecten de manera més significativa el rendiment de l'algorisme general.

7.2.1. ALGORISME DE CERCA SOBRE LLISTES ORDENADES

En la interfície de la classe *Iterador_gCc* s'hi ha inclòs una operació per a la cerca d'elements sobre una llista –mitjançant l'iterador esmentat-. Aquesta operació de cerca és necessària pel fet d'haver de fer insercions ordenades sobre les llistes, però també per a mantenir aquestes amb un nombre no molt elevat d'elements –ja que un element que es vol inserir en una llista només s'inserirà si encara no existeix dins de l'esmentada llista; per això, caldrà cercar-lo abans-.

El quadre 7.1 conté el codi de l'operació de cerca.

Quadre 7.1. Codi de l'operació de cerca

```
template <class Tipus_gC> Boolea iTn Iterador_gCc <Tipus_gC>:: Cercar_fm1c
(const Tipus_gC & Element_prT) {

    Natural iTn IndexI_vNn = 0;
    Natural iTn IndexM_vNn = 0;
    Natural iTn IndexS_vNn = 0;

    Node_gSc <Tipus_gC> * IndexI_vaNc = Referent_aaNc -> Posterior_aaNc;
    Node_gSc <Tipus_gC> * IndexM_vaNc = Referent_aaNc -> Posterior_aaNc;
    Node_gSc <Tipus_gC> * IndexS_vaNc = Referent_aaNc -> Posterior_aaNc;

    while (IndexS_vaNc != Referent_aaNc) {

        IndexS_vNn += 1;

        IndexS_vaNc = IndexS_vaNc -> Posterior_aaNc;

    }

    while (IndexI_vNn < IndexS_vNn) {

        while (IndexM_vNn < ((IndexI_vNn + IndexS_vNn) / 2)) {

            IndexM_vNn += 1;

            IndexM_vaNc = IndexM_vaNc -> Posterior_aaNc;

        }

    }

}
```

```

if ((* IndexM_vaNc -> Element_aaT) < Element_prT) {

    IndexI_vNn = IndexM_vNn + 1;

    IndexI_vaNc = IndexM_vaNc -> Posterior_aaNc;

}

else if ((* IndexM_vaNc -> Element_aaT) > Element_prT) {

    IndexS_vNn = IndexM_vNn;
    IndexS_vaNc = IndexM_vaNc;

}

else {

    IndexI_vNn = IndexM_vNn;
    IndexS_vNn = IndexM_vNn;
    IndexI_vaNc = IndexM_vaNc;
    IndexS_vaNc = IndexM_vaNc;

}

IndexM_vNn = IndexI_vNn;
IndexM_vaNc = IndexI_vaNc;

}

Referencia_aaNc = IndexM_vaNc;

if (Referencia_aaNc == Referent_aaNc)

    return (false);

else if ((* Referencia_aaNc -> Element_aaT) != Element_prT)

    return (false);

else

    return (true);

}

```

L'algorisme de cerca s'ha d'implementar sobre una llista creada dinàmicament –les llistes no són estàtiques ja que el nombre d'elements que poden contenir pot variar molt-; una llista creada estàticament –és a dir, un vector- amb un nombre variable d'elements, que pot arribar a ser molt alt, però també molt baix, condueix a un desaprofitament molt important d'espai de memòria.

La primera idea seria implementar una cerca seqüencial, però el seu cost és $\vartheta(n)$ i, a la vegada, es desaprofitaria una característica molt important d'aquestes llistes: s'han de mantenir ordenades en tot moment. Això condueix a pensar que potser la millor opció seria implementar un algorisme que simulés la cerca dicotòmica, tot i que no ho sigui realment, ja que la cerca dicotòmica necessita un vector estàtic per a funcionar amb un cost real $\vartheta(\log(n))$.

La idea és la següent:

- Es creen tres apuntadors sobre la llista referenciada per l'iterador juntament amb tres nombres associats –respectivament- a cada un dels apuntadors esmentats. Un dels apuntadors s'inicialitza al començament de la llista, un altre al final i el tercer farà de pivot –els tres nombres s'ajustaran per a que indiquin en tot moment les posicions dels seus apuntadors associats dins de la llista-.
- Aleshores, s'actua com en una cerca dicotòmica: es calcula l'element mig de la sub-llista indicada pels apuntadors inicial i final i s'hi col·loca l'apuntador pivot. Si aquest és més gran que l'element buscat, la següent sub-llista per a la cerca serà la que començarà amb l'apuntador inicial i acabarà amb l'element pivot; sinó, la següent sub-llista per a la cerca serà la que començarà amb l'element pivot i acabarà amb l'apuntador final.
- En cas que l'element pivot sigui l'element buscat, la cerca retorna *cert* i col·loca l'iterador sobre l'element cercat; si l'apuntador inicial i el final s'arriben a creuar –detectable mitjançant els nombres associats als apuntadors- i l'element indicat per l'apuntador pivot no és l'element cercat, es retorna *fals* i l'apuntador es col·loca en la posició immediatament posterior a la que ocuparia l'element cercat –fet que afavoreix una posterior inserció ordenada de l'element que s'ha cercat-.

Tot i simular una cerca dicotòmica, l'algorisme de cerca implementat té un cost $\vartheta(n)$, ja que, per a col·locar l'apuntador pivot en les posicions requerides, aquest ha de recorre seqüencialment les posicions que el separen del seu destí en cada iteració. Tot i això, una cerca seqüencial seria molt més costosa ja que, en aquest cas, l'algorisme hauria de fer una mitjana de n comparacions d'elements de la llista; en canvi, en l'algorisme implementat, el nombre mitjà de comparacions a fer és $\log(n)$ -a més, el cost temporal de les comparacions d'elements d'una llista és considerablement més gran que el cost de desplaçar un apuntador seqüencialment en la mateixa llista-.

Es podria dir que l'algorisme de cerca implementat és una versió millorada d'una cerca seqüencial que, tot i no arribar a tenir un cost $\vartheta(\log(n))$, millora substancialment el cost d'una cerca seqüencial.

Una altra solució podria haver estat no implementar llistes sinó AVL's –que permeten cerques amb cost $\vartheta(\log(n))$ sobre estructures dinàmiques-. Com que s'ha vist que un dels requeriments de l'enunciat del problema que indica que l'eina implementada ha de ser notablement més ràpida que el *Mathematica* es compleix sobradament –veure el capítol de resultats-, s'ha deixat aquesta opció com una possible millora d'aquest treball –veure el capítol de conclusions-.

7.2.2. ALGORISMES QUE IMPLEMENTEN OPERACIONS ARITMÈTIQUES

Entre les operacions més utilitzades dins del conjunt d'operacions de les diferents classes implementades, hi ha les operacions aritmètiques entre objectes d'aquestes classes. Concretament, les operacions de suma i producte són les més importants per la seva utilització constant –cosa que fa que una bona implementació de les mateixes tingui conseqüències positives en el rendiment de l'algorisme general-.

D'altra banda, cal comentar que s'ha optat per implementar totes les operacions aritmètiques mitjançant la sobrecàrrega d'operadors que modifiquin l'objecte al qual s'apliquen; d'aquesta manera, s'evita el fet de retornar valors molt grans en memòria –els resultats d'aquestes operacions-.

Les operacions detallades són les de suma d'un terme amb un sumand –ja siguin simbòlics, trigonomètrics o numèrics- i la de multiplicació de dos sumands numèrics:

- *Terme_Cst::operator += (const Sumand_Cst &).*
- *Terme_Ct::operator += (const Sumand_Ct &).*
- *Terme_Csn::operator += (const Sumand_Csn &).*
- *Sumand_Csn::operator *= (const Sumand_Csn &).*

El quadre 7.2 conté el codi d'aquestes operacions.

Quadre 7.2. Codi de les operacions aritmètiques més importants

```

Buit iTm Terme_Cst:: operator += (const Sumand_Cst & Operand_prSst) {
    Iterador_gCc <Sumand_Cst> Index_vlc (S_aLc);
    if (Index_vlc . Cercar_fmLc (Operand_prSst)) {
        Sumand_Cst * Auxiliar_vaSst = S_aLc . Consultar_fcLc (Index_vlc);
        * Auxiliar_vaSst += Operand_prSst;
        if (Auxiliar_vaSst -> EsBuit_fcSst ())
            S_aLc . Eliminar_fmLc (Index_vlc);
    }
    else if (! Operand_prSst . EsBuit_fcSst ()) {
        Sumand_Cst * Operand_vaSst = new Sumand_Cst (Operand_prSst);
        S_aLc . Inserir_fmLc (Index_vlc,
                             Operand_vaSst);
    }
}

Buit iTm Terme_Ct:: operator += (const Sumand_Ct & Operand_prSt) {
    Iterador_gCc <Sumand_Ct> Index_vlc (S_aLc);
    if (Index_vlc . Cercar_fmLc (Operand_prSt)) {
        Sumand_Ct * Auxiliar_vaSt = S_aLc . Consultar_fcLc (Index_vlc);
        * Auxiliar_vaSt += Operand_prSt;
        if (Auxiliar_vaSt -> EsBuit_fcSt ())
            S_aLc . Eliminar_fmLc (Index_vlc);
    }
    else if (! Operand_prSt . EsBuit_fcSt ()) {
        Sumand_Ct * Operand_vaSt = new Sumand_Ct (Operand_prSt);
        S_aLc . Inserir_fmLc (Index_vlc,
                             Operand_vaSt);
    }
}

```


Abans de donar l'explicació detallada del codi de les operacions del quadre anterior, cal explicar com s'organitzen els sumands dins les llistes que apareixen com a atributs dels termes. Aquestes llistes es mantenen ordenades en tot moment per uns criteris d'ordenació definits en les operacions $< i >$ de les diverses classes sumand. Aquests criteris d'ordenació són els següents:

- Els sumands simbòlics i numèrics tenen, cada un d'ells, com a atributs, un coeficient i una llista de factors –simbòlics i numèrics, respectivament-. Aquesta llista de factors –també ordenada segons els criteris d'ordenació definits en les operacions $< i >$ de les diverses classes factor- és l'atribut que defineix els criteris d'ordenació per a aquests dos tipus de sumand.
- Els sumands trigonomètrics tenen, com a atribut, un coeficient i dos exponents –un per al cosinus i un altre per al sinus-. Aquest parell d'exponents són els atributs que defineixen els criteris d'ordenació per a aquest tipus de sumand.

També s'assegura que, en tot moment, les llistes de sumands ordenades definides pels termes no continguin cap sumand duplicat –entenen per duplicació de sumands l'existència de dos o més sumands amb els valors dels atributs, utilitzats en els criteris d'ordenació, idèntics-. Això s'aconsegueix fent que qualsevol inserció d'un sumand en un terme s'implementi com una suma d'aquest sumand en aquell terme –aquest procediment s'aplica per tal d'evitar que les llistes de sumands creixin indefinidament-. Aquestes operacions de suma són les que es detallen a continuació –el codi d'aquestes operacions és el que apareix en el quadre 7.2-.

Es pot comprovar com l'algorisme de les tres operacions de suma del quadre 7.2 segueix el mateix esquema: donat el terme al qual s'hi ha d'afegir el sumand proporcionat com a paràmetre, es crea, prèviament, un iterador sobre la llista de sumands del terme; seguidament, es cerca sobre la llista el possible sumand que pugui sumar-se directament al sumand proporcionat com a paràmetre –per que un és un duplicat de l'altre, en el cas que només es diferenciïn en l'atribut coeficient-; si existeix, es consulta la llista per tal d'obtenir el sumand en qüestió i es realitza la suma –si el resultat és nul, s'elimina de la llista-; en canvi, si el sumand no existeix dins la llista del terme i el sumand proporcionat com a paràmetre no és nul, aquest últim s'insereix directament a la llista –mitjançant l'iterador utilitzat en la cerca, cosa que garanteix el manteniment de l'ordenació de la llista-.

Pel que fa a l'operació de multiplicació de dos sumands numèrics, primerament es multipliquen els coeficients; després, per cada factor numèric de l'operand passat com a paràmetre, se'n fa una còpia i s'insereix ordenadament a la llista de factors numèrics de l'operand original.

7.2.3. INTEGRACIÓ DE TERMES TRIGONOMÈTRICS

El funcionament i les característiques de la integració de termes trigonomètrics es tracten en aquesta secció, a part de la resta, donada la importància i l'impacte –per exemple, en temps d'execució- que aquesta operació té en l'algorisme general.

Tot seguit, s'exposa el procés matemàtic seguit per a trobar una solució al càlcul de la integral d'un terme trigonomètric. Aquesta integral d'un terme trigonomètric serà la suma de les integrals dels sumands trigonomètrics que formen el propi terme trigonomètric. Es defineix la integral d'un sumand trigonomètric com:

$$I_{p,q}(\varphi) = \int \cos^p(\varphi) \sin^q(\varphi) d\varphi$$

Aleshores, s'utilitza el mètode de la integració per parts per a trobar-hi una solució:

$$\begin{aligned} \bullet \quad I_{p,q}(\varphi) &= \int \cos^p(\varphi) \sin^q(\varphi) d\varphi \Rightarrow \\ &\Rightarrow I_{p,q}(\varphi) = \int \cos^{p-1}(\varphi) \cos(\varphi) \sin^q(\varphi) d\varphi \end{aligned}$$

$$\begin{aligned} \text{on: } u(\varphi) &= \cos^{p-1}(\varphi); \dot{u}(\varphi) = -(p-1)\cos^{p-2}(\varphi)\sin(\varphi)d\varphi; \\ v(\varphi) &= \frac{1}{(q+1)}\sin^{q+1}(\varphi); \dot{v}(\varphi) = \cos(\varphi)\sin^q(\varphi)d\varphi \end{aligned}$$

Llavors, seguint els passos del mètode de la integració per parts:

$$\begin{aligned}
 & \bullet I_{p,q}(\varphi) = \frac{1}{(q+1)} \cos^{p-1}(\varphi) \sin^{q+1}(\varphi) + \frac{(p-1)}{(q+1)} \int \cos^{p-2}(\varphi) \sin^{q+2}(\varphi) d\varphi \Rightarrow \\
 & \Rightarrow I_{p,q}(\varphi) = \frac{1}{(q+1)} \cos^{p-1}(\varphi) \sin^{q+1}(\varphi) + \\
 & + \frac{(p-1)}{(q+1)} \int \cos^{p-2}(\varphi) \sin^q(\varphi) \sin^2(\varphi) d\varphi \Rightarrow \\
 & \Rightarrow I_{p,q}(\varphi) = \frac{1}{(q+1)} \cos^{p-1}(\varphi) \sin^{q+1}(\varphi) + \\
 & + \frac{(p-1)}{(q+1)} \int \cos^{p-2}(\varphi) \sin^q(\varphi) (1 - \cos^2(\varphi)) d\varphi \Rightarrow \\
 & \Rightarrow I_{p,q}(\varphi) = \frac{1}{(q+1)} \cos^{p-1}(\varphi) \sin^{q+1}(\varphi) + \\
 & + \frac{(p-1)}{(q+1)} \int \cos^{p-2}(\varphi) \sin^q(\varphi) d\varphi - \\
 & - \frac{(p-1)}{(q+1)} \int \cos^p(\varphi) \sin^q(\varphi) d\varphi \Rightarrow \\
 & \Rightarrow \left(1 + \frac{(p-1)}{(q+1)}\right) I_{p,q}(\varphi) = \frac{1}{(q+1)} \cos^{p-1}(\varphi) \sin^{q+1}(\varphi) + \\
 & + \frac{(p-1)}{(q+1)} \int \cos^{p-2}(\varphi) \sin^q(\varphi) d\varphi \Rightarrow \\
 & \Rightarrow I_{p,q}(\varphi) = \frac{1}{(p+q)} \cos^{p-1}(\varphi) \sin^{q+1}(\varphi) + \frac{(p-1)}{(p+q)} I_{p-2,q}(\varphi)
 \end{aligned}$$

Si, en el primer pas del mètode de la integració per parts, es desglossa el terme $\sin^q(\varphi)$ en els termes $\sin^{q-1}(\varphi)$ i $\sin(\varphi)$ -enlloc de fer-ho amb el terme $\cos^p(\varphi)$ - s'arriba a aquesta altra solució, igualment vàlida:

$$\bullet I_{p,q}(\varphi) = -\frac{1}{(p+q)} \cos^{p+1}(\varphi) \sin^{q-1}(\varphi) + \frac{(q-1)}{(p+q)} I_{p,q-2}(\varphi)$$

Altrament, per a $p, q = 0$, la solució és:

$$\bullet I_{0,0}(\varphi) = \varphi$$

Així, resumint, per a $p, q \in \mathbb{N}$, la solució a la integral d'un sumand trigonomètric és de la forma:

$$I_{p,q}(\varphi) = \begin{cases} \frac{1}{(p+q)} \cos^{p-1}(\varphi) \sin^{q+1}(\varphi) + \frac{(p-1)}{(p+q)} I_{p-2,q}(\varphi); & p > 0 \text{ (cas a)} \\ -\frac{1}{(p+q)} \cos^{p+1}(\varphi) \sin^{q-1}(\varphi) + \frac{(q-1)}{(p+q)} I_{p,q-2}(\varphi); & q > 0 \text{ (cas b)} \\ \varphi; & p, q = 0 \text{ (cas c)} \end{cases} \quad (7.1)$$

De l'anàlisi de l'expressió (7.1), se'n dedueixen les següents afirmacions:

La solució de la integral en els casos a) i b) té un caràcter recursiu $-I_{p,q}(\varphi)$ depen de $I_{p',q'}(\varphi)$, amb $p' \leq p$ i $q' \leq q$.

La solució de la integral en el cas c) representa un dels casos base de la recursió; de la mateixa manera, si $p = 1$ -en el cas a)- o $q = 1$ -en el cas b)-, aleshores aquests casos resulten, també, casos base de la mateixa recursió.

La taula 7.1 resumeix, per a qualsevol combinació dels exponents p i q , quin és el cas a seguir, per part de l'algorisme que implementi el càlcul integral, de manera que aquest segueixi el camí més adient de cara a minimitzar el seu temps d'execució.

Taula 7.1. Casuística dels exponents p i q en l'algorisme del càlcul integral

	q = 0	q = 1	parell (q) q > 1	senar (q) q > 1
p = 0	cas base c)	cas base b)	cas recursiu b)	cas recursiu b)
p = 1	cas base a)	cas base a) cas base b)	cas base a)	cas base a)
parell (p) p > 1	cas recursiu a)	cas base b)	cas recursiu a) cas recursiu b)	cas recursiu b)
senar (p) p > 1	cas recursiu a)	cas base b)	cas recursiu a)	cas recursiu a) cas recursiu b)

Les decisions preses en aquest punt han estat les següents:

Si $p = 0$ i $q = 0$ o $p = 1$ o $q = 1$, aleshores l'algorisme tria, respectivament, els casos c), a) i b), per tal de situar-se en un cas base i acabar, així, la recursió; la resta de casos són recursius.

Si $q = 0$ i p és parell o p és senar, aleshores l'algorisme tria el cas a), ja que en el cas b) no és permesa la condició $q = 0$ -l'exponent de $\sin(\varphi)$ esdevindria negatiu-. A partir d'aquí, l'algorisme escollirà sempre el cas a) fins que $p = 1$ -partint d'un exponent p senar- o $p = 0$ -partint d'un exponent p parell-.

Si $p = 0$ i q és parell o q és senar, aleshores l'algorisme tria el cas b), ja que en el cas a) no és permesa la condició $p = 0$ -l'exponent de $\cos(\varphi)$ esdevindria negatiu-. A partir d'aquí, l'algorisme escollirà sempre el cas b) fins que $q = 1$ -partint d'un exponent q senar- o $q = 0$ -partint d'un exponent q parell-.

Si p és senar i q és parell, aleshores l'algorisme tria el cas a), ja que, d'aquesta manera, $p \rightarrow 1$ -fins arribar al cas base $p = 1$ -; de l'altra manera, $q \rightarrow 0$ però, llavors, s'hauria de continuar la recursió fins que $p = 1$.

Si q és senar i p és parell, aleshores l'algorisme tria el cas b), ja que, d'aquesta manera, $q \rightarrow 1$ -fins arribar al cas base $q = 1$ -; de l'altra manera, $p \rightarrow 0$ però, llavors, s'hauria de continuar la recursió fins que $q = 1$.

Si p i q són parells o p i q són senars, qualsevol dels dos casos -a) o b)- serà igualment vàlid per a arribar a un cas base: en el cas que p i q siguin senars, només caldrà que un dels dos exponents arribi a un cas base - $p = 1$ o $q = 1$ -; en el cas que p i q siguin parells, s'haurà de fer arribar tots dos exponents a un cas base - $p = 0$ i $q = 0$ - per a acabar la recursió -quan això passi, el cas base serà el cas c), que és el cas on apareixerà el paràmetre φ fora de l'argument de les funcions trigonomètriques $\sin(\varphi)$ i $\cos(\varphi)$ i, per tant, és on es generaran les possibles constants d'isocronia.

Tot i la naturalesa recursiva de l'algorisme del càlcul integral, a l'hora d'implementar aquest algorisme s'ha fet d'una manera únicament iterativa. Així, donats els valors dels exponents p i q del sumand trigonomètric a integrar, es generen dos sumands trigonomètrics nous –segons la casuística presentada en la taula 7.1-: el primer sumand –creat segons la fórmula matemàtica (7.1)- serà sumat a un terme trigonomètric que contindrà els sumands ja definitius com a resultat del càlcul integral –aquest és el primer dels sumands de les expressions de (7.1) en els casos a) o b)-; el segon sumand –creat, també, segons la fórmula matemàtica (7.1)- serà el que s'agafarà com a sumand trigonomètric a integrar en el següent pas de la iteració. Ara, els nous exponents p i q seran els exponents d'aquest segon sumand –aquest és el segon dels sumands de les expressions de (7.1) en els casos a) o b)-.

El quadre 7.3 conté el codi de l'algorisme d'integració.

Quadre 7.3. Codi de l'algorisme d'integració

```

Terme_Ct * Sumand_Ct:: Integrar_fmcSt (const Natural iTn & Subindex_prNn) const {

    Terme_Ct * Resultat_vaTt = new Terme_Ct (Subindex_prNn );
    Terme_Csn * Coeficient_vaTsn = new Terme_Csn (Coeficient_aTsn);

    Natural iTn Cosinus_vNn = Cosinus_aNn;
    Natural iTn Sinus_vNn = Sinus_aNn;

    while (CasBCr_pN (Cosinus_vNn, Sinus_vNn)) {                                     //cas b) o c)

        if (CasBr_pN (Cosinus_vNn, Sinus_vNn)) {                                   //cas b)

            * Coeficient_vaTsn /= (Cosinus_vNn + Sinus_vNn);

            Cosinus_vNn -= 1;
            Sinus_vNn += 1;

            Sumand_Ct * ParcialBr_vaSt = new Sumand_Ct (* Coeficient_vaTsn,
                                                         Cosinus_vNn,
                                                         Sinus_vNn);

            * Resultat_vaTt += * ParcialBr_vaSt; delete ParcialBr_vaSt;
            * Coeficient_vaTsn *= Cosinus_vNn;

            Cosinus_vNn -= 1;
            Sinus_vNn -= 1;

        }
    }
}

```

```

else if (CasCr_pN (Cosinus_vNn, Sinus_vNn)) {                                     //cas c

    * Coeficient_vaTsn /= (Cosinus_vNn + Sinus_vNn);

    Coeficient_vaTsn -> Negatiu_fmTsn ();

    Cosinus_vNn += 1;
    Sinus_vNn   -= 1;

    Sumand_Ct * ParcialCr_vaSt = new Sumand_Ct (* Coeficient_vaTsn,
                                                Cosinus_vNn,
                                                Sinus_vNn);

    * Resultat_vaTt += * ParcialCr_vaSt; delete ParcialCr_vaSt;
    * Coeficient_vaTsn *= Sinus_vNn;

    Coeficient_vaTsn -> Negatiu_fmTsn ();

    Cosinus_vNn -= 1;
    Sinus_vNn   -= 1;

}
}

if (CasAb_pN (Cosinus_vNn, Sinus_vNn)) {                                       //cas a

    * Resultat_vaTt += * Coeficient_vaTsn; delete Coeficient_vaTsn;

}

else if (CasBb_pN (Cosinus_vNn, Sinus_vNn)) {                                   //cas b

    * Coeficient_vaTsn /= (Cosinus_vNn + Sinus_vNn);

    Cosinus_vNn -= 1;
    Sinus_vNn   += 1;

    Sumand_Ct * ParcialBb_vaSt = new Sumand_Ct (* Coeficient_vaTsn,
                                                Cosinus_vNn,
                                                Sinus_vNn);

    * Resultat_vaTt += * ParcialBb_vaSt; delete ParcialBb_vaSt;
    delete Coeficient_vaTsn;

}

```

```

else if (CasCb_pN (Cosinus_vNn, Sinus_vNn)) { //cas c)

    * Coeficient_vaTsn /= (Cosinus_vNn + Sinus_vNn);

    Coeficient_vaTsn -> Negatiu_fmTsn ();

    Cosinus_vNn += 1;
    Sinus_vNn -= 1;

    Sumand_Ct * ParcialCb_vaSt = new Sumand_Ct (* Coeficient_vaTsn,
                                                Cosinus_vNn,
                                                Sinus_vNn);

    * Resultat_vaTt += * ParcialCb_vaSt; delete ParcialCb_vaSt;
                        delete Coeficient_vaTsn;

}

return (Resultat_vaTt);

}

```

7.3. INTRODUCCIÓ A LA LLIBRERIA *NTL*

La llibreria *NTL* per al treball amb grans nombres proveeix un seguit d'estructures de dades, classes i algorismes per a treballar amb objectes matemàtics com ara: nombres enters de mida arbitrària, vectors, matrius, polinomis, nombres reals, etc. En definitiva, proveeix una interfície consistent per a una gran varietat de classes que representen objectes matemàtics. La llibreria *NTL* és de lliure distribució, i es pot utilitzar d'acord amb els termes de la llicència *GNU*. El seu autor és *Victor Shoup* i la llibreria es pot aconseguir de la seva pàgina web -<http://shoup.net>-.

La llibreria està formada per una sèrie de mòduls; entre ells hi ha, per exemple:

- *GF2* → enters mòdul 2.
- *RR* → nombres reals.
- *ZZ* → nombres enters de llargada arbitrària.
- *matrix* → matrius de dos dimensions.
- *vectors* → macros per a treballar amb vectors de mida variable.
- etc.

Cada un d'aquests mòduls és accessible mitjançant l'arxiu de capçalera corresponent *-nom_modul.h-*.

El present treball ha fet ús de la classe *ZZ* –nombres enters de mida arbitrària- per tal d'implementar nombres reals, necessaris per la definició dels coeficients dels sumands trigonomètrics. Aquests nombres reals s'implementen com un quocient entre dos objectes de la classe *ZZ* –no s'ha utilitzat la classe *RR*, que representa nombres reals, per que aquesta arrodoneix els resultats en les operacions aritmètiques, cosa que fa que el resultat final no sigui vàlid-. Així mateix, també s'ha utilitzat l'operació *GCD* –màxim comú divisor- aplicada a dos nombres enters, per tal de mantenir reduïda al mínim l'expressió dels nombres reals implementats com a quocient –numerador i denominador es divideixen pel resultat de l'operació *GCD* en cada operació aritmètica realitzada sobre els coeficients reals-.

De totes les estructures de dades implementades en aquest treball, el paràmetre que té una ocupació de memòria més gran és precisament el coeficient real implementat a partir d'objectes de la classe *ZZ* –aquests nombres s'arriben a fer molt grans-. Això fa que s'hagi de tenir cura en la gestió de la mida d'aquests nombres per evitar que el rendiment de l'algorisme general se'n ressentixi.

Finalment, cal dir que, per a un bon funcionament de la llibreria *NTL*, cal invocar la macro *NTL_CLIENT* -proveïda per la mateixa llibreria- al començament del programa.

BIBLIOGRAFIA

Tot seguit, es presenta el material bibliogràfic utilitzat:

[1] G. PETIT BOIS, Tables of Indefinite Integrals. Dover Publications, 1961.

[2] J. LUÍS VARONA MALUMBRES, Métodos Clásicos de Resolución de Ecuaciones Diferenciales Ordinarias. Universidad de la Rioja, 1996.

[3] V. JIMÉNEZ LÓPEZ, Ecuaciones diferenciales (cómo aprenderlas, cómo enseñarlas). Servicio de Publicaciones de la Universidad de Murcia, 2000.

[4] M. A. LIAPUNOV, Problème général de la stabilité du mouvement. Ann. Of Math. Stud. 17, Princeton University Press, 1947.

[5] H. POINCARÉ, Mémoire sur les courbes définies par les équations différentielles. Journal de Mathématiques 37 (1881), 375-422; 8 (1882), 251-296; Oeuvres de Henri Poincaré, vol. I, Gauthier-Villars. Paris, 1951, pp. 3-84.

[6] J. CHAVARRIGA i J. GINÉ, Integrability of a linear center perturbed by homogeneous polynomial. "Proceedings of the 2nd Catalan Days on Applied Mathematics" (Odeillo, 1995), pp. 61-75. Collect. Études, Presses Univ. Perpignan. Perpignan, 1995.

[7] J. CHAVARRIGA, I. A. GARCÍA i J. GINÉ, Isochronous centers of a linear center perturbed by homogeneous polynomials. "Proceedings of the 3th Catalan Days on Applied Mathematics" (Lleida, 1996), pp. 65-80. Fundación Pública Instituto de Estudios Ilerdenses de la Diputación de Lleida. Lleida, 1996.

[8] J. CHAVARRIGA, I.A. GARCÍA i J. GINÉ, El problema del Centre-Focus. Varia Mathematica, 1 (2000), 175-191.

[9] BJARNE STROUSTROUP, The C++ Programming Language - Third Edition. Addison-Wesley, 1999.

[10] JOSEP M. RIBÓ, C++ Orientat a Objectes (apunts de classe). Universitat de Lleida, 1996.

[11] XAVIER FAUS TORÀ, Implementació i Optimització d'un Algorisme pel Càlcul de les Constants de Poincaré-Liapunov (TFC). Universitat de Lleida, 2000.

[12] RUBÉN QUADRAT GUIJARRO, Construcción de un Manipulador Algebraico para el Cómputo de las Constantes de Lyapunov (TFC). Universitat de Lleida, 1994.

[13] <http://www.shoup.net>