Beschreibung des mex-Interface zu radau5.f von Hairer und Wanner

von C. Ludwig

Inhaltsverzeichnis

1 Problemstellung

In diesem File wird die MATLAB-Anbindung des Integrators radau5.f von Hairer und Wanner beschrieben. Die Fortran-Files sind unter

`http://www.unige.ch/math/folks/hairer/prog/stiff.tar.gz`

erhältlich. Zusätzlich zu den Informationen im File radau5.f wird der Integrator auch in E. Hairer, G. Wanner: Solving Ordinary Differential Equations II beschrieben.

Es wurde bei der MATLAB-Anbindung darauf geachtet, möglichst alle features von radau5.f zu unterstützen. Insbesondere die Unterstützung von Matrizen mit Bandstrukturen wird ausführlich behandelt.

Betrachtet werden Differentialgleichungssysteme 1. Ordnung der Form

$$Mx' = f(t, x).$$

Dabei ist $x \in \mathbb{R}^d$ der Zustandsvektor und $M \in \mathbb{R}^{d \times d}$ eine konstante Massenmatrix. Im Fall $M = I$ heißt das System explizit, ansonsten implizit.

2 Aufruf bei allgemeinen Problemen

2.1 Ein- und Ausgabeargumente

Man kann radau5Mex mit 3 oder 4 Eingabeargumente und mit 2, 3 oder 4 Ausgabeargumente aufrufen:

```
[tGitter,xGitter,stats,taupred]=radau5Mex(f,t,x0,opt)
```

Dabei ist `f` ein String mit dem Funktionsnamen für die rechte Seite mit folgender Form:

```
function dx=f(t,x)
```

Bei `t` handelt es sich um einen Zeilenvektor mit den auszuwertenden Zeitpunkten. Er muss mindestens die Länge 2 (einen Start- und einen Endwert) haben. Bei mehr als zwei Werten wird auf dense output umgeschaltet. `x0` ist ein Spaltenvektor der Länge `d` mit den den Startwerten der Zustände. Das Argument `opt` ist optional. Falls vorhanden, muss es eine struct sein mit Optionen. Die möglichen Optionen werden in einem eigenen Abschnitt behandelt.

In `tGitter` befinden sich die Zeitpunkte, an denen die Lösung approximiert wurde. Falls `t` nur zwei Werte enthält, dann sind in `tGitter` diese zwei Werte und auch noch die Zwischenstellen, die sich der Integrator radau5 automatisch wählt. Enthält `t` mehr als zwei Werte, so hängt es von der Option `IncludeGridPointsInDenseOutput` ab, was in `tGitter` zu finden ist. Ist `IncludeGridPointsInDenseOutput=0`, so sind in `tGitter` genau die Punkte aus `t`. Ist `IncludeGridPointsInDenseOutput!=0`, so sind in `tGitter` zusätzlich zu den Werten aus `t` noch die von radau5 automatisch gewählten Hilfszeitpunkte.

In `xGitter` befinden sich die approximierten Zustände zu den jeweiligen Zeitpunkten in `tGitter`.

Im optionalen Argument `stats` befinden sich einige statistische Angaben. In der 1. Komponente ist der Rückgabewert `IDID` von radau5. Hier ein Auszug der möglichen Rückgabewerte von radau5:

1	erfolgreiche Integration
2	erfolgreiche Integration (vorzeitiger Abbruch durch OutputFunction)
< 0	erfolglose Integration
-1	inkonsistente Eingaben
-2	zu wenig Speicher (sollte mit dem mex-Interface nicht auftreten)
-3	Schrittweite wurde zu klein
-4	Matrix wiederholt singulär

In der 2. Komponente steht die Anzahl der f-Auswertungen, in der 3. Komponente die Anzahl der Jacobi-Auswertungen und in der 4./5. bzw. 6. Komponente die Anzahl der berechneten/akzeptierten bzw. verworfenen Schritte. In der 7. Komponente die Anzahl der benötigten LR-Zerlegungen und in der letzten Komponente die Anzahl der Vorwärts- und Rückwärtssubstitutionen.

Im optionalen Argument `taupred` wird der letzte Schrittweitenvorschlag gespeichert.

2.2 Optionen

2.2.1 RelTol

Hier kann die relative Toleranz angegeben werden. Entweder sind `RelTol` und `AbsTol` beide Skalare oder beide $(d, 1)$-double Vektoren für die einzelnen Zustands-Komponenten. Der Standardwert ist $1 \cdot 10^{-3}$. Bei radau5 heißt dieser Parameter `RTOL`.

2.2.2 AbsTol

Hier kann die absolute Toleranz angegeben werden. Entweder sind `RelTol` und `AbsTol` beide Skalare oder beide $(d, 1)$-double Vektoren für die einzelnen Zustands-Komponenten. Der Standardwert ist $1 \cdot 10^{-6}$. Bei radau5 heißt dieser Parameter `ATOL`.

2.2.3 eps

Hier kann man die Maschinengenauigkeit des verwendeten Rechners einstellen. Standardwert ist $1 \cdot 10^{-16}$. Bei radau5 heißt dieser Parameter `WORK(1)`.

2.2.4 rho

Ist der Sicherheitsfaktor bei der Schrittweitensteuerung. Der Standardwert ist 0,9. Bei radau5 heißt dieser Parameter `WORK(2)`.

2.2.5 StepSizeMinSelection

Ist die untere Grenze bei der Schrittweitensteuerung. Es gilt:

$$\texttt{StepSizeMinSelection} \leq \frac{h_{\text{new}}}{h_{\text{old}}}$$

Der Standardwert ist 0,2. Bei radau5 heißt dieser Parameter `WORK(8)`.

2.2.6 StepSizeMaxSelection

Ist die obere Grenze bei der Schrittweitensteuerung. Es gilt:

$$\frac{h_{\text{new}}}{h_{\text{old}}} \leq \texttt{StepSizeMaxSelection}$$

Der Standardwert ist 8,0. Bei radau5 heißt dieser Parameter `WORK(9)`.

2.2.7 MaxStep

Ist eine obere Schranke für die maximale Schrittweite. Der Standardwert ist $t_{\text{start}} - t_{\text{end}}$. Bei radau5 heißt dieser Parameter `WORK(7)`.

2.2.8 InitialStep

Dieser double-Wert gibt den Vorschlag für die erste Schrittweite an. Sollte in der `opt` struct `opt.InitialStep=0` sein, so wird die Schrittweite auf den Standardwert gesetzt. Der Standardwert ist $1 \cdot 10^{-6}$. Bei radau5 heißt dieser Parameter `H`.

2.2.9 MaxNumberOfSteps

Dieser Wert gibt eine Obergrenze für die Anzahl der Schritte an. Der Standardwert ist 100000. Bei radau5 heißt dieser Parameter `IWORK(2)`.

2.2.10 FreezeStepSizeLeftBound

Gilt

$$\texttt{FreezeStepSizeLeftBound} \leq \frac{h_{\text{new}}}{h_{\text{old}}} \leq \texttt{FreezeStepSizeRightBound},$$

dann wird die Schrittweite nicht verändert. Damit können für große Systeme LR-Zerlegungen gespart werden. Der Standardwert ist 1,0. Dieser Parameter heißt bei radau5 `WORK(5)`.

2.2.11 FreezeStepSizeRightBound

Gilt

$$\texttt{FreezeStepSizeLeftBound} \leq \frac{h_{\text{new}}}{h_{\text{old}}} \leq \texttt{FreezeStepSizeRightBound},$$

dann wird die Schrittweite nicht verändert. Damit können für große Systeme LR-Zerlegungen gespart werden. Der Standardwert ist 1,2. Dieser Parameter heißt bei radau5 `WORK(6)`.

2.2.12 StepSizeStrategy

Ist ein Switch, welcher angibt, welche Methode für die Schrittweitenwahl verwendet werden soll. Mögliche Werte sind hier 1 oder 2. 1 ist ein predictive controller nach Gustafsson, 2 ist die klassische Schrittweitenwahl. Der Standardwert ist 1. Dieser Parameter heißt bei radau5 `IWORK(8)`.

2.2.13 MaxNewtonIterations

Gibt die maximale Anzahl von Newton-Iterationen pro Schritt zur Lösung des impliziten Systems an. Der Standardwert ist 7. Dieser Parameter heißt bei radau5 `IWORK(3)`.

2.2.14 StartNewtonWithZeros

Flag, ob die Newton-Iteration mit dem Nullvektor gestartet werden soll. Für den Fall, dass `StartNewtonWithZeros!=0` ist, so wird die extrapolierte Kollokationslösung als Startwert verwendet. Der Standardwert ist 0. Bei radau5 heißt dieser Parameter `IWORK(4)`.

2.2.15 NewtonStopCriterion

Genauigkeitsschranke für den Abbruch bei der Newtoniteration. Der Standardwert wird mit $\max\big(10 \cdot \texttt{eps}/\texttt{RelTol}(1), \min\big(0{,}03, \sqrt{\texttt{RelTol(1)}}\big)\big)$ berechnet. Bei radau5 heißt dieser Parameter `WORK(4)`.

2.2.16 DimensionOfIndex1Vars

Dimension der Index 1 Variablen. Der Standardwert ist d. Dieser Parameter heißt bei radau5 `IWORK(5)`.

2.2.17 DimensionOfIndex2Vars

Dimension der Index 2 Variablen. Der Standardwert ist 0. Dieser Parameter heißt bei radau5 `IWORK(6)`.

2.2.18 DimensionOfIndex3Vars

Dimension der Index 3 Variablen. Der Standardwert ist 0. Dieser Parameter heißt bei radau5 `IWORK(7)`.

2.2.19 Mass

Falls die Massenmatrix die Identität ist, also das System explizit ist, so muss diese Option nicht angegeben werden. Ansonsten gibt es zwei Fälle, ist `MassLowerBandwidth=d`, so wird eine (vollbesetzte) (d, d)-Matrix erwartet. Ist `MassLowerBandwidth<d`, so bedeutet das, dass die Massenmatrix Bandstruktur hat. In diesem Fall wird bei `Mass` eine cell oder eine Matrix erwartet, die die Einträge der Bänder enthält. Matrizen mit Bandstrukturen werden in einem eigenen Kapitel behandelt. Der Standardwert ist keiner (also $M = I$). Bei radau5 heißt dieser Parameter `MAS` bzw. `IMAS`.

2.2.20 MassLowerBandwidth

Die untere Bandbreite der Massenmatrix. Falls `Mass` nicht angegeben wurde, wird diese Option auch nicht benötigt. Ansonsten bedeutet `MassLowerBandwidth=d`, dass die Massenmatrix vollbesetzt ist. In diesem Fall wird in `Mass` diese (d, d)-Matrix erwartet. Ist `MassLowerBandwidth<d`, so hat die Massenmatrix Bandstruktur. In `Mass` müssen dann die Einträge der Bänder stehen. Details zu Matrizen mit Bandstruktur gibt es in einem eigenen Kapitel darüber. Der Standardwert für diesen Parameter hängt von `Mass` ab. Ist `Mass` nicht angegeben, so ist 0 der Standardwert, ansonsten d. Bei radau5 heißt dieser Parameter `MLMAS`.

2.2.21 MassUpperBandwidth

Die obere Bandbreite der Massenmatrix. Falls `Mass` nicht angegeben wurde, wird diese Option nicht benötigt. Falls `MassLowerBandwidth=d`, so wird diese Option auch nicht benötigt. In diesem Fall wird ja von einer vollbesetzten Massenmatrix ausgegangen. Der

Standardwert ist d, sofern die Option benötigt wird. Details zu Matrizen mit Bandstruktur gibt es in einem eigenen Kapitel. Bei radau5 heißt dieser Parameter `MUMAS`.

2.2.22 Jacobian

Ein String mit dem Namen einer Funktion zur Berechnung der Jacobimatrix der rechten Seite. Diese Funktion muss von der Form:

```
function df=jacobian(t,x)
```

Wie der Rückgabewert auszusehen hat hängt noch zwei weiteren Parametern ab, nämlich `JacobianLowerBandwidth` und `JacobianUpperBandwidth`. Wird `Jacobian` nicht angegeben, so wird mit `JacobianLowerBandwidth` und `JacobianUpperBandwidth` die Besetzungsstruktur der Jacobimatrix ermittelt und die relevanten Einträge mit numerischer Differentation berechnet. Bei radau5 heißt dieser Parameter `JAC` bzw. `IJAC`.

2.2.23 JacobianLowerBandwidth

Gibt die untere Bandbreite der Jacobimatrix an, also wie viele untere Nebendiagonalen vorhanden sind. Bei `JacobianLowerBandwidth=d` ist die Jacobimatrix vollbesetzt. Falls bei `Jacobian` eine Funktion angegeben wurde, dann muss sie diese vollbesetzte Jacobimatrix zurückliefern. Für `JacobianLowerBandwidth<d` hat die Matrix Bandstruktur. In diesem Fall muss eine unter `Jacobian` angegebene Funktion die Jacobimatrix die Elemente der Bänder zurückliefern. Dazu gibt es ein eigenes Kapitel. Dieser Parameter heißt bei radau5 `MLJAC`.

2.2.24 JacobianUpperBandwidth

Gibt die obere Bandbreite der Jacobimatrix an. Diese Option wird nur benötigt, falls `JacobianLowerBandwidth<d` ist. Dieser Parameter heißt bei radau5 `MUJAC`.

2.2.25 RecomputeJACFactor

Ist Indikator, wann die Jacobimatrix neuberechnet werden soll. Je höher dieser Wert (etwa 0,1), desto seltener wird die Jacobimatrix ausgewertet. Wenn die Jacobimatrix einfach auszuwerten ist, etwa bei kleinen Systemen, so ist ein kleinerer Wert, wie 0,001, angebracht. Falls `RecomputeJACFactor<0` ist, so wird nach jedem akzeptierten Schritt die Jacobimatrix neu ausgewertet. Der Standardwert ist 0,001. Bei radau5 heißt dieser Parameter `WORK(3)`.

2.2.26 TransfromJACtoHess

Ist ein Flag, ob die Jacobimatrix auf Hessenberggestalt gebracht werden soll. Dies wird nur unterstützt, falls die Massenmatrix die Identität ist und `M1=0` und die Jacobimatrix nicht Bandstruktur hat.

2.2.27 M1

Ist bei allgemeinen Problemen 0.

2.2.28 M2

Ist bei allgemeinen Problemen 0.

2.2.29 IncludeGridPointsInDenseOutput

Falls `t` mehr als zwei Werte enthält, wird auf dense-Output umgeschaltet, damit an den angegeben Stellen ausgewertet werden kann. Für die Ausgabe `tGitter` und `xGitter` gibt es nun zwei Möglichkeiten: `IncludeGridPointsInDenseOutput=0`, dann enthält `tGitter xGitter` nur die Ausgabe an den vorgegebenen Stellen. Um auch noch die von radau5 generierten Gitterpunkte auszugegeben, setzt man `IncludeGridPointsInDenseOutput!=0`. Der Standardwert ist 0. Dies ist ein Parameter des Interfaces.

2.2.30 OutputFcn

Sei der Name der Output-Function `outfcn`. Dann wird zu Beginn der Integration aufgerufen: `outfcn([tStart,tEnd],x0,'init')`. Nach jedem erfolgreichen Integrationsschritt wird aufgerufen: `status=outfcn(t,x)`. Liefert `outfcn` einen `status=1`, so wird die Integration abgebrochen (auch wenn `tEnd` noch nicht erreicht). Sonst geht es weiter. Am Ende der Integration wird aufgerufen: `outfcn([],[],'done')`. Es gibt schon vorgefertigte Output-Functions: vgl. odeplot, odephas2, odephas3, odeprint. Der Standardwert ist keine Output-Function. Dies ist ein Parameter des Interface.

2.2.31 OutputCallMode

Ist nur bei dense-Output mit Output-Function interessant. Bestimmt wann `outfcn` aufgerufen wird:

1 bei den in t angegeben Stellen
2 bei den von radau5 erstellen Gitterpunkten
3 sowohl 1 und 2

Der Standardwert ist 1. Dies ist ein Parameter des Interface.

2.2.32 OptWarnMiss

Ist ein Flag, welches angibt, ob eine Warnung ausgegeben werden soll, wenn in der `opt`-struct eine benötigt Option fehlt. Der Standardwert ist 0, also keine Warnung.

2.2.33 OptWarnType

Ist ein Flag, welches angibt, ob eine Warnung ausgegeben werden soll, wenn in der `opt`-struct eine benötigte Option vorhanden ist, aber den falschen Typ hat. Der Standardwert ist 1, also Warnung ausgeben.

2.2.34 OptWarnSize

Ist ein Flag, welches angibt, ob eine Warnung ausgegeben werden soll, wenn in der `opt`-struct eine benötigte Option vorhanden ist, den richtigen Typ hat, aber die falsche Größe hat. Der Standardwert ist 1, also Warnung ausgeben.

3 Aufruf bei Problemen mit Spezialstruktur

Ein Problem hat Spezialsstruktur, falls es ein $m_1 > 0$, ein $m_2 > 0$ und eine natürliche Zahl mm gibt, so dass $m_1 = mm \cdot m_2$ und

$$x'_i = x_{i+m_2} \qquad \text{für } i = 1, \ldots, m_1 \tag{3.1}$$

gilt. Diese Struktur erlaubt eine viel schnellere Berechnung. Vieles läuft genauso, wie in Kapitel 2 beschrieben, ab. In diesem Kapitel werden nur die Unterschiede erläutert. Denn jezt haben einige Optionen und Funktionen eine etwas andere Bedeutung. Dem Interface und radau5 wird diese Struktur mit den Optionen `M1` und `M2` mitgeteilt.

3.1 Ein- und Ausgabeargumente

Der Aufruf lautet wieder:

```
[tGitter,xGitter,stats,taupred]=radau5Mex(f,t,x0,opt)
```

Dabei bezeichnet zwar `f` wieder eine Funktion der Form:

```
function dx=f(t,x)
```

aber diese rechte Seite muss nur den nichttrivialen Teil zurückliefern. In diesem Fall wird als `dx` also ein $(d - m_1, 1)$ Vektor erwartet.

3.2 Partitionierungen

Bei Problemen mit Spezialstruktur werden die Massenmatrix und die Jacobimatrix partitioniert. Die Massenmatrix M und die Jacobimatrix J haben folgende Form:

$$M = \left(\begin{array}{c|c} 1 & \\ \ddots & \\ 1 & \\ \hline & \tilde{M} \end{array}\right) \qquad \text{und} \qquad J = \left(\begin{array}{c} 1 \\ \ddots \\ 1 \\ \hline \tilde{J} \end{array}\right)$$

Dabei ist $\tilde{M} \in \mathbb{R}^{(d-m_1)\times(d-m_1)}$ der rechte untere $(d - m_1, d - m_1)$-Block der Massenmatrix M und $\tilde{J} \in \mathbb{R}^{(d-m_1)\times d}$ der untere $(d-m_1, d)$-Block der Jacobimatrix J. Alle Optionen beziehen sich nur noch auf die nichttrivialen Teile, also auf $\tilde{M}$ und $\tilde{J}$.

3.3 Optionen

3.3.1 M1

Dieser Parameter ist gemäß der Formel (3.1) zu setzen.

3.3.2 M2

Dieser Parameter ist gemäß der Formel (3.1) zu setzen.

3.3.3 Mass

Falls die Massenmatrix die Identität ist, also das System explizit ist, so muss diese Option nicht angegeben werden. Ansonsten gibt es zwei Fälle, ist `MassLowerBandwidth=d-m1`, so wird eine (vollbesetzte) $(d-m_1, d-m_1)$-Matrix erwartet. Ist `MassLowerBandwidth<d-m1`, so bedeutet das, dass die Massenmatrix $\tilde{M}$ Bandstruktur hat. In diesem Fall wird bei `Mass` eine cell oder eine Matrix erwartet, die die Einträge der Bänder enthält. Matrizen mit Bandstrukturen werden in einem eigenen Kapitel behandelt. Der Standardwert ist keiner (also $\tilde{M} = I$). Bei radau5 heißt dieser Parameter `MAS` bzw. `IMAS`.

3.3.4 MassLowerBandwidth

Die untere Bandbreite der Massenmatrix $\tilde{M}$. Falls `Mass` nicht angegeben wurde, wird diese Option auch nicht benötigt. Ansonsten bedeutet `MassLowerBandwidth=d-m1`, dass die Massenmatrix $\tilde{M}$ vollbesetzt ist. In diesem Fall wird in `Mass` diese $(d-m_1, d-m_1)$-Matrix erwartet. Ist `MassLowerBandwidth<d-m1`, so hat die Massenmatrix $\tilde{M}$ Bandstruktur. In `Mass` müssen dann die Einträge der Bänder stehen. Details zu Matrizen mit Bandstruktur gibt es in einem eigenen Kapitel darüber. Der Standardwert für diesen Parameter hängt von `Mass` ab. Ist `Mass` nicht angegeben, so ist 0 der Standardwert, ansonsten $d - m_1$. Bei radau5 heißt dieser Parameter `MLMAS`.

3.3.5 MassUpperBandwidth

Die obere Bandbreite der Massenmatrix $\tilde{M}$. Falls `Mass` nicht angegeben wurde, wird diese Option nicht benötigt. Falls `MassLowerBandwidth=d-m1`, so wird diese Option auch nicht benötigt. In diesem Fall wird ja von einer vollbesetzten Massenmatrix $\tilde{M}$ ausgegangen. Der Standardwert ist $d - m1$, sofern die Option benötigt wird. Details zu Matrizen mit Bandstruktur gibt es in einem eigenen Kapitel. Bei radau5 heißt dieser Parameter `MUMAS`.

3.3.6 Jacobian

Ein String mit dem Namen einer Funktion zur Berechnung der Jacobimatrix $\tilde{J}$ der rechten Seite. Diese Funktion muss von der Form:

```
function df=jacobian(t,x)
```

Wie der Rückgabewert auszusehen hat hängt noch zwei weiteren Parametern ab, nämlich `JacobianLowerBandwidth` und `JacobianUpperBandwidth`. Wird `Jacobian` nicht angegeben, so wird mit `JacobianLowerBandwidth` und `JacobianUpperBandwidth` die Besetzungsstruktur der Jacobimatrix ermittelt und die relevanten Einträge mit numerischer Differentation berechnet. Bei radau5 heißt dieser Parameter `JAC` bzw. `IJAC`.

3.3.7 JacobianLowerBandwidth

Bei `JacobianLowerBandwidth=d-m1` ist die Jacobimatrix $\tilde{J}$ vollbesetzt. Falls bei `Jacobian` eine Funktion angegeben wurde, dann muss sie eine vollbesetzte $(d - m_1, d)$ Matrix zurückliefern.

Der Fall `JacobianLowerBandwidth<d-m1` wird nur unterstützt, falls m_1 und m_2 zusätzlich die Bedingung $m_1 + m_2 = d$ erfüllen. Dann wird die Jacobimatrix $\tilde{J}$ in $mm + 1$ (beachte, dass $m_1 = mm \cdot m_2$ ist) Teilmatrizen der Blockgröße (m_2, m_2) unterteilt, also

$$\tilde{J} = (\ \tilde{J}_1 \,|\, \tilde{J}_2 \,|\ \cdots\ |\, \tilde{J}_{mm+1}\ ).$$

Jetzt beziehen sich `JacobianLowerBandwidth` und `JacobianUpperBandwidth` gleichermaßen auf alle Blöcke $\tilde{J}_i$. Diese Darstellung ist also nur möglich wenn jeder der Blöcke $\tilde{J}_i$ Bandstruktur mit denselben oberen bzw. unteren Bandbreiten besitzt. Falls bei `Jacobian` eine Funktion angegeben wurde, so muss diese eine cell, bestehend aus $mm + 1$ Komponenten, zurückliefern. Jede dieser cell-Komponenten beschreibt für sich in der gewohnten Weise (vgl. Kapitel 4) Matrizen mit Bandstrukturen.

Hier ein ganz kleines Beispiel. Es seien $m_1 = 4$, $m_2 = 2$ und

$$\tilde{J} = \begin{pmatrix} 1 & 3 & 4 & 6 & 7 & 9 \\ 0 & 2 & 0 & 5 & 0 & 8 \end{pmatrix},$$

dann sind mit `JacobianLowerBandwidth=0` und `JacobianUpperBandwidth=1` alle folgenden Möglichkeiten gültige Rückgabewerte der unter `Jacobian` angegebenen Funktion:

```
{{[3],[1,2]},{[6],[4,5]},{[9],[7,8]}}
  {[0,3;1,2], [0,6;4,5], [0,9;7,8]}
 {{[3],[1,2]},[0,6;4,5],{[9],[7,8]}}
```

Dieser Parameter heißt bei radau5 `MLJAC`.

3.3.8 JacobianUpperBandwidth

Gibt die obere Bandbreite der Jacobimatrix an. Diese Option wird nur benötigt, falls `JacobianLowerBandwidth<d-m1` ist. Nähere Erläuterungen stehen bei der Beschreibung von `JacobianLowerBandwidth`. Dieser Parameter heißt bei radau5 `MUJAC`.

4 Matrizen mit Bandstruktur

4.1 Begriffe

Es sei $M \in \mathbb{R}^{d\times d}$ eine Matrix. Die Hauptdiagonale ist die 0. Diagonale. Alle Nebendiagonalen werden auch als Diagonalen bezeichnet. Eine Diagonale heißt unbesetzt, falls alle ihre Elemente 0 sind, andernfalls heißt sie besetzt.

Sei l die Nummer der letzten unteren Nebendiagonale, die besetzt ist, dann gilt für radau5 und dieses Interface: M hat Bandstruktur, wenn $l < d$ gilt. l heißt auch die untere Bandbreite von M. Bezeichnet u die Nummer der letzten oberen Nebendiagonale, die besetzt ist, dann heißt u auch die obere Bandbreite von M.

So hat z. B. die Matrix

$$M = \begin{pmatrix} 0 & 1 & 2 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 1 & 3 & 0 & 1 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 5 \end{pmatrix}$$

obere Bandbreite 2 und untere Bandbreite 1.

4.2 Darstellung mit cells

Dieses Interface bietet an, Matrizen mit Bandstrukturen mit Hilfe von cells darzustellen. Sei dazu u bzw. l die obere bzw. untere Bandbreite der Matrix $M \in \mathbb{R}^{d\times d}$. Ferner habe M Bandstruktur, also sei $l < d$. Dann hat M genau $1+l+u$ Diagonalen, die besetzt sind. Um eine solche Matrix möglichst platzsparend anzugeben, kann man nun diese $1+l+u$ Diagonalen von rechts oben nach links unten als Vektoren in eine cell packen und diese cell übergeben.

Nehmen wir das Beispiel von oben, dann kann man M so schreiben:

```
{[2,3,1],[1,2,0,0],[0,1,3,4,5],[0,1,0,0]};
```

4.3 Darstellung mit Matrizen

Diese Interface bietet auch an, Matrizen mit Bandstrukturen mit Hilfe einer meist viel kleineren Matrix darzustellen. Sei dazu u bzw. l wieder die obere bzw. untere Bandbreite der Matrix $M \in \mathbb{R}^{d\times d}$. Ferner habe M Bandstruktur, also sei $l < d$. Dann kann M durch eine Matrix $N \in \mathbb{R}^{(1+l+u)\times d}$ darstellen. Dabei gilt der Zusammenhang

$$N(i-j+u+1, j) = M(i, j),$$

wobei alle Paare (i, j) zulässig sind, die in der Bandbreite von M liegen. Die Werte, die durch obige Festlegung noch nicht bestimmt sind, sind egal.

Nehmen wir wieder das Beispiel von oben, dann kann man M so schreiben:

$$N = \begin{pmatrix} * & * & 2 & 3 & 1 \\ * & 1 & 2 & 0 & 0 \\ 0 & 1 & 3 & 4 & 5 \\ 0 & 1 & 0 & 0 & * \end{pmatrix}$$

Dabei kennzeichnet $*$ Werte, die egal sind.

Diese zweite Version entspricht genau der internen Darstellung bei radau5 und wird daher etwas schneller durch das Interface behandelt.

4.4 Beispiele

Hier noch ein paar Beispiele zur Verdeutlichung:

$$M = \begin{pmatrix} 1 & 4 & 0 \\ 0 & 2 & 5 \\ 0 & 0 & 3 \end{pmatrix};$$

```
{[4,5],[1,2,3]};
```

$$N = \begin{pmatrix} * & 4 & 5 \\ 1 & 2 & 3 \end{pmatrix};$$

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 6 & 2 & 0 & 0 & 0 \\ 0 & 7 & 3 & 0 & 0 \\ 0 & 0 & 8 & 4 & 0 \\ 0 & 0 & 0 & 9 & 5 \end{pmatrix};$$

```
{[1,2,3,4,5],[6,7,8,9]};
```

$$N = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & * \end{pmatrix};$$

5 Beispiele

Zum Abschluß noch 5 Beispiele. Bei allen Beispielen steht der Aufruf des Interfaces im Vordergrund. Natürlich lassen sich die meisten Beispiele durch wenige Umformungen erheblich vereinfachen. Die Beispiele sind vom Schwierigkeitsgrad steigend angeordnet. Alle Beispiele gibt es auch als zip-File.

5.1 Beispiel 1

Betrachtet man das System

$$\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \qquad \text{mit} \qquad x_1(1) = 1, \quad x_2(1) = 1,$$

so, ist die rechte Seite

```
function dx=f(t,x)
dx=x;
```

und der Aufruf von radau5Mex könnte so lauten:

```
opt.RelTol=1e-6;opt.AbsTol=1e-6;
% Jacobimatrix hat Bandstruktur
% Jacobimatrix wird nicht explizit angegeben
%   => radau5 wird numerische Differentation verwenden
opt.JacobianLowerBandwidth=0;
opt.JacobianUpperBandwidth=0;
[tG,xG,stats]=radau5Mex('f',[1,2],[1;2],opt);
```

5.2 Beispiel 2

Beim System

$$\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = \begin{pmatrix} x_2 \\ x_1 \end{pmatrix} \qquad \text{mit} \qquad x_1(1) = 1, \quad x_2(1) = 2,$$

ist die rechte Seite auch ganz einfach:

```
function dx=f(t,x)
dx=[x(2);x(1)];
```

aber diesesmal wollen wir die Jacobimatrix explizit angeben:

```
function df=jac(t,x)
df=[0,1;1,0];
```

Der Aufruf lautet in etwa so:

```
opt.RelTol=1e-6;opt.AbsTol=1e-6;
% Jacobimatrix vollbesetzt
% Jacobimatrix wird explizit angegeben
opt.JacobianLowerBandwidth=2;
opt.Jacobian='jac';
% Jacobiauswertung ist so billig, deshalb
opt.RecomputeJACFactor=-1;

[tG,xG,stats]=radau5Mex('f',[1,2],[1;2],opt);
```

5.3 Beispiel 3

Jetzt nehmen wir mal ein System mit vollbesetzter Massenmatrix:

$$\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} x' = t \begin{pmatrix} x_2 \\ x_1 \end{pmatrix} \qquad \text{mit} \qquad x_1(1) = 1, \quad x_2(1) = 2$$

Auch hier ist die rechte Seite

```
function dx=f(t,x)
dx=t*[x(2);x(1)];
```

einfach und die Jacobimatrix auch:

```
function df=jac(t,x)
df=[0,t*x(2); t*x(1),0];
```

Der Aufruf ist das einzig interessante, denn da muss ja die Massenmatrix angegeben werden:

```
opt.RelTol=1e-6;opt.AbsTol=1e-6;
% Massenmatrix: keine Bandstruktur
opt.MassLowerBandwidth=2;
opt.Mass=[2,1;1,2];
% Jacobimatrix keine Bandstruktur
% Jacobimatrix wird explizit angegeben
opt.JacobianLowerBandwidth=2;
opt.Jacobian='jac';
% Jacobiauswertung ist so billig, deshalb
opt.RecomputeJACFactor=-1;

[tG,xG,stats]=radau5Mex('f',[1,2],[1;2],opt);
```

5.4 Beispiel 4

Nun wird es mal Zeit für ein System, bei dem Matrizen mit Bandstrukturen vorkommen:

$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \begin{pmatrix} x_1 - x_2 \\ x_2 - x_3 \\ t x_3 \end{pmatrix} \qquad \text{mit} \qquad x_1(1) = x_2(1) = x_3(1) = 1$$

Die rechte Seite lautet:

```
function dx=f(t,x)
dx=[x(1)-x(2); x(2)-x(3); t*x(3)];
```

Die Jacobimatrix hat Bandstruktur, also:

```
function df=jac(t,x)
df={[-1,-1],[1,1,t]};
```

Der Aufruf:

```
opt.RelTol=1e-6;opt.AbsTol=1e-6;
% Jacobimatrix Bandstruktur
% Jacobimatrix wird explizit angegeben
opt.JacobianLowerBandwidth=0;
opt.JacobianUpperBandwidth=1;
opt.Jacobian='jac';
% Jacobiauswertung ist so billig, deshalb
opt.RecomputeJACFactor=-1;

[tG,xG,stats]=radau5Mex('f',[1,2],[1;1;1],opt);
```

5.5 Beispiel 5

Jetzt ein System mit Spezialstruktur und Jacobiblöcken mit Bandstruktur:

$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \\ x_4' \\ x_5' \\ x_6' \end{pmatrix} = \begin{pmatrix} x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_1 + tx_2 + x_3 + x_5 + x_6 \\ 2x_2 + x_4 \end{pmatrix} \qquad \text{mit} \qquad x_i(1) = 1$$

Bei der rechten Seite darf nur der nichttriviale Teil übergeben werden:

```
function dx=f(t,x)
dx=[x(1)+t*x(2)+x(3)+x(5)+x(6); % x5'
         2*x(2)+x(4)];     % x6'
```

Für die Jacobiblöcke hat man:

```
function df=jac(t,x)
df={{t,[1,2]},{0,[1,1]},{1,[1,0]}};
return;
% Auch möglich wäre gewesen:
% df={[0,t;1,2],[0,0;1,1],[0,1;1,0]}
% oder
% df={{t,[1,2]},[0,0;1,1],[0,1;1,0]}
```

Und dann noch der Aufruf:

```
opt.RelTol=1e-6;opt.AbsTol=1e-6;
% System hat Spezialstruktur
```

```
opt.M1=4;opt.M2=2;
% nichttriviale Jacobiblöcke haben Bandstruktur
% Blöcke werden explizit angegeben.
opt.JacobianLowerBandwidth=0;
opt.JacobianUpperBandwidth=1;
opt.Jacobian='jac';
% Jacobiauswertung ist so billig, deshalb
opt.RecomputeJACFactor=-1;

[tG,xG,stats]=radau5Mex('f',[1,2],[1;1;1;1;1;1],opt);
```