UNIVERSITAT DE BARCELONA
Facultat de Matemàtiques i Informàtica

GRAU DE MATEMÀTIQUES

Treball final de grau

MÈTODES NUMÈRICS PER EQUACIONS DIFERENCIALS APLICATS A MECÀNICA CELESTE

Autor: Daniel Cuadrillero Moles

Director: Dr. Àngel Jorba i Monte
Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 12 de juny de 2023

Abstract

In this document, we will explore a selection of interesting numerical methods for ordinary differential equations. Mainly, we will explain the Taylor methods, the Runge-Kutta methods and the Extrapolation methods, providing respectively a theoretical basis, where we will delve into key aspects of numerical methods, including convergence or stability.

In general, our aim will be to guide the theory to the implementation of the methods in programs in C language using advanced control techniques on step, that will allow us to obtain results with errors below a predetermined tolerance. For this reason, our primary focus will be on the concrete methods of Taylor applying automatic differentiation, Runge-Kutta-Fehlberg methods and the Extrapolation Gragg-Burlisch-Stoer method.

To demonstrate the practicality of these approaches, we will apply them to some celestial mechanics problems, such as the Central Force Problem or a more general version, the N-Body Problem. By utilizing actual data from the Solar System, we will compare the accuracy and efficiency of these three methods drawing appropriate conclusions.

Resum

En aquest document, revisarem alguns dels mètodes numèrics més interessants per a equacions diferencials ordinàries. Explicarem principalment els mètodes de Taylor, els mètodes Runge-Kutta i els mètodes d'Extrapolació, donant respectivament un context teòric, aprofundint en alguns aspectes clau per a mètodes numèrics com són la convergència o la estabilitat del propi mètode.

En general, la idea serà orientar la teoria cara a la implementació dels mètodes amb programes en llenguatge C mitjançant l'ús de tècniques de control de pas, que ens permitiràn obtenir resultats amb errors per sota una tolerància fixada. Per aquest motiu, ens centrarem en els mètodes específics de Taylor aplicant diferenciació automàtica, mètodes Runge-Kutta-Fehlberg i el mètode d'Extrapolació Gragg-Burlisch-Stoer.

Per demostrar la seva utilitat, dirigirem la implementació a l'aplicació dels mètodes en problemes de mecànica celeste, com ara el Problema de la Força Central o una versió més general, el Problema dels N Cossos. Compararem directament els tres mètodes utilitzant dades reals del Sistema Solar, i veurem quin és el més precís i quin presenta una millor eficiència extraient unes conclusions adequades.

2020 Mathematics Subject Classification. 65L05, 65L06, 65L20, 70F15

Agraïments

Voldria donar les gràcies especialment al tutor del treball, Àngel Jorba i Monte, per la seva dedicació i esforç en ajudar-me a realitzar aquest document. Per totes les hores dedicades, la seva guia i el seu suport durant la realització del treball.

Índex

1 Introducció

Les Matemàtiques regeixen el món, i avui dia això no és cap misteri. S'apliquen a una gran quantitat de disciplines, des de tecnologia i ciències fins en àmbits més socials.

En concret, una de les necessitats més importants de les branques científiques és el poder modelitzar tot allò que ens envolta i que està sotmès a un canvi. Això ens porta naturalment al camp de les equacions diferencials ordinàries o EDOs i la seva resolució de forma numèrica, concepte que també es coneix com integració numèrica. Una manera de formular aquests models matemàtics i estudiar-los de forma eficient és amb l'ús dels ordinadors, emprant mètodes i algorismes per aproximar les solucions de forma numèrica. Quan es parla de problemes amb solucions de tipus numèrica, es fan referència a problemes en els quals trobar una solució explícita és gairebé impossible o molt costós a nivell de càlculs, i per tant s'opta per calcular directament la solució del problema numèricament de forma aproximada, sense necessitat d'obtenir la fórmula explícita en un primer moment. Però també té la seva part negativa, i és que al treballar amb ordinadors un dels principals inconvenients són els errors en les operacions aritmètiques, ja que aquests fan els càlculs amb un nombre finit de dígits, i afectarà en certa manera als nostres resultats.

En aquest document revisarem alguns dels mètodes numèrics més rellevants per a la resolució d'EDOs, intentant seguir el desenvolupament que es va començar amb la teoria i els problemes explicats a les assignatures de Mètodes Numèrics i Equacions en Derivades Parcials. De forma resumida, s'explicaràn els mètodes de Taylor, amb especial incís en l'aplicació amb Diferenciació Automàtica, Runge-Kutta, amb la implementació pràctica del Runge-Kutta-Fehlberg, i Extrapolació, amb el mètode de Gragg-Burlisch-Stoer com a principal referència. Per altra banda, adjuntarem uns programes adequats, en llenguatge C, per aplicar els mètodes que es vagin explicant en problemes coneguts de mecànica celeste, com per exemple, el Problema de la Força Central o el Problema dels N Cossos.

Els objectius que ens marcarem seràn llavors estudiar i entendre els tres tipus de mètodes que veurem i implementar-los adequadament amb programes que aplicarem en problemes de mecànica celeste.

En primer lloc i abans d'entrar en matèria, es farà una petita revisió sobre aspectes generals sobre el tema d'integració numèrica, explicant quin és l'objectiu i repassant conceptes que s'aniràn repetint i ampliant al llarg de tot el document, com ara, error d'un mètode d'integració, estabilitat i convergència. Rematarem la secció comentant el conegut mètode d'Euler, que ens servirà com a base per poder explicar i entendre millor els mètodes que anirem descrivint.

2 Integració numèrica d'equacions diferencials

2.1 Objectiu i mètode d'integració

En aquest document, ens centrarem en la resolució del problema del valor inicial següent:

$$\begin{cases} x'(t) = f(t, x(t)), \\ x(a) = x_0, \end{cases} \tag{2.1}$$

Per simplicitat suposarem que f, sovint anomenada camp vectorial (ja que associa un vector de l'espai a un altre), és analítica en el seu domini de definició i $x(t)$ està definida per $t \in [a, b]$, variable a la que anomenarem temporal, a diferència de x que serà la variable espacial. A la vegada, la variable espacial serà una funció que podrà tenir una dimensió qualsevol n, $x : \mathbb{R} \longrightarrow \mathbb{R}^n, t \longmapsto x(t)$. El nostre objectiu serà trobar una aproximació de la solució $x(t)$ per temps $t = b$.

Notem que podem escriure la següent expressió

$$x(b) - x(a) = \int_b^a x'(t)\ dt = \int_b^a f(t, x(t))\ dt,$$

pel teorema fonamental del càlcul. Això ens indica que calcular el valor de $x(b)$ és equivalent a calcular aquesta integral, ja que disposem del valor $x(a)$ com a dada incial del problema. Per tant, ja veiem que la resolució del problema del valor inicial està fortament lligada a l'objectiu de la integració numèrica, que serà aproximar el valor de la integral

$$\int_b^a f(t, x(t))\ dt,$$

Per resoldre el problema, s'empren els anomenats mètodes d'integració numèrica, que generalment es defineixen com mètodes que utilitzen diferents procediments matemàtics per aproximar el valor de la integral mitjançant combinacions d'avaluacions del camp vectorial, o de les seves funcions derivades, en punts triats estratègicament segons cada mètode. Gràficament, un cop posicionats a un punt inicial del retrat de fase, la idea general dels mètodes és seguir la trajectòria de la solució parametritzant-la pel temps.

Existeixen gran varietat de mètodes d'integració, com ara, mètodes lineals d'un pas o multipas, mètodes de sèries de Taylor, mètodes d'interpolació o extrapolació, mètodes Runge-Kutta, mètodes híbrids, mètodes d'integració geomètrica, etc... Aquí només revisarem alguns d'ells.

En general, si ens trobem davant d'un problema de valor inicial, per poder aplicar un mètode haurem de definir un nombre natural N, un nombre $h > 0$ real i suficientment petit anomenat pas i una sèrie de passos temporals $t_0 = a,\ t_1 = t_0 + h,\ t_2 = t_1 + h, ..., t_N = t_{N-1} + h = b$. Estem fent així el que s'anomena discretització de la variable temporal t, que és contínua. El nombre N marcarà el nombre de particions que feim a l'interval $[a, b]$ de distància h, per tant, es satisfarà: $(b - a)/N = h$.

Dit això, intentarem donar una definició matemàtica general de mètode numèric:

$$\tilde{x}_{i+1} = \sum_{k=1}^{m} \alpha_k \tilde{x}_{i+1-k} + h\phi(t_{i+1}, ..., t_{i+1-m}, \tilde{x}_{i+1}, ..., \tilde{x}_{i+1-m}, h), \tag{2.2}$$

on els $\tilde{x}_i$ són els valors de la solució a temps t_i que es van calculant amb el mètode. Les constants α_k i la funció ϕ variarien segons el mètode, a més del nombre natural m, que és el nombre que marcarà quants passos diferents empra el mètode, diferenciant així mètodes d'un pas o multipas. Per exemple, en el cas del mètode d'Euler que veurem més endavant, $m = 1$, $\alpha_1 = 1$ i $\phi(t_{i+1}, t_i, x_{i+1}, x_i, h) = f(x_i, t_i)$. Aquesta definició té en compte tant si el mètode és de tipus explícit com implícit. Recordem que la diferència principal entre aquests dos tipus és que els mètodes implícits empren a la fórmula per al següent pas el propi valor que es vol calcular, fet que provoca que s'hagi de resoldre un sistema d'equacions per poder calcular cada valor de la successió.

2.2 Error i convergència d'un mètode

Hem comentat anteriorment que el nostre objectiu és aproximar la solució numèrica, i no calcular-la explícitament, per tant, estarem cometent un error a l'hora de realitzar els càlculs. Existeixen diferents tipus d'errors que es poden produir al aplicar un mètode:

- Errors d'arrodoniment: Són un tipus d'error comès durant la realització d'operacions aritmètiques quan es treballa amb un nombre finit de dígits. Només cal adonar-se de la gran quantitat de dígits que es van necessitant per representar els resultats a mesura que s'incrementa el nombre de dígits dels nombres que s'estàn operant. Són errors que no es poden evitar, ja que es troben en la majoria d'operacions bàsiques, la divisió n'és un exemple clar, $1/3 = 0.333333... \approx 0.33$, si només féssim les operacions fins a dos dígits després de la coma. Els hem de tenir molt presents, perquè es poden anar acumulant a la llarga i suposar una greu imprecisió en les dades, a més de fer que el mètode sigui numèricament inestable.

- Errors de truncament o discretització: Són uns errors comesos a causa del reemplaçament de problemes contínus per problemes discrets. Per exemple, per calcular la integral d'una funció contínua, com és el nostre cas, necessitaríem avaluar el camp en tot l'interval d'integració, però a la pràctica només s'utilitzaria un nombre finit de punts.

Un altre tipus d'error que cal mencionar, són els errors comesos en alguns mètodes implícits, on s'utilitza una expressió que s'ha d'iterar fins que acaba convergint a la solució del problema. En molts casos, la convergència del mètode està més que demostrada, però suposant que es pogués fer la iteració infinitament, fet que no és possible. Així llavors, hi hauria un error a l'hora d'aturar aquest procés iteratiu.

Els errors són importants en un mètode, no es poden evitar però si es poden controlar, de tal manera que es pot aconseguir una precisió dessitjada en els càlculs, realitzant diferents tècniques de control. Mantenir un pas constant al llarg de tota la integració pot no ser convenient, ja que en certes regions pot passar que el camp sigui "suau", i per tant, es cometi un error molt petit usant un pas relativament gran; mentre que, al cap d'una estona, la trajectòria pot entrar en una regió on el camp variï molt, i si volem tenir poc error en la integració dins d'aquesta regió, ens cal prendre un pas molt petit. Si usem un pas constant durant tota la integració, i volem mantenir-nos per sota d'un nivell d'error, ens veurem obligats a usar un pas petit durant tota la trajectòria, cosa que pot alentir notablement el procés de càlcul.

Un concepte molt relacionat és l'eficiència del mètode, que marca la "qualitat"del mateix a partir dels recursos que s'empren per realitzar el mètode. Computacionalment,

l'eficiència es mesuraria per l'ús de la memòria de l'ordinador i pel temps que tardaria en executar l'algorisme del mètode. Un "bon"mètode hauria de ser capaç de combinar aquests dos aspectes d'una forma equilibrada.

Matemàticament, i seguint amb la notació de la secció anterior, definirem l'*error global de discretització* com:

$$E(h) = \max_{1 \leq i \leq N} |\tilde{x}_i - x(t_i)|,$$

on els $\tilde{x}_i$ són els valors de la solució a temps t_i que es van calculant amb el mètode, i $x(t_i)$ és el valor real de la solució del problema en aquest punt.

L'*error local de truncament* o discretització es podria definir de la mateixa manera però només tenint en un compte un pas fixe i:

$$R_{i+1}(h) = \tilde{x}_{i+1} - x(t_{i+1}),$$

on hem suposat localment que $\tilde{x}_i = x(t_i)$, és a dir, que la solució proporcionada pel mètode al pas anterior és exacta.

També direm que el mètode numèric és *convergent* si $E(h)$ convergeix a 0 al fer el límit quan h tendeix a 0, o equivalentment, N tendeix a ∞. Generalment, $E(h) = O(h^p)$ per un nombre p natural que variarà segons el mètode i que s'anomena *ordre de convergència*, o equivalentment $R_{i+1}(h) = O(h^{p+1})$ com veurem més endavant. Aquest, sol ser un bon indicador de la velocitat a la qual convergeix la successió del mètode iteratiu amb el que estem treballant.

Emprarem d'ara en endavant la notació $z = O(h^p)$ on p és un nombre natural, per representar que existeixen unes constants positives C i h_0 tal que

$$|z| \leq Ch^p, \text{ per } 0 < h < h_0,$$

Així que z convergeix a 0 quan $h \to 0$ amb ordre de convergència p.

2.3 Estabilitat d'un mètode

Un altre dels aspectes fonamentals d'un mètode numèric és la seva estabilitat. Per estabilitat, entenem que per canvis arbitràriament petits a les condicions inicials del problema, la solució proporcionada pel mètode sofreix també canvis en una mesura menor o igual a aquesta. L'estabilitat descriu com els errors es van propagant a mesura que es realitza l'algorisme. Un mètode inestable seria aquell mètode on aquests canvis a les condicions del problema produirien canvis suficientment grans a la solució.

Suposem que donat un mètode numèric amb pas h, el volem aplicar a la següent equació diferencial:

$$\begin{cases} x'(t) = \lambda x(t), \\ x(0) = 1, \end{cases} \tag{2.3}$$

amb $\lambda \in \mathbb{C}$. La solució exacta de (2.3) és clarament $x(t) = e^{\lambda t}$, i per tant, $\lim_{t\to\infty} x(t) = 0$ si i només si $Re(\lambda) < 0$. Direm que el mètode aplicat és *estable* si dona una successió de la solució $\{\tilde{x}_n\}_n$ que està acotada $\forall h \in \mathbb{R}$. Si volem estudiar l'estabilitat d'un mètode numèric on $x(t)$ és una funció $x : \mathbb{R} \longrightarrow \mathbb{R}^n$, llavors enlloc d'un valor λ, en aquest

cas tindríem una matriu A que diagonalitzaríem, i estudiaríem l'estabilitat component a component amb la definició explicada, però és clar que amb lambdes diferents segons els valors propis de la matriu.

Definirem també com el *domini d'estabilitat* $\mathcal{D}$ del mètode com el conjunt de tots els nombres $h\lambda \in \mathbb{C}$ tal que $\lim_{n\to\infty} \tilde{x}_n = 0$. Per tant, si $\mathcal{D}$ és tot $\mathbb{C}$, direm que el mètode en qüestió és estable.

Si volem que la successió dels valors de la solució sigui acotada s'ha de complir que $Re(\lambda) < 0$, i tenint en compte que generalment s'utilitzen passos temporals positius $h > 0$, llavors ens interessa especialment el que passi a la part del pla $\mathbb{C}^- := \{z \in \mathbb{C} : Re(z) < 0\}$. Per tant, en aquest cas no ens faria falta que el domini d'estabilitat fos tot $\mathbb{C}$, sinó que amb $\mathbb{C}^-$ ja podríem dir que el mètode és suficientment estable. Aquests mètodes reben el nom de *A-estables* (assimptòticament estables) i compleixen que $\mathbb{C}^- \subseteq \mathcal{D}$.

L'estabilitat d'un mètode serà un aspecte clau a mirar si després volem provar a variar les condicions inicials del problema, és una bona característica de robustesa del mètode enfront a petits canvis. A la pràctica, no cal que sempre treballem amb mètodes estables o A-estables des d'aquest punt de vista. Si presenten un domini d'estabilitat prou bo ja és suficient, perquè, si per exemple el domini d'estabilitat fós un disc de radi $r > 0$ centrat a l'orígen, només intentaríem aplicar el mètode amb passos h tals que es trobessin dins aquest domini, per assegurar-nos una bona estabilitat. El principal inconvenient que podria ocórrer és que podríem necessitar una quantitat de càlculs exageradament gran si emprem una h dins aquest domini, i aquesta és la situació dels problemes, o equacions diferencials, coneguts com "stiff". Per combatre aquest tipus de problemes, s'utilitzen més els anomenats mètodes implícits, que es coneix que són més estables que els explícits ja que presenten un domini d'estabilitat major. En aquest document no tractarem gaire aquest aspecte, però quan es parla d'estabilitat d'un mètode és interessant comentar-ho.

2.4 El mètode d'Euler

Per endinsar-nos al tema de mètodes numèrics per equacions diferencials començarem explicant un dels mètodes d'un pas més bàsics, el conegut mètode d'Euler, que ens servirà com un cas simple dels mètodes generals explicats posteriorment.

Si agafem la notació emprada anteriorment per representar un mètode amb solució $x(t)$ i passos temporals t_i per $i = 0, 1, 2, ...$, i apliquem sèries de Taylor, es pot desenvolupar cada expressió $x_i := x(t_i)$ com:

$$x(t_{i-1} + h) = x(t_{i-1}) + x'(t_{i-1})h + O(h^2),$$

Emprant la definició de la derivada en la equació anterior i negligint els termes d'ordre superior, tenim:

$$x(t_{i-1} + h) = x(t_{i-1}) + hf(t_{i-1}, x_{i-1}),$$

I per tant, podem definir així un mètode iteratiu

$$\tilde{x}_i = \tilde{x}_{i-1} + hf(t_{i-1}, \tilde{x}_{i-1}) \text{ per } i = 1, 2, 3, ...,$$

i $\tilde{x}_0 = x_0 = x(a)$, que és la dada inicial del problema. Aquest és l'anomenat mètode d'Euler, un mètode amb ordre de convergència 1, és a dir, és un mètode de primer ordre, i a més és no estable.

És interessant també definir la versió implícita del mètode:

$$\tilde{x}_i = \tilde{x}_{i-1} + hf(t_i, \tilde{x}_i) \text{ per } i = 1, 2, 3, ...$$

Gràficament estaríem seguint la trajectòria de la solució a partir de la recta tangent a la mateixa en cada pas temporal t_i. En cada pas, avançaríem una distància h sobre la tangent. És clar que amb passos més petits podem obtenir una major precisió, ja que ens mouríem menys sobre la tangent, però el nombre d'operacions incrementaria.

No justificarem el seu ordre de convergència ni l'estabilitat del mètode perquè després ho estudiarem del mètode de Taylor, i només caldrà mirar què passa en el cas senzill $p = 1$.

3 Mètode de Taylor

3.1 Idea general del mètode

Donades les dades del problema del valor inicial,

$$\begin{cases} x'(t) = f(t, x(t)), \\ x(a) = x_0, \end{cases} \tag{3.1}$$

la idea del mètode de Taylor és utilitzar sèries de Taylor (negligint termes d'ordre superior) per aproximar la solució al següent pas temporal $t_0 + h$ de la forma descrita a continuació:

$$\tilde{x}_1 = x_0 + x'(t_0)h + \frac{x''(t_0)}{2!}h^2 + ... + \frac{x^{(p)}(t_0)}{p!}h^p,$$

on p és un nombre natural positiu que marca l'ordre de la sèrie. Posteriorment, s'aniria aplicant l'algorisme amb cada pas $\tilde{x}_i$ per poder obtenir el pas següent $\tilde{x}_{i+1}$, per $i = 0, 1, 2...$ A partir d'ara denotarem aquest mètode com TS(p). Notem que el mètode TS(1) equivaldria al mètode d'Euler, com hem vist al capítol anterior.

Aquesta expressió seria una bona aproximació si poguéssim calcular els valors de les derivades avaluades als punts corresponents de forma eficient. Però el problema que s'ens presenta és que la funció f té dues variables, i totes dues depenen de la variable temporal t, i a l'hora de derivar acaben sortint moltes derivades parcials per la regla de la cadena. Com per exemple, per calcular la segona derivada respecte t:

$$\begin{aligned} x'(t_0) &= f(t_0, x(t_0)), \\ x''(t_0) &= f_t(t_0, x(t_0)) + f_x(t_0, x(t_0))x'(t_0), \end{aligned}$$

i així successivament amb les derivades posteriors. Per tant, el principal inconvenient és l'excés de cost en els càlculs per derivar i avaluar aquestes derivades, solució? Emprar la Diferenciació Automàtica. Aquest procediment ens permet una avaluació més ràpida de les derivades d'una funció fins a un ordre p arbritàriament alt. De totes maneres, necessitarem calcular un ordre òptim fins el qual calcular aquesta sèrie de Taylor, i un pas h adequat per estalviar-nos tots els càlculs possibles, sempre intentant obtenir un error per sota d'una tolerància establerta ε.

Un inconvenient del mètode de Taylor amb Diferenciació Automàtica és que la funció f ha de pertànyer a una classe especial de funcions. No aprofunditzarem gaire en aquest aspecte, perquè afortunadament aquesta classe conté moltes de les funcions que apareixen en la gran part d'aplicacions. Per més informació, consultar [JZ05].

3.2 Diferenciació automàtica

La diferenciació automàtica és un procés recursiu que calcula el valor de les derivades de certes funcions en un punt donat. Les funcions que considerarem seràn aquelles obtingudes per suma, producte, quocient i/o composició de funcions elementals. Les funcions elementals inclouen polinomis, funcions trigonomètriques, potències reals, exponencials, i logarítmiques.

En primer lloc, donarem la següent definició.

Definició: Sigui $u : I \subset \mathbb{R} \longrightarrow \mathbb{R}, t \mapsto u(t)$, una funció que té derivades contínues fins a un cert ordre suficientment gran, definim la seva enèssima derivada normalitzada com:

$$u^{[n]}(t) = \frac{1}{n!} u^{(n)}(t), \tag{3.2}$$

on $u^{(n)}(t)$ denota la derivada enèsima respecte la variable temporal t.

A continuació demostrarem una sèrie de propietats de les derivades normalitzades que ens serviràn posteriorment per a l'aplicació del mètode de Taylor.

Proposició 3.1. *Suposem que $u(t) = F(v(t), w(t))$, on v i w són de classe C^n i coneixem els valors de $v^{[j]}(t)$ i $w^{[j]}(t)$, amb $j = 0, ..., n$ per un temps donat. Llavors, per $\alpha \in \mathbb{R}\backslash\{0\}$, tenim:*

(1) Si $u(t) = v(t) \pm w(t)$, llavors

$$u^{[n]}(t) = v^{[n]}(t) \pm w^{[n]}(t),$$

(2) Si $u(t) = v(t)w(t)$, llavors

$$u^{[n]}(t) = \sum_{j=0}^{n} v^{[n-j]}(t) w^{[j]}(t),$$

(3) Si $u(t) = \frac{v(t)}{w(t)}$, llavors

$$u^{[n]}(t) = \frac{1}{w^{[0]}(t)} \left[v^{[n]}(t) - \sum_{j=1}^{n} w^{[j]}(t) u^{[n-j]}(t) \right],$$

(4) Si $u(t) = v(t)^{\alpha}$, llavors

$$u^{[n]}(t) = \frac{1}{n v^{[0]}(t)} \sum_{j=0}^{n-1} (n\alpha - j(\alpha + 1)) v^{[n-j]}(t) u^{[j]}(t),$$

(5) Si $u(t) = e^{v(t)}$, llavors

$$u^{[n]}(t) = \frac{1}{n} \sum_{j=0}^{n-1} (n - j) u^{[j]}(t) v^{[n-j]}(t),$$

(6) Si $u(t) = \ln v(t)$, llavors

$$u^{[n]}(t) = \frac{1}{v^{[0]}(t)} \left[v^{[n]}(t) - \frac{1}{n} \sum_{j=1}^{n-1} (n - j) v^{[n]}(t) u^{[n-j]}(t) \right],$$

(7) Si $u(t) = \cos w(t)$ i $v(t) = \sin w(t)$, llavors

$$u^{[n]}(t) = -\frac{1}{n} \sum_{j=1}^{n} j v^{[n-j]}(t) w^{[j]}(t),$$

$$v^{[n]}(t) = \frac{1}{n} \sum_{j=1}^{n} j u^{[n-j]}(t) w^{[j]}(t),$$

Demostració. (1) És evident de les propietats de la derivada:

$$u(t) = v(t) \pm w(t), \text{ derivant n cops} \longrightarrow u^{(n)}(t) = v^{(n)}(t) \pm w^{(n)}(t),$$

i dividint a cada banda per $\frac{1}{n!}$ tenim el resultat.

(2) Emprant la fórmula de la derivada enèssima de Leibniz i la definició de nombre combinatori:

$$u^{[n]}(t) = \frac{1}{n!}u^{(n)}(t) = \frac{1}{n!}\sum_{j=0}^{n}\binom{n}{j}v^{(n-j)}(t)w^{(j)}(t) =$$

$$= \frac{1}{n!}\sum_{j=0}^{n}\frac{n!}{j!(n-j)!}v^{(n-j)}(t)w^{(j)}(t) = \sum_{j=0}^{n}v^{[n-j]}(t)w^{[j]}(t),$$

(3) De la definició de u tenim que $v(t) = u(t)w(t)$, i ara aplicant (2) obtenim:

$$v^{[n]}(t) = \sum_{j=0}^{n}u^{[n-j]}(t)w^{[j]}(t) = w^{[0]}(t)u^{[n]}(t) + \sum_{j=1}^{n}u^{[n-j]}(t)w^{[j]}(t),$$

Finalment, aïllant $u^{[n]}(t)$ de la expressió anterior:

$$u^{[n]}(t) = \frac{1}{w^{[0]}(t)}\left[v^{[n]}(t) - \sum_{j=1}^{n}u^{[n-j]}(t)w^{[j]}(t)\right],$$

(4) Tenim $u(t) = v(t)^{\alpha}$, si apliquem logarítmes i derivem:

$$\ln u(t) = \alpha \ln v(t) \longrightarrow \frac{u'(t)}{u(t)} = \alpha\frac{v'(t)}{v(t)},$$

Acabem obtenint: $u'(t)v(t) = \alpha v'(t)u(t)$. Si apliquem (2) a cada banda:

$$\sum_{j=0}^{n}(u')^{[n-j]}v^{[j]} = \sum_{j=0}^{n}\alpha u^{[n-j]}(v')^{[j]},$$

Utilitzant la següent igualtat

$$(u')^{[n-j]} = \frac{1}{(n-j)!}(u')^{(n-j)} = \frac{1}{(n-j)!}u^{(n-j+1)} =$$

$$\frac{(n-j+1)!}{(n-j)!}u^{[n-j+1]} = (n-j+1)u^{[n-j+1]}, \tag{3.3}$$

ens queda la expressió anterior com

$$\sum_{j=0}^{n}(n-j+1)u^{[n-j+1]}v^{[j]} = \sum_{j=0}^{n}\alpha(j+1)u^{[n-j]}v^{[j+1]},$$

Si s'ordenen totes dues igualtats per termes en el mateix bàndol de la igualtat ens queda l'expressió (es pot fer una comprovació coeficient a coeficient):

$$\sum_{j=0}^{n+1}(j-(n-j+1)\alpha)u^{[j]}v^{[n-j+1]} = 0,$$

Ara, extraient el darrer terme per aïllar $u^{[n+1]}$, obtenim:

$$u^{[n+1]}(t) = \frac{1}{(n+1)v^{[0]}(t)} \sum_{j=0}^{n} ((n+1)\alpha - j(\alpha+1))v^{[n-j+1]}(t)u^{[j]}(t),$$

que és el que volíem veure però pel cas $n+1$.

(5) Si apliquem logarítmes i derivem com abans, obtenim $u'(t) = u(t)v'(t)$. Després, emprant també (2) i la definició de derivada normalitzada:

$$(n+1)u^{[n+1]}(t) = \sum_{j=0}^{n} (n-j+1)u^{[j]}(t)v^{[n-j+1]}(t),$$

que és el que volíem demostrar per $n+1$.

(6) Aplicant logarítmes i derivant l'expressió inicial: $u'(t)v(t) = v'(t)$. Ara, tornant a utilitzar (2) i la definició de derivada normalitzada:

$$(n+1)v^{[n+1]}(t) = \sum_{j=0}^{n} (n-j+1)v^{[j]}(t)u^{[n-j+1]}(t),$$

Aïllant $u^{[n+1]}(t)$, acabem tenint:

$$u^{[n+1]}(t) = \frac{1}{v^{[0]}(t)} \left[v^{[n+1]}(t) - \frac{1}{n+1} \sum_{j=1}^{n} (n-j+1)v^{[j]}(t)u^{[n-j+1]}(t) \right],$$

que era el que es volia veure en el cas $n+1$.

(7) Procedint de mateixa manera que els casos anteriors, s'arriba a $u'(t) = -v(t)w'(t)$ i $v'(t) = u(t)w'(t)$. Emprant un altre cop la propietat (2) s'obté l'expressió corresponent.

□

Pel que es pot veure de la forma que tenen aquestes expressions, no és difícil arribar a la conclusió que el nombre d'operacions aritmètiques necessàries per avaluar les derivades normalitzades d'ordre n és com a molt $O(n^2)$ (n del sumatori, multiplicat per n si és el cas que necessita les $n-1$ derivades normalitzades anteriors), fet que fa que l'algorisme emprant Diferenciació Automàtica sigui prou eficient.

3.3 Control de pas i ordre òptim

Cada expansió en potències de la solució $x(t)$ a temps $t = t_i$ emprant sèries de Taylor tindrà un radi de convergència diferent per cada temps, i un mètode d'integració ha de tenir-ho en compte. Per poder obtenir una determinada precisió en els resultats, i realitzar el menor nombre d'operacions aritmètiques possibles, per cada temps, anirem calculant els valors adequats per al pas h, el més gran possible, i ordre p, el més petit possible. Per tant, donada una precisió ε, volem trobar els valors corresponents per h_i i p_i en cada pas temporal.

Imaginem que ens trobem davant un problema de valor inicial com (2.1), que ja s'han fet i passos de l'algorisme, i que denotem a la solució exacta que satisfà $x(t_i) = \tilde{x}_i$ com x_i, on $\tilde{x}_i$ és el valor proporcionat pel mètode en aquest pas. Utilitzant sèries de Taylor:

$$x_i(t+h) = \sum_{j=0}^{\infty} x_i^{[j]}(t) h^j,$$

Ara, utilitzant el mètode TS(p_i) amb pas h_i, tindrem:

$$\tilde{x}_{i+1} = \sum_{j=0}^{p_i} x_i^{[j]}(t_i) h_i^j,$$

i volem imposar per $t_{i+1} = t_i + h_i$:

$$||x_i(t_{i+1}) - \tilde{x}_{i+1}|| \leq \varepsilon, \tag{3.4}$$

Per determinar aquests valors de pas i ordre òptims, ens recolzarem en la següent proposició.

Proposició 3.2. *Suposem que la funció $z \mapsto x(t_i + z)$ és analítica en un disc de radi ρ_i, i que existeix una constant positiva A_i tal que*

$$|x_i^{[j]}| \leq \frac{A_i}{\rho_i^j}, \quad \forall j \in \mathbb{N},$$

Llavors, els valors h_i i p_i que donen la precisió preestablerta ε i minimitzen el nombre d'operacions aritmètiques tendeixen a:

$$h_i = \frac{\rho_i}{e^2}, \qquad i \qquad p_i = -\frac{1}{2} \ln\left(\frac{\varepsilon}{A_i}\right) - 1, \tag{3.5}$$

Demostració. Es pot consultar a l'article [JZ05]. □

Una vegada la precisió és triada, l'ordre òptim p_i és l'únic que garanteix que es compleixi aquesta precisió, ja que aquest depèn de ε i h_i no.

Per altra banda, calcular els valors de les constants A_i i els radis de convergència de les sèries seria molt costós, ja que és informació que no es pot obtenir fàcilment, així que s'implementa d'una altra manera. Una vegada fixat un error ε, es calcula l'ordre òptim com:

$$p_i = \left\lceil -\frac{1}{2} \ln \varepsilon + 1 \right\rceil,$$

on $\lceil * \rceil$ és la notació per la funció "sostre"o ceiling (arrodoniment a l'alça). Estaríem aplicant la proposició (3.2) amb $A_i = 1$ i p_i dos unitats major, fet que justificarem després.

Per determinar un pas h_i una vegada ja fixat l'ordre p, distingirem dos casos:

- Si $||x_i||_\infty \leq 1$, definirem

$$\rho_i^{(j)} = \left(\frac{1}{||x_i^{[j]}||_\infty}\right)^{1/j}, \quad 1 \leq j \leq p,$$

- I si $||x_i||_\infty > 1$

$$\rho_i^{(j)} = \left(\frac{||x_i||_\infty}{||x_i^{[j]}||_\infty} \right)^{1/j}, \quad 1 \leq j \leq p,$$

Aquestes definicions són deduïdes de la proposició (3.2) i del criteri de l'arrel. Sigui quin sigui el cas en el que ens trobem, estimarem el radi de convergència com el mínim dels dos darrers termes:

$$\rho_i = \min \left\{ \rho_i^{(p-1)}, \rho_i^{(p)} \right\}, \tag{3.6}$$

i llavors emprant la proposició (3.2), ens queda que el pas seria

$$h_i = \frac{\rho_i}{e^2},$$

La raó per la qual es fan aquestes tries dels paràmetres és per a que els darrers termes de la sèrie de Taylor de la solució siguin gairebé menyspreables, al ser més petits que la tolerància fixada des d'un inici. Aquesta idea, ve justificada amb la següent proposició.

Proposició 3.3. *Suposant que tenim valors de pas, ordre, i radi de convergència com els definits prèviament, tornem a distingit dos casos:*

(1) Si $||x_i||_\infty \leq 1$*, es compleixen*

$$||x_i^{[p_i-1]} h_i^{p_i-1}||_\infty \leq \varepsilon, \quad ||x_i^{[p_i]} h_i^{p_i}||_\infty \leq \frac{\varepsilon}{e^2},$$

(2) Si $||x_i||_\infty > 1$*, es compleixen*

$$\frac{||x_i^{[p_i-1]} h_i^{p_i-1}||_\infty}{||x_i||_\infty} \leq \varepsilon, \quad \frac{||x_i^{[p_i]} h_i^{p_i}||_\infty}{||x_i||_\infty} \leq \frac{\varepsilon}{e^2},$$

Demostració. En primer lloc, de la definició de $p_i = \left\lceil -\frac{1}{2} \ln \varepsilon + 1 \right\rceil$ deduïm que:

$$p_i \geq -\frac{1}{2} \ln \varepsilon + 1 \longrightarrow -2(p_i - 1) \leq \ln \varepsilon \longrightarrow e^{-2(p_i-1)} \leq \varepsilon \tag{3.7}$$

Per tant, emprant la definició del pas h_i, obtenim la igualtat

$$||x_i^{[p_i-1]} h_i^{p_i-1}||_\infty = \frac{||x_i^{[p_i-1]} \rho_i^{p_i-1}||_\infty}{e^{2(p_i-1)}} \leq \varepsilon,$$

Pel que hem vist anteriorment a (3.7), si es compleix que $||x_i^{[p_i-1]} \rho_i^{p_i-1}||_\infty \leq 1$, ja ho tindríem demostrat, així que vegem-ho.

$$||x_i^{[p_i-1]} \rho_i^{p_i-1}||_\infty \leq ||x_i^{[p_i-1]} \left(\frac{1}{||x_i^{[p_i-1]}||} \right)^{(p_i-1)/(p_i-1)} ||_\infty = 1,$$

per la definició de la funció mínim. Per demostrar la segona desigualtat, farem un raonament semblant,

$$||x_i^{[p_i]} h_i^{p_i}||_\infty = \frac{||x_i^{[p_i]} \rho_i^{p_i}||_\infty}{e^{2p_i}} = \frac{||x_i^{[p_i]} \rho_i^{p_i}||_\infty}{e^{2(p_i-1)+2}} \leq \frac{\varepsilon}{e^2},$$

Per demostrar (2) es fa anàlogament, però emprant la definició corresponent del radi de convergència per al cas $||x_i||_\infty > 1$.

□

En conclusió, utilitzant aquestes definicions dels paràmetres, podem menysprear la resta de termes de la sèrie de Taylor d'ordre major a p_i i acabarem aconseguint la precisió que volíem des d'un principi, és a dir, es complirà (3.4). Notem que la proposició ens diu que tant l'error absolut com l'error relatiu dels darrers termes són menors o iguals a la tolerància ε.

3.4 Convergència i estabilitat del mètode

En aquest apartat ens dedicarem a justificar quin ordre de convergència té el $TS(p)$ i a trobar les seves característiques d'estabilitat.

Justificació de l'ordre de convergència:

Ens recolzarem principalment en una demostració del llibre [Cob84]. Demostrarem que l'error global de discretització és $E(h) = O(h^p)$:

$$E(h) = \max_{1 \leq i \leq N} |\tilde{x}_i - x(t_i)|,$$

Suposem que la funció f és suficientment bona i les seves derivades parcials respecte la variable t són acotades. Ara, emprant la definició del mètode i sèries de Taylor, tenim

$$\tilde{x}_i = \tilde{x}_{i-1} + \tilde{x}'(t_{i-1})h + \frac{\tilde{x}''(t_{i-1})}{2!}h^2 + ... + \frac{\tilde{x}^{(p)}(t_{i-1})}{p!}h^p,$$

$$x(t_i) = x(t_{i-1}+h) = x(t_{i-1}) + x'(t_{i-1})h + \frac{x''(t_{i-1})}{2!}h^2 + ... + \frac{x^{(p)}(t_{i-1})}{p!}h^p + R_i(h),$$

on en aquest cas l'error de truncament és $R_i(h) = \frac{1}{(p+1)!}h^{p+1}x^{(p+1)}(\xi)$, per $\xi \in (t_{i-1}, t_i)$. A causa de que hem suposat que la funció f és suficientment bona, podem suposar que existeix una constant $|x^{(p+1)}(t)| \leq C$ per tot $t \in (t_{i-1}, t_i)$, i per tant:

$$|R_i(h)| \leq \frac{1}{(p+1)!}Ch^{p+1},$$

i $R_i(h) = O(h^{p+1})$. Per tant, si definim l'error acumulat $e_i = \tilde{x}_i - x(t_i)$, i utilitzem $x'(t) = f(t, x(t))$, obtenim

$$e_i = \tilde{x}_{i-1} - x(t_{i-1}) + f(t_{i-1}, \tilde{x}_{i-1})h - f(t_{i-1}, x(t_{i-1}))h + \frac{df}{dt}(t_{i-1}, \tilde{x}_{i-1})\frac{h^2}{2!}$$

$$-\frac{df}{dt}(t_{i-1}, x(t_{i-1}))\frac{h^2}{2!} + ... + R_i(h) =$$

$$= e_{i-1} + h[f(t_{i-1}, \tilde{x}_{i-1}) - f(t_{i-1}, x(t_{i-1}))]\cdot$$
$$\cdot\frac{h^2}{2!}\left[\frac{df}{dt}(t_{i-1}, \tilde{x}_{i-1}) - \frac{df}{dt}(t_{i-1}, x(t_{i-1}))\right] + ... + R_i(h),$$

Si apliquem el teorema del valor mig en diverses variables en cada diferència de les funcions f i derivades, obtindrem una sèrie de cotes de les diferències per algunes $0 < c_j < 1$:

$$|f^{(j)}(t_{i-1}, \tilde{x}_{i-1}) - f^{(j)}(t_{i-1}, x(t_{i-1}))| =$$
$$\left|\frac{df^{(j)}}{dx}(t_{i-1}, c_j\tilde{x}_{i-1} + (1-c_j)x(t_{i-1}))(\tilde{x}_{i-1} - x(t_{i-1}))\right| \leq M_j e_{i-1} \quad \text{per } j = 0, 1, ..., p,$$

al tenir les derivades parcials acotades per $M_j > 0$, i on $f^{(j)}$ indica la j-èssima derivada de la funció respecte la variable t. Per tant, ajuntant aquesta expressió amb l'anterior, obtenim que

$$|e_i| \leq \left(1 + M_1 h + M_2\frac{h^2}{2!} + ... + M_p\frac{h^p}{p!}\right)|e_{i-1}| + R_i(h),$$

Si posem $c = \left(1 + M_1 h + M_2\frac{h^2}{2!} + ... + M_p\frac{h^p}{p!}\right)$ i anem repetint aquest procés successivament,

$$|e_i| \leq c|e_{i-1}| + R_i(h) \leq ... \leq$$
$$\leq c^{i-1}|e_1| + c^{i-2}R_i(h) + ... + c\ R_i(h) + R_i(h),$$

on $|e_1| = |\tilde{x}_1 - x(t_1)| = R_i(h) = O(h^{p+1})$, error local de truncament. Si enlloc de l'error e_i, empram l'error final e_N, ens quedarien $N = (b-a)/h$ termes $O(h^{p+1})$. Per tant, la suma seria $O(h^p)$ i es satisfaria $E(h) = O(h^p)$.

Per demostrar aquesta darrera part, faltaria veure que c^N és acotat quan $h \to 0$, que emprant la definició del nombre e:

$$\lim_{h\to 0}\left(1 + M_1 h + M_2\frac{h^2}{2!} + ... + M_p\frac{h^p}{p!}\right)^{(b-a)/h} = e^{(b-a)M_1},$$

Justificació de l'estabilitat

Ara demostrarem que el mètode de Taylor no és estable, per tant, hem de demostrar que el seu domini d'estabilitat no és tot $\mathbb{C}$. Apliquem el mètode a l'equació diferencial (2.3):

$$\tilde{x}_{n+1} = \tilde{x}_n + h\lambda\tilde{x}_n + \frac{h^2}{2!}\lambda^2\tilde{x}_n + ... + \frac{h^p}{p!}\lambda^p\tilde{x}_n = \left(1 + \lambda h + \frac{(h\lambda)^2}{2!} + ... + \frac{(h\lambda)^p}{p!}\right)\tilde{x}_n,$$

Per tant, si repetim el procés n cops ens queda,

$$\tilde{x}_n = \left(1 + \lambda h + \frac{(h\lambda)^2}{2!} + ... + \frac{(h\lambda)^p}{p!}\right)^n,$$

La successió $\{\tilde{x}_n\}_n$ complirà que $\lim_{n\to\infty} \tilde{x}_n = 0$ si i només si el que està elevat a n dins el parèntesi és menor que 1. Per tant, podem concloure que:

$$\mathcal{D}_{Taylor} = \left\{ z \in \mathbb{C} : \left| 1 + z + \frac{z^2}{2!} + ... + \frac{z^p}{p!} \right| < 1 \right\},$$

Aquest conjunt és diferent de $\mathbb{C}$ ja que si $z = 1$ no es compleix la desigualtat, per tant el mètode no és estable. I tampoc és A-estable, ja que sigui quin sigui el nombre p, sempre es pot trobar un nombre real molt negatiu z que no satisfaci la desigualtat.

4 Mètodes Runge-Kutta

4.1 Definició del mètode

La idea de generalitzar el mètode d'Euler avaluant un cert nombre de vegades la derivada de la solució en un mateix pas, és generalment atribuïda a Runge (1895). Posteriorment, Kutta (1901) i Heun (1900) també feren les seves aportacions al tema.

Els anomenats mètodes Runge-Kutta o RK, milloren una dels aspectes que se'ls recrimina als mètodes de Taylor, que és el càlcul de les diferencials del camp. Aquests mètodes intenten aproximar les diferencials per combinacions d'avaluacions del camp en allò que podríem dir punts estratègics.

Donades les dades del problema de valor inicial (2.1), amb pas $h = (b-a)/N$ i temps $t_{n+1} = t_n + h, t_0 = a, t_N = b$ per $n = 0, ..., N-1$, N nombre de particions de l'interval $[a, b]$, definirem el mètode general Runge-Kutta de s etapes o $RK(s)$ com:

$$\tilde{x}_{n+1} = \tilde{x}_n + h \sum_{i=1}^{s} b_i k_i, \tag{4.1}$$

i les variables k_i són calculades emprant el camp:

$$k_i = f\left(t_n + c_i h, \tilde{x}_n + h \sum_{j=1}^{s} a_{i,j} k_j \right), \quad \text{per } i = 1, ..., s,$$

on els valors $a_{i,j}$, b_i i c_i depenen del mètode de Runge-Kutta que s'estigui emprant i es dedueixen imposant condicions de convergència, com per exemple, que el mètode tingui l'error local de truncament amb ordre més elevat possible. Notem que s és el nombre d'avaluacions del camp que caldrà fer a cada pas. Per altra banda, es pot apreciar que els mètodes Runge-Kutta són mètodes d'un pas, i que per $s = 1$, s'obté el mètode d'Euler o equivalentment $RK(1)$.

Es solen representar els mètodes RK(s) visualment amb el que s'anomena taula de Butcher, on hi consten tots els coeficients del mètode ordenats.

$$\begin{array}{c|cccc} c_1 & a_{1,1} & a_{1,2} & \dots & a_{1,s} \\ c_2 & a_{2,1} & a_{2,2} & \dots & a_{2,s} \\ \vdots & \vdots & \vdots & & \\ c_s & a_{s,1} & a_{s,2} & \dots & a_{s,s} \\ \hline & b_1 & b_2 & \dots & b_s \end{array}$$

Figura 1: **Representació dels mètodes RK(s) amb una taula de Butcher**

A la definició general proporcionada s'inclouen tant la versió explícita com implícita del mètode. Si $a_{i,j} = 0$ per tot $j \geq i$ (la matriu $A = (a_{i,j})$ és estrictament triangular inferior) i $c_1 = 0$ el mètode serà explícit, altrament serà implícit. En aquest document ens centrarem sobretot en els mètodes explícits.

A continuació farem un exemple de com es dedueixen aquestes constants del mètode per $s = 2$ i versió explícita, per tant, $c_1 = 0$ i $a_{1,1}, a_{1,2}, a_{2,2} = 0$. Siguin $\tilde{x}_{n+1}$ la solució proporcionada pel mètode i $x(t_{n+1})$ la solució exacta, suposem localment que $\tilde{x}_n = x(t_n)$, llavors:

$$\tilde{x}_{n+1} = \tilde{x}_n + h(b_1k_1 + b_2k_2) =$$

$$\tilde{x}_n + h(b_1 f(t_n + c_1h, \tilde{x}_n + h(a_{1,1}k_1 + a_{1,2}k_2)) + b_2 f(t_n + c_2h, \tilde{x}_n + h(a_{2,1}k_1 + a_{2,2}k_2))) =$$

$$\tilde{x}_n + h(b_1 f(t_n, \tilde{x}_n) + b_2 f(t_n + c_2h, \tilde{x}_n + ha_{2,1} f(t_n, \tilde{x}_n))),$$

i per altra banda emprant sèries de Taylor,

$$x(t_{n+1}) = x(t_n) + hx'(t_n) + \frac{h^2}{2!}x''(t_n) + \frac{h^3}{3!}x'''(t_n) + O(h^4),$$

on aquesta darrera expressió també es pot posar en termes de la funció f de la següent manera:

$$x(t_{n+1}) = x(t_n) + hf(t_n, x(t_n)) + \frac{h^2}{2!}(f_t(t_n, x(t_n)) + f_x(t_n, x(t_n))f(t_n, x(t_n))) +$$

$$+\frac{h^3}{3!}(f_{tt}(t_n, x(t_n)) + 2f_{tx}(t_n, x(t_n))f(t_n, x(t_n)) + f_t(t_n, x(t_n))f_x(t_n, x(t_n)) +$$

$$+f_{xx}(t_n, x(t_n))f^2(t_n, x(t_n)) + f_x^2(t_n, x(t_n))f(t_n, x(t_n))) + O(h^4),$$

f_t i f_x indicant les respectives derivades parcials de la funció. La primera expressió també es pot reescriure emprant el desenvolupament de Taylor en diverses variables

$$\tilde{x}_{n+1} = \tilde{x}_n + hb_1 f(t_n, \tilde{x}_n) + hb_2[f(t_n, \tilde{x}_n) + c_2hf_t(t_n, \tilde{x}_n) + ha_{2,1}f_x(t_n, \tilde{x}_n)f(t_n, \tilde{x}_n) +$$

$$+\frac{(c_2h)^2}{2!}f_{tt}(t_n, \tilde{x}_n) + c_2a_{2,1}h^2 f_{tx}(t_n, \tilde{x}_n)f(t_n, \tilde{x}_n) + \frac{(a_{2,1}h)^2}{2!}f_{xx}(t_n, \tilde{x}_n)f^2(t_n, \tilde{x}_n) + O(h^3)],$$

Tot seguit feim la resta per calcular l'error local de truncament, anul·lant els termes que són iguals amb la suposició local $\tilde{x}_n = x(t_n)$:

$$R_{n+1}(h) = h(b_1 + b_2 - 1)f(t_n, \tilde{x}_n) +$$

$$+h^2\left[\left(b_2c_2 - \frac{1}{2}\right)f_t(t_n, \tilde{x}_n) + \left(b_2a_{2,1} - \frac{1}{2}\right)f_x(t_n, \tilde{x}_n)f(t_n, \tilde{x}_n)\right]$$

$$+h^3[\left(\frac{c_2^2b_2}{2} - \frac{1}{6}\right)f_{tt}(t_n, \tilde{x}_n) + \left(\frac{a_{2,1}^2b_2}{2} - \frac{1}{6}\right)f_{xx}(t_n, \tilde{x}_n)f^2(t_n, \tilde{x}_n) +$$

$$+\left(b_2c_2a_{2,1} - \frac{1}{3}\right)f_{tx}(t_n, \tilde{x}_n)f(t_n, \tilde{x}_n)$$

$$-\frac{1}{6}\left(f_t(t_n, x(t_n))f_x(t_n, x(t_n)) + f_x^2(t_n, x(t_n))f(t_n, x(t_n))\right)] + O(h^4),$$

Com veiem a l'expressió anterior, podem imposar les condicions:

$$\begin{cases} b_1 + b_2 = 1, \\ b_2c_2 = \dfrac{1}{2}, \\ b_2a_{2,1} = \dfrac{1}{2}, \end{cases}$$

i d'aquesta manera tindrem que $R_{n+1}(h) = O(h^3)$. Això també ens diu que l'error global de discretització del mètode hauria de ser $E(h) = O(h^2)$, i que per tant, el mètode té ordre de convergència 2. No podem anul·lar el següent coeficient perquè el darrer terme no s'anul·la mai independentment de les variables que es triïn. Aquest sistema no lineal té infinitat de solucions, i totes elles donen lloc a un Runge-Kutta diferent amb 2 etapes. Potser el RK(2) més conegut és el que s'obté prenent $b_1 = b_2 = 1/2$ i $c_2 = a_{2,1} = 1$:

$$x_{n+1} = x_n + h\left(\frac{1}{2}f(t_n, x_n) + \frac{1}{2}f_t(t_n + h, x_n + hf(t_n, x_n))\right),$$

per $n = 0, ..., N-1$ i condició inicial $x_0 = x(a)$.

Podríem definir $b_2 = \theta \neq 0$, i d'aquesta manera obtindríem tota la família de mètodes Runge-Kutta de dues etapes, que es representarien amb la taula de Butcher:

$$\begin{array}{c|cc} 0 & 0 & \\ a & a & 0 \\ \hline & 1-\theta & \theta \end{array} \qquad a = 1/(2\theta)$$

Figura 2: **Representació dels mètodes RK(2)**

Un mètode Runge-Kutta es caracteritza llavors per una tria particular dels paràmetres $a_{i,j}$, b_i i c_i que proporcionin un algorisme eficient, en el sentit d'aconseguir unes bones característiques de convergència, o minimitzar el nombre s d'avaluacions del camp en cada pas per aconseguir una ordre de precisió fixat p.

Si imposem condicions de convergència per mètodes amb nombre d'etapes superior, el nombre d'equacions del sistema i de variables també augmentaria. Les condicions algebraiques sobre els coeficients del mètode s'anirien complicant cada vegada més. El patró darrera aquestes condicions és conegut, i per explicar-ho hauríem d'entrar en teoria de combinatòria i grafs, ja que s'associa a cada mètode Runge-Kutta un arbre amb certes propietats, cosa que no farem, però es pot trobar més informació a les referències emprades en aquest capítol [But16], [GH10], [Ise08] i [Lam72]. Algo que si remarcarem és que un mètode RK d'ordre de convergència p necessita almenys $s = p$ avaluacions del camp. De fet, els RK d'ordre 4 són els mètodes d'ordre més gran pels quals el nombre d'avaluacions del camp s coincideix amb l'ordre. Per construir un RK d'ordre 5, s ja ha de valer 6, i per tant, aquests dos valors no han de coincidir sempre obligatòriament.

4.2 Mètodes Runge-Kutta-Fehlberg i control de pas

Com amb tot mètode d'integració, l'elecció d'un bon pas d'integració és un dels aspectes clau per tenir una bona eficiència. Mantenir un pas constant al llarg de tot el procés ja hem argumentat prèviament que no era convenient, així que en aquesta secció ens dedicarem a explicar un tipus de mètodes Runge-Kutta que fan un control del pas de forma automàtica una vegada fixat una fita per l'error ε. Aquests mètodes s'anomenen Runge-Kutta-Fehlberg, desenvolupats per el matemàtic alemany Fehlberg basant-se en els mètodes RK.

La idea principal d'aquests mètodes és la següent. Suposem localment que ens trobem en el punt $x(t_n) = \tilde{x}_n$ i que donat un pas h_n volem calcular la solució $x(t_n + h_n)$. En

primer lloc, calcularem una solució $\bar{x}_{n+1}$ amb un mètode Runge-Kutta d'un cert ordre i després calcularem una altra $\hat{x}_{n+1}$ amb un RK d'un ordre superior. Si $|\bar{x}_{n+1} - \hat{x}_{n+1}| \leq \varepsilon$, voldrà dir que les dues solucions coincideixen prou, en concret, la solució proporcionada pel mètode de major ordre serà encara més bona aproximació i agafarem aquesta per continuar amb la iteració. Si en canvi no es complís la desigualtat, el que faríem seria reduir el pas h_n i tornar a calcular aquestes dues solucions, repetint el procés quantes vegades faci falta fins complir la condició. Finalment, farem una predicció del nou pas h_{n+1} i passarem a la següent iteració, i procedim així successivament fins haver acabat l'integració.

Fins aquí podríem dir que no té res en especial, més enllà de la deducció del pas h_{n+1} que l'explicarem tot seguit, però la tria en concret de dos mètodes RK d'ordre consecutiu ens permet estalviar avaluacions del camp, ja que el mètode d'ordre superior aprofitarà els càlculs que hagi realitzat el d'ordre inferior. En conclusió, tindrem un cost computacional equivalent a si només haguéssim realitzat el mètode d'ordre més alt.

A continuació, vegem com obtenim una fórmula del pas òptim, basant-nos en el llibre [JM04]. Suposem que tenim dos mètodes RK d'ordres p i $p+1$ que proporcionen dues solucions $\bar{x}_{n+1}$, $\hat{x}_{n+1}$ i un pas h_n. El nostre objectiu és trobar un h_{n+1} tal que:

$$|\bar{x}_{n+2} - \hat{x}_{n+2}| \leq \varepsilon, \tag{4.2}$$

on aquestes són les solucions que proporcionen els mètodes al següent pas $t_n + h_{n+1}$, i ε és una tolerància fixada. Per l'ordre de convergència dels mètodes, sabem que, negligint termes d'ordre superior a $p+1$:

$$\bar{x}_{n+1} - \hat{x}_{n+1} = (\bar{x}_{n+1} - x(t_{n+1})) - (\hat{x}_{n+1} - x(t_{n+1})) =$$

$$= R^p_{n+1}(h_n) - R^{p+1}_{n+1}(h_n) = N(t_n)h_n^{p+1} + O(h^{p+2}),$$

on R^p i R^{p+1} indiquen els errors locals de discretització, que són respectivament $O(h^{p+1})$ i $O(h^{p+2})$, i $N(t_n)$ seria el coeficient d'ordre $p+1$ de $R^p_{n+1}(t_n)$. D'aquí deduim:

$$|N(t_n)| = \frac{|\bar{x}_{n+1} - \hat{x}_{n+1}|}{h_n^{p+1}},$$

I per tant, amb un raonament similar, imposant (4.2),

$$|\bar{x}_{n+2} - \hat{x}_{n+2}| = |N(t_{n+1})|h_{n+1}^{p+1} \leq \varepsilon,$$

Ara, emprant el desenvolupament de Taylor amb $N(t_{n+1})$ al voltant de t_n, tenim:

$$N(t_{n+1}) = N(t_n + h_{n+1}) = N(t_n) + O(h_{n+1}),$$

la qual cosa implica que

$$|N(t_n)|h_{n+1}^{p+1} + O(h_{n+1}^{p+2}) \leq \varepsilon,$$

i negligint termes d'ordre $O(h_{n+1}^{p+2})$ ens queda

$$h_{n+1}^{p+1} \leq \frac{\varepsilon}{|N(t_n)|} = \frac{h_n^{p+1}\varepsilon}{|\bar{x}_{n+1} - \hat{x}_{n+1}|},$$

Agafant el màxim h_{n+1} que doni la precisió demanda, obtenim la *fórmula de predicció del nou pas*:

$$|h_{n+1}| = |h_n| \sqrt[p+1]{\frac{\varepsilon}{|\bar{x}_{n+1} - \hat{x}_{n+1}|}},$$

on s'acostuma a posar un factor de seguretat, per exemple 0.9, per assegurar que si el pas ha de créixer, ho faci només si es compleix aquest marge. En conclusió, donat uns valors inicials t_n, x_n, h_n, l'algorisme ens proporcionarà uns valors $t_{n+1} = t_n + h_n$, $\tilde{x}_{n+1} = \hat{x}_{n+1}$, i una predicció del nou pas h_{n+1} amb la fórmula deduida.

A la representació del mètode RKF amb les taules de Butcher s'afegeixen les constants c_i i $a_{i,j}$ compartides, i les constants b_i que són independents de cada mètode. Més endavant, a la secció d'aplicacions es veurà un exemple de com s'implementa.

i \ j	c_i	a_{ij} 1	2	3	4	5	6	7	8
1	0								
2	$\frac{1}{18}$	$\frac{1}{18}$							
3	$\frac{1}{6}$	$\frac{-1}{12}$	$\frac{1}{4}$						
4	$\frac{2}{9}$	$\frac{-2}{81}$	$\frac{4}{27}$	$\frac{8}{81}$					
5	$\frac{2}{3}$	$\frac{40}{33}$	$\frac{-4}{11}$	$\frac{-56}{11}$	$\frac{54}{11}$				
6	1	$\frac{-369}{73}$	$\frac{72}{73}$	$\frac{5380}{219}$	$\frac{-12285}{584}$	$\frac{2695}{1752}$			
7	$\frac{8}{9}$	$\frac{-8716}{891}$	$\frac{656}{297}$	$\frac{39520}{891}$	$\frac{-416}{11}$	$\frac{52}{27}$	0		
8	1	$\frac{3015}{256}$	$\frac{-9}{4}$	$\frac{-4219}{78}$	$\frac{5985}{128}$	$\frac{-539}{384}$	0	$\frac{693}{3328}$	
$\bar{b}_j$		$\frac{3}{80}$	0	$\frac{4}{25}$	$\frac{243}{1120}$	$\frac{77}{160}$	$\frac{73}{700}$		
b_j		$\frac{57}{640}$	0	$\frac{-16}{65}$	$\frac{1377}{2240}$	$\frac{121}{320}$	0	$\frac{891}{8320}$	$\frac{2}{35}$

Figura 3: **Representació d'un mètode Runge-Kutta-Fehlberg d'ordres 5-6**

És conegut que un dels principals inconvenients d'aquests mètodes a partir de $p > 4$ es troba a la hora de resoldre problemes del tipus

$$x'(t) = f(t) + \varepsilon_0 g(x(t)),$$

amb ε_0 molt petit o directament 0, ja que l'estimació de l'error serà quasi idènticament zero. Això passa perquè, per la tria dels paràmetres, el mètode d'ordre p té exactament el mateix error de truncament que el mètode d'ordre $p+1$ per aquest problema, de fet, l'error local de truncament és el mateix terme de $O(h^{p+2})$ en els dos mètodes. A causa d'això, cada pas del mètode serà acceptat i probablement serà augmentat en mida.

Una solució a aquest problema podria ser simplement tractar aquest tipus de problemes amb una fórmula per a l'estimació de l'error diferent, així només caldria detectar quan un problema és d'aquesta forma i tractar-lo en conseqüència, però això ja no ho veurem (consultar [Ver78]).

4.3 Estabilitat dels mètodes RK

La noció d'estabilitat segueix sent la mateixa pels mètodes Runge-Kutta, enfront a petits errors o canvis en les dades d'entrada, la solució proporcionada pel mètode manté aquests

errors controlats per sota una tolerància. Matemàticament, s'estudia el comportament del mètode aplicat a l'equació diferencial descrita en (2.3), i és el que anem a fer ara amb els mètodes RK:

$$\tilde{x}_{n+1} = \tilde{x}_n + h\sum_{i=1}^{s} b_i k_i,$$

on calculem els valors k_i:

$$k_i = f\left(t_n + c_i, \tilde{x}_n + h\sum_{j=1}^{s} a_{i,j}k_j\right) = \lambda(\tilde{x}_n + h\sum_{j=1}^{s} a_{i,j}k_j),$$

Si expressem el mètode RK emprat en forma de matrius i vectors, tal que $\mathbf{c} = (c_1, ..., c_s)^t$, $\mathbf{b} = (b_1, ..., b_s)^t$ i $A = (a_{i_j})$ de dimensió $s \times s$, i si definim $\mathbf{k} = (k_1, ..., k_s)^t$ i $\mathbf{e} = (1, ..., 1)^t$, llavors:

$$\mathbf{k} = \lambda(\mathbf{e}\tilde{x}_n + hA\mathbf{k}),$$

i aïllant el vector $\mathbf{k}$, ens queda:

$$\mathbf{k} = \lambda(Id - h\lambda A)^{-1}\mathbf{e}\tilde{x}_n,$$

on Id és la matriu identitat $s \times s$. Per tant, substituint a la definició del mètode tenim

$$\tilde{x}_{n+1} = \tilde{x}_n + h\mathbf{b}^t\mathbf{k} = \tilde{x}_n + h\lambda\mathbf{b}^t(Id - h\lambda A)^{-1}\mathbf{e}\tilde{x}_n = (1 + h\lambda\mathbf{b}^t(Id - h\lambda A)^{-1}\mathbf{e})\tilde{x}_n,$$

Si suposem que el mètode és explícit, la matriu A serà triangular inferior, el que implica que és nilpotent, és a dir, $A^s = 0$, i per tant, ens permet expressar la inversa de la fórmula com:

$$(Id - h\lambda A)^{-1} = Id + h\lambda A + (h\lambda)^2A^2 + ... + (h\lambda)^{s-1}A^{s-1},$$

Substituint ara a l'expressió que teníem del mètode:

$$\begin{aligned}\tilde{x}_{n+1} &= (1 + h\lambda\mathbf{b}^t(Id + h\lambda + ... + (h\lambda)^{s-1}A^{s-1})\mathbf{e})\tilde{x}_n = \\ &= (1 + h\lambda\mathbf{b}^t\mathbf{e} + (h\lambda)^2\mathbf{b}^tA\mathbf{e} + ... + (h\lambda)^s\mathbf{b}^tA^s\mathbf{e})\tilde{x}_n, \end{aligned} \tag{4.3}$$

Ara, del fet que el mètode té ordre p, s'obtenen les condicions d'ordre per a les variables $\mathbf{b}^tA^{i-2}\mathbf{c} = \frac{1}{i!}$ i $\mathbf{b}^tA^{i-1}\mathbf{e} = \frac{1}{i!}$ per $i = 1, ..., p$ (fet que no hem vist però es pot trobar a [But16]). Si prenem p que sigui almenys com s, és a dir, que el nombre d'etapes del mètode sigui menor o igual a l'ordre de convergència del mètode, que com hem comentat anteriorment, això només passa amb ordres fins a $p = 4$ inclòs, l'expressió queda amb $z = h\lambda$ com un polinomi de grau s:

$$\tilde{x}_{n+1} = \left(1 + z + \frac{z^2}{2!} + ... + \frac{z^s}{s!}\right)\tilde{x}_n = \left(1 + z + \frac{z^2}{2!} + ... + \frac{z^s}{s!}\right)^{n+1}, \tag{4.4}$$

i per tant per mètodes explícits amb $s \leq p$ tenim,

$$\mathcal{D}_{RK(s)} = \left\{z \in \mathbb{C} : \left|1 + z + \frac{z^2}{2!} + ... + \frac{z^s}{s!}\right| < 1\right\},$$

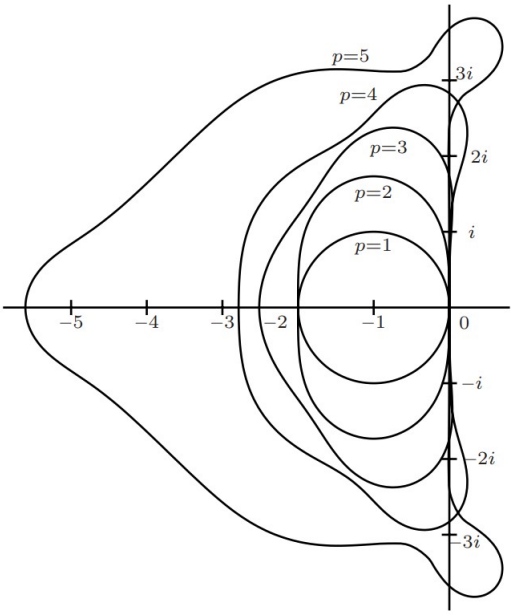


Figura 4: **Dominis d'estabilitat de mètodes RK amb diferents ordres p**

és a dir, presenta el mateix domini d'estabilitat que el mètode TS(s), que com ja vam argumentar a la secció anterior, era diferent de $\mathbb{C}$ i $\mathbb{C}^-$, i per tant el mètode no era ni estable ni A-estable.

A partir de $p \geq 5$ i $s \geq 6$, l'expressió (4.4) segueix sent de grau s però els coeficients dels termes superiors a p depenen segons el mètode RK. Si $s = 6$, i $p = 5$, el dibuix de la figura 4 seria un exemple amb domini:

$$\mathcal{D} = \left\{ z \in \mathbb{C} : \left| 1 + z + \frac{z^2}{2!} + \frac{z^3}{3!} + \frac{z^4}{4!} + \frac{z^5}{5!} + \frac{z^6}{1280} \right| < 1 \right\},$$

De fet, es pot demostrar el següent fet més general:

Lema 4.1. *No existeix cap mètode Runge-Kutta explícit que sigui A-estable.*

Demostració. Donat un mètode RK explícit, de (4.3) es pot deduir que si expressem el seu domini d'estabilitat com

$$\mathcal{D} = \{ z \in \mathbb{C} : |r(z)| < 1 \},$$

la funció $r(z)$ haurà de ser sempre un polinomi no constant que compleix $r(0) = 1$. També sabem que cap polinomi, excepte del constant $r(z) = c \in (-1, 1)$, estarà uniformement acotat pel valor unitat en $\mathbb{C}^-$, i això implica que el mètode no pot ser A-estable. □

Per altra banda, l'estabilitat dels mètodes RK implícits es força caòtica, només en donarem un exemple sense entrar en més detalls. Estudiarem l'estabilitat del mètode, aplicant-lo a l'equació diferencial (2.3), representat per la taula de la figura 5.

Aprofitant la fórmula que hem deduït abans, amb aquests coeficients tenim:

$$\tilde{x}_n = 1 + z \begin{pmatrix} 1/2 & 1/2 \end{pmatrix} \left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - z \begin{pmatrix} 1/4 & 1/4 - \sqrt{3}/6 \\ 1/4 + \sqrt{3}/6 & 1/4 \end{pmatrix} \right)^{-1} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$\begin{array}{c|cc} \frac{1}{2}-\frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4}-\frac{\sqrt{3}}{6} \\ \frac{1}{2}+\frac{\sqrt{3}}{6} & \frac{1}{4}+\frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Figura 5: **Mètode RK implícit de dues etapes**

del que segueix, emprant la fórmula del càlcul de matrius inverses de dimensió 2:

$$A^{-1} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

$$\tilde{x}_n = 1 + z\begin{pmatrix} 1/2 & 1/2 \end{pmatrix}\begin{pmatrix} \frac{-3z+12}{z^2-6z+12} & \frac{-2\sqrt{3}z+3z}{z^2-6z+12} \\ \frac{2\sqrt{3}z+3z}{z^2-6z+12} & \frac{-3z+12}{z^2-6z+12} \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = 1 + \frac{12z}{z^2-6z+12} = \frac{z^2+6z+12}{z^2-6z+12},$$

I en conclusió, el domini d'estabilitat d'aquest mètode serà

$$\mathcal{D} = \left\{ z \in \mathbb{C} : \left| \frac{z^2+6z+12}{z^2-6z+12} \right| < 1 \right\},$$

Agafant $z = 0$ no es compleix la desigualtat, el que implica que no és un mètode estable, podria ser A-estable? Demostrem la següent proposició d'anàlisi complexa per veure-ho:

Proposició 4.2. *Suposem que r es una funció racional arbitrària que no és constant. Llavors, $|r(z)| < 1,\ \forall z \in \mathbb{C}^-$ si i només si tots els pols de r tenen part real positiva i $|r(it)| \leq 1,\ \forall t \in \mathbb{R}$.*

Demostració. Si $|r(z)| < 1$, per tot $z \in \mathbb{C}^-$, llavors és clar que també es compleix per continuïtat $|r(z)| \leq 1$, per tot $z \in \mathbb{C}^-$. En particular, r no pot tenir pols en el pla tancat dels complexos amb part real negativa, i es compleix $|r(it)| \leq 1,\ \forall t \in \mathbb{R}$.

Recordem que z_0 és un pol de la funció r si

$$\lim_{z \to z_0} |r(z)| = +\infty,$$

Per provar l'altra implicació, si sabem que tots els pols tenen part real positiva, llavors la funció racional r és analítica en el tancat $\mathbb{C}^-$ i també holomorfa. Per tant, com r no és constant, aplicant el principi del mòdul màxim, sabem que assolirà el seu màxim en la frontera del conjunt. En altres paraules $|r(it)| \leq 1$ per tot $t \in \mathbb{R}$, implica que $|r(z)| < 1$ per tot $z \in \mathbb{C}^-$, i acabem la demostració. □

Per tant, com els pols de la nostra funció són $3 \pm i\sqrt{3}$, que són els punts on s'anul·la el denominador, i tenen part real positiva, i a més es compleix $|r(it)| = 1,\ \forall t \in \mathbb{R}$, podem afirmar que el mètode és assimptòticament estable.

5 Mètodes d'Extrapolació

En aquesta primera secció del capítol, ens oblidarem per un moment de que estem tractant amb equacions diferencials, i ens centrarem en la resolució de la integral

$$\int_b^a f(x)\ dx,$$

per posteriorment aplicar les estratègies que vegem en la resolució del problema de valor inicial comentat al capítol 2. Dit això, una bona forma de calcular integrals definides és emprant mètodes de discretització, els quals aproximen la integral per sumes finites que corresponen a una partició de l'interval d'integració $[a, b]$. Hi ha gran quantitat de mètodes d'aquest tipus, com ara els que empren la regla de Simpson, o la regla del trapezi, que generalment es coneixen com fórmules de Newton-Cotes tancades (al ser $[a, b]$ un interval tancat). Repassem un poc la teoria general d'aquests mètodes coneguts com mètodes d'interpolació.

5.1 Interpolació

Considerem una partició uniforme de l'interval $[a, b]$ donada per $x_i = a + ih$, per $i = 0, 1, ..., N$, amb un pas $h = (b - a)/N$, on $N > 0$ és un nombre natural. Les fórmules de Newton-Cotes són obtingudes reemplaçant el camp f per un polinomi interpolador adequat $P(x)$, i $\int_b^a P(x)\ dx$ s'agafa com a aproximació de la integral que volíem calcular. Aquest polinomi, que pot ser de grau N o menys, ha de complir:

$$P_N(x_i) = f(x_i) =: f_i, \text{ per } i = 0, ..., N, \tag{5.1}$$

Per la fórmula d'interpolació de Lagrange, podem expressar aquest polinomi com:

$$P_N(x) = \sum_{i=0}^{N} f_i L_i(x), \text{ on } L_i(x) = \prod_{k=0,\ k\neq i}^{N} \frac{x - x_k}{x_i - x_k},$$

Recordem que el polinomi de Lagrange L_i té la propietat especial de que val 0 si $x \neq x_i$, i 1 si $x = x_i$, a més de ser l'únic polinomi que ho verifica, per tant, el polinomi interpolador que hem definit compleix les condicions (5.1). Si feim un canvi de variable $x = a + ht$, llavors:

$$L_i(x) = \phi_i(t) := \prod_{k=0,\ k\neq i}^{N} \frac{t - k}{i - k},$$

Sigui N un nombre natural, si definim $\alpha_i := \int_0^N \phi_i(t)\ dt$, que anomenarem *pesos*, les fórmules de Newton-Cotes que aproximen la integral (5.1) es poden escriure com:

$$\int_b^a P_N(x)\ dx = h \sum_{i=0}^{N} f_i \alpha_i, \quad f_i = f(a + ih), \quad h = (b - a)/N,$$

Per exemple, si $N = 1$, la fórmula resultant es coneix com la regla del trapezi:

$$\int_a^b P_N(x)\ dx = \frac{h}{2}(f(a) + f(b)),$$

Les fórmules de Newton-Cotes normalment no s'apliquen a l'interval d'integració sencer, s'apliquen a subintervals en els que s'ha dividit aquest interval. Llavors, la integral s'aproxima per la suma de totes les aproximacions en els subintervals. Aquest procediment s'anomena *regla de composició*. Seguint l'exemple anterior amb $N = 1$, si feim la divisió de l'interval $[a, b]$ en els subintervals $[x_i, x_{i+1}]$ on $x_i = a + ih, i = 0, 1, ..., N$ i $h = (b - a)/N$, obtenim la fórmula per a l'aproximació de tot l'interval com:

$$T(h) := \sum_{i=0}^{N-1} \frac{h}{2}[f(x_i) + f(x_{i+1})] = h\left[\frac{f(a)}{2} + f(a + h) + ... + f(b - h) + \frac{f(b)}{2}\right],$$

que li direm regla del trapezi composta.

Ara, una pregunta molt natural seria demanar-nos quant de precises són aquestes aproximacions, i la resposta ve donada en forma del següent teorema:

Teorema 5.1. *Fórmula d'Euler-MacLaurin. Sigui T(h) la fórmula dels trapezis composta i* $f \in C^{2m+2}([a, b])$*, llavors és certa la igualtat:*

$$T(h) = \int_a^b f(x)\ dx + \tau_1 h^2 + \tau_2 h^4 + ... + \tau_m h^{2m} + \alpha_{m+1} h^{2m+2},$$

$$on\ \ \tau_k := \frac{B_{2k}}{(2k)!}(f^{(2k-1)}(b) - f^{(2k-1)}(a)), \quad k = 1, ..., m$$

$$\alpha_{m+1}(h) = \frac{B_{2m+2}}{(2m + 2)!}(b - a)f^{(2m+2)}(\xi(h)),\ a < \xi(h) < b,$$

i B_k *són els nombres de Bernoulli:* $B_2 = \frac{1}{6}$, $B_4 = -\frac{1}{30}$, $B_6 = \frac{1}{42}$, $B_8 = -\frac{1}{30}$, ...

Demostració. En la demostració s'empren els anomenats polinomis de Bernoulli i algunes de les seves propietats. Es pot trobar a la referència [SB02]. □

La fórmula d'Euler-MacLaurin expandeix la fórmula dels trapezis composta en termes del pas h, i aquesta expressió serà la clau per obrir la porta als anomenats mètodes d'extrapolació. Expansions d'aquesta forma, s'anomenen *expansions assimptòtiques en h* si els coeficients τ_k no depenen de h, com és aquest cas, i juntament amb α_{m+1} compleixen la mateixa igualtat del teorema.

Com $f^{(2m+2)}$ és contínua per hipòtesi en l'interval tancat $[a, b]$, existeix una cota L tal que $|f^{(2m+2)(x)}| \leq L$ per tot $x \in [a, b]$, llavors existeix una constant M_{m+1} tal que

$$|\alpha_{m+1}(h)| \leq M_{m+1},$$

per tot $h = (b - a)/n, n = 1, 2, ...$ Això demostra que l'error de l'expansió assimptòtica proporcionada pel teorema esdevé relativament petita a mesura que h decreix. Per tant, l'expressió es comporta com un polinomi en h^2 que val per $h = 0$ el valor exacte de la integral.

Enlloc de solucionar el problema d'interpolació tot d'un cop calculant el polinomi de Lagrange, es podria solucionar també el problema per certs subconjunts dels punts x_i, i després intentar ajuntar-ho tot per trobar la solució del problema general. Aquesta va ser la idea del matemàtic Eric Harold Neville, que la va expressar de la següent manera.

Suposem que $P_{i,j}$ denota el polinomi de grau $j-i$ tal que passa pels punts $f(x_k)$ per $k = i, ..., j$, per tant, és el polinomi interpolador d'aquest subconjunt del conjunt global definit a (5.1). Llavors el polinomi $P_{i,j}$ satisfarà la relació de recurrència:

$$P_{i,i}(x) = f(x_i), \quad 0 \leq i \leq N,$$

$$P_{i,j}(x) = \frac{(x - x_i)P_{i+1,j}(x) - (x - x_j)P_{i,j-1}(x)}{x_j - x_i}, \quad 0 \leq i < j \leq n,$$

Provar la primera condició és trivial, ja que el polinomi interpolador d'un sol punt és ell mateix. Per demostrar la segona fórmula, denotem $R(x)$ com la banda dreta de la igualtat, i veurem que R compleix les mateixes característiques que el polinomi interpolador $P_{i,j}$, i al ser aquest únic, seràn el mateix.

Per les definicions de $P_{i,j-1}$ i $P_{i+1,j}$, tenim:

$$R(x_i) = P_{i,j-1}(x_i) = f_i, \text{ i } R(x_j) = P_{i+1,j}(x_j) = f_j,$$

$$\text{A més,} \quad R(x_k) = \frac{(x_k - x_i)f_k - (x_k - x_j)f_k}{x_j - x_i} = f_k,$$

per $k = i+1, i+2, ..., j-2, j-1$. I demostrem així la igualtat. D'aquesta manera obtenim un algorisme recursiu per obtenir el polinomi $P_{0,N}$ que serà el polinomi interpolador sobre l'interval global, que era el volíem en un principi. Aquest algorisme es coneix com *Algorisme de Neville*.

Tot el que hem vist en aquesta secció sobre interpolació ens servirà per poder definir i entendre els mètodes d'integració per extrapolació.

5.2 Integració per Extrapolació

En moltes situacions en anàlisi numèric, tenim l'objectiu d'avaluar una expressió matemàtica, però només som capaços de càlcular una aproximació de la mateixa emprant la tècnica de discretització. En general, podem suposar que aquesta expressió, que li direm $A(h)$, té una expansió assimptòtica de la forma:

$$A(h) = A_0 + A_1 h + A_2 h^2 + ... + A_N h^N + R_N(h), \tag{5.2}$$

on $R_N(h) = O(h^{N+1})$ a mesura que $h \to 0$, i els coeficients A_i no depenen de h. Com el pas de discretització és h, el natural seria agafar A_0 com el valor exacte del que volem, ja que al fer $h \to 0$, seria com fer que el pas fós molt petit i ens estaríem apropant a la solució cada vegada més.

Agafant un pas adient h_0, calcularíem $A(h_0)$ i tindríem l'expressió

$$A(h_0) = A_0 + A_1 h_0 + A_2 h_0^2 + ... + A_N h_0^N + O(h_0^{N+1}),$$

Si a la vegada també calculem $A(\frac{1}{2}h_0)$, obtindríem:

$$2A(\frac{1}{2}h_0) - A(h_0) = A_0 - \frac{1}{2}A_2h_0^2 + ... = A_0 + O(h_0^2),$$

que és una aproximació millor que considerant només $A(h_0)$ i $A(\frac{1}{2}h_0)$ per separat, idea que es comenta a [HW76].

Per tant, d'aquesta idea i de la fórmula d'Euler-MacLaurin, en el cas de la regla dels trapezis composta, es podria suggerir el següent mètode per trobar el valor de la integral.

Denotem per $H := (b-a) > 0$, el pas general de la integració, i definim una seqüència de m enters positius

$$n_0 < n_1 < n_2 < ... < n_m,$$

i definim els seus passos corresponents $h_0 > h_1 > ... > h_m$ per $h_i = H/n_i$. Ara, per cada pas h_i es calculen els valors amb la regla dels trapezis composta:

$$T_{i,0} := T(h_i), \quad i = 0, 1, ..., m,$$

Si construïm el polinomi interpolador en h^2 d'aquests punts, tindrà una forma

$$\tilde{T}_{m,m}(h) := a_0 + a_1h^2 + ... + a_mh^{2m}, \quad \text{tal que compleix}$$
$$\tilde{T}_{m,m}(h_i) = T(h_i), \quad i = 0, 1, ..., m,$$

i si prenem el valor extrapolat $\tilde{T}_{m,m}(0)$ com l'aproximació de la integral, haurem construït uns mètodes que es coneixen com mètodes d'extrapolació. Depenent de la successió de valors n_i que s'agafin, es poden obtenir diferents mètodes, si agafem $n_i = 2^i$, per tant $h_i = H/2^i$, obtindrem l'anomenat *mètode de Romberg.*

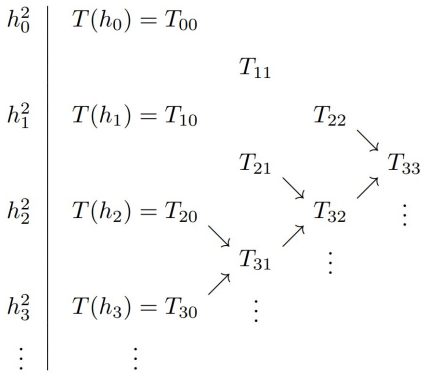


Figura 6: **Esquema recursiu que representa l'algorisme d'Aitken-Neville**

Per calcular aquest polinomi interpolador, podem emprar l'algorisme recursiu de Neville, de tal manera que podem definir $\tilde{T}_{i,k}(h)$ com el polinomi interpolador de grau com a màxim k, $1 \leq k \leq i \leq m$, en h^2 tal que $\tilde{T}_{i,k}(h_j) = T(h_j)$ per $j = i-k, i-k+1, ..., i$. D'aquesta manera, podem anar definit els valors $T_{i,k} := \tilde{T}_{i,k}(0)$ i emprar la fórmula recursiva de Neville amb $x_i = h_i^2$:

$$T_{i,k} = T_{i,k-1} + \frac{T_{i,k-1} - T_{i-1,k-1}}{\left[\frac{h_{i-k}}{h_i}\right]^2 - 1}, \quad 1 \leq k \leq i \leq m,$$

que rep el nom d'*Algorisme d'Aitken-Neville*, en honor als matemàtics que el van idear en els anys 1932-1934, basats en idees prèvies del matemàtic Jordan en 1928.

De forma més general encara, enlloc d'agafar la regla dels trapezis composta i la seva expansió per a l'error, es pot agafar un altre mètode de discretització $T(h)$ amb una expansió assimptòtica de l'error

$$T(h) = \tau_0 + \tau_1 h^{\gamma_1} + \tau_2 h^{\gamma_2} + ... + \tau_m h^{\gamma_m} + \alpha_{m+1}(h) h^{\gamma_{m+1}}, \tag{5.3}$$

on els exponents γ_i compleixen $0 < \gamma_1 < \gamma_2 < ... < \gamma_{m+1}$ i no fa falta que siguin tots enters. Els coeficients τ_i són independents de h, la funció $\alpha_{m+1}(h)$ està acotada per $h \to 0$, i $\tau_0 = \lim_{h\to 0} T(h)$ és la solució exacta de la integral. Llavors, els polinomis interpoladors del mètode d'extrapolació també tindrien coeficients i exponents diferents, però la idea seria la mateixa que s'ha explicat amb la regla dels trapezis composta:

$$\tilde{T}_{i,k}(h) = b_0 + b_1 h^{\gamma_1} + ... + b_k h^{\gamma_k},$$

$$\text{i compleixen } \tilde{T}_{i,k}(h_j) = T(h_j),\ j = i-k, i-k+1, ..., i,$$

Per mètodes d'extrapolació basats en expansions assimptòtiques de l'error com l'anterior, es podria demostrar la proposició següent:

Proposició 5.2. *Si l'expressió assimptòtica de l'error de T es comporta com (5.3), i els exponents són de la forma $\gamma_k = \gamma k$, per un k fixat, la seqüència $T_{i,k}$ aproxima la integral $\int_a^b f(x)dx$ com un mètode d'ordre $\gamma(k+1)$.*

Demostració. Suposem que $\gamma_k = \gamma k$. Simplificarem la demostració utilitzant una altra notació:

$$z := h^\gamma, \quad z_j := h_j^\gamma, \quad j = 0, 1, ..., m,$$

Per tant, el polinomi interpolador passa a ser:

$$\tilde{T}_{k,k}(h) = b_0 + b_1 z + b_2 z^2 + ... + b_k z^k,$$

Ara, aplicant la fórmula d'interpolació de Lagrange i l'expressió assimptòtica de l'error del mètode T, per $z = 0$ es compleix

$$T_{k,k} = \tilde{T}_{k,k}(0) = \sum_{j=0}^{k} c_j T(h_j) = \sum_{j=0}^{k} c_j [\tau_0 + \tau_1 z_j + ... + \tau_k z_j^k + \alpha_{k+1}(h_j) z_j^{k+1}], \tag{5.4}$$

$$\text{amb} \qquad c_j := \prod_{\sigma=0, \sigma\neq j}^{k} \frac{z_\sigma}{z_\sigma - z_j},$$

$$\text{i on} \qquad \alpha_{k+1}(h_j) z_j^{k+1} = \tau_{k+1} h_j^{\gamma_{k+1}} + ... + \tau_m h_j^{\gamma_m} + \alpha_{m+1}(h_j) h_j^{\gamma_{m+1}},$$

Per la unicitat del polinomi interpolador i la fórmula de Lagrange, per qualsevol polinomi p de grau $\leq k$, es compleix

$$p(0) = \sum_{j=0}^{k} c_j p(z_j),$$

Triant p adequadament com z^β per $\beta = 0, 1, ..., k$, i $z^{k+1} - (z - z_0)(z - z_1) \cdots (z - z_k)$ per $\beta = k + 1$, llavors tenim:

$$\sum_{j=0}^{k} c_j z_j^\beta = \begin{cases} 1 & si\ \beta = 0, \\ 0 & si\ \beta = 1, 2, ..., k, \\ (-1)^k z_0 z_1 \cdots z_k & si\ \beta = k + 1, \end{cases}$$

Per veure una justificació completa, consultar pàgina 168 de [SB02]. Per tant, emprant aquesta expressió en (5.4), obtenim:

$$T_{k,k} = \tau_0 + \sum_{j=0}^{k} c_j \alpha_{k+1}(h_j) z_j^{k+1}, \tag{5.5}$$

Per $k < m$, acabem obtenint

$$T_{k,k} - \tau_0 = (-1)^k z_0 z_1 \cdots z_k \alpha_{k+1}(h_j),$$

on $\alpha_{k+1}(h_j) = \tau_{k+1} + O(h_0^\gamma)$. Si enlloc de fer-ho amb $T_{k,k}$, ho feim amb $T_{i,k}$, només hem de canviar els z_j anteriors per $z_{i-k}, z_{i-k+1}, ..., z_i$ i llavors per $k < m$ i $i \geq k$ tenim

$$T_{i,k} - \tau_0 = (-1)^k h_{i-k}^\gamma h_{i-k+1}^\gamma \cdots h_i^\gamma (\tau_{k+1} + O(h_{i-k}^\gamma)),$$

En conseqüència, per un k fixat, i $i \to \infty$,

$$|T_{i,k} - \tau_0| = O(h_{i-k}^{(k+1)\gamma}),$$

i hem acabat la demostració. □

En altres paraules, en cada columna, els valors $T_{i,k}$ convergeixen a la solució com un mètode d'ordre $\gamma(k + 1)$. Si haguéssim agafat un mètode amb diferent expressió assimptòtica, amb $\gamma = 1$, haguéssim obtingut un mètode amb un ordre de convergència la meitat de bo que amb la obtinguda amb un de potències parelles. En conclusió, els mètodes d'extrapolació amb expansions assimptòtiques en potències de h^2 són particularment efectius.

Si el mètode $T(h)$ fós un mètode d'ordre de convergència p, llavors l'expansió assimptòtica de l'error seria de la forma

$$T(h) = \tau_0 + \tau_p h^p + \tau_{p+1} h^{p+1} + ... + \tau_{p+m-1} h^{p+m-1} + \alpha_{p+m}(h) h^{p+m}, \tag{5.6}$$

I respectivament tenim la proposició:

Proposició 5.3. *Si el mètode T es comporta com un mètode amb ordre de convergència p, per un k fixat, la seqüència $T_{i,k}$ aproxima la integral com un mètode d'ordre $p+k-1$.*

Demostració. Per l'expressió (5.4), obtindrem el polinomi interpolador següent

$$\tilde{T}_{i,k}(h) = a_0 + a_p h^p + ... + a_{p+k-1}h^{p+k-1} + O(h^{p+k}),$$

$$\text{que compleixen } \tilde{T}_{i,k}(h_j) = T(h_j),\ j = i-k, i-k+1, ..., i,$$

La solució exacta del problema plantejat és τ_0, i la solució proporcionada pel mètode en aquest pas és a_0, per tant, si veiem que l'error local compleix $|\tau_0 - a_0| \leq O(H^{p+k})$, llavors tindrem que l'error global de truncament serà $O(H^{p+k-1})$, i per tant es tracta d'un mètode d'ordre de convergència $p+k-1$.

Per veure-ho, començarem avaluant el polinomi interpolador a $h = h_j$

$$\tilde{T}_{i,k}(h_j) = a_0 + a_p h_j^p + ... + a_{p+k-1}h_j^{p+k-1} + O(h_j^{p+k}),$$

i com per definició sabem que $\tilde{T}_{i,k}(h_j) = T(h_j)$, utilitzant l'expressió assimptòtica de l'error del mètode T, ens queda l'expressió

$$\tau_0+\tau_p h_j^p+\tau_{p+1}h_j^{p+1}+...+\tau_{p+m-1}h_j^{p+m-1}+\alpha_{p+m}(h_j)h_j^{p+m} = a_0+a_p h_j^p+...+a_{p+k-1}h_j^{p+k-1}+O(h_j^{p+k}),$$

que reordenant i agrupant termes, i tenint en compte que $k \leq m$, obtenim

$$(\tau_0-a_0)+(\tau_p-a_p)h_j^p+(\tau_{p+1}-a_{p+1})h_j^{p+1}+...+(\tau_{p+k-1}-a_{p+k-1})h_j^{p+k-1} = O(h_j^{p+k}) = O(H^{p+k}),$$

ja que $h_j \leq H$ per qualsevol $j = i-k, ..., i$. Si ens fixem bé, aquesta expressió és un sistema lineal d'equacions amb $k+1$ incògnites $(\tau_0 - a_0), (\tau_p - a_p)H^p, ..., (\tau_{p+k-1} - a_{p+k-1})H^{p+k-1}$ i matriu del sistema

$$A = \begin{pmatrix} 1 & \frac{1}{n_{i-k}^p} & \cdots & \frac{1}{n_{i-k}^{p+k-1}} \\ \vdots & \vdots & & \vdots \\ 1 & \frac{1}{n_i^p} & \cdots & \frac{1}{n_i^{p+k-1}} \end{pmatrix}$$

Si multipliquem el vector d'incògnites per aquesta matriu, obtenim la mateixa expressió que tenim a dalt, ja que per definició dels passos: $h_j = H/n_j$. Ara, si la matriu A fós invertible i anomenem com y al vector de les $k+1$ incògnites, aplicant normes del suprem a l'expressió matricial del sistema obtindríem

$$|\tau_0 - a_0| \leq ||y||_\infty \leq ||A^{-1}||_\infty O(H^{p+k}) = O(H^{p+k}),$$

i per tant ja ho tindríem demostrat. No acabarem de justificar perquè la matriu A és invertible, però la prova es pot trobar a la pàgina 225 de [HNW00]. □

5.3 Mètode Gragg-Burlisch-Stoer i control de pas

A partir d'ara, tornem a treballar amb equacions diferencials, i el nostre objectiu torna a ser la resolució del problema del valor inicial:

$$\begin{cases} x'(t) = f(t, x(t)), \\ x(a) = x_0, \end{cases}$$

Donada la definició general dels mètodes d'extrapolació, vegem un exemple d'algorisme en concret que després implementarem amb un programa. A la pràctica, s'utilitza especialment la funció de Gragg $S(t;h)$, que donades les dades del problema del valor inicial f, $[a,b]$ i $N,n>0$, es defineix $H:=(b-a)/N$ com el pas general bàsic del problema (si volem un pas constant), i $h:=H/n$ el pas de la iteració. Llavors, el valor de la funció a $\bar{t}=t_0+H$ es calcula de la següent manera:

$$\begin{aligned}&\eta_0:=x_0,\\&\eta_1:=\eta_0+hf(t_0,\eta_0),\ t_1:=t_0+h,\end{aligned}$$

i per $j=1,2,...,n-1$:

$$\begin{aligned}&\eta_{j+1}:=\eta_{j-1}+2hf(t_j,\eta_j),\ t_{j+1}:=t_j+h,\\&S(\bar{t};h):=\frac{1}{2}[\eta_n+\eta_{n-1}+hf(t_n,\eta_n)],\end{aligned}$$

Aquesta funció té una particularitat especial, i és que la seva expansió assimptòtica també és amb potències parelles de h, com al cas de la regla dels trapezis composta, a més de tenir unes propietats bastant satisfactòries d'estabilitat, fet que va demostrar William B. Gragg en una prova molt llarga i complicada que no explicarem [HNW00]. Com s'ha comentat anteriorment, per un mètode d'extrapolació s'ha de definir una seqüència de m nombres enters tal que

$$F=\{n_0,n_1,n_2,...,n_m\},\quad 0<n_0<n_1<n_2<...<n_m,$$

i també els passos $h_i:=H/n_i$, que ens serviràn per calcular els valors de $S(\bar{t},h_i)$. Degut a les característiques de la funció S, F només pot tenir o nombres parells o nombres senars, perquè l'expansió de l'error de la funció presenta un terme oscil·lador $(-1)^{n_i}$ i podria causar cancel·lacions. Normalment, s'acostuma a agafar la successió de nombres parells

$$F=\{2,4,6,8,12,16,...\},\quad n_i:=2n_{i-2}\text{ per }i\geq 3,$$

encara que també es poden agafar altres. Aquests valors fixaràn la primera columna de valors $T_{i,k}$, i després mitjançant l'algorisme d'Aitken-Neville s'anirien calculant els altres. Com s'ha explicat a la secció anterior, el que es vol calcular és el polinomi interpolador

$$\begin{aligned}\tilde{T}_{i,k}(h):=a_0+a_1h^2+...+a_mh^{2m},&\quad\text{tal que compleix}\\\tilde{T}_{i,k}(h_j)=S(\bar{t};h_j),&\quad j=i-k,i-k+1,...,i,\end{aligned}$$

Cada columna convergirà a $x(\bar{t})$:

$$\lim_{i\to\infty}T_{i,k}=x(\bar{t})\ \text{ per }k=0,1,...$$

I en particular, convergiràn com un mètode d'ordre $(2k+2)$ tal i com ja hem vist.

Per fer la tria del següent pas $H>0$ en la iteració (en cas de voler fer control de pas), tal i com hem fet amb els anteriors mètodes per fer un control del pas de forma automàtica, utilitzarem una fórmula de predicció del nou pas molt semblant a la que hem deduït amb els mètodes Runge-Kutta:

$$|H_k|=|H|\sqrt[2k+1]{\frac{\varepsilon}{|T_{k-1,k-1}-T_{k,k}|}},$$

on ε és la precisió objectiu, i s'agafa aquesta arrel $(2k+1)$ perquè és l'ordre del mètode $T_{k-1,k-1}$ més una unitat, seguint el mateix raonament que vam fer al cas Runge-Kutta-Fehlberg i menyspreant els termes d'ordre superior a $2k+1$.

Per tant, també d'una forma similar, aniríem calculant aquests valors $T_{k-1,k-1}$ i $T_{k,k}$ i si la seva diferència és menor a ε, llavors acceptem el pas agafant com aproximació el valor proporcionat per $T_{k,k}$ i deduïm el següent pas H per a la pròxima iteració del mètode. Si la diferència resulta ser major, actualitzem el pas amb la fórmula igualment però tornem a repetir aquesta iteració amb el nou pas, que ara serà menor.

Un altre tria que hauríem de fer seria la del valor k, que indica quants valors inicials de la funció S hem de calcular per després aplicar la fórmula recursiva i obtenir l'aproximació de la integral. Per fer la predicció del nou valor de k, definirem una nova variable que representi la quantitat de "treball"que es realitza en el càlcul de $T_{k,k}$:

$$W_k = \frac{A_k}{H_k},$$

on H_k és el pas de la iteració, de la fórmula anterior, i A_k serà el nombre d'avaluacions de la funció f que es fan segons el nombre k, per tant, no és difícil veure que per definició de S:

$$A_1 = n_1 + 1, \quad A_k = A_{k-1} + n_k, \quad \text{per } k > 1$$

La idea seria triar un nombre k de tal manera que W_k sigui el mínim possible. La definició de W_k té aquesta forma perquè sempre s'intenten fer el mínim nombre d'avaluacions del camp possibles, ja que podrien portar molta feina i errors, encara que això es podria veure compensat si el pas és suficientment gran, ja que en total es faràn menys. En conclusió, seria raonable emprar la fórmula de predicció:

$$\bar{k} = \begin{cases} k-1, & si \;\; W_{k-1} < 0.9W_k \\ k+1, & si \;\; W_k < 0.9W_{k-1} \\ k, & altrament \end{cases} \tag{5.7}$$

on posem el factor 0.9 per assegurar-nos que les desigualtats es compleixen amb suficient marge. Intuïtivament, si el treball per k ha estat major que per $k-1$ llavors reduïm el seu valor, i si ha estat menor l'augmentem. Si els valors són prou semblants, no el canviem.

6 Aplicacions a mecànica celeste

En aquest apartat aplicarem els tres mètodes explicats a les seccions anteriors al problema dels N Cossos, un problema molt conegut de mecànica celeste, però abans els aplicarem al problema de la Força Central, que és una simplificació d'aquest, per introduir-nos. Sempre es farà també una breu explicació sobre el problema en qüestió i alguna demostració, si escau, d'alguna propietat que anem a emprar. Ens basarem principalment en el llibre [Pol66].

6.1 Problema de la Força Central

El problema de la Força Central és potser un dels problemes més simples de mecànica celeste. Descriu el moviment d'una partícula de massa m que es atreta per un centre fix amb una força $mf(r)$, que és proporcional a la massa del cos i depèn només de la distància r entre l'objecte en qüestió i el centre O. S'assumeix que la funció f, que anomenarem llei d'atracció, està ben definida per tot $r > 0$, nombre real.

Matemàticament, si denotem com $\mathbf{r}$ el vector director des del centre a la partícula, segons la segona llei de Newton, podem formular el problema com

$$m\ddot{\mathbf{r}} = \frac{-mf(r)}{r}\mathbf{r},$$

on $\ddot{\mathbf{r}}$ indica la segona derivada, i r és la norma del vector (euclidiana). Per tant, $\frac{r}{\mathbf{r}}$ és el vector unitari de la posició de la partícula. Si $\dot{\mathbf{r}}$ denota la velocitat, l'equació també es pot escriure

$$\begin{cases} \dot{\mathbf{r}} = \mathbf{v}, \\ \dot{\mathbf{v}} = \dfrac{-f(r)}{r}\mathbf{r}, \end{cases} \tag{6.1}$$

El problema s'ha convertit ara en estudiar el parell de vectors posició i velocitat que simultàniament satisfan (6.1) en un interval de temps. Observem que la massa és irrellevant per a resoldre el problema, ja que es cancel·la als dos costats de la igualtat. Tal vegada, el cas més important a estudiar seria quan la llei d'atracció fós la llei de gravitació de Newton, és a dir, $f(r) = \frac{\mu}{r^2}$, on μ, que s'anomena paràmetre gravitacional estàndard, és una constant positiva que depèn de les unitats triades i del centre d'atracció. Normalment, per conveni s'agafa $\mu = GM$, on G és la constant de gravitació i M és la massa del cos central del problema.

A partir de la segona equació, es pot deduir

$$\mathbf{r} \times \dot{\mathbf{v}} = \frac{-f(r)}{r}(\mathbf{r} \times \mathbf{r}) = 0,$$

ja que el producte vectorial d'un vector amb si mateix és zero. Llavors, la derivada de $\mathbf{r} \times \mathbf{v}$, que és $\mathbf{r} \times \dot{\mathbf{v}} + \mathbf{v} \times \mathbf{v}$, és idènticament 0, i per tant l'expressió queda:

$$\mathbf{r} \times \mathbf{v} = \mathbf{c},$$

i $\mathbf{c}$ és un vector constant. Ens referirem a aquesta variable com *moment angular*, i l'equació anterior és coneguda com la llei de conservació del moment angular. Si $c \neq 0$, voldrà dir

que el vector posició i el de velocitat formaràn un pla perpendicular al vector $\mathbf{c}$, que és constant, per tant la partícula es mourà sempre sobre un pla fixe que passa per l'origen i és perpendicular a $\mathbf{c}$.

Per altra banda, hi ha una segona constant del moviment, que és l'energia. Per deduir-la, hem de multiplicar la segona equació de (6.1) pel vector $\mathbf{v}$, i obtenim:

$$\mathbf{v} \cdot \dot{\mathbf{v}} = \frac{-f(r)}{r}(\mathbf{r} \cdot \mathbf{v}) = \frac{-f(r)r\dot{r}}{r} = -f(r)\frac{dr}{dt},$$

on hem emprat que, com $r^2 = \mathbf{rr}$, llavors $r \cdot \dot{r} = \mathbf{r} \cdot \dot{\mathbf{r}}$ derivant a cada costat. Ara, al fer el mateix amb la variable v, reordenar termes i integrant a cada banda, ens queda l'expressió

$$\int v \; dv = \int -f(r) \; dr,$$

del que resulta

$$\frac{1}{2}v^2 = F(r) + h, \tag{6.2}$$

on $F(r)$ és una funció amb derivada $-f(r)$ i h és la constant d'integració. Per tant, la funció $F(r)$ es calcula convencionalment de la manera següent:

$$F(r) = \int_r^a f(x) \; dx,$$

(1) Si la integral convergeix amb a com ∞, llavors es fa aquesta tria. (2) Si la primera opció dona una integral divergent i triant $a = 0$ dona una que és convergent, llavors la triem d'aquesta manera. (3) Es triarà $a = 1$ si qualsevol de les opcions anteriors falla.

Per tant, si f és de la forma $f(r) = \frac{\mu}{r^p}$, llavors si $p > 1 \to a = \infty$; si $p < 1 \to a = 0$; i si $p = 1 \to a = 1$, conseqüència del criteri de convergència d'integrals impròpies.

En el cas de seguir la llei de gravitació de Newton, la funció $mF(r)$ és coneix com *energia potencial* i es denota amb el símbol U, la quantitat $T := \frac{1}{2}v^2m$ s'anomena *energia cinètica*, i $M = hm$ és l'*energia total* o mecànica. En conclusió, (6.2) esdevé

$$T - U = M,$$

que és conegut com el principi de conservació de l'energia. L'energia total M és mantendrà constant encara que la velocitat i posició de la partícula canviïn.

Per acabar, veurem que també existeix un altre vector que roman constant al llarg del moviment de la partícula, aquest serà l'*eix d'excentricitat*. Suposarem que la partícula es mou segons la llei de gravitació de Newton, per tant les equacions seràn

$$\begin{cases} \dot{\mathbf{r}} = \mathbf{v}, \\ \dot{\mathbf{v}} = \dfrac{-\mu}{r^3}\mathbf{r}, \end{cases}$$

En primer lloc, si $r \neq 0$, llavors podem deduir la següent fórmula

$$\frac{d}{dt}\frac{\mathbf{r}}{r} = \frac{r\dot{\mathbf{r}} - \dot{r}\mathbf{r}}{r^2} =$$

Ara, multiplicant per r a dalt i baix, i emprant la fórmula que hem deduït abans $r \cdot \dot{r} = \mathbf{r} \cdot \dot{\mathbf{r}}$, obtenim

$$= \frac{(\mathbf{r} \cdot \mathbf{r})\dot{\mathbf{r}} - (\mathbf{r} \cdot \dot{\mathbf{r}})\mathbf{r}}{r^3} = \frac{(\mathbf{r} \times \mathbf{v}) \times \mathbf{r}}{r^3} = \frac{\mathbf{c} \times \mathbf{r}}{r^3},$$

on hem emprat la fórmula del producte vectorial

$$(a \times b) \times c = (a \cdot c)b - (c \cdot b)a,$$

De la fórmula es pot deduir que si $c = 0$, llavors $\mathbf{r}/r$ és un vector constant, és a dir, que el vector unitari de $\mathbf{r}$ és constant, i per tant, la partícula del problema és mourà només sobre una recta que passa per lórigen. Multiplicant als dos costats per $-\mu$, tindrem

$$-\mu \frac{d}{dt} \frac{\mathbf{r}}{r} = \mathbf{c} \times (\frac{-\mu}{r^3}\mathbf{r}),$$

que es pot reescriure com

$$\mu \frac{d}{dt} \frac{\mathbf{r}}{r} = \dot{\mathbf{v}} \times \mathbf{c},$$

Integrant als dos costats (a la esquerra per la variable $\frac{\mathbf{r}}{r}$ i la dreta per la variable temporal t), arribem a

$$\mu \left(\mathbf{e} + \frac{\mathbf{r}}{r} \right) = \mathbf{v} \times \mathbf{c},$$

on $\mathbf{e}$ és la constant d'integració. Com abans, si $c \neq 0$, com es compleix $\mathbf{r} \cdot \mathbf{c} = 0$, llavors $\mathbf{e} \cdot \mathbf{c} = 0$, ja que sinó el vector $\mathbf{e} + \mathbf{r}/r$ no seria perpendicular a $\mathbf{c}$. Per tant $\mathbf{e}$ i $\mathbf{c}$ són perpendiculars i $\mathbf{e}$ es troba en el pla on es mou la partícula. De la mateixa manera, si $c = 0$, llavors $\mathbf{r}/r = -\mathbf{e}$, el que implica que $\mathbf{e}$ es troba també en la recta on és mou la partícula.

Ara multiplicant per $\mathbf{r}$ als dos costats, i recordant que $r^2 = \mathbf{r} \cdot \mathbf{r}$, ens queda

$$\mu(\mathbf{e} \cdot \mathbf{r} + r) = \mathbf{r} \cdot \mathbf{v} \times \mathbf{c} = \mathbf{r} \times \mathbf{v} \cdot \mathbf{c} = \mathbf{c} \cdot \mathbf{c},$$

on s'ha emprat la definició del producte mixt, el que implica

$$\mathbf{e} \cdot \mathbf{r} + r = c^2/\mu,$$

Si $e = 0$, llavors $r = c^2/\mu$, el que implica que la posició respecta l'origen de la partícula no creix ni decreix al llarg del moviment, i per tant, es tracta d'un moviment circular.

En canvi, suposem ara que $e \neq 0$. Denotem per $(\mathbf{r}, \theta)$ com el vector posició i l'angle respecte l'eix x per presentar la posició Q de la partícula, i denotem com ω l'angle de $\mathbf{e}$ respecte també l'eix x. Notem que també podem representar la partícula com $(\mathbf{r}, f)$ on $f = \theta - \omega$, que seria com emprar el vector $\mathbf{e}$ com l'eix de coordenades. Llavors, per la definició de producte escalar $\mathbf{e} \cdot \mathbf{r} = er\cos f$, l'equació anterior es converteix en

$$r = e(\frac{c^2}{\mu e} - r\cos f),$$

Aquesta equació ens indica que la distància de la partícula Q al centre és e vegades la diferència entre la partícula i la recta L, una recta a una distància $\frac{c^2}{\mu e}$ des de l'origen i perpendicular a $\mathbf{e}$. La definició anterior és la mateixa del que és coneix a geometria per

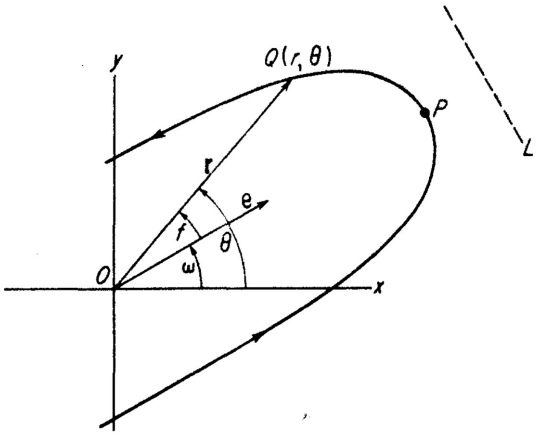


Figura 7: **Representació del moviment de la partícula i l'excentricitat**

excentricitat d'una secció cònica, i per tant, si $0 < e < 1$ l'òrbita de la partícula serà una el·lipse, si $e = 1$, una paràbola, i si $e > 1$, una hipèrbola, i serà constant al llarg de tot el moviment, hem demostrat llavors la primera llei de Kepler.

Segons la definició anterior, si $e > 0$, el valor de r és més petit quan $f = 0$, llavors el punt P de la figura és el més proper al focus O, i es coneix com el pericentre.

Aquests tres paràmetres que es mantenen constants al llarg del recorregut de la partícula ens serviràn com a indicador de la presició dels càlculs que anem fent. Com teòricament s'haurien de mantenir fixes, les variacions que sofreixin haurien de ser molt petites si volem considerar l'implementació del mètode prou bona.

6.1.1 Exemple de resolució mitjançant el mètode de Taylor

Una vegada entès el problema, el resoldrem aplicant el mètode de Taylor amb Diferenciació Automàtica vist al capítol 3. Recordem que el nostre objectiu era trobar una solució al problema del valor inicial (2.1). Denotarem r_1, r_2, r_3 com els tres components del vector r, i v_1, v_2, v_3 les del v. El problema

$$\begin{cases} \dot{\mathbf{r}} = \mathbf{v}, \\ \dot{\mathbf{v}} = \dfrac{-\mu}{r^3}\mathbf{r}, \end{cases}$$

el podem reescriure component a component com

$$\begin{cases} \dot{r}_1 = v_1, \\ \dot{r}_2 = v_2, \\ \dot{r}_3 = v_3, \\ \dot{v}_1 = \dfrac{-\mu r_1}{(\sqrt{r_1^2 + r_2^2 + r_3^2})^3}, \\ \dot{v}_2 = \dfrac{-\mu r_2}{(\sqrt{r_1^2 + r_2^2 + r_3^2})^3}, \\ \dot{v}_3 = \dfrac{-\mu r_3}{(\sqrt{r_1^2 + r_2^2 + r_3^2})^3}, \end{cases}$$

El que farem en primer lloc serà descompondre l'equació en una seqüència d'operacions simples per poder aplicar les propietats de la diferenciació vistes a la proposició (3.1) i anar calculant els valors de les derivades de forma recursiva. Una vegada tinguem tots els valors de les derivades normalitzades, calcularem la sèrie de Taylor amb l'algorisme de Horner, i així obtindrem la solució en el següent pas temporal.

Simplificarem l'equació del problema en diferents operacions component a component, definint les noves variables:

$$\begin{cases} u_1 = r_1,\ u_2 = r_2,\ u_3 = r_3, \\ u_4 = v_1,\ u_5 = v_2,\ u_6 = v_3, \\ u_7 = u_1u_1,\ u_8 = u_2u_2,\ u_9 = u_3u_3, \\ u_{10} = u_7 + u_8 + u_9,\ u_{11} = (u_{10})^{-3/2}, \\ u_{12} = u_{11}u_1,\ u_{13} = u_{11}u_2,\ u_{14} = u_{11}u_3, \\ \dot{r}_1 = u_4,\ \dot{r}_2 = u_5,\ \dot{r}_3 = u_6, \\ \dot{v}_1 = -\mu u_{12},\ \dot{v}_2 = -\mu u_{13},\ \dot{v}_3 = -\mu u_{14}, \end{cases}$$

Si apliquem les propietats de la proposició (3.1), obtenim les fórmules per a les derivades normalitzades

$$\begin{cases} u_1^{[n]}(t) = r_1^{[n]}(t),\ u_2^{[n]}(t) = r_2^{[n]}(t),\ u_3^{[n]}(t) = r_3^{[n]}(t), \\ u_4^{[n]}(t) = v_1^{[n]}(t),\ u_5^{[n]}(t) = v_2^{[n]}(t),\ u_6^{[n]}(t) = v_3^{[n]}(t), \\ u_7^{[n]}(t) = \displaystyle\sum_{i=0}^{n} u_1^{[n-i]}(t)u_1^{[i]}(t),\ u_8^{[n]}(t) = \sum_{i=0}^{n} u_2^{[n-i]}(t)u_2^{[i]}(t),\ u_9^{[n]}(t) = \sum_{i=0}^{n} u_3^{[n-i]}(t)u_3^{[i]}(t), \\ u_{10}^{[n]}(t) = u_7^{[n]}(t) + u_8^{[n]}(t) + u_9^{[n]}(t),\ u_{11}^{[n]}(t) = \dfrac{1}{nu_{10}^{[0]}(t)} \displaystyle\sum_{j=0}^{n-1} (-n\frac{3}{2} - j(-\frac{3}{2}+1))u_{11}^{[n-j]}(t)u_{10}^{[j]}(t), \\ u_{12}^{[n]}(t) = \displaystyle\sum_{i=0}^{n} u_{11}^{[n-i]}(t)u_1^{[i]}(t),\ u_{13}^{[n]}(t) = \sum_{i=0}^{n} u_{11}^{[n-i]}(t)u_2^{[i]}(t),\ u_{14}^{[n]}(t) = \sum_{i=0}^{n} u_{11}^{[n-i]}(t)u_3^{[i]}(t), \\ \dot{r}_1^{[n+1]}(t) = \dfrac{1}{n+1}u_4^{[n]}(t),\ \dot{r}_2^{[n+1]}(t) = \dfrac{1}{n+1}u_5^{[n]}(t),\ \dot{r}_3^{[n+1]}(t) = \dfrac{1}{n+1}u_6^{[n]}(t), \\ \dot{v}_1^{[n+1]}(t) = -\dfrac{\mu}{n+1}u_{12}^{[n]}(t),\ \dot{v}_2^{[n+1]}(t) = -\dfrac{\mu}{n+1}u_{13}^{[n]}(t),\ \dot{v}_3^{[n+1]}(t) = -\dfrac{\mu}{n+1}u_{14}^{[n]}(t), \end{cases}$$

on hem emprat la propietat del producte, de la suma, de l'exponent i les sis darreres equacions es dedueixen directament de la definició de la derivada normalitzada. Si tenim els valors $r_i^{[0]}(t) = r_i(t)$ i $v_i^{[0]}(t) = v_i(t)$ per $i = 1, 2, 3$, que són les condicions inicials en aquest pas, podem obtenir els valors de les derivades $r_i^{[1]}(t)$ i $v_i^{[1]}(t)$, i si calculem recursivament, podem obtenir tots els valors fins un ordre p òptim, el que ens proporciona la fórmula deduïda a l'apartat (3.2) donada una tolerància ε. Llavors, donat un pas h, haurem obtenint el conjunt de derivades normalitzades per calcular la sèrie de Taylor de la solució al següent pas temporal $\bar{t} := t + h$,

$$\bar{r}_i = \sum_{j=0}^{p} r_i^{[j]}(t)h^j,$$

$$\bar{v}_i = \sum_{j=0}^{p} v_i^{[j]}(t)h^j,$$

per $i = 1, 2, 3$. Per avaluar aquest polinomi utilitzarem l'algorisme de Horner, que es coneix per ser un algorisme òptim. Recordem que consistia en aplicar la fórmula iterativa a un polinomi $p(h)$ de grau n amb coeficients a_i:

$$p(h) = a_0 + h(a_1 + h(a_2 + ... + h(a_{n-1} + ha_n)...)),$$

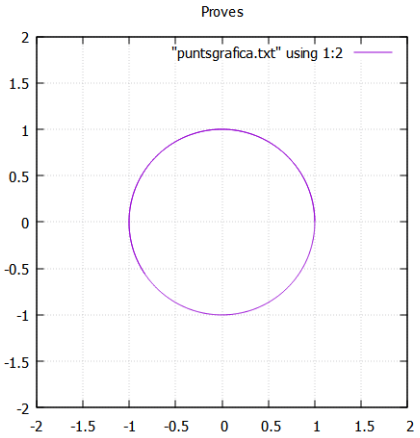
Després de calcular el valor de la solució al següent pas temporal, ens quedaria emprar les fórmules vistes al capítol (3.2) per deduir un pas per a la següent iteració, i així seguir calculant els valors de la solució fins temps $t = b$, que era el nostre objectiu.

A l'annex, hi constarà el programa en C que implementa el mètode de Taylor amb Diferenciació Automàtica. La funció principal admet com a paràmetres d'entrada: un fitxer on escriurà els resultats, una variable *optim* que marcarà 1 si es vol emprar l'ordre òptim proporcionat per la fórmula o 0 sinó, en tot cas s'empraria l'ordre p, una variable *control* que valdrà 1 si es vol fer control de pas de forma automàtica amb la fórmula deduïda o 0 sinó, que s'utilitzaria el pas fixe proporcionat h, les variables a i b que marquen l'interval d'integració, la variable μ del problema, la variable *error* que serà la tolerància, i els vectors r_0 i v_0, que són les condicions inicials del problema.

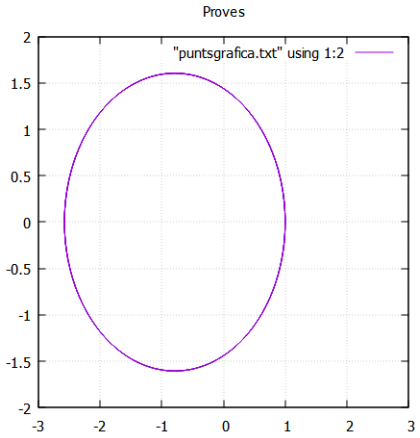
6.1.2 Proves i Resultats

En aquesta secció, anem a realitzar algunes proves sobre la integració realitzada amb el mètode de Taylor. El programa en C que hem implementat proporciona un fitxer amb els punts de l'òrbita resultant que pintarem amb el programa Gnuplot. Per tant, demanant una precisió prou bona (per exemple 10^{-12}), integrant en un temps suficientment gran per poder visualitzar l'òrbita sencera o una part important ($t = 100$), sense emprar control de pas però si d'ordre amb un pas constant $h = 0.1$ (ja que ens interessa dibuixar la gràfica, i sinó ens sortirien molt pocs punts), i amb diverses condicions inicials, es poden obtenir diferents seccions còniques.

Si $r_0 = (1, 0, 0)$ i $v_0 = (0, 1, 0)$, obtenim (a) amb excentricitat: 0, per tant, es tracta d'una circumferència. Si $r_0 = (1, 0, 0)$ i $v_0 = (0, 1.2, 0)$, obtenim (b) amb excentricitat: 0.44, és a dir, una el·lipse.



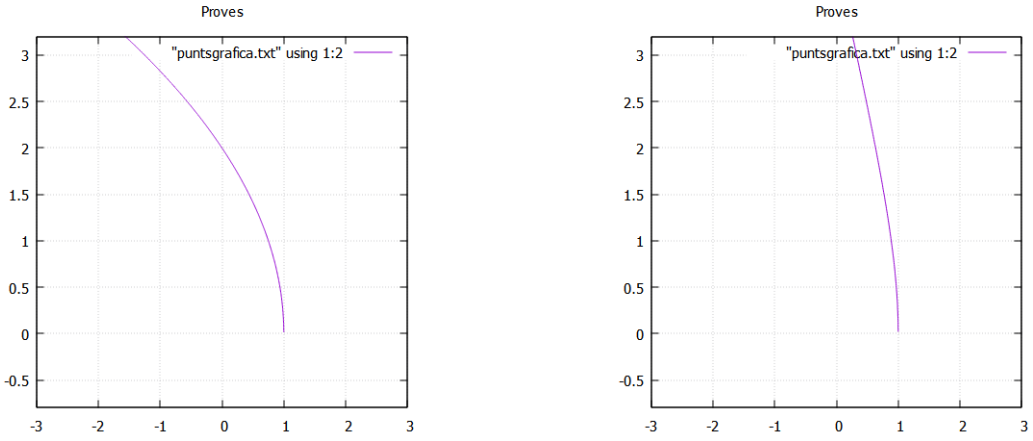
(a) Excentricitat 0



(b) Excentricitat 0.44

Figura 8

Si $r_0 = (1, 0, 0)$ i $v_0 = (0, \sqrt{2}, 0)$, obtenim (a) amb excentricitat: 1, una paràbola. Si $r_0 = (1, 0, 0)$ i $v_0 = (0, 2, 0)$, obtenim (b) amb excentricitat: 3, per tant, una hipèrbola.



(a) Excentricitat 1

(b) Excentricitat 3

Figura 9

Per calcular les excentricitats hem emprat la fórmula més simple:

$$e = \sqrt{1 + \frac{2Mc^2}{\mu^2}},$$

on M és l'energia total, c el moment angular i μ el paràmetre de gravitació estàndard. Per veure una justificació de la fórmula es pot consultar [AM08].

Els dos darrers exemples són casos de òrbites d'escapament, ja que la partícula tendeix a escapar de la força exercida pel cos central, en canvi, els dos primers es consideren òrbites estables ja que tendeixen a orbitar al voltant del cos indefinidament si no intervenen forces externes.

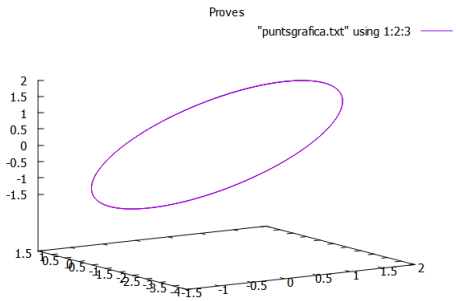


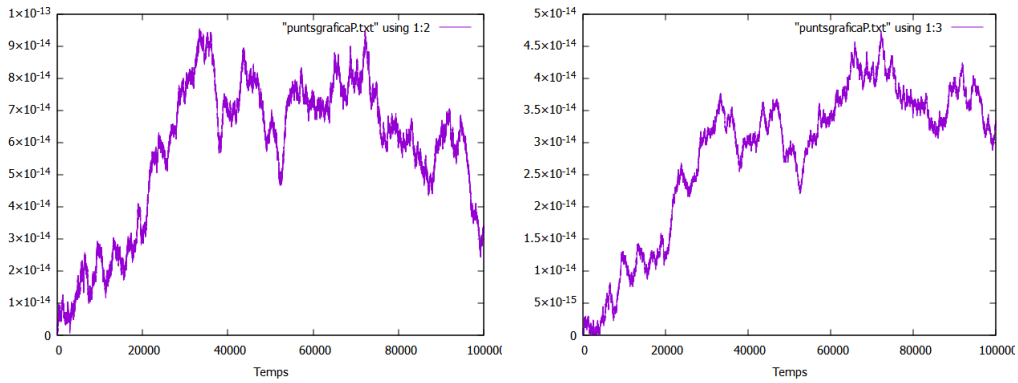
Figura 10

Vistos els diferents tipus de trajectòries que pot fer la partícula, ara anem a fixar unes condicions inicials $r_0 = (1, 1, 1)$ i $v_0 = (-0.5, 0.5, 05)$ i veurem si es conserven l'energia, el moment angular i l'excentricitat, que com hem vist, si suposem que no hi ha cap força exterior que influeix sobre la partícula, aquestes s'haurien de mantenir constants

al llarg del moviment. Demanarem una precisió $\varepsilon = 10^{-18}$ (per la fórmula de l'ordre òptim estareim calculant la sèrie de Taylor fins $p = 22$) i ho farem amb control de pas, per comprovar que l'estratègia funciona. Haurem de triar també un temps suficientment gran, per poder apreciar si hi ha algun tipus d'error acumulatiu, com per exemple $t = 10^5$.

En primer lloc, observem en la figura 10 que l'òrbita serà una el·lipse al tenir excentricitat 0.4365764.

Després, si ens fixem en els passos que va calculant, es pot veure que el pas h va variant entre 0.3 i 1.8 aproximadament, realitzant passos més petits quan es troba aprop del periheli. A continuació mostrarem uns gràfics per veure com varien l'energia, el moment i l'excentricitat respectivament dibuixant els seus errors relatius. L'eix d'abscisses representarà sempre el temps en dies (fins a 10^5).



(a) **Variació relativa d'energia total** (b) **Variació relativa de moments angulars**

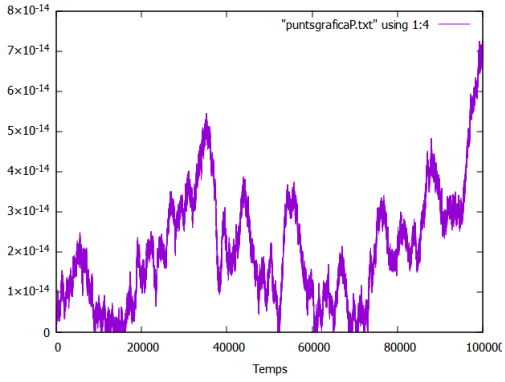


Figura 11: **Variació relativa d'excentricitat**

Respectivament presenten errors relatius finals d'aproximadament $3.278267 \cdot 10^{-14}$, $3.281493 \cdot 10^{-14}$, i $6.993309 \cdot 10^{-14}$. En conclusió, es pot veure que les gràfiques no tendeixen a créixer o decréixer massa, al menys en el temps introduït, i mantenen els errors relativament estables. En aquest exemple hem hagut de demanar $\varepsilon = 10^{-18}$ i ens ha proporcionat errors de 10^{-14}, això és perquè hem triat un temps força elevat, amb un temps bastant menor, es respectaria molt més la tolerància introduïda.

Si volguéssim comparar les estratègies d'emprar control de pas i mantenir un pas constant al llarg de la integració, com per exemple $h = 1$, ens donaríem compte que l'estratègia sense control de pas manté els errors de les energies, moments i excentricitats per sota un marge al llarg de tota la òrbita excepte quan s'apropa al periheli. Això és

degut a que la partícula adquireix en aquest punt la seva velocitat màxima, i per tant, pateix molts canvis de direcció en poc temps, i si porta un pas massa elevat pot comportar a desviacions, canviant així l'òbita i cometent molts errors.

6.2 Problema dels N Cossos

Una vegada que hem entès el problema de la Força Central, un problema més complicat que podríem afrontar seria el problema dels dos cossos, o una versió més general encara, el problema de N Cossos.

En el problema, tindrem n cossos o partícules ($n \geq 2$) que es mouen a la vegada en un espai, amb masses m_i per $i = 1, ..., n$ respectivament, que s'atreuen unes a les altres en parelles amb una força $\frac{Gm_jm_k}{r_{jk}^2}$, on r_{jk} és la distància entre la partícula k i la j.

Si representem O com l'origen de l'espai, i denotem com r_i, v_i els vectors de posició i velocitat de la partícula i, llavors per la segona llei de Newton, cada partícula k de l'espai satisfarà:

$$m_k\ddot{\mathbf{r}}_k = \sum_{j=1,j\neq k}^{n} \frac{Gm_jm_k}{r_{jk}^3}(\mathbf{r}_j - \mathbf{r}_k), \quad k = 1, ..., n, \tag{6.3}$$

on la part dreta de la igualtat representa la força total que exerceixen les altres $n-1$ partícules sobre la partícula k. Notem que es pot representar el problema com el sistema

$$\begin{cases} \dot{\mathbf{r}}_k = \mathbf{v}_k, \\ \dot{\mathbf{v}}_k = \displaystyle\sum_{j=1,j\neq k}^{n} \frac{Gm_j}{r_{jk}^3}(\mathbf{r}_j - \mathbf{r}_k), \end{cases}$$

Aquest sistema té $6n$ incògnites, les tres components de posició, les tres de velocitat i això per cada partícula $k = 1, ..., n$. Si feim un doble sumatori sobre totes les partícules, obtenim l'equació

$$\sum_{k=1}^{n} m_k\ddot{\mathbf{r}}_k = \sum_{k=1}^{n}\sum_{j=1,j\neq k}^{n} \frac{Gm_jm_k}{r_{jk}^3}(\mathbf{r}_j - \mathbf{r}_k) = 0,$$

que val 0 perquè per cada terme $\mathbf{r}_j - \mathbf{r}_k$ apareix el terme negatiu $\mathbf{r}_k - \mathbf{r}_j$ amb el mateix coeficient. Si anomenem $M = \sum_{k=1}^{n} m_k$ com la massa total del sistema i $\mathbf{r}_c = \frac{\sum_{k=1}^{n} m_k\mathbf{r}_k}{M}$ com el centre de masses del conjunt de n cossos, llavors de l'expressió anterior deduïm que $\ddot{\mathbf{r}}_c = 0$. Això implica que $\dot{\mathbf{r}}_c$ és constant i $\mathbf{r}_c = \mathbf{a}t + \mathbf{b}$, on $\mathbf{a}$ i $\mathbf{b}$ són dos vectors constants que es poden calcular donades les condicions inicials del problema, és a dir, valors de posició i velocitat en un temps inicial. Aquesta equació es coneix com el principi de conservació del moment lineal, ja que el centre de masses es mou uniformement en línia recta.

Com una vegada donades les condicions inicials, el centre de masses queda determinat, el problema es converteix a estudiar el moviment dels cossos en relació al centre de masses. Per tant, és convenient moure l'origen de l'espai de coordenades al centre de masses, i com es compleix $\ddot{\mathbf{r}}_c = 0$, l'equació del problema no es veurà afectada si posem $\mathbf{r}_k - \mathbf{r}_c$

enlloc de $\mathbf{r}_k$, més enllà de canviar les posicions de les partícules però no afectarà al seu moviment. Així, a part de les equacions principals del problema, també es compliràn les condicions

$$\sum_{k=1}^{n} m_k \mathbf{r}_k = 0, \quad \text{i} \quad \sum_{k=1}^{n} m_k \mathbf{v}_k = 0,$$

el que ens indica que enlloc de ser un sistema de $6n$ incògnites, serà de $6n-6$ si feim aquest canvi.

Com al cas del problema de la Força Central, aquí també tindrem constants que es conserven al llarg del moviment, com són l'energia total o el moment angular.

En aquest cas, denotem U com la energia potencial del conjunt de partícules, i que definirem així

$$U = \sum_{1 \leq j < k \leq n} \frac{G m_j m_k}{r_{jk}},$$

Si denotem el gradient de U en la direcció $\mathbf{r}_k = (x_k, y_k, z_k)$ com

$$\frac{dU}{d\mathbf{r}_k} = \left[\frac{dU}{dx_k}, \frac{dU}{dy_k}, \frac{dU}{dz_k} \right],$$

llavors, podem reescriure (6.3) com

$$m_k \ddot{\mathbf{r}}_k = \frac{dU}{d\mathbf{r}_k},$$

ja que $r_{jk} = |\mathbf{r}_j - \mathbf{r}_k|$ i la seva derivada respecte $\mathbf{r}_k$ és $-\frac{\mathbf{r}_j - \mathbf{r}_k}{r_{jk}}$. Ara, fent el sumatori per totes les partícules a cada costat i multiplicant per $\dot{\mathbf{r}}_k = d\mathbf{r}_k/dt$, acabem obtenint

$$\sum_{k=1}^{n} m_k \dot{\mathbf{r}}_k \ddot{\mathbf{r}}_k = \sum_{k=1}^{n} \frac{dU}{d\mathbf{r}_k} \frac{d\mathbf{r}_k}{dt},$$

El costat dret de la igualtat és la derivada total de U respecte la variable temporal t (derivem U respecte $\mathbf{r}_k$ i aquest respecte t, per la regla de la cadena, i sumem per cada partícula), i com $v_k^2 = \dot{\mathbf{r}}_k \cdot \dot{\mathbf{r}}_k$, tenim l'expressió

$$\frac{d}{dt} \frac{1}{2} \sum_{k=1}^{n} m_k v_k^2 = \frac{dU}{dt},$$

Definint T la energia total de les n partícules com $\frac{1}{2}\sum_{k=1}^{n} m_k v_k^2$, llavors l'expressió anterior es pot reescriure com

$$T = U + M,$$

on M és una constant que anomenarem energia total. Per tant, acabem de demostrar que l'energia és una constant del moviment, com passava també amb el problema de la Força Central. Degut a aquesta constant, el sistema que teníem passa de $6n-6$ a $6n-7$ variables, i encara el podrem reduir a tres menys amb la constància del moment angular $\mathbf{c}$. Per arribar a aquesta conclusió, feim el producte vectorial a cada banda de (6.3) per

$\mathbf{r}_k$, i a causa de la propietat distributiva del producte i que el producte vectorial d'un vector per si mateix és nul, llavors

$$\sum_{k=1}^{n} m_k(\mathbf{r}_k \times \ddot{\mathbf{r}}_k) = \sum_{k=1}^{n} \sum_{j=1, j\neq k}^{n} \frac{Gm_j m_k}{r_{jk}^3}(\mathbf{r}_j \times \mathbf{r}_k),$$

Com per cada terme $\mathbf{r}_j \times \mathbf{r}_k$ apareix el terme $\mathbf{r}_k \times \mathbf{r}_j$, el costat dret de la igualtat val zero, ja que per la propietat anticonmutativa del producte $\mathbf{a} \times \mathbf{b} = -(\mathbf{b} \times \mathbf{a})$. Per tant,

$$\sum_{k=1}^{n} m_k(\mathbf{r}_k \times \ddot{\mathbf{r}}_k) = 0,$$

Ara, integrant l'expressió anterior respecte la variable t, i del fet que la derivada de $\mathbf{r} \times \mathbf{v}$ és $\mathbf{r} \times \dot{\mathbf{v}}$ (que hem vist al capítol del problema de la Força Central), obtenim

$$\mathbf{c} = \sum_{k=1}^{n} m_k(\mathbf{r}_k \times \mathbf{v}_k),$$

on $\mathbf{c}$ és un vector constant, el moment angular total. Com abans, aquests paràmetres que es mantenen constants al llarg del moviment, seràn un indicador de quant de precís serà el nostre mètode, segons la variació d'aquestes a mesura que integrem.

6.2.1 Exemple de resolució mitjançant els mètodes RK Felhberg i GBS

Abans hem aplicat el mètode de Taylor al problema de la Força Central, i ara aplicarem un mètode Runge-Kutta-Fehlberg d'ordres 5-6, i el mètode GBS d'extrapolació, explicats a les seccions anteriors, al problema dels N Cossos. Aquests dos mètodes s'implementen de tal forma que només canviant la funció f i les condicions inicials es puguin aplicar a qualsevol altra problema.

En el nostre cas, la funció f del problema de valor inicial (2.1) és una funció tal que donat un temps t, i la solució avaluada en aquest temps $x(t) = (\mathbf{r}_1(t), \mathbf{v}_1(t), ..., \mathbf{r}_n(t), \mathbf{v}_n(t))$, que tindrà $2n$ components amb 3 coordenades cada un, retorna per $i = 1, ..., 2n$

$$f_i = \begin{cases} \mathbf{v}_{(i+1)/2}, & \text{per i senar} \\ \displaystyle\sum_{j=1, j\neq i/2}^{n} \frac{Gm_j}{r_{j,i/2}^3}(\mathbf{r}_j - \mathbf{r}_{i/2}), & \text{per i parell} \end{cases}$$

on $f(t, x(t)) = (f_1, ..., f_{2n})$. Per tant, una vegada tenim clar a quina funció hem d'aplicar els mètodes d'integració, només cal seguir els passos explicats als apartats corresponents de cada mètode.

En primer lloc, aplicarem el mètode Runge-Kutta-Fehlberg d'ordres 5-6 representat amb la taula de Butcher de la figura 3. És a dir, que donat un pas h_n, un temps t_n, i el valor de la solució x_n, per obtenir les variables k haurem de calcular:

$$
\begin{aligned}
k_1 &= h_n f(t_n, x_n),\\
k_2 &= h_n f(t_n + \frac{1}{18}h_n, x_n + \frac{1}{18}k_1),\\
k_3 &= h_n f(t_n + \frac{1}{6}h_n, x_n - \frac{1}{12}k_1 + \frac{1}{4}k_2),\\
k_4 &= h_n f(t_n + \frac{2}{9}h_n, x_n - \frac{2}{81}k_1 + \frac{4}{27}k_2 + \frac{8}{81}k_3),\\
k_5 &= h_n f(t_n + \frac{2}{3}h_n, x_n + \frac{40}{33}k_1 - \frac{4}{11}k_2 - \frac{56}{11}k_3 + \frac{54}{11}k_4),\\
k_6 &= h_n f(t_n + h_n, x_n - \frac{369}{73}k_1 + \frac{72}{73}k_2 + \frac{5380}{219}k_3 - \frac{12285}{584}k_4 + \frac{2695}{1752}k_5),\\
k_7 &= h_n f(t_n + \frac{8}{9}h_n, x_n - \frac{8716}{891}k_1 + \frac{656}{297}k_2 + \frac{39520}{891}k_3 - \frac{416}{11}k_4 + \frac{52}{27}k_5),\\
k_8 &= h_n f(t_n + h_n, x_n + \frac{3015}{256}k_1 - \frac{9}{2}k_2 - \frac{4219}{78}k_3 + \frac{5985}{128}k_4 - \frac{539}{384}k_5 + \frac{693}{3328}k_7),
\end{aligned}
$$

i a partir d'aquests valors podem calcular l'estimació dels dos mètodes RK d'ordres 5 i 6 respectivament com

$$\bar{x}_{n+1} = x_n + \frac{3}{80}k_1 + \frac{4}{25}k_3 + \frac{243}{1120}k_4 + \frac{77}{160}k_5 + \frac{73}{700}k_6,$$

$$\hat{x}_{n+1} = x_n + \frac{57}{640}k_1 - \frac{16}{65}k_3 + \frac{1377}{2240}k_4 + \frac{121}{320}k_5 + \frac{891}{8320}k_7 + \frac{2}{35}k_8,$$

Després, calcularem l'error $|\bar{x}_{n+1} - \hat{x}_{n+1}|$ i si compleix que sigui més petit o igual que una tolerància prèviament fixada ε, llavors acceptarem com a solució de la iteració, la solució estimada pel mètode d'ordre superior, és a dir, $\hat{x}_{n+1}$. Si no es complís, reduiríem el pas h_n utilitzant la fórmula de predicció del nou pas deduïda a l'apartat (4.2) per al cas $p = 5$, és a dir

$$|h| = |h_n| \sqrt[6]{\frac{\varepsilon}{|\bar{x}_{n+1} - \hat{x}_{n+1}|}},$$

i tornaríem a calcular els valors corresponents del mètodes, i així successivament fins complir la desigualtat. Notem que tant si es compleix la desigualtat de la diferència dels mètodes com si no, la fórmula proporciona un nou pas adequat, per tant, quan es compleixi, podrem donar una predicció del pas h_{n+1} per a la següent iteració emprant la mateixa expressió.

Per al cas de la resolució del problema de N Cossos amb el mètode d'extrapolació Gragg-Burlisch-Stoer, necessitarem, a més d'un pas H_n, un nombre k_n que marcarà el nombre de vegades que calculem la funció S del mètode per després calcular el polinomi interpolador.

Per començar, haurem de calcular la funció S per a la successió de valors

$$F = \{2, 4, 8, 12, 16, ...\},\ n_i := 2n_{i-2}\ \ per\ 3 \leq i \leq k_n,$$

Tot seguit, passem a calcular el polinomi interpolador d'aquests valors en h^2 amb l'algorisme d'Aitken-Neville, que ens proporcionarà el valor del mètode T_{k_n,k_n}, que acceptarem com a solució si la diferència entre aquest valor i el valor calculat T_{k_n-1,k_n-1} és menor o igual que una tolerància fixada ε. De la mateixa manera que amb el mètode RKF,

utilitzarem la fórmula de predicció del nou pas, tant si l'hem d'actualitzar per tornar a repetir l'iteració en el cas de no complir la desigualtat, com per predir el següent pas per seguir amb la integració si es compleix (si es desitja fer control de pas). A més, també podrem calcular un nou valor k amb l'expressió (5.5) i emprar-lo d'una manera similar.

A l'annex proporcionarem la implementació tant del mètode RKF 5-6, com la del GBS, i també la implementació del mètode de Taylor, que emprarem en el capítol posterior. Les funcions principals dels programes sempre proporcionen un paràmetre d'entrada amb el nom del fitxer on guardar la informació de la solució, una variable *control* que si es posa amb valor 1 es fa control de pas, i si es posa igual a 0 no, unes variables a, i b per marcar els límits d'integració, una tolerància *error*, un pas inicial h, el vector de condicions inicials x_0, el camp vectorial f del problema, i en aquest cas, passem per paràmetre les masses dels cossos del problema de n cossos, encara que també es podrien definir defora com a constants generals i així ens estalviaríem aquesta part. En el cas del mètode de Taylor, hi ha les respectives variables de control i valor inicial per a l'ordre p de la sèrie de Taylor, i equivalentment pel mètode GBS amb el nombre k de avaluacions de la funció S.

6.3 Simulació del Sistema Solar

En aquesta secció, a diferència de les proves realitzades amb el Problema de la Força Central, ho farem sobre cossos reals amb magnituds. Intentarem fer una simulació del Sistema Solar, incloent els planetes Mercuri, Venus, la Terra, Mart, Júpiter, Saturn, Urà, Neptú, Plutó i també afegirem la Lluna i el Sol, a un temps molt elevat per veure què passa i com es conserven els errors d'energia i moment angular en aquest cas, a més del centre de masses.

Una feina prèvia que cal fer és llavors obtenir les dades del Sistema Solar, tant les masses dels cossos com les posicions i velocitats respecte un sistema de referència. Per trobar aquesta informació utilitzarem l'aplicació web Horizons que es troba a la pàgina oficial de la Nasa. Triarem com a unitat de distància la unitat astronòmica o UA, que recordem que 1 UA equival a $0.149597870691 \cdot 10^9\ km$, i com a unitat de temps el dia. Les dades correspondràn al dia 1 de gener de l'any 2000 a les 0:00h. L'aplicació també proporciona la informació referent a les masses, però ens dona directament el producte de la massa del cos M multiplicada per la constant de gravitació universal G, això és perquè la constant de gravitació resulta molt difícil d'aproximar a l'estudiar els planetes. Es coneixen amb molta major precisió com es comporten les òrbites dels cossos i la magnitud de les forces exercides entre ells, així que és més fàcil obtenir amb precisió el producte $G \cdot M$ que la constant o la massa per separades. En quant a posicions i velocitats es refereix, l'aplicació proporciona diferents sistemes de referència per obtenir les dades, aquí utilitzarem com a sistema de referència el de coordenades equatorials. Com a centre de coordenades tindrem el centre de masses, tant el referent a posicions com al de velocitats. El pla x-y o $z = 0$ serà un pla que passi per l'equador de la Terra, amb eix de les x la recta que va des del centre de coordenades fins el que s'anomena el punt Àries, és a dir, que l'eix x serà la recta intersecció entre el pla eclíptic (pla format per el moviment de la Terra la voltant del Sol) i el pla que passa per l'equador quan ocorre l'equinocci de primavera, i per tant, el centre de coordenades està contingut en aquesta recta. L'eix y serà la recta perpendicular a l'eix x continguda en el pla equatorial i que passa pel centre, i l'eix z la recta perpendicular als altres dos eixos i en sentit cap al pol nord terrestre.

Inicialment, si calculem el centre de masses, observarem que no es troba exactament al centre de coordenades, perquè hem hagut de menysprear tota la resta de cossos del

Sistema Solar que també influeixen en el moviment. Per aquest motiu, abans de començar amb la integració haurem de fer una petita translació de tots els cossos, per així situar el centre de masses al centre de coordenades.

Una vegada disposem de tota la informació necessària, realtizarem diverses proves. En primer lloc, veurem com de precisos són els tres mètodes, simplement portant cada mètode a l'extrem per veure quins errors es cometen en la conservació d'energia, moment angular i centre de massa. Recordem que les variables *double*, que són les que emprarem, tenen una precisió fins a 15 dígits decimals, i per tant, la precisió màxima que podrem assolir serà de $1 \cdot 10^{-15}$ en quant a errors relatius es refereix. Per altra banda, mirarem quins són els més eficients en quant a temps de còmput es refereix. No compararem els mètodes entre les seves versions amb i sense control de pas, perquè ja ho hem comentat en el cas del Problema de la Força Central i els resultats tampoc difereixen massa.

Els paràmetres que fixarem seràn: una tolerància suficientment petita per obtenir els resultats més precisos segons cada mètode, un temps final d'integració prou elevat, com per exemple de 365250 dies (1000 anys terrestres aproximadament), i sempre emprarem les versions amb control de pas, amb un pas inicial de 0.1 dies.

Si intentem obtenir la màxima precisió en cada mètode i observant els temps d'execució dels programes, que recordem que variaràn segons les especificacions de l'ordinador en el que s'executin, obtenim els resultats següents:

Mètode de Taylor: Provant amb toleràncies menors a $\varepsilon = 10^{-11}$, es pot comprovar que el mètode va proporcionant errors cada vegada més petits fins a $\varepsilon = 10^{-16}$. A partir d'aquesta tolerància els errors ja no varien gaire. Per tant, acaba cometent errors relatius de conservació d'energia, moment angular i centre de masses de respectivament $4.27 \cdot 10^{-15}$, $9.75 \cdot 10^{-14}$ i $1.14 \cdot 10^{-14}$ en un temps d'aproximadament 26 segons.

Mètode RKF: Realitzant unes proves similars a les del mètode de Taylor, es pot veure que el mètode Runge-Kutta assoleix abans la seva precisió màxima, ja que a partir de toleràncies $\varepsilon = 10^{-13}$ ja gairebé no s'observen canvis ens els errors, més enllà del augment considerable del temps d'execució. Amb aquesta tolerància, el mètode fa errors relatius de $1.14 \cdot 10^{-13}$, $1.27 \cdot 10^{-13}$ i $8.05 \cdot 10^{-14}$ en un temps de 145 segons.

Mètode GBS: Per acabar, el mètode d'Extrapolació GBS es comporta d'una manera semblant. Provant toleràncies $\varepsilon = 10^{-11}, 10^{-12}, 10^{-13}$, ja es pot comprovar que els errors no canvien gaire, arribant a tenir una precisió de $4.34 \cdot 10^{-13}$, $4.98 \cdot 10^{-13}$ i $2.28 \cdot 10^{-13}$ en un temps d'aproximadament 134 segons i nombre d'avaluacions de la funció S: $k = 7$. Es poden fer proves amb diferents nombres k, però els errors no són prou millors, i els temps comencen a augmentar considerablement a partir de valors més grans que 10, ja que el nombre d'avaluacions de la funció S va creixent pràcticament com a potències de 2.

Per tant, amb les proves realitzades, podem concloure que el mètode més ràpid i precís és el mètode de Taylor amb una diferència prou notòria. El mètode Runge-Kutta-Fehlberg proporciona errors relatius un poc millors que el mètode d'Extrapolació GBS. A partir de toleràncies $\varepsilon = 10^{-14}$, tots dos mètodes augmenten massa els seus temps d'execució, però els del mètode GBS creixen més ràpidament. Llavors, de les estratègies que hem vist, la millor en aquests aspectes és la d'expandir la sèrie de Taylor de la solució fins a un ordre determinat, ja que és la que proporciona més dígits de precisió. A diferència dels altres dos, que tenien la idea de combinar avaluacions de la funció f del problema en punts estratègics o simplement calcular la solució amb diferents passos, i després calcular el polinomi interpolador per extrapolar un valor de la solució.

En quant a òrbites i comportament dels planetes es refereix, en aquests 1000 anys no s'observen canvis entre mètodes. Els planetes orbiten infinitament en trajectòries el·líptiques i el Sol es manté pràcticament immòvil, només movent-se lleugerament al voltant del centre de masses. Les òrbites dels planetes tampoc són sempre les mateixes, acaben patint algunes deformacions, però són efectes totalment normals degut al moviment del Sol, que es veu influenciat per les forces principalment dels planetes més grans, com Júpiter o Saturn.

7 Conclusions

En conclusió, hem estudiat alguns dels mètodes numèrics per equacions diferencials més rellevants, com són el mètode de Taylor, els mètodes Runge-Kutta o els mètodes d'Extrapolació, i hem demostrat algunes de les seves propietats de convergència i estabilitat. Per tant, hem assolit els objectius plantejats en quant a coneixement teòric del tema es refereix.

Per altra banda, hem entès els problemes de mecànica celeste de la Força Central i dels N Cossos. Hem vist algunes de les seves característiques generals, i hem demostrat que existeixen un conjunt de constants del moviment. Aquestes propietats les hem emprat en els apartats corresponents d'aplicacions dels mètodes, ja que mesuràvem la precisió d'un mètode com la variació de la constant durant la integració.

Per a la implementació dels mètodes, ho hem fet aplicant les diferents tècniques de control de pas comentades, i pels resultats obtinguts a les simulacions, podem dir que es tracten de tècniques realment efectives. En l'apartat anterior, hem comparat directament les versions dels tres mètodes explicats, analitzant la seva precisió i eficiència. Hem acabat concloent que el mètode de Taylor és superior en aquests aspectes envers als altres dos. Cal comentar que existeixen millors versions, en quant a eficiència i precisió es refereix, dels mètodes Runge-Kutta-Fehlberg i d'Extrapolació GBS però que no hem estudiat. A les respectives referències de cada capítol es pot trobar més informació d'aquests mètodes. En resum, hem assolit també els objectius que ens havíem marcat en la realització dels programes.

Referències

[AM08] R. Abraham and J.E. Marsden. *Foundations of Mechanics*. AMS Chelsea publishing. AMS Chelsea Pub./American Mathematical Society, 2008.

[But16] J. C. Butcher. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, Ltd, 2016.

[Cob84] S. M. Cobb. An introduction to numerical methods for differential equations, by james m. ortega and william g. poole, jr. pp 329. Â£10Â·50. 1983. isbn 0-273-01637-7 (pitman). *The Mathematical Gazette*, 68(444):143â143, 1984.

[GH10] D. Griffiths and D. Higham. *Numerical Methods for Ordinary Differential Equations: Initial Value Problems*. 01 2010.

[HNW00] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving ordinary differential equations I. Nonstiff problems*, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second revised edition, 2000.

[HW76] G. Hall and J. M. Watt. *Modern numerical methods for ordinary differential equations*. Clarendon Press, Oxford, 1976.

[Ise08] A. Iserles. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2 edition, 2008.

[JM04] À. Jorba and J. Masdemont. *Introducció a la simulació*. e-politext. Universitat Politecnica de Catalunya. Iniciativa Digital Politecnica, 2004.

[JZ05] À. Jorba and M. Zou. A software package for the numerical integration of ODEs by means of high-order Taylor methods. *Exp. Math.*, 14(1):99–117, 2005.

[Lam72] J. D. Lambert. *Computational methods in ordinary differential equations*. John Wiley, 1972.

[Pol66] H. Pollard. *Mathematical introduction to celestial mechanics*. Prentice-Hall, Inc., Englewood Cliffs, N.J., 1966.

[SB02] J. Stoer and R. Bulirsch. *Introduction to numerical analysis*, volume 12 of *Texts in Applied Mathematics*. Springer-Verlag, New York, third edition, 2002.

[Ver78] J. H. Verner. Explicit runge-kutta methods with estimates of the local truncation error. *SIAM Journal on Numerical Analysis*, 15(4):772–790, 1978.

Apèndix

A Codi

En aquesta secció adjuntarem els codis dels programes en C emprats al document.

```
// Metode de Taylor aplicat al Problema de la Forca Central

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

// Funcions auxiliars per calcular les derivades normalitzades
double multiplicar(double *v, double *w, int i);
double exponent(double *v, double *w, int i);

// Funcio principal
void taylor_CentralForceProblem(char nom_fitxer[], int optim, int control,
    double a, double b, double mu, double error, double h, int p, double r0
    [3], double v0[3]);

// Funcio main
int main(void) {

  double r0[3], v0[3];

  // Condicions inicials
  r0[0] = 1;
  r0[1] = 1;
  r0[2] = 1;
  v0[0] = -0.5;
  v0[1] = 0.5;
  v0[2] = 0.5;

  taylor_CentralForceProblem("puntsgrafica.txt", 1, 1, 0, 100000, 1, 1e-18,
      0.1, 8, r0, v0);

  return 0;
}

/*
  Fixar parametres:
  nom_fitxer: nom del fitxer on es van guardant les dades
  optim : 1 -> S'utilitza l'ordre optim de la formula, 0 -> S'utilitza l'
      ordre passat com a paramentre p
  control: 1 -> Es fa control de pas, 0 -> No es fa control de pas
  a -> Temps Inicial
  b -> Temps Final
  mu -> Constant de la formula del Problema de la Forca Central
  error -> Precisio que es vol aconseguir
  h -> Pas inicial, important si no es vol fer control de pas
  p -> Ordre maxim de la sÃ¨rie de Taylor
  r0, v0 -> Condicions inicials del problema
*/

void taylor_CentralForceProblem(char nom_fitxer[], int optim, int control,
    double a, double b, double mu, double error, double h, int p, double r0
```

```
  [3], double v0[3]){

int i = 0, j = 0, ordre = 0;
double m = 0;
FILE *puntsgrafica;
puntsgrafica = fopen(nom_fitxer, "w");

// Ordre Optim
// Formula:
if(optim == 1){

  ordre = -0.5*log(error) + 1;
  ordre += 1;

  printf("Ordre Optim: %d\n", ordre);

  p = ordre;
}

// Cream els vectors de variables auxiliars

double u[14][p+1], r[3][p+1], v[3][p+1], moment1[3], moment2[3], resultat
    [6], e, U, T, M;

// CONDICIONS INICIALS

r[0][0] = r0[0];
r[1][0] = r0[1];
r[2][0] = r0[2];
v[0][0] = v0[0];
v[1][0] = v0[1];
v[2][0] = v0[2];

// Calcul de moment angular a temps = a:

moment1[0] = (r[1][0] * v[2][0]) - (r[2][0] * v[1][0]);
moment1[1] = (r[2][0] * v[0][0]) - (r[0][0] * v[2][0]);
moment1[2] = (r[0][0] * v[1][0]) - (r[1][0] * v[0][0]);

suma1 = sqrt(moment1[0]*moment1[0] + moment1[1]*moment1[1] + moment1[2]*
    moment1[2]);

// Calcul de l'energia total a temps = a:

T = 0.5*(v[0][0]*v[0][0] + v[1][0]*v[1][0] + v[2][0]*v[2][0]);
U = -mu/sqrt(r[0][0]*r[0][0] + r[1][0]*r[1][0] + r[2][0]*r[2][0]);
M = T + U;

// Calcul de l'excentricitat a temps = a:

e = sqrt(1 + (2*M*(moment1[0]*moment1[0] + moment1[1]*moment1[1] + moment1
    [2]*moment1[2])/(mu*mu)));

printf("Excentricitat: %le\n", e);

fprintf(puntsgrafica, "%17.16le\t%17.16le\t%17.16le\n", M, suma1, e);

for(m=a; m<b; m+=h){
  // Diferenciaci\'o Autom\`{a}tica

  for(i=0; i<p; i++){
```

```
    u[0][i] = r[0][i];
    u[1][i] = r[1][i];
    u[2][i] = r[2][i];
    u[3][i] = v[0][i];
    u[4][i] = v[1][i];
    u[5][i] = v[2][i];

    u[6][i] = multiplicar(u[0], u[0], i);

    u[7][i] = multiplicar(u[1], u[1], i);

    u[8][i] = multiplicar(u[2], u[2], i);

    u[9][i] = u[6][i] + u[7][i] + u[8][i];

    if(i>0){
      u[10][i] = exponent(u[10], u[9], i);
    }
    else{
      u[10][i] = pow(u[9][i], -1.5);
    }

    u[11][i] = multiplicar(u[10], u[0], i);

    u[12][i] = multiplicar(u[10], u[1], i);

    u[13][i] = multiplicar(u[10], u[2], i);

    v[0][i+1] = -mu*u[11][i]/(i+1);
    v[1][i+1] = -mu*u[12][i]/(i+1);
    v[2][i+1] = -mu*u[13][i]/(i+1);

    r[0][i+1] = u[3][i]/(i+1);
    r[1][i+1] = u[4][i]/(i+1);
    r[2][i+1] = u[5][i]/(i+1);

  }

  for(int i=0; i<6; i++){
    resultat[i] = 0;
  }

  // Taylor amb avaluacio del polinomi amb Horner
  for(i=p; i>-1; i--){
    resultat[0] = resultat[0]*h + r[0][i];
    resultat[1] = resultat[1]*h + r[1][i];
    resultat[2] = resultat[2]*h + r[2][i];
    resultat[3] = resultat[3]*h + v[0][i];
    resultat[4] = resultat[4]*h + v[1][i];
    resultat[5] = resultat[5]*h + v[2][i];
  }

  r[0][0] = resultat[0];
  r[1][0] = resultat[1];
  r[2][0] = resultat[2];
  v[0][0] = resultat[3];
  v[1][0] = resultat[4];
  v[2][0] = resultat[5];

  // CONTROL DE PAS
```

```
  if (control == 1){
    int option = 1;
    double max = fabs(r[0][0]), max1, max2, v1, v2;

    for(i = 0; i < 3; i++){
      if(max < fabs(r[i][0])){
        max = fabs(r[i][0]);
      }
      if(max < fabs(v[i][0])){
        max = fabs(v[i][0]);
      }
    }

    if(max > 1){
      option = 2;
    }

    max2 = r[0][p-1];
    for(i = 0; i < 3; i++){
      if(max2 < fabs(r[i][p-1])){
        max2 = fabs(r[i][p-1]);
      }
      if(max2 < fabs(v[i][p-1])){
        max2 = fabs(v[i][p-1]);
      }
    }

    max1 = r[0][p];
    for(i = 0; i < 3; i++){
      if(max1 < fabs(r[i][p])){
        max1 = fabs(r[i][p]);
      }
      if(max1 < fabs(v[i][p])){
        max1 = fabs(v[i][p]);
      }
    }

    if(option == 1){
      v1 = pow(1./max2, 1./(p-1));
      v2 = pow(1./max1, 1./(p));
    }
    else{
      v1 = pow(max/max2, 1./(p-1));
      v2 = pow(max/max1, 1./(p));
    }

    double min = fmin(v1, v2);

    h = min/exp(2);
  }

  // Escrivim els punts a un fitxer
  // fprintf(puntsgrafica, "%lf\t%lf\t%lf\t%lf\t%lf\t%lf\n", r[0][0], r
      [1][0], r[2][0], v[0][0], v[1][0], v[2][0]);

}

// CALCUL DE L'ERROR DEL METODE

// Moment a temps = b: Vector mom
```

```
  moment2[0] = (r[1][0] * v[2][0]) - (r[2][0] * v[1][0]);
  moment2[1] = (r[2][0] * v[0][0]) - (r[0][0] * v[2][0]);
  moment2[2] = (r[0][0] * v[1][0]) - (r[1][0] * v[0][0]);

  // Calcul de l'energia total a temps = b:

  double T2, U2, M2;

  T2 = 0.5*(v[0][0]*v[0][0] + v[1][0]*v[1][0] + v[2][0]*v[2][0]);
  U2 = -mu/sqrt(r[0][0]*r[0][0] + r[1][0]*r[1][0] + r[2][0]*r[2][0]);
  M2 = T2 + U2;

  // Calcul de l'excentricitat a temps = b:

  double e2;

  e2 = sqrt(1 + (2*M2*(moment2[0]*moment2[0] + moment2[1]*moment2[1] +
      moment2[2]*moment2[2])/(mu*mu)));

  // Resta de moments:
  if(moment1[0] != 0){
    moment2[0] = fabs(moment2[0] - moment1[0])/moment1[0];
  }
  else{
    moment2[0] = fabs(moment2[0] - moment1[0]);
  }
  if(moment1[1] != 0){
    moment2[1] = fabs(moment2[1] - moment1[1])/moment1[1];
  }
  else{
    moment2[1] = fabs(moment2[1] - moment1[1]);
  }
  if(moment1[2] != 0){
    moment2[2] = fabs(moment2[2] - moment1[2])/moment1[2];
  }
  else{
    moment2[2] = fabs(moment2[2] - moment1[2]);
  }

  // Norma euclidiana de la resta:

  double suma = 0;

  for(i=0; i<3; i++){
    suma += moment2[i]*moment2[i];
  }
  suma = sqrt(suma);

  printf("Error relatiu de moments angulars del metode: %le\n", suma);

  printf("Error relatiu d'energies totals del metode: %le\n", (M2-M)/M);

  printf("Error relatiu d'excentricitats del metode: %le\n", (e2-e)/e);

  fclose(puntsgrafica);

}

// Funcions auxiliars per realitzar la diferenciacio automatica

double multiplicar(double *v, double *w, int i){
```

```
  int j;
  double suma = 0;
  for(j=0; j<i+1; j++){
    suma += v[i-j]*w[j];
  }
  return suma;
}

double exponent(double *v, double *w, int i){
  int j;
  double suma = 0;
  for(j=0; j<i; j++){
    suma += (-i*1.5 + j*0.5) * w[i-j] * v[j];
  }
  suma *= 1./(i * w[0]);
  return suma;
}
```

```
// Metode de Taylor aplicat al Problema dels N cossos

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

// Nombre de cossos del problema
#define N  11

// Masses del Sistema Solar
#define SUN  2.959122082855911e-04
#define MERCURY  4.912547451450812e-11
#define VENUS  7.243452486162703e-10
#define EARTH  8.887692390113510e-10
#define MOON  1.093189565989891e-11
#define MARS  9.549535105779258e-11
#define JUPITER  2.825345909524226e-07
#define SATURN  8.459715185680659e-08
#define URANUS  1.292024916781969e-08
#define NEPTUNE  1.524358900784276e-08
#define PLUTO  2.188699765425970e-12


double multiplicar(double *v, double *w, int i);
double exponent(double *v, double *w, int i);
void taylor_NBodyProblem(char nom_fitxer[], int optim, int control, double a
    , double b, double error, double h, int p, double r0[3][N], double v0
    [3][N], double mas[N]);

int main(void) {

  double v0[3][N], r0[3][N], mas[N];

  // CONDICIONS INICIALS (Sistema Solar)
  // SUN
  r0[0][0] = -7.139147120601119e-03;
  r0[1][0] = -2.643642827742669e-03;
  r0[2][0] = -9.214328974937250e-04;
  v0[0][0] = 5.374261885473962e-06;
  v0[1][0] = -6.761946839502383e-06;
  v0[2][0] = -3.034366584882792e-06;
```

```
// MERCURY
r0[0][1] = -1.478672244638274e-01;
r0[1][1] = -4.006284788220161e-01;
r0[2][1] = -1.989142284217030e-01;
v0[0][1] = 2.117424560897133e-02;
v0[1][1] = -5.515502052846831e-03;
v0[2][1] = -5.141099487603652e-03;

// VENUS
r0[0][2] = -7.257693636228210e-01;
r0[1][2] = -3.966769810278080e-02;
r0[2][2] = 2.790149931923123e-02;
v0[0][2] = 5.189070601827868e-04;
v0[1][2] = -1.851509568109535e-02;
v0[2][2] = -8.362180624996631e-03;

// EARTH
r0[0][3] = -1.756637992089585e-01;
r0[1][3] = 8.861993027720751e-01;
r0[2][3] = 3.844346475329568e-01;
v0[0][3] = -1.722857155938858e-02;
v0[1][3] = -2.766250533342586e-03;
v0[2][3] = -1.199380179087664e-03;

// MOON
r0[0][4] = -1.777871599453381e-01;
r0[1][4] = 8.846186349429087e-01;
r0[2][4] = 3.840156824524305e-01;
v0[0][4] = -1.690468993697088e-02;
v0[1][4] = -3.189750216558814e-03;
v0[2][4] = -1.384021708498301e-03;

// MARS
r0[0][5] = 1.383221919028136e+00;
r0[1][5] = -8.149426100955265e-03;
r0[2][5] = -4.104002827718646e-02;
v0[0][5] = 7.533013358251248e-04;
v0[1][5] = 1.380715975238168e-02;
v0[2][5] = 6.312746435988354e-03;

// JUPITER
r0[0][6] = 3.996320681110833e+00;
r0[1][6] = 2.730993728174368e+00;
r0[2][6] = 1.073274468938447e+00;
v0[0][6] = -4.558099510764580e-03;
v0[1][6] = 5.878007648708349e-03;
v0[2][6] = 2.630568703876106e-03;

// SATURN
r0[0][7] = 6.401418058908803e+00;
r0[1][7] = 6.170249258316296e+00;
r0[2][7] = 2.273030044850830e+00;
v0[0][7] = -4.285743775521677e-03;
v0[1][7] = 3.522771577940802e-03;
v0[2][7] = 1.639335306091836e-03;

// URANUS
r0[0][8] = 1.442338133008372e+01;
r0[1][8] = -1.251013434083655e+01;
r0[2][8] = -5.683123856249092e+00;
v0[0][8] = 2.683753457289004e-03;
```

```
v0[1][8] = 2.455012942924642e-03;
v0[2][8] = 1.037270738064569e-03;

// NEPTUNE
r0[0][9] = 1.680362717435433e+01;
r0[1][9] = -2.298358750883521e+01;
r0[2][9] = -9.825655628595147e+00;
v0[0][9] = 2.584745162555314e-03;
v0[1][9] = 1.661540410385484e-03;
v0[2][9] = 6.157289666404291e-04;

// PLUTO
r0[0][10] = -9.883996976624029e+00;
r0[1][10] = -2.798093194013836e+01;
r0[2][10] = -5.753974189979520e+00;
v0[0][10] = 3.034076757014200e-03;
v0[1][10] = -1.134492916122669e-03;
v0[2][10] = -1.268191592371952e-03;

// Masses dels cossos
mas[0] = SUN;
mas[1] = MERCURY;
mas[2] = VENUS;
mas[3] = EARTH;
mas[4] = MOON;
mas[5] = MARS;
mas[6] = JUPITER;
mas[7] = SATURN;
mas[8] = URANUS;
mas[9] = NEPTUNE;
mas[10] = PLUTO;

// Calcul del Centre de Masses:

double M=0, r[3], v[3];

r[0]=0;
r[1]=0;
r[2]=0;
v[0]=0;
v[1]=0;
v[2]=0;

for(int i=0; i<N; i++){
  M += mas[i];
}
for(int i=0; i<3; i++){
  for(int j=0; j<N; j++){
    r[i] += mas[j]*r0[i][j];
    v[i] += mas[j]*v0[i][j];
  }
  r[i] = r[i]/M;
  v[i] = v[i]/M;
}

// Translacio:
for(int i=0; i<3; i++){
  for(int j=0; j<N; j++){
    r0[i][j] -= r[i];
    v0[i][j] -= v[i];
  }
```

```
  }

  taylor_NBodyProblem("puntsgraficaT.txt", 1, 1, 0, 365250, 1e-16, 0.1, 15,
      r0, v0, mas);

  return 0;
}

/*
    Fixar parametres:
    optim: 1 -> Fa Taylor fins l'ordre optim, 0 -> Fa Taylor fins l'ordre
        fixat
    control: 1 -> Es fa control de pas, 0 -> No es fa control de pas
    a -> Temps Inicial
    b -> Temps Final
    error -> Precisio que es vol aconseguir
    h -> Pas incial, important si no es vol fer control de pas
    p -> Ordre de la serie de Taylor
    r0, v0 -> Condicions inicials
  */

void taylor_NBodyProblem(char nom_fitxer[], int optim, int control, double a
    , double b, double error, double h, int p, double r0[3][N], double v0
    [3][N], double mas[N]){
  int i = 0, j = 0, k = 0, ordre = 0;
  double m = 0, moment1[3][N], suma_moments[3], moment2[3][N], suma_moments2
      [3];

  FILE *puntsgrafica;
  puntsgrafica = fopen(nom_fitxer, "w");

  // Ordre Optim
  // Formula de l'error
  if(optim == 1){

    ordre = -0.5*log(error) + 1;
    ordre += 1;
    // printf("Ordre optim: %d\n", ordre);

    p = ordre;

  }

  // Cream els vectors de variables auxiliars
  double u[6][N][p+1], r[3][N][p+1], v[3][N][p+1], x[N][3][N][p+1], y[N][3][
      N][p+1], s[N][3][N][p+1], d[N][3][N][p+1], z[N][N][p+1], q[N][N][p+1],
       f[3][N];

  // CONDICIONS INICIALS
  for(i = 0; i < N; i++){
    for(j = 0; j < 3; j++){
      r[j][i][0] = r0[j][i];
      v[j][i][0] = v0[j][i];
    }
  }

  // Calcul de moments angulars

  for(i = 0; i < N; i++){
    moment1[0][i] = (r[1][i][0] * v[2][i][0]) - (r[2][i][0] * v[1][i][0]);
    moment1[1][i] = (r[2][i][0] * v[0][i][0]) - (r[0][i][0] * v[2][i][0]);
```

```
  moment1[2][i] = (r[0][i][0] * v[1][i][0]) - (r[1][i][0] * v[0][i][0]);
}

for(j = 0; j < 3; j++){
  suma_moments[j] = 0;
}

for(i = 0; i < N; i++){
  for(j = 0; j < 3; j++){
    suma_moments[j] += mas[i]*moment1[j][i];
  }
}

double suma1 = sqrt(suma_moments[0]*suma_moments[0] + suma_moments[1]*
    suma_moments[1] + suma_moments[2]*suma_moments[2]);

// Calcul de l'energia total:

double U=0, T=0, E=0, norma[N][N], norm[N];
for(j=0; j<N; j++){
  for(int k=j+1; k<N; k++){
    norma[j][k] = 0;
    for(int i=0; i<3; i++){
      norma[j][k] += (r[i][j][0] - r[i][k][0])*(r[i][j][0] - r[i][k][0]);
    }
    norma[j][k] = sqrt(norma[j][k]);
    U += mas[j]*mas[k]/(norma[j][k]);
  }
}

for(j=0; j<N; j++){
  norm[j] = 0;
  for(int i=0; i<3; i++){
    norm[j] += v[i][j][0]*v[i][j][0];
  }
  norm[j] = sqrt(norm[j]);
  T += mas[j]*norm[j]*norm[j];
}
T *= 0.5;
E = T - U;

for(m=a; m<b; m+=h){
  // Diferenciacio Automatica

  for(i=0; i<p; i++){
    for(k=0; k<N; k++){
      // Variables inicials
      u[0][k][i] = r[0][k][i];
      u[1][k][i] = r[1][k][i];
      u[2][k][i] = r[2][k][i];
      u[3][k][i] = v[0][k][i];
      u[4][k][i] = v[1][k][i];
      u[5][k][i] = v[2][k][i];

      for(j=0; j<N; j++){
        if((j-1) == (k-1)){
          continue;
        }
        for(l=0; l<3; l++){
              d[j][l][k][i] = u[l][k][i] - r[l][j][i];
              x[j][l][k][i] = multiplicar(d[j][l][k], d[j][l][k], i);
```

```
                }
        }

        for(j=0; j<N; j++){
          if((j-1) == (k-1)){
            continue;
          }
          z[j][k][i] = x[j][0][k][i] + x[j][1][k][i] + x[j][2][k][i];
        }

        for(j=0; j<N; j++){
          if((j-1) == (k-1)){
            continue;
          }
          if(i>0){
            q[j][k][i] = exponent(q[j][k], z[j][k], i);
          }
          else{
            q[j][k][i] = pow(z[j][k][i], -1.5);
          }
        }

        for(j=0; j<N; j++){
          if((j-1) == (k-1)){
            continue;
          }
          for(l=0; l<3; l++){
                s[j][l][k][i] = r[l][j][i] - u[l][k][i];
                y[j][l][k][i] = multiplicar(q[j][k], s[j][l][k], i);
                  }
        }

        for(l=0; l<3; l++){
             f[l][k] = 0;
             for(j=0; j<N; j++){
                  if((j-1) == (k-1)){
                      continue;
                  }
                  f[l][k] += mas[j]*y[j][l][k][i];
           }
        }

        for(l=0; l<3; l++){
             v[l][k][i+1] = f[l][k]/(i+1);
             r[l][k][i+1] = u[l+3][k][i]/(i+1);
        }

      }
    }

    double resultat[6][N];
    for(k=0; k<N; k++){
      for(int i=0; i<6; i++){
        resultat[i][k] = 0;
      }
    }

    // Taylor per avaluacio del polinomi amb Horner
    for(k=0; k<N; k++){
      for(i=p; i>-1; i--){
        resultat[0][k] = resultat[0][k]*h + r[0][k][i];
```

```
      resultat[1][k] = resultat[1][k]*h + r[1][k][i];
      resultat[2][k] = resultat[2][k]*h + r[2][k][i];
      resultat[3][k] = resultat[3][k]*h + v[0][k][i];
      resultat[4][k] = resultat[4][k]*h + v[1][k][i];
      resultat[5][k] = resultat[5][k]*h + v[2][k][i];
    }
  }

  for(k=0; k<N; k++){
    r[0][k][0] = resultat[0][k];
    r[1][k][0] = resultat[1][k];
    r[2][k][0] = resultat[2][k];
    v[0][k][0] = resultat[3][k];
    v[1][k][0] = resultat[4][k];
    v[2][k][0] = resultat[5][k];
  }

  // CONTROL DE PAS
  if(control == 1){
    int option = 1;
    double max = fabs(r[0][0][0]), max1, max2;
    for(k=0; k<N; k++){
      for(i = 0; i < 3; i++){
        if(max < fabs(r[i][k][0])){
          max = fabs(r[i][k][0]);
        }
        if(max < fabs(v[i][k][0])){
          max = fabs(v[i][k][0]);
        }
      }
    }


    if(max > 1){
      option = 2;
    }

    double v1, v2;

    max2 = r[0][0][p-1];
    for(k=0; k<N; k++){
      for(i = 0; i < 3; i++){
        if(max2 < fabs(r[i][k][p-1])){
          max2 = fabs(r[i][k][p-1]);
        }
        if(max2 < fabs(v[i][k][p-1])){
          max2 = fabs(v[i][k][p-1]);
        }
      }
    }

    max1 = r[0][0][p];
    for(k=0; k<N; k++){
      for(i = 0; i < 3; i++){
        if(max1 < fabs(r[i][k][p])){
          max1 = fabs(r[i][k][p]);
        }
        if(max1 < fabs(v[i][k][p])){
          max1 = fabs(v[i][k][p]);
        }
      }
```

```
      }

      if(option == 1){
        v1 = pow(1./max2, 1./(p-1));
        v2 = pow(1./max1, 1./(p));
      }
      else{
        v1 = pow(max/max2, 1./(p-1));
        v2 = pow(max/max1, 1./(p));
      }

      double min = fmin(v1, v2);

      h = min/exp(2);

    }

}

// Calcul del Centre de Masses

double M=0, r1[3], v1[3];

r1[0]=0;
r1[1]=0;
r1[2]=0;
v1[0]=0;
v1[1]=0;
v1[2]=0;

for(int i=0; i<N; i++){
  M += mas[i];
}

for(int i=0; i<3; i++){
  for(int j=0; j<N; j++){
      r1[i] += mas[j]*r[i][j][0];
      v1[i] += mas[j]*v[i][j][0];
  }
  r1[i] = r1[i]/M;
  v1[i] = v1[i]/M;
}

// Calcul dels moments angulars finals

for(i = 0; i < N; i++){
  moment2[0][i] = (r[1][i][0] * v[2][i][0]) - (r[2][i][0] * v[1][i][0]);
  moment2[1][i] = (r[2][i][0] * v[0][i][0]) - (r[0][i][0] * v[2][i][0]);
  moment2[2][i] = (r[0][i][0] * v[1][i][0]) - (r[1][i][0] * v[0][i][0]);
}

for(j = 0; j < 3; j++){
  suma_moments2[j] = 0;
}

for(i = 0; i < N; i++){
  for(j = 0; j < 3; j++){
    suma_moments2[j] += mas[i]*moment2[j][i];
  }
}
```

```
// Resta de moment:
if(suma_moments[0] != 0){
  suma_moments2[0] = (suma_moments2[0] - suma_moments[0])/suma_moments[0];
}
else{
  suma_moments2[0] = (suma_moments2[0] - suma_moments[0]);
}
if(suma_moments[1] != 0){
  suma_moments2[1] = (suma_moments2[1] - suma_moments[1])/suma_moments[1];
}
else{
  suma_moments2[1] = (suma_moments2[1] - suma_moments[1]);
}
if(suma_moments[2] != 0){
  suma_moments2[2] = (suma_moments2[2] - suma_moments[2])/suma_moments[2];
}
else{
  suma_moments2[2] = (suma_moments2[2] - suma_moments[2]);
}

// Calcul de l'energia total:

double U2=0, T2=0, E2 = 0, norma2[N][N], norm2[N];
for(j=0; j<N; j++){
  for(int k=j+1; k<N; k++){
    norma2[j][k] = 0;
    for(int i=0; i<3; i++){
      norma2[j][k] += (r[i][j][0] - r[i][k][0])*(r[i][j][0] - r[i][k][0]);
    }
    norma2[j][k] = sqrt(norma2[j][k]);
    U2 += mas[j]*mas[k]/(norma2[j][k]);
  }
}

for(j=0; j<N; j++){
  norm2[j] = 0;
  for(int i=0; i<3; i++){
    norm2[j] += v[i][j][0]*v[i][j][0];
  }
  norm2[j] = sqrt(norm2[j]);
  T2 += mas[j]*norm2[j]*norm2[j];
}
T2 *= 0.5;
E2 = T2 - U2;

printf("Error relatiu d'energies: %le\n", (E-E2)/E);

// Norma del moment:

double suma = 0;

for(i=0; i<3; i++){
  suma += suma_moments2[i]*suma_moments2[i];
}
suma = sqrt(suma);

printf("Error relatiu de moments angulars: %le\n", suma);

printf("Centre de masses al final de la integracio: %le \n", sqrt(r1[0]*r1
    [0] + r1[1]*r1[1] + r1[2]*r1[2]));
```

```
  fclose(puntsgrafica);

}

// Funcions auxiliars per a la diferenciacio automatica

double multiplicar(double *v, double *w, int i){
  int j;
  double suma = 0;
  for(j=0; j<i+1; j++){
    suma += v[i-j]*w[j];
  }
  return suma;
}

double exponent(double *v, double *w, int i){
  int j;
  double suma = 0;
  for(j=0; j<i; j++){
    suma += (-i*1.5 + j*0.5) * w[i-j] * v[j];
  }
  suma *= 1./(i * w[0]);
  return suma;
}
```

```
// Metode de Runge-Kutta-Fehlberg aplicat a el Problema de N Cossos

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

// Nombre de cossos del problema
#define N  11
// Dimensio del sistema d'equacions
#define n N*6

void f(double t, double* x, double *mas, double K[n]);
void rkf_56_NBodyProblem(char nom_fitxer[], int control, double a, double b,
     double error, double h, double x0[n], double mas[N], void f(double t,
    double* x, double *mas, double K[n]));

#define C2  1./18
#define C3  1./6
#define C4  2./9
#define C5  2./3
#define C6  1
#define C7  8./9
#define C8  1

#define B_1  3./80
#define B_2  0
#define B_3  4./25
#define B_4  243./1120
#define B_5  77./160
#define B_6  73./700

#define B1  57./640
#define B2  0
#define B3  -16./65
#define B4  1377./2240
```

```
#define B5   121./320
#define B6   0
#define B7   891./8320
#define B8   2./35

#define A21   1./18
#define A31   -1./12
#define A41   -2./81
#define A51   40./33
#define A61   -369./73
#define A71   -8716./891
#define A81   3015./256

#define A32   1./4
#define A42   4./27
#define A52   -4./11
#define A62   72./73
#define A72   656./297
#define A82   -9./4

#define A43   8./81
#define A53   -56./11
#define A63   5380./219
#define A73   39520./891
#define A83   -4219./78

#define A54   54./11
#define A64   -12285./584
#define A74   -416./11
#define A84   5985./128

#define A65   2695./1752
#define A75   52./27
#define A85   -539./384

#define A76   0
#define A86   0

#define A87   693./3328

/*
   Fixar parametres:
   control: 1 -> Es fa control de pas, 0 -> No es fa control de pas
   a -> Temps Inicial
   b -> Temps Final
   error -> Precisio que es vol aconseguir
   h -> Pas incial, important si no es vol fer control de pas
   mas -> Vector de masses dels cossos
   x0 -> Condicio inicial del sistema
   f -> Camp vectorial del problema
*/

void rkf_56_NBodyProblem(char nom_fitxer[], int control, double a, double b,
     double error, double h, double x0[n], double mas[N], void f(double t,
    double* x, double *mas, double K[n])){
  int i = 0, j = 0, k = 0;
  double m = 0, moment1[3][N], suma_moments[3], moment2[3][N], suma_moments2
      [3];
  double K[8][n], p[n], x[n], y[n];
  FILE *puntsgrafica;
  puntsgrafica = fopen(nom_fitxer, "w");
```

```
// CONDICIONS INICIALS
for(i = 0; i < n; i++){
  p[i] = x0[i];
}

// Calcul de moments angulars

for(i = 0; i < N; i++){
  moment1[0][i] = (p[(6*i) + 1] * p[(6*i) + 5]) - (p[(6*i) + 2] * p[(6*i)
      + 4]);
  moment1[1][i] = (p[(6*i) + 2] * p[(6*i) + 3]) - (p[(6*i)] * p[(6*i) +
      5]);
  moment1[2][i] = (p[(6*i)] * p[(6*i) + 4]) - (p[(6*i) + 1] * p[(6*i) +
      3]);
}

for(j = 0; j < 3; j++){
  suma_moments[j] = 0;
}

for(i = 0; i < N; i++){
  for(j = 0; j < 3; j++){
    suma_moments[j] += mas[i]*moment1[j][i];
  }
}

double suma1 = sqrt(suma_moments[0]*suma_moments[0] + suma_moments[1]*
    suma_moments[1] + suma_moments[2]*suma_moments[2]);

// Calcul de l'energia total:

double U=0, T=0, E=0, norma[N][N], norm[N];
for(j=0; j<N; j++){
  for(int k=j+1; k<N; k++){
    norma[j][k] = 0;
    for(int i=0; i<3; i++){
      norma[j][k] += (x0[(6*j)+i] - x0[(6*k)+i])*(x0[(6*j)+i] - x0[(6*k)+i
          ]);
    }
    norma[j][k] = sqrt(norma[j][k]);
    U += mas[j]*mas[k]/norma[j][k];
  }
}

for(j=0; j<N; j++){
  norm[j] = 0;
  for(int i=0; i<3; i++){
    norm[j] += x0[(6*j)+i+3]*x0[(6*j)+i+3];
  }
  norm[j] = sqrt(norm[j]);
  T += mas[j]*norm[j]*norm[j];
}
T *= 0.5;
E = T - U;


// RUNGE-KUTTA-FEHLBERG

m = a;
while(m < b){
```

```
f(m + C1*h, p, mas, K[0]);

for(k=0; k<n; k++){
  x[k] = p[k] + h*A21*K[0][k];
}

f(m + C2*h, x, mas, K[1]);

for(k=0; k<n; k++){
  x[k] = p[k] + h*(A31*K[0][k] + A32*K[1][k]);
}

f(m + C3*h, x, mas, K[2]);

for(k=0; k<n; k++){
  x[k] = p[k] + h*(A41*K[0][k] + A42*K[1][k] + A43*K[2][k]);
}

f(m + C4*h, x, mas, K[3]);

for(k=0; k<n; k++){
  x[k] = p[k] + h*(A51*K[0][k] + A52*K[1][k] + A53*K[2][k] + A54*K[3][k
      ]);
}

f(m + C5*h, x, mas, K[4]);

for(k=0; k<n; k++){
  x[k] = p[k] + h*(A61*K[0][k] + A62*K[1][k] + A63*K[2][k] + A64*K[3][k]
      + A65*K[4][k]);
}

f(m + C6*h, x, mas, K[5]);

for(k=0; k<n; k++){
  x[k] = p[k] + h*(A71*K[0][k] + A72*K[1][k] + A73*K[2][k] + A74*K[3][k]
      + A75*K[4][k] + A76*K[5][k]);
}

f(m + C7*h, x, mas, K[6]);

for(k=0; k<n; k++){
  x[k] = p[k] + h*(A81*K[0][k] + A82*K[1][k] + A83*K[2][k] + A84*K[3][k]
      + A85*K[4][k] + A86*K[5][k] + A87*K[6][k]);
}

f(m + C8*h, x, mas, K[7]);

// RK-5
for(k=0; k<n; k++){
  x[k] = p[k] + h*(B_1*K[0][k] + B_2*K[1][k] + B_3*K[2][k] + B_4*K[3][k]
      + B_5*K[4][k] + B_6*K[5][k]);
}

// RK-6
for(k=0; k<n; k++){
  y[k] = p[k] + h*(B1*K[0][k] + B2*K[1][k] + B3*K[2][k] + B4*K[3][k] +
     B5*K[4][k] + B6*K[5][k] + B7*K[6][k] + B8*K[7][k]);
}
```

```
  // Resta de solucions:
  for(k=0; k<n; k++){
    x[k] = y[k] - x[k];
  }

  // Norma de la resta:
  double suma = 0;
  for(k=0; k<n; k++){
    suma += x[k]*x[k];
  }

  suma = sqrt(suma);

  // Calcul del nou pas:

  if(control == 1){
      if(suma != 0){
          h = 0.9*h*pow(error/suma, 1./6);
      }
      if(suma <= error){
          m += h;
          for(k=0; k<n; k++){
              p[k] = y[k];
          }
      }
  }
  else{
      m += h;
      for(k=0; k<n; k++){
        p[k] = y[k];
      }
  }

  // Fi bucle
}

// Calcul del centre de masses

double M=0, r[3], v[3];

r[0]=0;
r[1]=0;
r[2]=0;
v[0]=0;
v[1]=0;
v[2]=0;

for(int i=0; i<N; i++){
  M += mas[i];
}
for(int i=0; i<3; i++){
  for(int j=0; j<N; j++){
      r[i] += mas[j]*p[(6*j)+i];
      v[i] += mas[j]*p[(6*j)+3+i];
  }
  r[i] = r[i]/M;
  v[i] = v[i]/M;
}

// Calculs dels moments angulars
```

```
for(i = 0; i < N; i++){
  moment2[0][i] = (p[(6*i) + 1] * p[(6*i) + 5]) - (p[(6*i) + 2] * p[(6*i)
      + 4]);
  moment2[1][i] = (p[(6*i) + 2] * p[(6*i) + 3]) - (p[(6*i)] * p[(6*i) +
      5]);
  moment2[2][i] = (p[(6*i)] * p[(6*i) + 4]) - (p[(6*i) + 1] * p[(6*i) +
      3]);
}

for(j = 0; j < 3; j++){
  suma_moments2[j] = 0;
}

for(i = 0; i < N; i++){
  for(j = 0; j < 3; j++){
    suma_moments2[j] += mas[i]*moment2[j][i];
  }
}

// Resta de moment:
if(suma_moments[0] != 0){
  suma_moments2[0] = (suma_moments2[0] - suma_moments[0])/suma_moments[0];
}
else{
  suma_moments2[0] = (suma_moments2[0] - suma_moments[0]);
}
if(suma_moments[1] != 0){
  suma_moments2[1] = (suma_moments2[1] - suma_moments[1])/suma_moments[1];
}
else{
  suma_moments2[1] = (suma_moments2[1] - suma_moments[1]);
}
if(suma_moments[2] != 0){
  suma_moments2[2] = (suma_moments2[2] - suma_moments[2])/suma_moments[2];
}
else{
  suma_moments2[2] = (suma_moments2[2] - suma_moments[2]);
}

// Calcul de l'energia total:

double U2=0, T2=0, E2 = 0, norma2[N][N], norm2[N];
for(j=0; j<N; j++){
  for(int k=j+1; k<N; k++){
    norma2[j][k] = 0;
    for(int i=0; i<3; i++){
      norma2[j][k] += (p[(6*j)+i] - p[(6*k)+i])*(p[(6*j)+i] - p[(6*k)+i]);
    }
    norma2[j][k] = sqrt(norma2[j][k]);
    U2 += mas[j]*mas[k]/norma2[j][k];
  }
}

for(j=0; j<N; j++){
  norm2[j] = 0;
  for(int i=0; i<3; i++){
    norm2[j] += p[(6*j)+i+3]*p[(6*j)+i+3];
  }
  norm2[j] = sqrt(norm2[j]);
  T2 += mas[j]*norm2[j]*norm2[j];
}
```

```
  T2 *= 0.5;
  E2 = T2 - U2;

  printf("Error relatiu d'energies: %le\n", (E-E2)/E);

  // Norma del moment:

  double suma = 0;

  for(i=0; i<3; i++){
    suma += suma_moments2[i]*suma_moments2[i];
  }
  suma = sqrt(suma);

  printf("Error relatiu de moments angulars: %le\n", suma);

  printf("Centre de masses al final de la integracio: %le \n", sqrt(r[0]*r
      [0] + r[1]*r[1] + r[2]*r[2]));

  fclose(puntsgrafica);

}

// Camp vectorial del problema N cossos

void f(double t, double* x, double* mas, double K[n]){
  double y[n];
  for(int i=0; i<N; i++){

    y[(6*i)] = x[(6*i)+3];
    y[(6*i)+1] = x[(6*i)+4];
    y[(6*i)+2] = x[(6*i)+5];
    y[(6*i)+3] = 0;
    y[(6*i)+4] = 0;
    y[(6*i)+5] = 0;

    for(int j=0; j<N; j++){
      if(j == i){
        continue;
      }

      // Calcul de la norma:
      double norma = 0;
      for(int k=0; k<3; k++){
        norma += (x[(6*j) + k] - x[(6*i) + k])*(x[(6*j) + k] - x[(6*i) + k])
            ;
      }
      norma = mas[j]*pow(norma, -1.5);

      y[(6*i)+3] += norma*(x[(6*j)] - x[(6*i)]);
      y[(6*i)+4] += norma*(x[(6*j) + 1] - x[(6*i) + 1]);
      y[(6*i)+5] += norma*(x[(6*j) + 2] - x[(6*i) + 2]);

    }
  }

  for(int i=0; i<n; i++){
    K[i] = y[i];
  }

}
```

```
// Metode d'Extrapolacio GBS aplicat a el Problema de N Cossos

#include <math.h>
#include <stdio.h>
#include <stdlib.h>

// Nombre de cossos del problema
#define N 11
// Dimensio del sistema d'equacions
#define n N * 6

void S(double t, double *x, double h, int k, double *mas, double sol[n]);
void extrapolationGBS_NBodyProblem(char nom_fitxer[], int control, int extra
    , int e, double error, double a, double b, double H, double x0[n],
    double mas[N], void f(double t, double *x, double *mas, double K[n]));

/*
  Fixar parametres:
  nom_fitxer -> Nom del fitxer on es guardaran les dades
  control -> 1 Si es vol fer control de pas, 0 si no07
  extra -> 1 Si es vol control del pas d'extrapolacio, 0 si no
  e -> Pas d'extrapolacio inicial
  error -> Tolerancia per a les dades proporcionades
  a -> Temps Inicial
  b -> Temps Final
  H -> Pas del metode general
  mas -> Vector de masses dels cossos
  x0 -> Condicio inicial del sistema
  f -> Camp vectorial del problema

  ATENCIO:
  Si els parametres d'error i nombre de passos d'extrapolacio no estan "
      equilibrats", el programa petara si es vol fer control de pas
  ja que si la diferencia entre els resultats amb passos e i e-1 es molt mes
       petita que l'error, llavors els passos H creixeran massa i la
      variable sobrepasara el seu limit
*/

void extrapolationGBS_NBodyProblem(char nom_fitxer[], int control, int extra
    , int e, double error, double a, double b, double H, double x0[n],
    double mas[N], void f(double t, double *x, double *mas, double K[n])) {
  long i = 0, j = 0, k = 0, l = 0;
  double m = 0, e2 = 0, resta, A = 1, W1 = 0, W2 = 0, moment1[3][N],
      suma_moments[3], moment2[3][N], suma_moments2[3];
  double p[n], T[25][25][n], s[25][n], h[25], q[n];
  FILE *puntsgrafica;
  puntsgrafica = fopen(nom_fitxer, "w");

  double step = H;

  // CONDICIONS INICIALS
  for (i = 0; i < n; i++) {
    p[i] = x0[i];
  }

  // Calcul de moments angulars

  for (i = 0; i < N; i++) {
    moment1[0][i] = (p[(6 * i) + 1] * p[(6 * i) + 5]) - (p[(6 * i) + 2] * p
```

```
      [(6 * i) + 4]);
  moment1[1][i] = (p[(6 * i) + 2] * p[(6 * i) + 3]) - (p[(6 * i)] * p[(6 *
      i) + 5]);
  moment1[2][i] = (p[(6 * i)] * p[(6 * i) + 4]) - (p[(6 * i) + 1] * p[(6 *
      i) + 3]);
}

for (int j = 0; j < 3; j++) {
  suma_moments[j] = 0;
}

for (i = 0; i < N; i++) {
  for (int j = 0; j < 3; j++) {
    suma_moments[j] += mas[i] * moment1[j][i];
  }
}

double suma1 = sqrt(suma_moments[0]*suma_moments[0] + suma_moments[1]*
    suma_moments[1] + suma_moments[2]*suma_moments[2]);

// Calcul de l'energia total:

double U1 = 0, T1 = 0, E1 = 0, norma[N][N], norm[N];
for (j = 0; j < N; j++) {
  for (int k = j + 1; k < N; k++) {
    norma[j][k] = 0;
    for (int i = 0; i < 3; i++) {
      norma[j][k] += (x0[(6 * j) + i] - x0[(6 * k) + i]) * (x0[(6 * j) + i
          ] - x0[(6 * k) + i]);
    }
    norma[j][k] = sqrt(norma[j][k]);
    U1 += mas[j] * mas[k] / norma[j][k];
  }
}

for (j = 0; j < N; j++) {
  norm[j] = 0;
  for (int i = 0; i < 3; i++) {
    norm[j] += x0[(6 * j) + i + 3] * x0[(6 * j) + i + 3];
  }
  norm[j] = sqrt(norm[j]);
  T1 += mas[j] * norm[j] * norm[j];
}
T1 *= 0.5;
E1 = T1 - U1;

// GBS Extrapolation

m = a;
while (m < b) {

      for (i = 0; i < n; i++) {
      x0[i] = p[i];
      }

  A = 1;
  j = 2;
  k = 2;
  l = 3;

  // Calcul dels valors de la funcio S:
```

```
for (int i = 0; i < e; i++) {
  if (i == 0) {
    h[i] = j;
    A += j;
    S(m, p, H / j, j, mas, s[i]);
  } else if (i % 2 == 1) {
    k = k * 2;
    h[i] = k;
    A += k;
    S(m, p, H / k, k, mas, s[i]);
  } else {
    l = l * 2;
    h[i] = l;
    A += l;
    S(m, p, H / l, l, mas, s[i]);
  }
}

for (int i = 0; i < e; i++) {
  for (int j = 0; j < n; j++) {
    T[i][0][j] = s[i][j];
  }
}

// Formula de Aitken-Neville:

for (int i = 1; i < e; i++) {
  for (int j = 1; (j < e) && (i >= j); j++) {
    for (int k = 0; k < n; k++) {
            T[i][j][k] = T[i][j - 1][k] + (T[i][j - 1][k] - T[i - 1][j -
                1][k])/((h[i] / h[i - j]) * (h[i] / h[i - j]) - 1);
      }
    }
}

resta = 0;
for (int i = 0; i < n; i++) {
  p[i] = T[e - 1][e - 1][i];
  q[i] = T[e - 2][e - 2][i];
  resta += (q[i] - p[i])*(q[i] - p[i]);
}

// Control del nombre d'avaluacions de la funcio S

e2 = e;
if (extra == 1) {
  W2 = A/H;
  if(W1 < 0.9*W2 && e > 2){
    e -= 1;
  }
  else if(W1 > 0.9*W2){
    e += 1;
  }
  W1 = W2;
}

// Control del pas d'integracio

if (control == 1) {
  resta = sqrt(resta);
  if(resta > 0){
```

```
        H = 0.9 * H * pow(error/resta, 1./(2*e2 + 1));
      }
      if(resta <= error){
        m += H;
      }
      else{
        for (i = 0; i < n; i++) {
            p[i] = x0[i];
        }
      }
    }
    else{
        m += H;
    }

}

// Calcul del centre de masses

double M=0, r[3], v[3];

r[0]=0;
r[1]=0;
r[2]=0;
v[0]=0;
v[1]=0;
v[2]=0;

for(int i=0; i<N; i++){
  M += mas[i];
}
for(int i=0; i<3; i++){
  for(int j=0; j<N; j++){
      r[i] += mas[j]*p[(6*j)+i];
      v[i] += mas[j]*p[(6*j)+3+i];
  }
  r[i] = r[i]/M;
  v[i] = v[i]/M;
}

// Calcul dels moments angulars

for (i = 0; i < N; i++) {
  moment2[0][i] = (p[(6 * i) + 1] * p[(6 * i) + 5]) - (p[(6 * i) + 2] * p
      [(6 * i) + 4]);
  moment2[1][i] = (p[(6 * i) + 2] * p[(6 * i) + 3]) - (p[(6 * i)] * p[(6 *
       i) + 5]);
  moment2[2][i] = (p[(6 * i)] * p[(6 * i) + 4]) - (p[(6 * i) + 1] * p[(6 *
       i) + 3]);
}

for (j = 0; j < 3; j++) {
  suma_moments2[j] = 0;
}

for (i = 0; i < N; i++) {
  for (j = 0; j < 3; j++) {
    suma_moments2[j] += mas[i] * moment2[j][i];
  }
}
```

```
// Resta de moment:
if (suma_moments[0] != 0) {
  suma_moments2[0] = (suma_moments2[0] - suma_moments[0]) / suma_moments
      [0];
} else {
  suma_moments2[0] = (suma_moments2[0] - suma_moments[0]);
}
if (suma_moments[1] != 0) {
  suma_moments2[1] = (suma_moments2[1] - suma_moments[1]) / suma_moments
      [1];
} else {
  suma_moments2[1] = (suma_moments2[1] - suma_moments[1]);
}
if (suma_moments[2] != 0) {
  suma_moments2[2] = (suma_moments2[2] - suma_moments[2]) / suma_moments
      [2];
} else {
  suma_moments2[2] = (suma_moments2[2] - suma_moments[2]);
}

// Calcul de l'energia total:

double U2 = 0, T2 = 0, E2 = 0, norma2[N][N], norm2[N];
for (j = 0; j < N; j++) {
  for (int k = j + 1; k < N; k++) {
    norma2[j][k] = 0;
    for (int i = 0; i < 3; i++) {
      norma2[j][k] += (p[(6 * j) + i] - p[(6 * k) + i]) * (p[(6 * j) + i]
          - p[(6 * k) + i]);
    }
    norma2[j][k] = sqrt(norma2[j][k]);
    U2 += mas[j] * mas[k] / norma2[j][k];
  }
}

for (j = 0; j < N; j++) {
  norm2[j] = 0;
  for (int i = 0; i < 3; i++) {
    norm2[j] += p[(6 * j) + i + 3] * p[(6 * j) + i + 3];
  }
  norm2[j] = sqrt(norm2[j]);
  T2 += mas[j] * norm2[j] * norm2[j];
}
T2 *= 0.5;
E2 = T2 - U2;

printf("Error relatiu d'energies: %le\n", (E1 - E2) / E1);

// Norma del moment:

double suma = 0;

for (i = 0; i < 3; i++) {
  suma += suma_moments2[i] * suma_moments2[i];
}
suma = sqrt(suma);

printf("Error relatiu de moments angulars: %le\n", suma);

printf("Centre de masses al final de la integracio: %le \n", sqrt(r[0]*r
    [0] + r[1]*r[1] + r[2]*r[2]));
```

```
  fclose(puntsgrafica);
}

void S(double t, double *x, double h, int k, double *mas, double sol[n]) {
  double s[k + 1][n], y[n];

  for (int i = 0; i < n; i++) {
    s[0][i] = x[i];
  }

  f(t, s[0], mas, y);
  for (int j = 0; j < n; j++) {
    s[1][j] = s[0][j] + h * y[j];
  }
  t = t + h;

  for (int i = 2; i < k + 1; i++) {
    f(t, s[i - 1], mas, y);
    for (int j = 0; j < n; j++) {
      s[i][j] = s[i - 2][j] + 2 * h * y[j];
    }
    t = t + h;
  }

  f(t, s[k], mas, y);
  for (int i = 0; i < n; i++) {
    y[i] = 0.5 * (s[k][i] + s[k - 1][i] + h * y[i]);
  }

  for (int i = 0; i < n; i++) {
    sol[i] = y[i];
  }

}
```