Radau5 : résolution d'ODE raides

Violaine Louvet, Marc Massot

22 octobre 2018

Table des matières

1 Equations Différentielles Ordinaires, rappels

On considère le problème vectoriel suivant :

$$y' = f(t, y),\ y(t_0) = y_0 \tag{1}$$

1.1 Méthodes de Runge-Kutta

On considère une subdivision $t_0 < t_1 < \ldots < t_N = T$ de l'intervalle $[t_0, T]$, et on note $h_n = t_{n+1} - t_n$. On calcule l'approximation $y_n \approx y(t_n)$:

$$y_{n+1} = y_n + h_n \phi(t_n, y_n, h_n) \tag{2}$$

La plupart des méthodes numériques à un pas s'obtiennent à partir de l'intégration de (1) sur $[t_0, t_0 + h]$:

$$y(t_0 + h) = y_0 + \int_{t_0}^{t_0+h} f(\tau, y(\tau)) d\tau \tag{3}$$

pour laquelle on approche l'intégrale par une formule de quadrature d'ordre élevé.

Une méthode de Runge-Kutta à s étages est ainsi donnée par :

$$\begin{array}{rcc} k_1 & = & f(t_0, y_0) \\ k_2 & = & f(t_0 + c_2 h, y_0 + h a_{21} k_1) \\ k_3 & = & f(t_0 + c_3 h, y_0 + h(a_{31} k_1 + a_{32} k_2)) \\ & \ldots & \\ k_s & = & f(t_0 + c_s h, y_0 + h(a_{s1} k_1 + \cdots + a_{ss-1} k_{s-1})) \\ y_1 & = & y_0 + h(b_1 k_1 + \cdots + b_s k_s) \end{array} \tag{4}$$

où c_i, a_{ij}, b_j sont des coefficients.

D'une manière générale, on suppose que les c_i satisfont :

$$c_1 = 0,\ c_i = \sum_{j=1}^{i-1} a_{ij},\ i = 2, \ldots, s \tag{5}$$

2 Méthodes multi-pas

Ces méthodes utilisent les approximations successives $y_n, y_{n-1}, ...y_{n-k+1}$ pour obtenir une approximation plus précise de y_{n+1}.

2.1 Méthodes d'Adams explicites

On intègre l'équation (1) entre t_n et t_{n+1} :

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(\tau, y(\tau)) d\tau \tag{6}$$

pour laquelle on remplace $f(\tau, y(\tau))$ par le polynôme de degré $k-1$ qui vérifie :

$$p(t_j) = f_j = f(t_j, y(t_j)) \text{ pour } \ j = n, n-1, \ldots, n-k+1 \tag{7}$$

Ce polynôme peut s'exprimer en termes des différences régressives de f :

$$\nabla^0 f_n = f_n,\ \nabla^{j+1} f_n = \nabla^j f_n - \nabla^j f_{n-1} \tag{8}$$

et on a donc :

$$p(t) = p(x_n + sh) = \sum_{j=0}^{k-1} (-1)^j \begin{pmatrix} -s \\ j \end{pmatrix} \nabla^j f_n \tag{9}$$

On obtient ainsi l'approximation :

$$y_{n+1} = y_n + h \sum_{j=0}^{k-1} \gamma_j \nabla^j f_n \tag{10}$$

avec :

$$\gamma_j = (-1)^j \int_0^1 \begin{pmatrix} -s \\ j \end{pmatrix} ds \tag{11}$$

2.2 Méthodes d'Adams implicites

Il faut noter que la formule (10) est obtenue par intégration du polynôme p entre x_n et x_{n+1} c'est-à -dire en dehors de l'intervalle d'interpolation $[x_{n-k+1}, x_n]$. Ceci peut entraîner des erreurs importantes. Pour éviter ces erreurs, l'idée est donc de remplacer le polynôme p par un polynôme p^* de degré k satisfaisant :

$$p^*(t_j) = f_j = f(t_j, y(t_j)) \text{ pour } \ j = n+1, n, n-1, \ldots, n-k+1 \tag{12}$$

Dans ce cas, f_{n+1} est encore inconnue.
On obtient ainsi la méthode implicite suivante :

$$y_{n+1} = y_n + h \sum_{j=0}^{k} \gamma_j^* \nabla^j f_{n+1} \tag{13}$$

avec :

$$\gamma_j^* = (-1)^j \int_0^1 \begin{pmatrix} -s+1 \\ j \end{pmatrix} ds \tag{14}$$

2.3 Méthodes BDF, Backward Differentiation formulas

Au lieu de remplacer $f(t, y(t))$ par un polynôme d'interpolation $p(t)$, on va s'attacher maintenant à considérer le polynôme $q(t)$ de degré k défini par :

$$q(t_j) = y_j \text{ pour } \ j = n+1, n, n-1, \ldots, n-k+1 \tag{15}$$

et on détermine y_{n+1} de la façon suivante :

$$q'(t_{n+1}) = f(t_{n+1}, q(t_{n+1})) \tag{16}$$

On obtient ainsi les formules implicites :

$$\sum_{j=1}^{k} \frac{1}{j} \nabla^j y_{n+1} = h f_{n+1} \tag{17}$$

Il faut noter que pour $k > 6$, les méthodes BDF sont instables.

3 Cas des EDO raides : méthode de Runge-Kutta implicites

3.1 A-stabilité

Pour comprendre la raison pour laquelle les problèmes raides posent des difficultés de résolution aux méthodes précédentes, il faut introduire la notion de A-stabilité des méthodes.

On considère pour cela le problème test de Dahlquist :

$$y' = \lambda y \tag{18}$$

Sa solution exacte est

$$y(t) = e^{\lambda t} C \tag{19}$$

et elle reste bornée pour $t \geq 0$ si $\Re\lambda \leq 0$ ($\Re$ désigne la partie réelle). La solution numérique d'une méthode de RungeKutta ou d'une méthode multipas, appliquée avec des pas constants au problème (19), ne dépend que du produit $h\lambda$. Il est alors intéressant d'étudier pour quelle valeur de $h\lambda$ la solution numérique reste bornée. On obtient ainsi la définition de la A-stabilité :

Définition 1 (A-stabilité) *Considérons une méthode dont la solution numérique $y_{n n\geq 0}$ pour l'équation de test 19 est une fonction de $z = h\lambda$. Alors l'ensemble*

$$S := \{z \in \mathbb{C}; \{y_n\}_{n\geq 0} \ \ est\ born\acute{e}e\} \tag{20}$$

s'appelle domaine de stabilité de la méthode. On dit que la méthode est A-stable si

$$S \supset \mathbb{C}^- \quad o\grave{u} \ \ \mathbb{C}^- = \{z \in \mathbb{C}; \Re z \leq 0\} \tag{21}$$

Dans le cas des méthodes à un pas, le domaine de stabilité S est borné et la condition de stabilité $h\lambda \in S$ impose une restriction sévère à h. Ces méthodes ne sont donc pas appropriées pour les problèmes raides.

Les méthodes d'Adams explicites et implicites (à l'exception de la méthode d'Euler implicite et de la règle du trapèze) ont toutes un domaine de stabilité borné et petit. Ces méthodes ne sont donc pas utilisables pour des problèmes raides.

Les méthodes BDF, par contre, ont un domaine de stabilité plus important, ce qui explique qu'elles sont beaucoup utilisées pour résoudre des problèmes raides. La méthode BDF avec $k = 2$ est même A-stable.

La *barrière de Dahlquist* dit que l'ordre d'une méthode multipas A-stable ne peut être plus grand que 2. Ce résultat a contribué à la détermination d'autre méthodes d'intégration permettant de combiner A-stabilité et ordre élevé.

3.2 Méthodes de Runge-Kutta implicites

Comme on l'a introduit en 1.1, les méthodes de Runge-Kutta peuvent s'écrire de façon générale sous la forme suivante.

Définition 2 (Méthode de Runge-Kutta à s étages) *Soient b_i, a_{ij} $(i, j = 1 \ldots, s)$ des réels. La méthode de Runge-Kutta à s étages s'écrit :*

$$\begin{array}{rcl} k_i & = & f(x_0 + c_i h, y_0 + h \sum_{j=1}^{s} a_{ij} k_j \; i = 1, \ldots, s \\ y_1 & = & y_0 + h \sum_{i=1}^{s} b_i k_i \end{array} \tag{22}$$

Quand $a_{ij} = 0$ pour $i \geq j$, la méthode est explicite. Si $a_{ij} = 0$ pour $i > j$, et qu'au moins un des $a_{ii} \neq 0$, la méthode est dite diagonalement implicite (diagonal implicit RK, DIRK). Si de plus tous les termes diagonaux sont égaux $a_{ii} = \gamma$ pour $i = 1, \ldots, s$, on parle de "singly diagonal implicit method" (SDIRK). Dans tous les autres cas, la méthode est implicite.

Contrairement au cas des méthodes explicites, pour les méthodes implicites, les k_i ne peuvent plus être évalué de façon successives, et le système (22) constitue un système d'équations implicites pour la détermination des k_i.

Dans le cas des méthodes totalement implicites, ce système comprend sn inconnues (chaque k_i de dimension respective n chacun).

Certaines de ces méthodes de Runge Kutta implicites possèdent d'excellentes conditions de stabilité sous les hypothèses suivantes :

$$\begin{array}{lll} B(p): & \sum_{i=1}^{s} b_i c_i^{q-1} = \dfrac{1}{q} & q = 1, \ldots, p \\ C(\eta): & \sum_{j=1}^{s} a_{ij} c_j^{q-1} = \dfrac{c_i^q}{q} & i = 1, \ldots, s, \; q = 1, \ldots, \eta \\ D(\zeta): & \sum_{i=1}^{s} b_i c_i^{q-1} a_{ij} = \dfrac{b_j}{q}(1 - c_j^q) & j = 1, \ldots, s, \; q = 1, \ldots, \zeta \end{array} \tag{23}$$

La condition $B(p)$ signifie simplement que la formule de quadrature (b_i, c_i) est d'ordre p.

On a alors le théorème suivant :

Théorème 1 (Butcher) *Si les coefficients b_i, c_i, et a_{ij} de la méthode de RK vérifie les hypothèses $B(p)$, $C(\eta)$, et $D(\zeta)$ avec $p \leq \eta + \zeta + 1$ et $p \leq 2\eta + 2$ alors la méthode est d'ordre p.*

Butcher introduit des méthodes de Runge-Kutta implicite basée sur les formules de quadrature de Radau et Lobatto. Elles sont de type I, II ou III selon que les $c_1, \ldots, c_s$ sont les zéros des fonctions :

$$\begin{array}{lll} I: & \dfrac{d^{s-1}}{dx^{s-1}}(x^s(x-1)^{s-1}), & \text{Radau à gauche} \\ II: & \dfrac{d^{s-1}}{dx^{s-1}}(x^{s-1}(x-1)^s), & \text{Radau à droite} \\ III: & \dfrac{d^{s-2}}{dx^{s-2}}(x^{s-1}(x-1)^{s-1}), & \text{Lobatto} \end{array} \tag{24}$$

Cependant, ces méthodes ne sont pas A-stables. Ehle reprend les idées de Butcher et construit des méthodes de type I, II et III avec d'excellentes conditions de stabilité.

La méthode d'Ehle de type II, appelée méthode de Radau IIA, s'obtient par application de la condition $C(s)$. On peut montrer qu'elle constitue ainsi une méthode de collocation basé sur les zéros de la fonction définie en (24). Elle est d'ordre $2s-1$ et est A-stable.

C'est cette méthode, pour les valeurs $s = 3$ et $p = 5$, qui est implémentée dans le code radau5.

4 Le programme RADAU5

On a vu que la résolution du système d'équations différentielles par une méthode de RK implicite conduit à la résolution d'un système non-linéaire de taille ns pour les inconnues $k_1, \ldots, k_s$.

Une résolution efficace de ces systèmes est l'un des problème majeur de l'implémentation des méthodes de RadauIIA.

4.1 Reformulation du système non linéaire

La méthode peut se réécrire sous la forme :

$$\begin{array}{rcl} g_i & = & y_0 + h\sum_{j=1}^{s} a_{ij} f(x_0 + c_j h, g_j) \quad i = 1, \ldots, s \\ y_1 & = & y_0 + h\sum_{j=1}^{s} b_j f(x_0 + c_j h, g_j) \end{array} \tag{25}$$

On choisit de travailler avec des quantités plus petites afin de réduire les problèmes d'erreurs d'arrondis :

$$z_i = g_i - y_0 \tag{26}$$

ce qui permet de réécrire la première équation de (25) :

$$z_i = h\sum_{j=1}^{s} a_{ij} f(x_0 + c_j h, y_0 + z_j) \quad i = 1, \ldots, s \tag{27}$$

Ainsi, lorsque les variables $z_1, \ldots, z_s$ sont connues, la deuxième équation de (25) devient explicite pour y_1. Une application directe de ce raisonnement nécessite s évaluations de fonctions additionnelles.

On peut éviter ces calculs dans le cas où la matrice $A = (a_{ij})$ n'est pas singulière. On peut alors réécrire (27) sous la forme :

$$\begin{pmatrix} z_1 \\ \vdots \\ z_s \end{pmatrix} = A \begin{pmatrix} hf(x_0 + c_1 h, y_0 + z_1) \\ \vdots \\ hf(x_0 + c_s h, y_0 + z_s) \end{pmatrix}$$

Et ainsi

$$y_1 = y_0 + \sum_{i=1}^{s} d_i z_i \tag{28}$$

avec

$$(d_1, \ldots, d_s) = (b_1, \ldots, b_s) A^{-1} \tag{29}$$

Dans le cas qui nous intéresse, c'est-à -dire $s = 3$, $d = (0, 0, 1)$, puisque $b_i = a_{si}\ \forall i$.

4.2 Itérations de Newton simplifiées

La résolution du système non linéaire (27) se fait de façon itérative par une méthode de Newton, qui conduit à chaque itération à l'inversion de :

$$\begin{pmatrix} I - ha_{11}\dfrac{\partial f}{\partial y}(x_0 + c_1 h, y_0 + z_1) & \cdots & -ha_{1s}\dfrac{\partial f}{\partial y}(x_0 + c_s h, y_0 + z_s) \\ \vdots & & \vdots \\ -ha_{s1}\dfrac{\partial f}{\partial y}(x_0 + c_1 h, y_0 + z_1) & \cdots & I - ha_{ss}\dfrac{\partial f}{\partial y}(x_0 + c_s h, y_0 + z_s) \end{pmatrix}$$

Si on remplace le jacobien par une approximation :

$$J \approx \frac{\partial f}{\partial y}(x_0, y_0)$$

les itérations de Newton simplifiées deviennent :

$$\begin{aligned} (I - hA \otimes J)\Delta Z^k &= -Z^k + h(A \otimes I)F(Z^k) \\ Z^{k+1} &= Z^k + \Delta Z^k \end{aligned} \tag{30}$$

où $Z^k = (z_1^k, \ldots, z_s^k)^T$ est la $k$ème approximation de la solution et $\Delta Z^k = (\Delta z_1^k, \ldots, \Delta z_s^k)^T$ sont les incréments, $F(Z^k) = (f(x_0 + c_1 h, y_0 + z_1^k), \ldots, f(x_0 + c_s h, y_0 + z_s^k))^T$.

On peut constater que chaque itération nécessite s évaluations de f, et la résolution d'un système linéaire de taille ns. Par contre, la matrice $(I - hA \otimes J)$ est la même pour toutes les itérations. Sa décomposition LU n'est donc réalisée qu'une seule fois.

4.2.1 Conditions initiales pour la méthode de Newton

Comme la solution exacte de (27) vérifie $z_i = \mathcal{O}(h)$, le choix le plus simple est :

$$z_i^0 = 0 \; i = 1, \ldots, s \tag{31}$$

On peut faire des choix plus efficaces. Si q est le polynôme d'interpolation de degré s définit par :

$$q(0) = 0, \; q(c_i) = z_i, \; i = 1 \ldots, s$$

alors la condition initiale :

$$\begin{array}{rcl} z_i^0 & = & q(1 + wc_i) + y_0 - y_1 \; i = 1, \ldots, s \\ w & = & \dfrac{h_{new}}{h_{old}} \end{array} \tag{32}$$

donne une convergence numérique plus rapide que la précédente (31).

4.2.2 Estimation d'erreur et critère d'arrêt des itérations de Newton

Comme la convergence est linéaire, on a :

$$\|\Delta Z^{k+1}\| \leq \Theta \|\Delta Z^k\|$$

en espérant que $\Theta < 1$.

Si on applique l'inégalité triangulaire au développement (Z est la solution exacte de (27)) $Z^{k+1} - Z^* = (Z^{k+1} - Z^{k+2}) + (Z^{k+2} - Z^{k+3}) + \ldots$, on obtient l'estimation d'erreur :

$$\|Z^{k+1} - Z^*\| \leq \frac{\Theta}{1 - \Theta} \|\Delta Z^k\|$$

Le taux de convergence peut être estimé à partir de quantités calculées :

$$\Theta_k = \frac{\|\Delta Z^k\|}{\|\Delta Z^{k-1}\|}$$

L'erreur numérique ne doit pas être supérieure à l'erreur de discrétisation locale proche de Tol. Les itérations sont donc arrêtées quand :

$$\eta_k \|\Delta Z^k\| \leq \kappa . Tol \text{ avec } \eta_k = \frac{\Theta_k}{1 - \Theta_k} \tag{33}$$

Cela ne peut s'appliquer qu'après 2 itérations de la méthode. Pour pouvoir arrêter le calcul après une itération, on choisit pour $k = 0$:

$$\eta_0 = (\max(\eta_{old}, Uround))^{0.8}$$

où η_{old} correspond au dernier η_k du pas précédent.

Il reste encore à choisir le paramètre κ dans (33). Des expériences numériques montrent une meilleure efficacité du code pour des valeurs de κ autour de 10^{-1} ou 10^{-2}.

De même, il semble que le code soit plus efficace quand le nombre maximum d'itérations k_{max} est élevé ($k_{max} = 7$ ou 10). Durant ces itérations, le calcul est interrompu et redémarré avec un pas plus petit (par exemple avec $h := h/2$) dans le cas où on se trouve dans une des situations suivantes :

- $\Theta_k \geq 1$ pour un k,
- Pour quelques k,

$$\frac{\Theta_k^{k_{max}-k}}{1-\Theta_k}\|\Delta Z^k\| > \kappa.Tol$$

Si une seule itération de Newton est suffisante pour satisfaire (33) ou si le dernier $\Theta_k \leq 10^{-3}$, le jacobien n'est pas recalculé au pas suivant. Ainsi, pour des problèmes linéaires à coefficients constants, le jacobien n'est calculé qu'une seule fois, tant qu'aucun pas n'est rejeté.

4.3 Résolution du système linéaire

La résolution de (30) se fait en exploitant la structure particulière de la matrice $I - hA \otimes J$. L'idée est de multiplier (30) par $(hA)^{-1} \otimes I$ (en supposant que A est inversible) et de transformer A^{-1} en une matrice simple (diagonal par bloc, triangulaire ...) :

$$T^{-1}A^{-1}T = \Lambda$$

Si on considère le changement de variables $W^k = (T^{-1} \otimes I)Z^k$, (30) se réécrit :

$$\begin{aligned} &(h^{-1}\Lambda \otimes I - I \otimes J)\Delta W^k = -h^{-1}(\Lambda \otimes I)W^k + (T^{-1} \otimes I)F((T \otimes I)W^k) \\ &W^{k+1} = W^k + \Delta W^k \end{aligned} \tag{34}$$

Si on suppose que A^{-1} a une valeur propre réel $\hat{\gamma}$ et une paire de valeurs propres complexes conjuguées $\hat{\alpha} \pm i\hat{\beta}$ (ce qui est le cas pour Radau IIA), la matrice de (34) se réécrit :

$$\begin{pmatrix} \gamma I - J & 0 & -\beta I \\ 0 & \alpha I - J & 0 \\ 0 & \beta I & \alpha I - J \end{pmatrix} \tag{35}$$

avec $\gamma = h^{-1}\hat{\gamma}$, $\alpha = h^{-1}\hat{\alpha}$, $\beta = h^{-1}\hat{\beta}$.

Ainsi, le système (34) peut se décomposer en deux systèmes linéaires de taille n et $2n$.

D'autres idée sont possibles pour exploiter la structure particulière de la matrice $2n \times 2n$. La plus efficace semble être de transformer le système réel de dimension $2n$ en un système complexe de taille n :

$$((\alpha + i\beta)I - J)(u + iv) = a + ib$$

et lui appliquer un simple pivot de Gauss.

Remarque Pour les très grands systèmes avec une matrice jacobienne pleine, on peut gagner en temps de calcul en la transformant sous la forme d'une matrice de Hessenberg. Cette tranformation est d'autant plus avantageuse si le jacobien ne change pas durant plusieurs pas.

4.4 Contrôle du pas

Comme la méthode est d'ordre optimale, on ne peut pas se baser sur une autre plus précise pour estimer l'erreur. Donc, on considère une méthode d'ordre moins élevé de la forme :

$$\hat{y_1} = y_0 + h(\hat{b_0} f(x_0, y_0) + \sum_{i=1}^{3} \hat{b_i} f(x_0 + c_i h, g_i))$$

où les g_1, g_2, g_3 sont les valeurs obtenues pour la méthode de Radau IIA, et $\hat{b_0} \neq 0$.

On peut alors écrire la différence entre les solutions des deux méthodes (on fait le choix de $\hat{b_0} = \gamma_0 = \hat{\gamma^{-1}}$) :

$$\begin{aligned} \hat{y_1} - y_1 &= \gamma_0 h f(x_0, y_0) + \sum_{i=1}^{3} (\hat{b_i} - b_i) h f(x_0 + c_i h, g_i) \\ &= \gamma_0 h f(x_0, y_0) + e_1 z_1 + e_2 z_2 + e_3 z_3 \end{aligned} \tag{36}$$

dont on se sert pour l'évaluation de l'erreur :

$$err = (I - h\gamma_0 J)^{-1} (\hat{y_1} - y_1) \tag{37}$$

Dans le cas du premier pas, et après chaque pas rejeté pour lequels $\|err\| > 1$, l'erreur est calculé par :

$$\widetilde{err} = (I - h\gamma_0 J)^{-1} (\gamma_0 h f(x_0, y_0 + err) + e_1 z_1 + e_2 z_2 + e_3 z_3) \tag{38}$$

4.4.1 Contrôle standard

Les expressions (37) et (38) se comporte en $\mathcal{O}(h^4)$, la prédiction du pas s'écrit donc :

$$h_{new} = fac.h_{old}.\|err\|^{-1/4} \tag{39}$$

avec

$$\|err\| = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (\frac{err_i}{sc_i})^2}$$

et $sc_i = Atol_i + \max(|y_{0i}|, |y_{1i}|).Rtol_i$. Le coefficient fac dépend du nombre d'itérations de Newton $Newt$ de la façon suivante :

$$fac = 0.9 \times (2k_{max} + 1)/(2k_{max} + Newt)$$

Pour limiter le nombre de décomposition LU de la matrice (35), si le jacobien n'est pas recalculé et si $c_1 h_{old} < h_{new} < c_2 h_{old}$ avec $c_1 = 1.0$ et $c_2 = 1.2$, alors on conserve h_{old} pour le pas suivant.

4.4.2 Contrôle prédictif

Dans le cas de systèmes très raides, la décroissance du pas de temps peut devoir être extrêmement rapide alors que la précédente est limitée par le coefficient fac.

Si on note err_{n+1} l'erreur donné par (38) ou (37) et si h_n correspond au pas du calcul de l'étape n, la prédiction du pas est issue de la formule asymptotique :

$$\|err_{n+1}\| = C_n h_n^4 \tag{40}$$

L'évaluation de h_{new} dans (39) est basé sur l'hypothèse que $C_{n+1} \approx C_n$. Un meilleur modèle peut être obtenu en considérant que $\log C_n$ est une fonction linéaire de n, et donc que $\log C_{n+1} - \log C_n$ est constant ou encore :

$$C_{n+1}/C_n \approx C_n/C_{n-1}$$

On obtient ainsi une nouvelle estimation du pas :

$$h_{new} = fac.h_n(\frac{1}{\|err_{n+1}\|})^{1/4}.\frac{h_n}{h_{n+1}}(\frac{\|err_n\|}{\|err_{n+1}\|})^{1/4} \tag{41}$$

Dans le code, le nouveau pas conrrespond au minimum des résultats de (39) et (41).

4.5 Utilisation du code

On considère ici le cas d'un système explicite s'écrivant sous la forme générique :

$$Y' = F(X, Y)$$

La déclaration d'appel de la routine de résolution est :

```
SUBROUTINE RADAU5(N,FCN,X,Y,XEND,H,
                  RTOL,ATOL,ITOL,
                  JAC ,IJAC,MLJAC,MUJAC,
                  MAS ,IMAS,MLMAS,MUMAS,
                  SOLOUT,IOUT,
                  WORK,LWORK,IWORK,LIWORK,RPAR,IPAR,IDID)
```

4.5.1 Paramètres d'entrée généraux

n Taille du système

FCN routine externe pour le calcul de $F(X, Y)$. La déclaration de cette routine doit être la suivante :

```
SUBROUTINE FCN(N,X,Y,F,RPAR,IPAR)
   DOUBLE PRECISION X,Y(N),F(N)
   DOUBLE PRECISION RPAR(*)
   INTEGER IPAR(*)
```

X Valeur initiale pour la variable X

Y(N) Valeurs initiales pour l'inconnu Y

XEND Valeur finale pour laquelle on cherche à évaluer Y

H taille initiale du pas. Par défaut, si $H = 0$, le code prend $H = 10^-6$.

RTOL, ATOL, ITOL tolérances relatives et absolues, qui peuvent être scalaires (cas $ITOL = 0$) ou vectorielles de taille n (cas $ITOL = 1$). Le code vérifie que l'erreur locale sur $Y(I)$ est bien majorée par $RTOL \times |Y(I)) + ATOL|$.

JAC, IJAC routine externe pour le calcul du jacobien qui est appelée lorsque $IJAC = 1$. Quand $IJAC = 0$, le jacobien est évalué numériquement. La déclaration de la routine JAC doit avoir la structure suivante :

```
SUBROUTINE JAC(N,X,Y,DFY,LDFY,RPAR,IPAR)
   DOUBLE PRECISION X,Y(N),DFY(LDFY,N)
```

MLJAC, MUJAC : définissent la structure bande du jacobien. Si $MLJAC = n$, la matrice jacobienne est pleine et l'algèbre linéaire est réalisée par pivot de Gauss. Si $0 \leq MLJAC < n$, il définit la largeur de bande inférieure de la matrice. $MUJAC$ correspond à la largeur de bande supérieure du jacobien.

MAS, IMAS, MLMAS, MUMAS ont les mêmes significations que les données précédentes pour le cas où le problème à traiter comporte une matrice de masse.

SOLOUT, IOUT est le nom de la routine externe qui permet de récupérer la solution au cours du calcul. Si $IOUT = 1$, cette routine est appelée après chaque pas réussi. Sa déclaration doit être la suivante :

```
SUBROUTINE SOLOUT (NR,XOLD,X,Y,CONT,LRC,N,
   RPAR,IPAR,IRTRN)
   DOUBLE PRECISION X,Y(N),CONT(LRC)
```

Elle fournit la valeur de la solution Y sur le NRième point de la grille des X.

WORK, IWORK, LWORK, LIWORK tableaux de travail permettant d'affiner le passage de certains paramètres. Ceux-ci sont détaillés ci-dessous.

4.5.2 Paramètres d'entrées avancés

La taille des tableaux de travail varie selon le cas à traiter. Le tableau de doubles $WORK$ doit être de taille $LWORK$ et le tableau d'entiers $IWORK$ doit être de taille $LIWORK$.

— $LWORK \geq n \times (LJAC + LMAS + 3 \times LE + 12) + 20$ avec
  - • $LJAC = N$ et $LE = N$ si $MLJAC = n$ (matrice jacobienne pleine)
    • $LJAC = MLJAC + MUJAC + 1$ et $LE = 2 \times MLJAC + MUJAC + 1$ si $MLJAC < N$ (structure bande du jacobien)
  - $LMAS = 0$ si $IMAS = 0$ (cas considéré ici).
- $LIWORK \geq 3 \times N + 20$

Les tableaux $RPAR$ et $IPAR$ peuvent être utilisés pour passer des données entre le programme et les routines appelées par *RADAU5*.

La plupart des paramètres sont définis par défaut en imposant nulles les 20 premières entrées de $WORK$ et de $IWORK$. On peut affiner ces données en leur affectant des valeurs non nulles :

IWORK(1) Si $IWORK(1) = 1$, le code transforme le jacobien en matrice de Hessenberg (uniquement sur des matrices jacobiennes pleines)

IWORK(2) Nombre maximal de pas. La valeur par défaut est 10000.

IWORK(3) Nombre maximal d'itérations de Newton à chaque pas. La valeur par défaut est 7.

IWORK(4) Si $IWORK(4) = 0$, les conditions initiales considérées correspondent à (32), sinon elles sont prises vérifiant (31). Dans le cas où la méthode de Newton a du mal à converger, cette dernière solution est préférable. La valeur par défaut est $IWORK(4) = 0$.

IWORK(5,6,7) permettent de définir les dimensions du système dans le cas où il est d'index supérieur à 1.

IWORK(8) permet de définir la stratégie de contrôle du pas. Si $IWORK(8) = 1$, la stratégie décrite en 4.4.2 est appliquée, et si $IWORK(8) = 2$, c'est celle décrite en 4.4.1. Le choix par défaut est $IWORK(8) = 1$.

WORK(1) arrondi, valeur par défaut 10^{-16}.

WORK(2) correspond au facteur fac dans les équations (39) et (41). Par défaut, $fac = 0.9$.

WORK(3) définit si le jacobien doit être recalculé ou non.
— Lorsque le calcul du jacobien est coà»teux, cette valeur peut être augmentée (jusqu'à 0.1 par exemple).
— Pour de petits systèmes, elle peut être diminuée (0.001 par exemple).
— Lorsqu'elle est négative, le jacobien est réévalué à chaque étape.
La valeur par défaut est 0.001.

WORK(4) définit le critère d'arrêt pour la méthode de Newton, choisi normalement < 1. De petites valeurs de $WORK(4)$ rendent le code plus long mais plus sà»r. La valeur par défaut est $\min(0.03, \sqrt{RTOL})$.

WORK(5) et WORK(6) Si $WORK(5) < h_{new}/h_{old} < WORK(6)$, alors le pas reste inchangé. Cela permet, lorsque $WORK(3)$ est suffisament grand, d'éviter le recalcul de la décomposition LU. Pour de petits systèmes, on peut choisir $WORK(5) = 1$ et $WORK(6) = 1,2$, tandis que pour de grands sytèmes, il est préférable de prendre $WORK(5) = 0,99$ et $WORK(6) = 2$. Les valeurs par défaut sont $WORK(5) = 1$ et $WORK(6) = 1,2$.

WORK(7) Taille maximale du pas, par défaut $XEND - X$.

WORK(8), WORK(9) permettent de restreindre le nouveau pas qui est choisit de telle sorte que $WORK(8) \leq h_{new}/h_{old} \leq WORK(9)$. Par défaut, $WORK(8) = 0,2$ et $WORK(9) = 8$.

4.5.3 Paramètres de sortie

X correspond à la valeur de X pour laquelle la solution a été calculée. Si tout s'est bien passé, $X = XEND$.

Y(N) correspond à la solution calculée

H correspond au dernier pas calculé

IDID est un indice de réussite du calcul :
- $IDID = 1$ succès
- $IDID = 2$ succès (interrompu par un appel à Solout)
- $IDID = -1$ les entrées ne sont pas consistantes
- $IDID = -2$ $NMAX$ (nombre maximum d'itérations de Newton) doit être plus grand
- $IDID = -3$ le pas devient trop petit
- $IDID = -4$ la matrice est singulière de façon répétée.

IWORK(14)=NFCN nombre d'évaluations de la fonction (sans compter ceux qui ont servi au calcul du jacobien numérique)

IWORK(15)=NJAC nombre d'évaluation du jacobien

IWORK(16)=NSTEP nombre de pas calculés

IWORK(17)=NACCPT nombre de pas acceptés

IWORK(18)=NREJCT nombre de pas rejetés à cause du test d'erreur

IWORK(19)=NDEC nombre de décomposition LU des deux matrices

IWORK(20)=NSOL nombre de Forward/Backward substitutions des deux systèmes