RA

stichting
mathematisch
centrum

MC

REKENAFDELING

NR 25/72 APRIL

RA

K. DEKKER

EEN ALGOL 60 VERSIE VAN EXPONENTIEEL AANGEPASTE RUNGE-KUTTA METHODEN

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat 49, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

Voorwoord

Dit verslag bevat een ALGOL 60 programma voor het oplossen van beginwaarde problemen.
Het programma en de voorbeelden zijn getest op de Electrologica X8 computer van het Mathematisch Centrum te Amsterdam.
De schrijver wil zijn dank betuigen aan Dr. P. J. van der Houwen voor diens waardevolle suggesties en aan A. C. IJsselstein voor diens aanwijzingen bij het maken van de tekeningen.

Inhoud

1. Inleiding

Beschouw beginwaarde-problemen van het type

$$(1.1) \qquad \frac{dU}{dt} = H(t,U), \quad U(t_0) = U_0,$$

waarin H een (vector) functie is van U en de variabele t, en U een (vector) functie van t.
De traditionele methoden voor het numeriek integreren van dit soort problemen blijken niet goed toegepast te kunnen worden op (stelsels) stijve differentiaal-vergelijkingen.
Daarom zijn de laatste jaren een aantal nieuwe ideeën ontwikkeld. Pope [1] introduceerde in 1963 het begrip exponentiële aanpassing. Na hem hebben o.a. Liniger en Willoughby [2], Lawson [3], en Fowler en Warten [4] methoden gepubliceerd die gebaseerd zijn op deze techniek.
Op het Mathematisch Centrum is getracht exponentiële aanpassing in Taylor- en Runge-Kutta-methoden te realiseren. Enkele twee-punts benaderingen van de exponentiële operator worden in [6] gegeven, terwijl in [7] een 1^e-orde, exponentieel gefitte Taylor-methode beschreven staat.
In dit verslag geven we een ALGOL 60-versie van exponentiëel aangepaste expliciete Runge-Kutta-formules met een orde van nauwkeurigheid p, p = 1, 2, 3.

In het volgende hoofdstuk geven we een afleiding van de parameters van de methode en zijn stabiliteits-eigenschappen.
De ALGOL-60 versie van de procedure exponential fitted runge kutta staat in het derde hoofdstuk beschreven.
In het vierde hoofdstuk bespreken we een aantal testresultaten. Allereerst behandelen we een lineair stelsel van Fowler en Warten, dat zich zeer goed leent voor exponentiële methoden. Dit voorbeeld kozen we tevens om de interne instabiliteit van meer-punts formules te illustreren.
Bij het tweede voorbeeld, een derde-orde differentiaal-vergelijking, heeft de Jacobiaan complex-toegevoegde eigenwaarden.
Het derde voorbeeld, een niet-lineaire vergelijking, kozen we om het voordeel van een geschikte keuze van de orde aan te tonen.

2. De theorie van exponentieel aangepaste Runge-Kutta schema's

In dit hoofdstuk behandelen we de theoretische achtergronden van de procedure exponential fitted runge kutta. Hierbij maken we gebruik van de theorie van gestabiliseerde Runge-Kutta methoden die uitvoerig beschreven wordt in [8].

2.1. Algemene structuur van het Runge-Kutta schema

Beschouw de algemene n-punts formule

$$
(2.1)\qquad \begin{cases} u_0 = U_0, \\ u_{k+1} = u_k + \theta_0\, r_k^{(0)} + \ldots + \theta_{n-1}\, r_k^{(n-1)}, \quad k = 0, 1, \ldots, \\ r_k^{(0)} = \tau_k\, H(t_k, u_k), \\ r_k^{(j)} = \tau_k\, H(t_k+\mu_j\tau_k, u_k+\lambda_{j0} r_k^{(0)}+\ldots+\lambda_{jj-1} r_k^{(k-1)}), \\ \qquad\qquad\qquad\qquad\qquad\qquad j = 1, 2, \ldots, n-1, \end{cases}
$$

waarin u_k de numerieke oplossing is op het tijdstip t_k.
Verder geldt per definitie $\tau_k = t_{k+1}-t_k$.
Schema (2.1) kan compact weergegeven worden door de matrix

$$
(2.2)\qquad R = \begin{pmatrix} \mu_1 & \lambda_{10} & 0 & \cdots & 0 \\ \mu_2 & \lambda_{20} & \lambda_{21} & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ \mu_{n-1} & \lambda_{n-10} & \lambda_{n-11} & \cdots & \lambda_{n-1\,n-2} \\ \theta_0 & \theta_1 & \theta_2 & \cdots & \theta_{n-1} \end{pmatrix}.
$$

2.2. Stabiliteit

Beschouw eerst het lineaire beginwaarde-probleem

(2.3) $$\frac{dU}{dt} = DU, \quad U(t_0) = U_0,$$

waarin D een matrix is met eigenwaarden δ_i waarvoor geldt

$$Re(\delta_i) \leq 0, \quad \forall i.$$

We definiëren het stabiliteitspolynoom $P_n(z)$ door

$$P_n(z) = 1 + \beta_1 z + \ldots + \beta_n z^n,$$

waarin de parameters β_j, $j = 1, 2, \ldots, n$, functies zijn van de Runge-Kutta parameters μ_l en λ_{li} (2.2), die in paragraaf 2.6. gegeven worden.
Bij dit polynoom kunnen we een Runge-Kutta schema opstellen.
Wanneer voor alle $z_i = \tau\delta_i$ geldt

$$|P_n(z_i)| \leq 1,$$

dan noemen we dit schema stabiel.
Het beginwaarde-probleem (1.1) kunnen we locaal lineariseren. Nu noemen we het schema stabiel, indien voor de eigenwaarden δ_i van de Jacobiaan $J = (\frac{\partial H}{\partial U})$ met $Re(\delta_i) \leq 0$ geldt

$$|P_n(\tau\delta_i)| \leq 1.$$

2.3. Afleiding van de stabiliteitspolynomen

Bij stelsels stijve differentiaal-vergelijkingen liggen de eigenwaarden met negatief reëel deel vaak in enkele ver uit elkaar liggende clusters.
Het is daarom van belang stabiliteitspolynomen $P_n(z)$ te kiezen, zodanig dat voor elke cluster Cl geldt

$$\max_{\delta \in Cl} |P_n(\tau\delta)| \leq 1,$$

waarin τ de integratiestap voorstelt. (De coëfficiënten van P kunnen functies van τ zijn.)

In het volgende beschrijven we polynomen voor twee- en drie-cluster-methoden, waarbij één van de clusters bij de oorsprong van het eigenwaarden-vlak ligt, terwijl de andere cirkels zijn met als middelpunt twee complex toegevoegde punten; deze kunnen eventueel samenvallen. Beschouw het $z = \tau\delta$-vlak voor een gegeven integratiestap τ. Gevraagd wordt een polynoom P_n, zodanig dat voor een zo groot mogelijke ρ_1 geldt

$$\max_{|z-z_1|\leq\rho_1} |P_n(z)| \leq 1,$$

onder de nevenvoorwaarden

$$\text{(2.3)} \qquad \text{graad } (P_n) = n,\ P_n(0) = 1,\ \frac{dP_n(0)}{dz} = \beta_1,\ \ldots,\ \frac{d^r P_n(0)}{r!dz^r} = \beta_r.$$

Voor grote waarden van z_1 blijkt een optimale keuze te zijn [9]

$$\text{(2.4)} \qquad P_n(z) = |z_1|^{-2q}\,(z-z_1)^q(z-\bar{z}_1)^q\,R_r(z),\ q = \frac{n-r}{2}\ .$$

Hierin is $R_r(z)$ een polynoom van de graad r, waarvan de coëfficiënten eenduidig bepaald worden door voorwaarde (2.3).
De bij het polynoom $P_n(z)$ behorende ρ_1 wordt voor de twee-cluster-methode gegeven door

$$\text{(2.5)} \qquad \rho_1 = \left(\frac{|z_1|^{2q-r}}{\beta_r}\right)^{\frac{1}{2q}}$$

en in het geval van de drie-cluster-methode door

$$\text{(2.6)} \qquad \rho_1 = \frac{1}{2\sin\phi}\left(\frac{|z_1|^{q-r}}{\beta_r}\right)^{\frac{1}{q}}, \qquad \phi = \arg(z_1).$$

Voor kleine waarden van z_1 willen we dat $P_n(z)$ zich gedraagt als $\exp(z)$. Dit bereiken we met exponentiële aanpassing.
De voorwaarden hiervoor luiden voor de twee-clusters-methode

$$\text{(2.7)} \qquad P_n^{(j)}(z_1) = e^{z_1}, \qquad j = 0, \ldots, n-r-1,$$

en voor de drie-cluster-methode

$$P_n^{(j)}(z_1) = e^{z_1}, \qquad j = 0, \ldots, q-1. \tag{2.8}$$

Voor $|z_1| \to \infty$ leidt (2.7) of (2.8) juist tot de polynoom voorstelling (2.4).

Omdat bij de berekening van de coëfficiënten van $P_n(z)$ voor zeer kleine waarden van z_1 singulariteiten kunnen optreden (bijvoorbeeld als in voorwaarde 2.3 $\beta_j \neq 1/j!$), hebben we voor $|z_1| < 1$ voor P_n een n-de orde Padé-approximatie gekozen.

2.4. Constructie van de twee-cluster-methode

Bij de constructie van het stabiliteitspolynoom voor de twee-cluster-methode moeten we het stelsel vergelijkingen (2.7) oplossen onder de voorwaarde (2.3); we vinden dan de coëfficiënten $\beta_{r+1}, \ldots, \beta_n$. Stellen we

$$m = n-r,$$

$$\gamma_k = \sum_{j=k}^{m-1} \frac{1}{k!} \frac{1}{(j-k)!} (-z_1)^{j-k} e^{z_1}, \qquad 0 \leq k \leq n,$$

dan kan de oplossing expliciet geschreven worden:

$$\beta_k = \gamma_k + \sum_{j=0}^{r} z_1^{j-k} (\gamma_j - \beta_j) \prod_{\substack{i=r+1 \\ i \neq k}}^{n} \left(\frac{j-i}{k-i}\right), \qquad r+1 \leq k \leq n. \tag{2.9}$$

Voor het bewijs maken we gebruik van twee hulpstellingen.

Hulpstelling 1

Zij β_k gegeven door

$$\beta_k = z_1^{s-k} \prod_{\substack{i=r+1 \\ i \neq k}}^{r+m} \left(\frac{s-i}{k-i}\right),$$

dan voldoet $P(z) = \sum_{k=r+1}^{r+m} \beta_k z^k$ aan de vergelijkingen

$$P^{(j)}(z_1) = \frac{d^j}{dz^j} z_1^s , \qquad 0 \leq j \leq m-1.$$

Bewijs

Voor j = 0 geldt de bewering, immers

$$P(z_1) = \sum_{k=r+1}^{r+m} \beta_k z_1^k = z_1^s \sum_{k=r+1}^{r+m} \prod_{\substack{i=r+1 \\ i\neq k}}^{r+m} \left(\frac{s-i}{k-i}\right) =$$

$$= z_1^s \sum_{k=0}^{m-1} \prod_{\substack{i=0 \\ i\neq k}}^{m-1} \left(\frac{s-i-r-1}{k-i}\right).$$

$\prod_{\substack{i=0 \\ i\neq k}}^{m-1} \left(\frac{s-i-r-1}{k-i}\right) = L_k^m(s-r-1)$ zijn juist de Lagrange interpolatie coëfficiënten; hun som is 1.

Evenzo geldt (mits $m \geq 2$) voor $\beta_k^* = \frac{k-r-m}{s-r-m} \beta_k$

$$\sum_{k=r+1}^{r+m-1} \beta_k^* z_1^k = z_1^s .$$

Stel nu dat de bewering juist is voor alle $j \leq j0 \leq m-2$.

Schrijven we

$$\beta_k \left(\frac{d^j}{dz^j} z^k\right)_{z_1} = c_k z_1^{k-j}$$

dan volgt uit

$$\sum_{k=r+1}^{r+m} c_k z_1^{k-j} = \left(\frac{d^j}{dz^j} z^s\right)_{z_1}$$

en

$$\sum_{k=r+1}^{r+m-1} c_k^* z_1^{k-j} = \left(\frac{d^j}{dz^j} z^s\right)_{z_1} ,$$

dat

$$\sum_{k=r+1}^{r+m} \left(\frac{d^{j+1}}{dz^{j+1}} \beta_k z^k\right)_{z_1} = \frac{r+m-j}{z_1} \sum_{k=r+1}^{r+m} c_k z_1^{k-j} +$$

$$+ \frac{s-r-m}{z_1} \sum_{k=r+1}^{r+m-1} c_k^* z_1^{k-j} = \left(\frac{d^{j+1}}{dz^{j+1}} z^s\right)_{z_1} .$$

De bewering van de stelling geldt dus ook voor alle $j \leq j0+1$. Hieruit volgt door volledige inductie dat de bewering geldt voor alle $j \leq m-1$.

Hulpstelling 2

Zij γ_k gegeven door

$$\gamma_k = \sum_{j=k}^{m-1} \frac{1}{k!} \frac{1}{(j-k)!} (-z_1)^{j-k} e^{z_1}, \qquad 0 \leq k \leq m-1,$$

dan voldoet $Q(z) = \sum_{k=0}^{m-1} \gamma_k z^k$ aan de vergelijkingen

$$Q^{(j)}(z_1) = e^{z_1}, \qquad 0 \leq j \leq m-1.$$

Bewijs

$Q(z)$ kan geschreven worden als

$$Q(z) = e^{z_1} (1+(z-z_1)+\ldots+\frac{1}{(m-1)!}(z-z_1)^{m-1});$$

hieruit volgt direkt de bewering van de stelling.

Zij nu

$$\beta_{1k} = - \sum_{j=0}^{r} z_1^{j-k} \beta_j \prod_{\substack{i=r+1 \\ i\neq k}}^{r+m} \left(\frac{j-i}{k-i}\right), \qquad r+1 \leq k \leq r+m,$$

$$\beta_{2k} = \gamma_k + \sum_{j=0}^{r} z_1^{j-k} \gamma_j \prod_{\substack{i=r+1 \\ i\neq k}}^{r+m} \left(\frac{j-i}{k-i}\right), \qquad r+1 \leq k \leq r+m,$$

$$P_0(z) = \sum_{k=0}^{r} \beta_k z^k,$$

$$P_1(z) = \sum_{k=r+1}^{r+m} \beta_{1k} z^k,$$

$$P_2(z) = \sum_{k=r+1}^{r+m} \beta_{2k} z^k.$$

Uit hulpstelling 1 volgt voor $0 \leq j \leq m-1$

$$P_1^{(j)}(z_1) = - P_0^{(j)}(z_1)$$

en

$$P_2^{(j)}(z_1) = \quad Q^{(j)}(z_1),$$

en uit hulpstelling 2

$$Q^{(j)}(z_1) = e^{z_1},$$

dus $P_n(z) = P_0(z) + P_1(z) + P_2(z)$ voldoet aan (2.7).

2.5. Constructie van de drie-cluster-methode

De drie-cluster-methode verkrijgen we door de reële en imaginaire delen van (2.8) te beschouwen; dit levert 2q vergelijkingen in de onbekenden $\beta_{r+1}, \ldots, \beta_n$:

$$(\vec{a}_{ij})\ (|z_1|^{n-r-1}\ \beta_n, \ldots, |z_1|\beta_{r+2}, \beta_{r+1})^t = (\vec{r}_i), \tag{2.10}$$

waarin

$$a_{2i,j} = \sin(n-r-j)\phi * \frac{(n+1-j)!}{(n+1-j+i-q)!},$$

$$a_{2i-1,j} = \cos(n-r-j)\phi * \frac{(n+1-j)!}{(n+1-j+i-q)!},$$

$$r_{2i} = \frac{e^{|z_1|}}{|z_1|^{r+1}} \sin(b\ \sin\phi - (r-q+1+i)\phi) + \sum_{j=0}^{r} \frac{j!}{(j+i-q)!}\ \beta_j\ |z_1|^{j-r-1}\ \sin(r+1-j)\phi,$$

$$r_{2i-1} = \frac{e^{|z_1|}}{|z_1|^{r+1}} \cos(b\ \sin\phi - (r-q+1+i)\phi) - \sum_{j=0}^{r} \frac{j!}{(j+i-q)!}\ \beta_j\ |z_1|^{j-r-1}\ \cos(r+1-j)\phi.$$

We zijn er niet in geslaagd hiervoor een analytische oplossing te geven. Daarom lossen we het stelsel (2.10) op met een LU-ontbinding met behulp van de standaard procedures det en sol [10].

2.6. Genererende matrix van Runge-Kutta schema's van orde p, p = 1,2,3

De coëfficiënten van de genererende matrix (2.2) kunnen voor de gevallen p = 1, 2, 3 zo gekozen worden, dat bij de berekeningen volgens het bijbehorende Runge-Kutta schema een beperkt gebruik van de geheugenruimte gemaakt wordt [8].
De waarden van de coëfficiënten, uitgedrukt in de parameters β_j, j = 1, ..., n, zijn

$$
\left\{
\begin{array}{lll}
\theta_0 = \begin{cases} 0 & , \\ \frac{1}{4} & , \end{cases} & & \begin{array}{l} p = 1, 2, \\ p = 3, \end{array} \\
\theta_j = 0 \quad , & & \\
\theta_{n-1} = 1 - \theta_0 , & & \\
\mu_{n-1} = \beta_2/\theta_{n-1} , & & \\
\mu_{n-j} = \dfrac{\beta_{j+1}}{\beta_j} \dfrac{\mu_{n-j+1}}{\mu_{n-j+1} - \theta_0} , & & j = 1, 2, \ldots, n-1, \\
\lambda_{jl} = \begin{cases} \theta_0 & , \\ \mu_1 & , \\ \mu_j - \theta_0 , & \\ 0 & , \end{cases} & & \begin{array}{l} 2 \le j \le n-1,\ l = 0, \\ j = 1,\ l = 0, \\ 2 \le j \le n-1,\ l = j-1, \\ 3 \le j \le n-1,\ 1 \le l \le j-2. \end{array}
\end{array}
\right. \tag{2.11}
$$

Voor de gevallen p = 1, 2 krijgen we op deze manier

$$\mu_j = \lambda_{jj-1} = \frac{\beta_{n+1-j}}{\beta_{n-j}}, \qquad j = 1, 2, \ldots, n-1,$$

met in het bijzonder

$$\mu_{n-1} = \frac{1}{2} \quad \text{als} \quad p = 2.$$

Voor p = 3 krijgen we

$$\mu_{n-1} = \frac{2}{3},\ \mu_{n-2} = \frac{8}{15},\ \lambda_{n-1\ n-2} = \frac{5}{12},\ \lambda_{n-2\ n-3} = \frac{17}{60}.$$

2.7. Interne stabiliteit

Bij meerpunts-formules kan numerieke instabiliteit optreden, hoewel het schema globaal gezien stabiel is. Deze wordt veroorzaakt doordat een fout bij de berekening van de tussenpunten opbouwt. In gevallen waarbij veel tussenpunten worden gebruikt, kan deze fout opbouw zeer hinderlijk worden.
Dit blijkt duidelijk, wanneer we het schema (2.1) met de formules (2.11) op het lineaire beginwaarde-probleem (2.3) toepassen.
De formule voor de tussenpunten $u_k^{(j)}$ wordt dan

$$u_k^{(j)} = u_k^{(0)} + \theta_0 \tau D u_k^{(0)} + \lambda_{jj-1}\, \tau D u_k^{(j-1)}, \qquad 1 \leq j \leq n-1,$$

waarin we $\lambda_{10} = \mu_1 - \theta_0$ hebben genomen om de notatie te vereenvoudigen.
Stellen we

$$P_s(z) = (1+\theta_0 z)(1+\lambda_{n-1\ n-2} z(1+\ldots(1+\lambda_{n-2+1\ n-s} z)\ldots)$$

en

$$Q_s(z) = \lambda_{n-1\ n-2} \cdots \lambda_{n-s\ n-s-1} z^s, \qquad 1 \leq s \leq n-1,$$

dan kunnen we $u_k^{(n-1)}$ schrijven als

$$u_k^{(n-1)} = P_s(\tau D)\ u_k^{(0)} + Q_s(\tau D)\ u_k^{(n-s-1)}.$$

Hieruit volgt

$$u_{k+1} = (1+\theta_0 \tau D+\theta_{n-1} P_s(\tau D))\ u_k^{(0)} + \theta_{n-1} \tau D\ Q_s(\tau D)\ u_k^{(n-s-1)}.$$

De factor $\theta_{n-1} \tau D\ Q_s(\tau D)$ is in het algemeen niet klein; voor sommige waarden van s en grote integratiestappen kan deze uitdrukking zelfs zeer groot worden.
We stellen

(2.12) $$M(z) = \max_s |\theta_{n-1}\ zQ_s(z)|.$$

In het geval $\theta_{n-1} = 1$ hangt dit maximum niet van l af; een schatting voor M(z) is dan

$$M(z) = \beta_r \cdot z^r.$$

Indien $\theta_{n-1} = \frac{3}{4}$, dan geldt bij benadering

$$M(z) = \tfrac{1}{2}\beta_r\ z^r\ (\tfrac{z}{4})^{l-1}, \quad l = n-r.$$

De interne instabiliteit neemt in dit geval toe, als l groter wordt.
Om interne instabiliteit te vermijden, formuleren we de eis

(2.13) $$M(z) \leq tol/\varepsilon, \qquad \text{voor alle } z = \tau\delta,$$

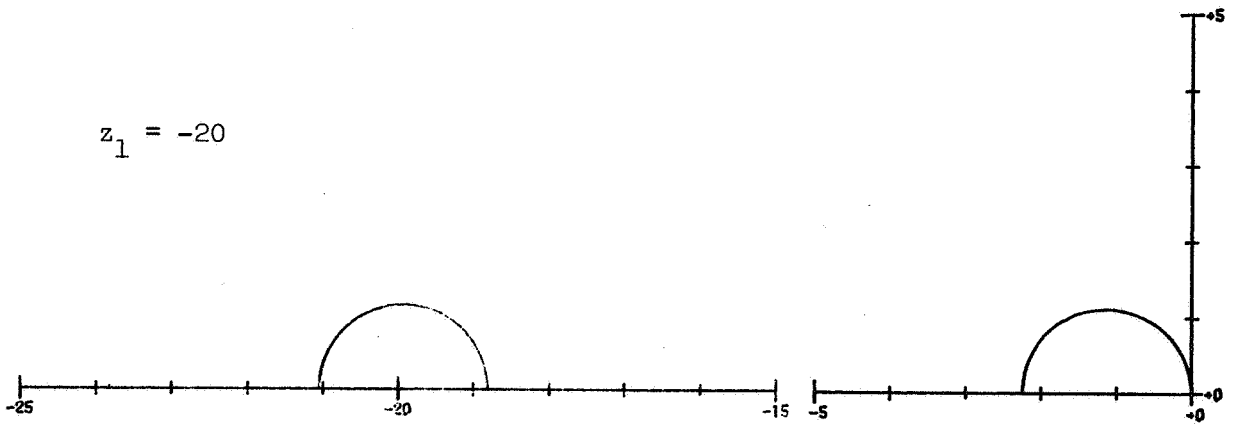
met δ eigenwaarde van de Jacobiaan;
ε is hierin de machine precisie, tol de gewenste precisie in de resultaten.

2.8. Stabiliteitsgebieden

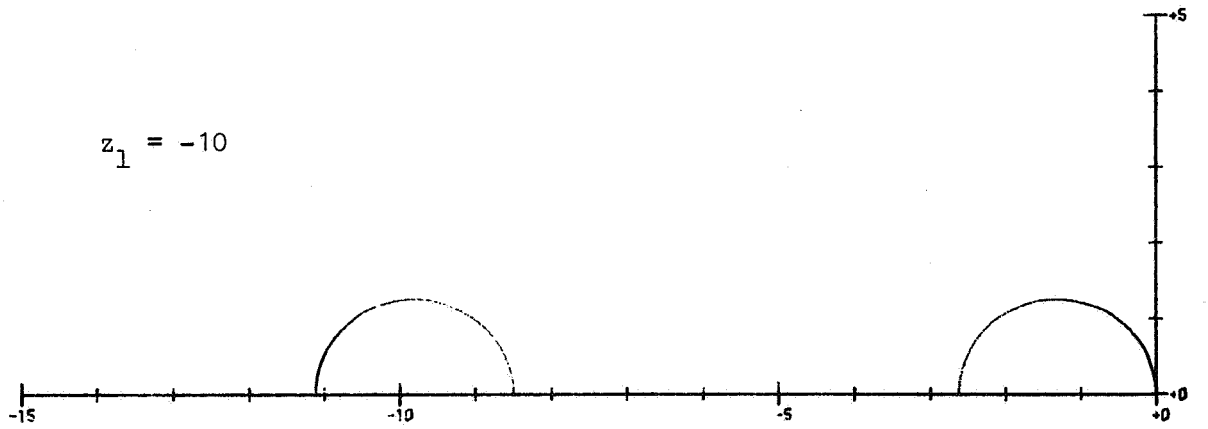
In paragraaf 2.3 is afgeleid, dat het linker stabiliteitsgebied voor grote waarden van $|z_1|$ een cirkel is, waarvan de straal door uitdrukking (2.5) wordt gegeven. Voor kleinere waarden van $|z_1|$ wordt deze cirkel gevormd, totdat samensmelting van het rechter-stabiliteitsgebied optreedt. Dit wordt geïllustreerd door de volgende tekeningen van stabiliteitsgebieden voor de twee-cluster-methode, waarbij $z_1 = -20, -10, -5$ gekozen is. Bovendien blijkt duidelijk dat de straal van de cirkel sterk afhankelijk is van de keuze van r en n. (Als beginstuk van het stabiliteitspolynoom is een r-de orde Padé-approximatie van e^z genomen.)

Stabiliteitsgebieden voor n = 2, r = 1.

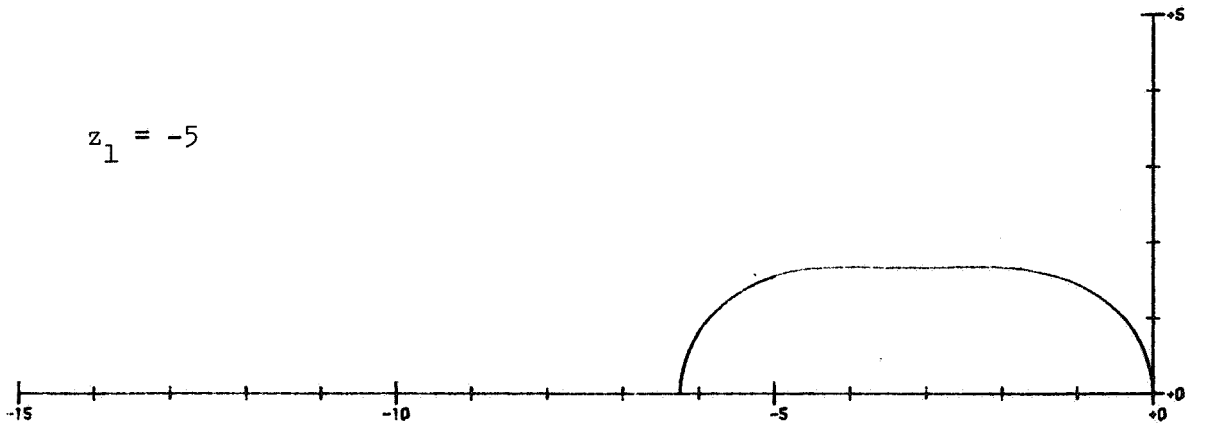
$z_1 = -20$



$z_1 = -10$

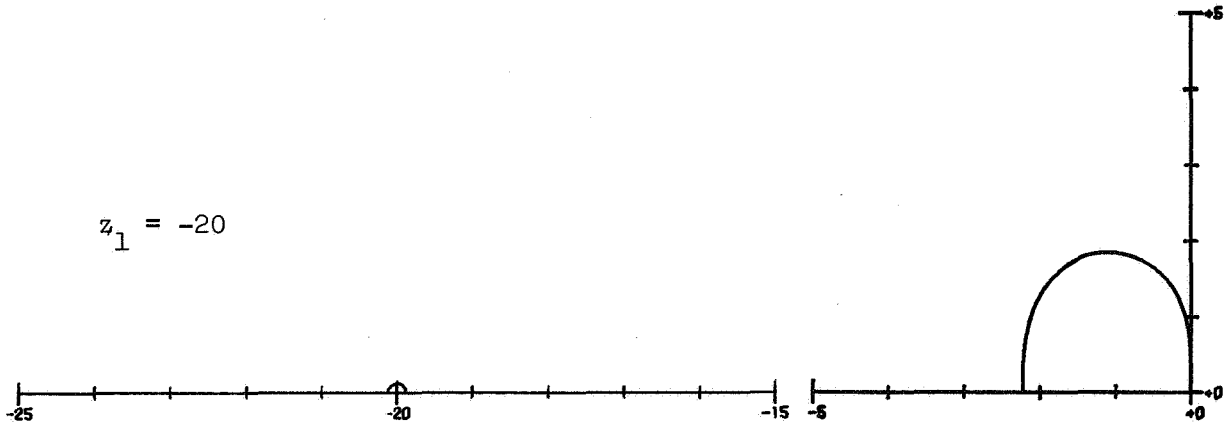


$z_1 = -5$

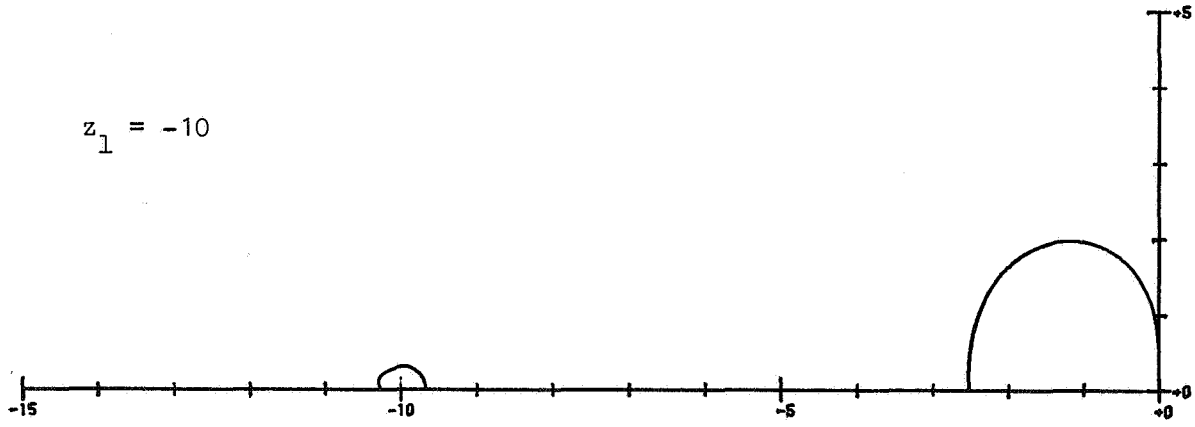


Stabiliteitsgebieden voor n = 3, r = 2.

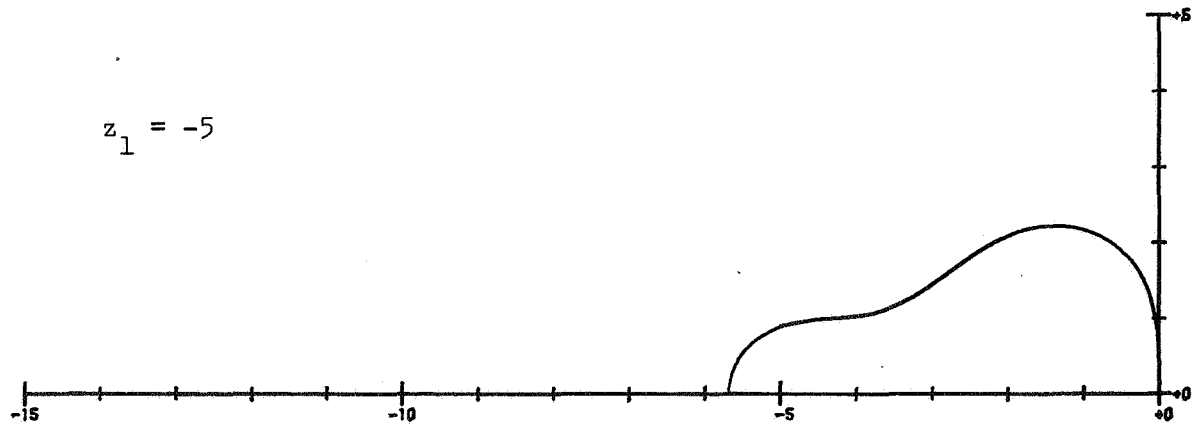
$z_1 = -20$



$z_1 = -10$

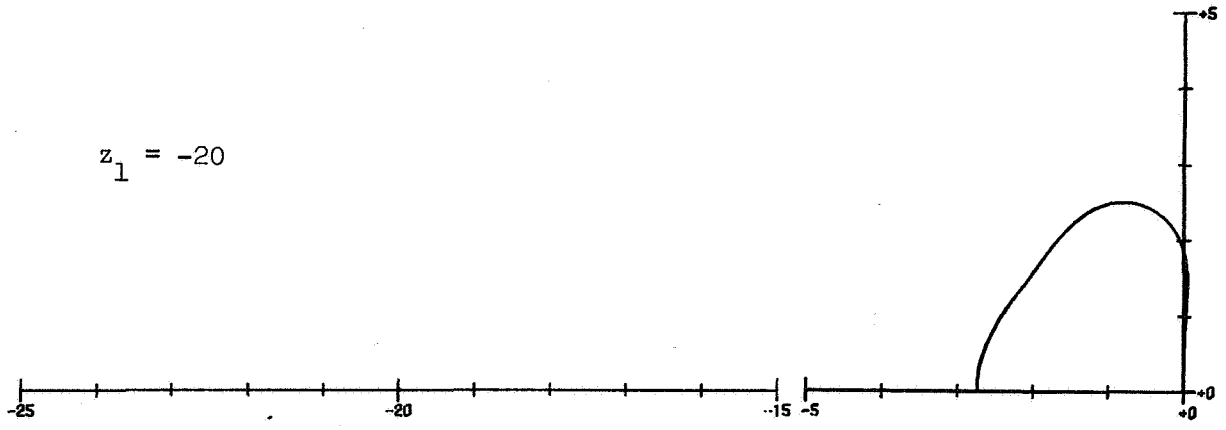


$z_1 = -5$

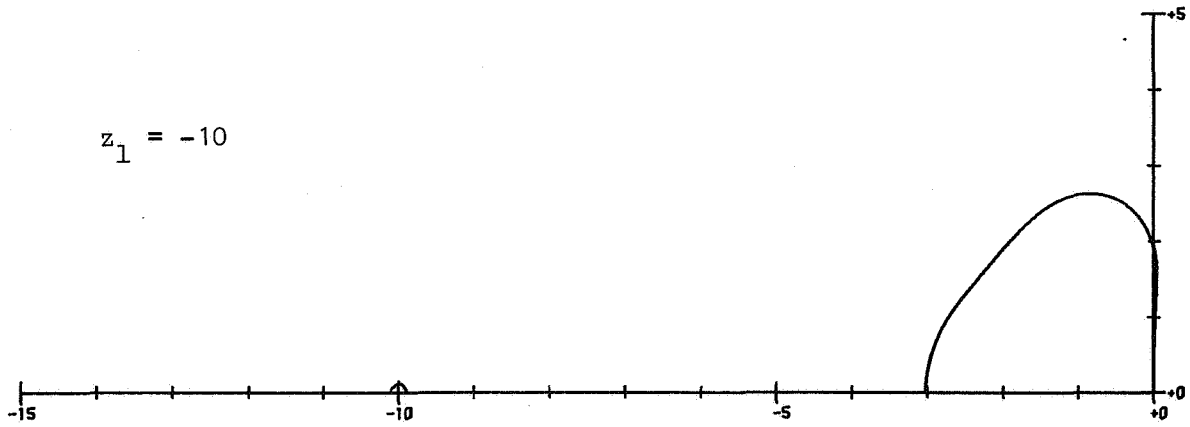


Stabiliteitsgebieden voor n = 4, r = 3.

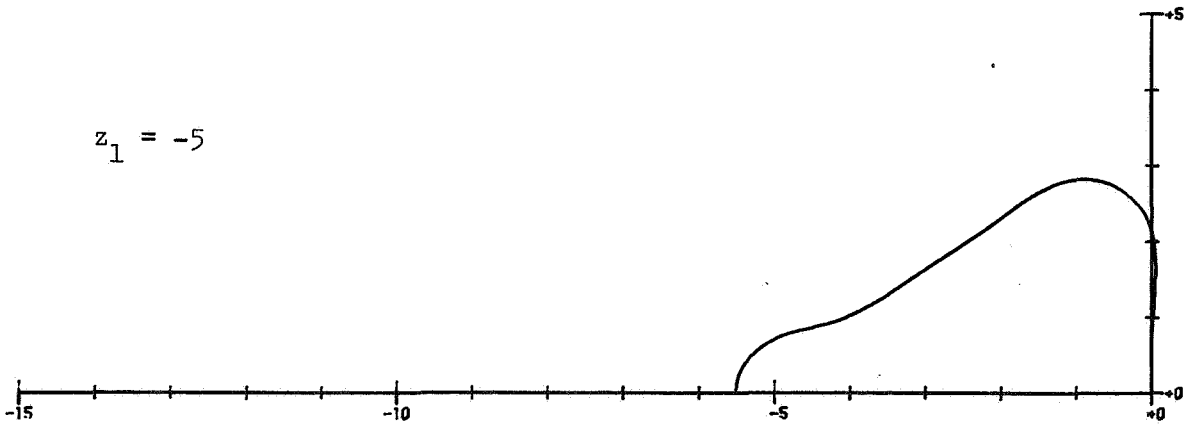
$z_1 = -20$



$z_1 = -10$

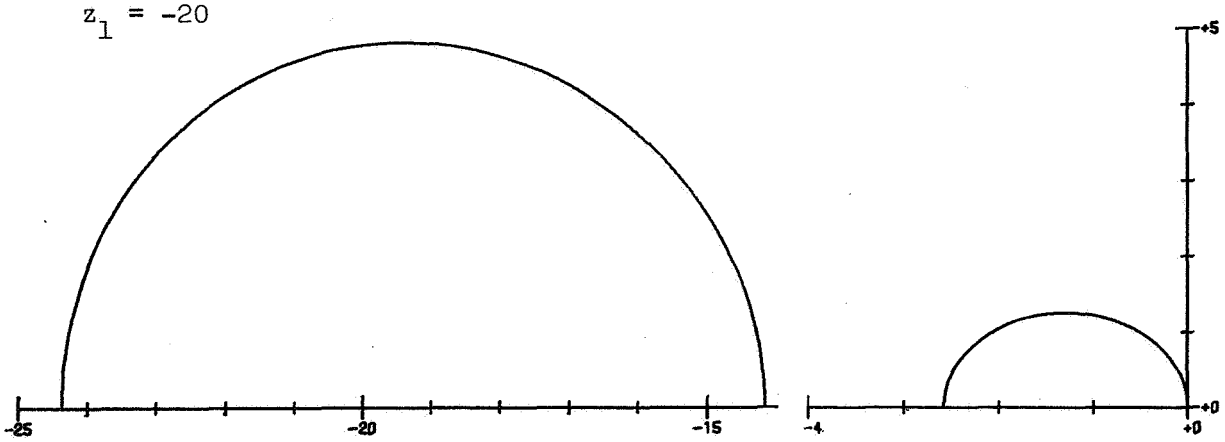


$z_1 = -5$

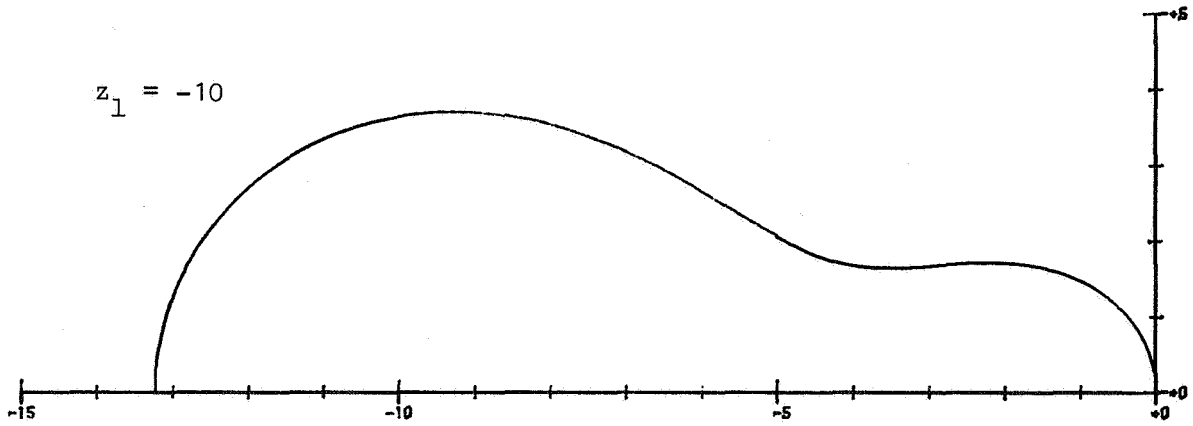


Stabiliteitsgebieden voor n = 3, r = 1.

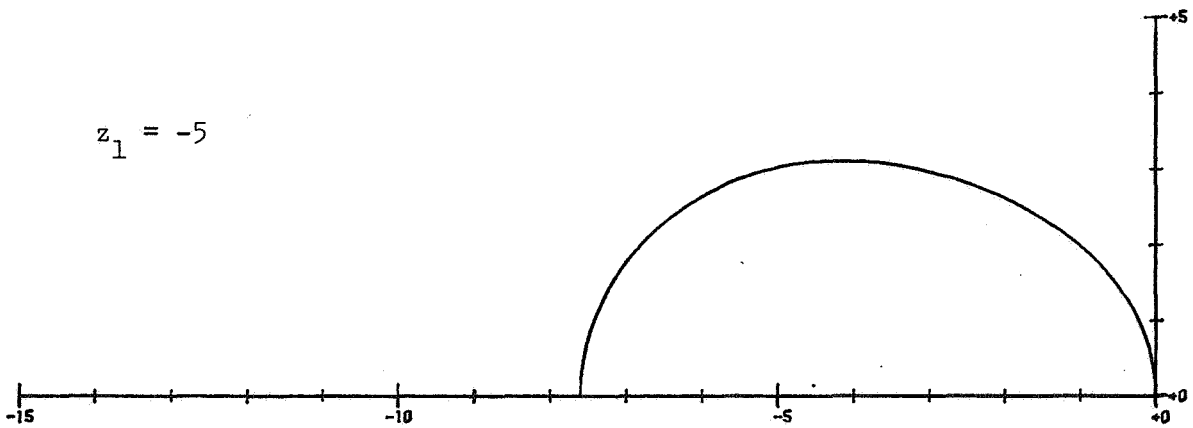
$z_1 = -20$



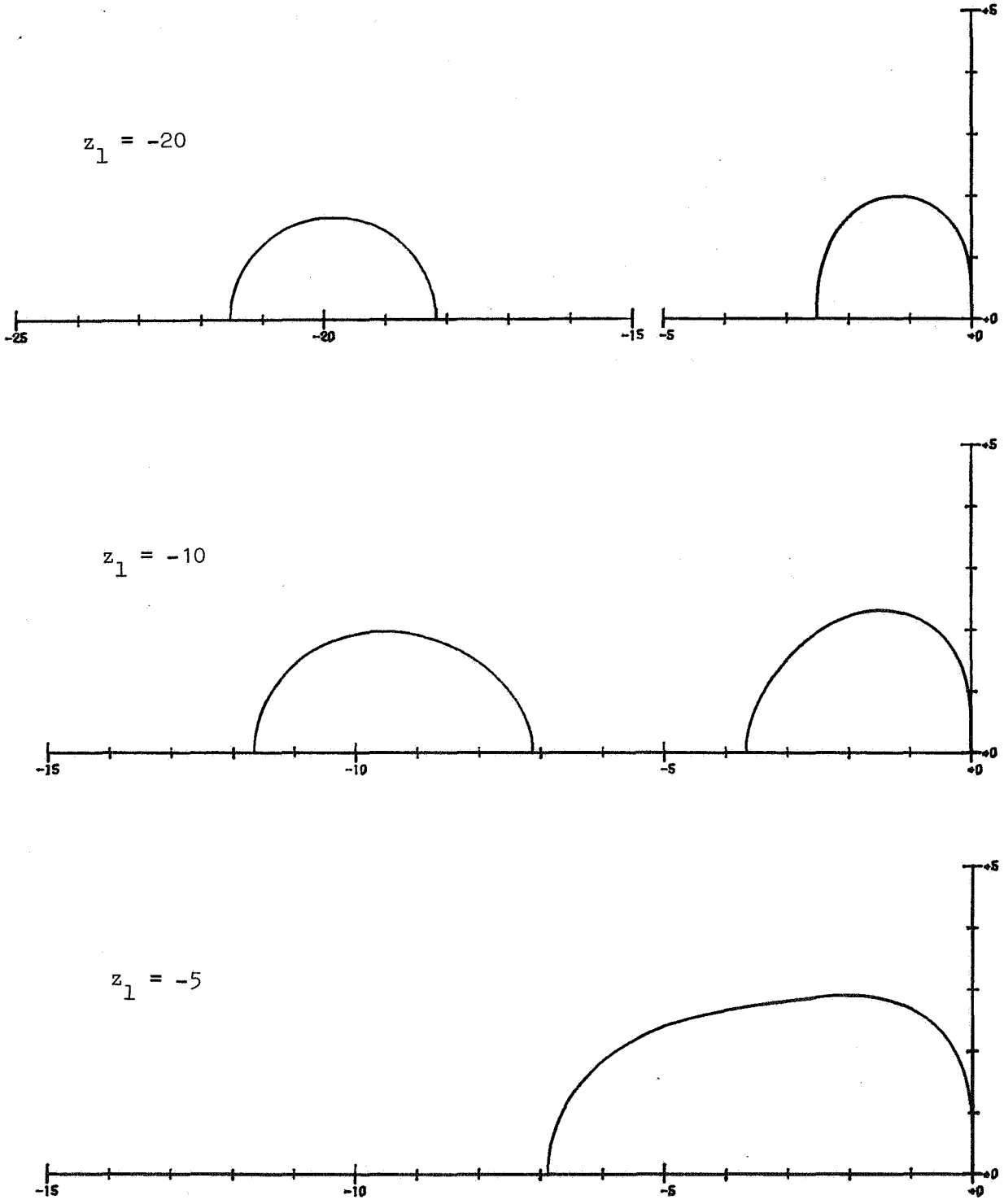
$z_1 = -10$



$z_1 = -5$

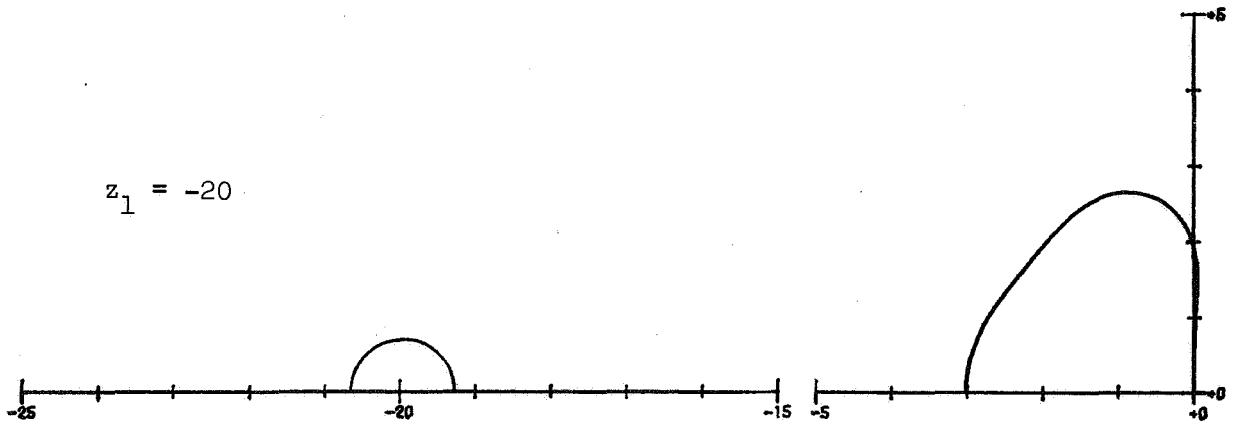


Stabiliteitsgebieden voor n = 4, r = 2.

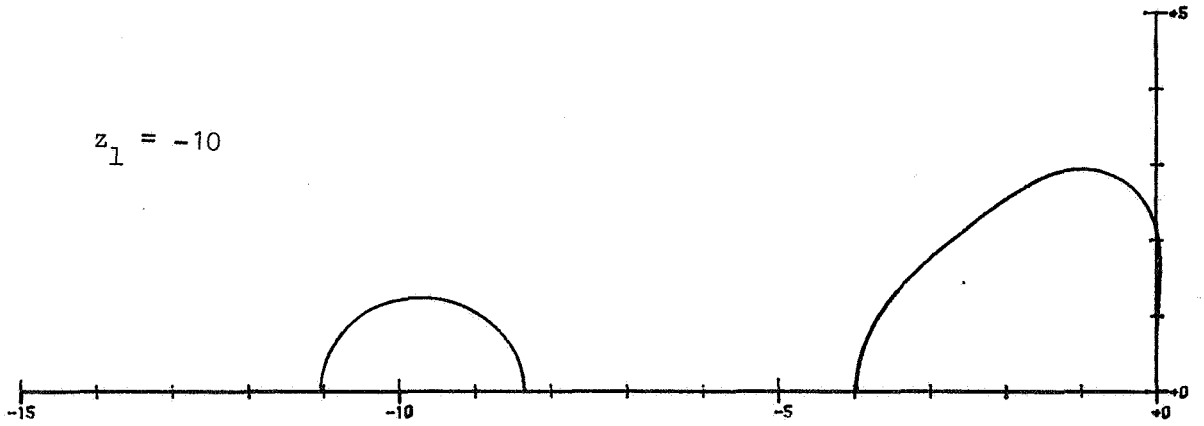


Stabiliteitsgebieden voor n = 5, r = 3.

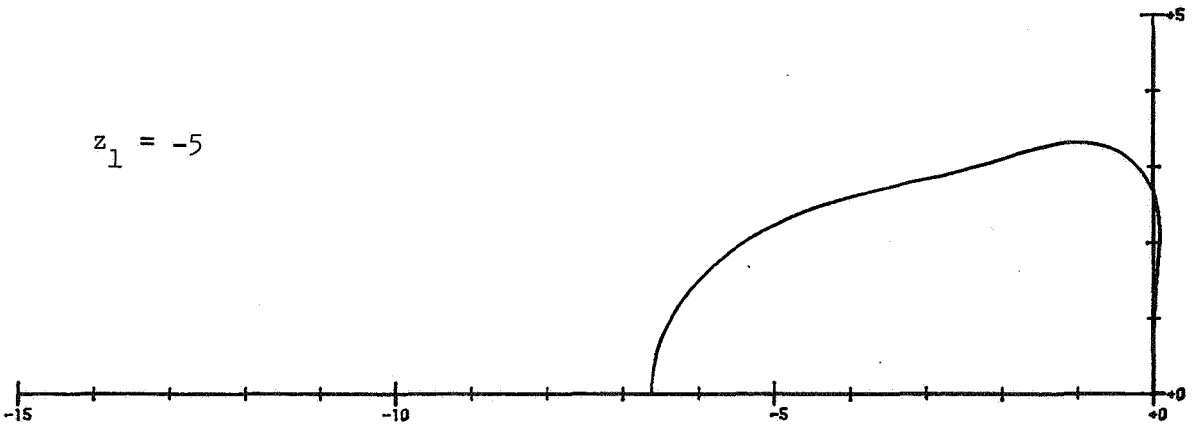
$z_1 = -20$



$z_1 = -10$



$z_1 = -5$



3. De procedure exponential fitted runge kutta

In de volgende paragraaf geven we de heading van exponential fitted runge kutta en de betekenis van de parameters.

3.1. Heading en parameters van exponential fitted runge kutta

```
procedure exponential fitted runge kutta (t, te, m0, m,
                    u, sigma, phi, diameter, derivative,
                    k, step, r, l, beta, thirdorder,
                    tol, output);
integer m0, m, k, r, l;
real t, te, sigma, phi, diameter, step, tol;
array u, beta;
boolean thirdorder;
procedure derivative, output;
```

Parameters:

t : <variable>;
t wordt gebruikt als Jensen parameter voor de onafhankelijke variabele; bij een aanroep van exponential fitted runge kutta moet t de waarde van de onafhankelijke variabele hebben;

te : <expression>;
de eindwaarde van t;

m0, m : <expression>;
indices van de eerste en de laatste vergelijking van het stelsel differentiaal-vergelijkingen;

u : een één-dimensionaal array u[m0:m];
bij een aanroep van exponential fitted runge kutta moet dit array de beginwaarden van $U(t_0)$ bevatten;

sigma : <expression>;
de modulus van het punt in het complexe vlak waarin de exponentiële aanpassing gewenst wordt;

phi : <expression>;
het argument van het complexe punt;

diameter : <expression>;
de diameter van de cluster in het linker halfvlak waarin exponentiële aanpassing gewenst wordt;

derivative : een procedure die als volgt door de gebruiker moet worden gegeven:
procedure derivative(x,v); real x; array v;

<body>;

na uitvoering van deze procedure moet het array v de componenten van H(t,U(t)) bevatten;

k : <variabele>;
telt het aantal integratiestappen;
bij de eerste aanroep van exponential fitted runge kutta moet k een door de gebruiker gegeven waarde hebben (bijvoorbeeld 0);
k kan worden gebruikt als Jensen variabele in de expressies voor te, step en in de procedure output;

step : <expression>;
geeft de lengte van de integratie-stap aan die de gebruiker zou willen nemen op grond van nauwkeurigheid;

r : <expression>;
de graad van het beginstuk van $P_n(z)$;

l : <expression>;
de orde van de exponentiële aanpassing;
de som van r en l is juist de graad van het stabiliteitspolynoom;
indien phi $\neq \pi$, dan moet l even zijn;

beta : een één-dimensionaal array beta[0:r+l];
bij een aanroep van exponential fitted runge kutta moet het array beta[0:r] de coëfficiënten van het beginstuk van $P_n(z)$ bevatten;

thirdorder : <expression>;
indien thirdorder de waarde true heeft, dan is de procedure 3-de orde exakt (mits het array beta[0:r] geschikt gekozen is), anders hoogstens 2-de orde exakt.

tol : <expression>;
de gewenste precisie in de resultaten;

output : een procedure die door de gebruiker moet worden gegeven;
procedure output;

<body>;

in deze procedure kan men uitvoer vragen van bijvoorbeeld de waarde van t, u[m0], ..., u[m], sigma, phi, diameter, k, beta[0], ..., beta[n], step.

In de volgende paragrafen geven we de body van exponential fitted runge kutta en een bespreking van de procedures die daarin voorkomen.

3.2. De body van exponential fitted runge kutta

```
procedure exponential fitted runge kutta(t, te, m0, m, u, sigma, phi,
                                         diameter, derivative, k,
                                         step, r, l, beta, thirdorder,
                                         tol, output);
integer m0, m, k, r, l;
real t, te, sigma, phi, diameter, step, tol;
array u, beta;
boolean thirdorder;
procedure derivative, output;

begin integer n;
    real theta0,thetanm1,tau,b,b0,phi0,phil,betar;
    boolean first,complex,change;
    integer array p[1:l];
    real array mu,labda[0:r l-1],pt[0:r],fac,betac[0:l-1],rl[m0:m],
              a[1:l,1:l];

    procedure formconstants;
    begin integer i;
        first:=false;
        fac[0]:=1;
        for i:=1 step 1 until l-1 do fac[i]:=i×fac[i-1];
        pt[r]:=l×fac[l-1];
        for i:=1 step 1 until r do pt[r-i]:=pt[r-i+1]×(l+i)/i
    end formconstants;
```

```
procedure formbeta;
begin integer i,j; real bb,c,d;
    if first then formconstants;
    d:=c:=exp(-b);
    betac[l-1]:=d/fac[l-1];
    for i:=1 step 1 until l-1 do
    begin c:=b×c/i; d:=d+c; betac[l-1-i]:=d/fac[l-1-i] end;
    bb:=1;
    for i:=r+1 step 1 until r+1 do
    begin c:=0;
        for j:=0 step 1 until r do
        c:=(beta[j]-(if j<l then betac[j] else 0))×pt[j]/(i-j)-c/b;
        beta[i]:=c/b/fac[l+r-i]/fac[i-r-1]+
                    (if i<l then bb×betac[i] else 0);
        bb:=bb×b
    end
end formbeta;

procedure solutionofcomplexequations;
begin integer i,j,c1,c3;
    real c2,e,b1,zii,cosphi,sinphi,cosiphi,siniphi,cosphil;
    real array d[1:l];

    procedure elements of matrix;
    begin phil:=phi0;
        cosphi:=cos(phil); sinphi:=sin(phil);
        cosiphi:=1; siniphi:=0;
        for i:=0 step 1 until l-1 do
        begin c1:=r+1+i; c2:=1;
            for j:=l-1 step -2 until 1 do
            begin a[j,l-i]:=c2×cosiphi;
                a[j+1,l-i]:=c2×siniphi;
                c2:=c1×c2; c1:=c1-1
            end;
            cosphil:=cosiphi×cosphi-siniphi×sinphi;
            siniphi:=cosiphi×sinphi+siniphi×cosphi;
            cosiphi:=cosphil
        end;
        det(a,l,p)
    end el of mat;

    procedure right hand side;
    begin e:=exp(b×cosphi);
        b1:=b×sinphi-(r+1)×phil;
        cosiphi:=e×cos(b1); siniphi:=e×sin(b1);
        b1:=1/b; zii:=b1↑r;
        for j:=l step -2 until 2 do
        begin d[j]:=zii×siniphi;
            d[j-1]:=zii×cosiphi;
            cosphil:=cosiphi×cosphi-siniphi×sinphi;
            siniphi:=cosiphi×sinphi+siniphi×cosphi;
            cosiphi:=cosphil;
            zii:=zii×b
        end;
```

```
        cosiphi:=zii:=1; siniphi:=0;
        for i:=r step -1 until 0 do
        begin c1:=i; c2:=beta[i];
            c3:=if 2xi>l-2 then 2 else l-2xi;
            cosphil:=cosiphixcosphi-siniphixsinphi;
            siniphi:=cosiphixsinphi+siniphixcosphi;
            cosiphi:=cosphil;
            for j:=l step -2 until c3 do
            begin d[j]:=d[j]+ziixc2xsiniphi;
                d[j-1]:=d[j-1]-ziixc2xcosiphi;
                c2:=c2xc1; c1:=c1-1
            end;
            zii:=ziixb1
        end
    end right hand side;

    if phi0≠phil then elements of matrix;
    right hand side;
    sol(a,l,p,d);
    for i:=1 step 1 until l do beta[r+i]:=d[l+1-i]xb1
end solofcomeq;

procedure coefficient;
begin integer j,k; real c;
    b0:=b; phi0:=phi;
    if b>1 then
    begin if complex then solution of complex equations
        else formbeta
    end;
    labda[0]:=mu[0]:=0;
    if thirdorder then
    begin theta0:=.25; thetanm1:=.75;
        if b<1 then
        begin c:=mu[n-1]:=2/3; labda[n-1]:=5/12;
            for j:=n-2 step -1 until 1 do
            begin c:=mu[j]:=c/(c-.25)/(n-j+1);
                labda[j]:=c-.25
            end
        end else
        begin c:=mu[n-1]:=beta[2]x4/3; labda[n-1]:=c-.25;
            for j:=n-2 step -1 until 1 do
            begin c:=mu[j]:=c/(c-.25)xbeta[n-j+1]/beta[n-j]/
                        (if j<1 then b else 1);
                labda[j]:=c-.25
            end
        end
    end else
    begin theta0:=0; thetanm1:=1;
        if b<1 then
        begin for j:=n-1 step -1 until 1 do mu[j]:=labda[j]:=1/(n-j+1)
        end else
        begin labda[n-1]:=mu[n-1]:=beta[2];
            for j:=n-2 step -1 until 1 do
            mu[j]:=labda[j]:=beta[n-j+1]/beta[n-j]/
                                    (if j<1 then b else 1)
        end
    end
end coefficient;
```

```
    procedure stepsize;
    begin real d,taustab,taustabint;
        tau:=step;
        complex:= abs(phi-3.1416)>.01;
        if complex∧even(l)=-1 then goto end of efrk;
        d:= if complex then (sigma/diameter/sin(phi))↑(.5×l/r)
            else (2×sigma/diameter)↑(l/r);
        taustab:=abs(d/betar/sigma);
        d:= if thirdorder then (2×10↑12×tol/beta[r]×4↑(l-1))↑(1/(n-1))
            else (10↑12×tol)↑(1/r)/betar;
        taustabint:= abs(d/sigma);
        if tau>taustab then tau:=taustab;
        if tau>taustabint then tau:=taustabint;
        if t+tau>te×(1-10↑-11) then tau:=te-t;
        if tau<t×10↑-12 then goto end of efrk;
        b:=tau×sigma; d:=.1×diameter×tau; d:=d×d;
        change:= b0=0 ∨ ((b-b0)×(b-b0)+b×b0×(phi-phi0)×(phi-phi0)>d)
    end stepsize;

    procedure difference scheme;
    begin integer i,j; real mt,lt,tht;
        i:=-1;
        next term:
        i:=i+1; mt:=mu[i]×tau; lt:=labda[i]×tau;
        for j:=m0 step 1 until m do rl[j]:=u[j]+lt×rl[j];
        derivative(t+mt,rl);
        if i=0∨i=n-1 then
        begin tht:=if i=0 then theta0×tau else thetanm1×tau;
            for j:=m0 step 1 until m do u[j]:=u[j]+tht×rl[j]
        end;
        if i<n-1 then goto next term;
        t:=t+tau
    end difference scheme;

    n:=r+1; first:=true; b0:=phi0:=0; betar:=beta[r]↑(1/r);
next level:
    stepsize;
    if change then coefficient;
    k:=k+1;
    difference scheme;
    output;
    if t<te then goto next level;
end of efrk:
end exponential fitted runge kutta;
```

3.3. De procedure formconstants

In deze procedure worden fac[j] = j! en pt[j] = $\frac{(r+1-j)!}{(r-j)!}$ berekend. Deze arrays worden gebruikt bij de berekening van beta[r+1], ..., beta[r+1].

3.4. De procedure formbeta

De coëfficiënten beta[r+1], ..., beta[r+1] van het stabiliteitspolynoom worden berekend volgens formule (2.9).

3.5. De procedure solution of complex equations

In deze procedure worden de coëfficiënten beta[r+1], ..., beta[r+1] berekend door het stelsel (2.10) op te lossen. De elementen van de matrix (a_{ij}) worden in de procedure elements of matrix bepaald, de rechterleden in right hand side.
De oplossing geschiedt met behulp van det en sol [10].

3.6. De procedure coefficient

De parameters θ_0, θ_{n-1}, λ_j en μ_j, $j = 1, 2, \ldots, n-1$, worden berekend volgens (2.11); we maken hierbij gebruik van de waarden van beta[0:n], die door solution of complex equation of formbeta berekend zijn.
Indien tau * sigma < 1, dan nemen we beta[j] = $\frac{1}{j!}$; (2.11) krijgt dan een bijzonder eenvoudige vorm.

3.7. De procedure stepsize

Deze procedure berekent de integratiestap, uitgaande van de waarde van step, zodat aan de voorwaarden (2.5) of (2.6) en (2.13) voldaan is. Voor de ε uit (2.13) is 10^{-12} genomen, de relatieve precisie van de Electrologica X8.
Bovendien wordt bepaald of z_1 zo veel is veranderd, dat een nieuwe berekening van de coëfficiënten nodig is. Als criterium hiervoor hebben we genomen $|dz_1| > .1 * \text{tau} * \text{diameter}$.

3.8. De procedure difference scheme

In deze procedure worden de waarden van u[j], $j = 0, \ldots, m$, de componenten van de numerieke oplossing u_k, vervangen door de componenten van u_{k+1}. Hierbij wordt derivative aangeroepen.

4. Testresultaten

In dit hoofdstuk bespreken we een drietal problemen waarmee we exponential fitted runge kutta hebben getest, te weten een lineair stelsel van Fowler en Warten [4], een derde-orde differentiaal-vergelijking en een niet-lineaire vergelijking. Resultaten met deze problemen van Taylor methode staan in [7] beschreven.

4.1. Een lineair stelsel van Fowler en Warten

Fowler en Warten behandelden als illustratie van een stijve differentiaal-vergelijking het beginwaarde-probleem

$$\dot{U} = D.U + F, \quad U(0) = U_0 \tag{4.1}$$

waarin

$$D = \begin{pmatrix} -500.5 & 499.5 \\ 499.5 & -500.5 \end{pmatrix}, \; F = \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \; U_0 = \begin{pmatrix} -.1 \\ .1 \end{pmatrix} .$$

De analytische oplossing van (4.1) luidt

$$U = 2(1-e^{-t}) \begin{pmatrix} 1 \\ 1 \end{pmatrix} r + .1\, e^{-1000t} \begin{pmatrix} -1 \\ 1 \end{pmatrix} .$$

De matrix D heeft de eigenwaarden $\delta_1 = -1000$ en $\delta_r = -1$. Zij liggen dus in twee ver verwijderde clusters in het linker halfvlak, terwijl bovendien de diameter van de linker cluster 0 is, zodat voorwaarde (2.6) geen beperking oplegt aan de staplengte. Een eerste orde aanpassing is daarom voldoende; verder kozen we r = 1, 2, 3, 4, 5 en beta[j] = $\frac{1}{j!}$, $j \leq r$. In dit speciale geval betekent dit, dat de methode r-de orde nauwkeurig is.
We hebben probleem (4.1) opgelost door integratie met constante staplengte over het interval [0,1]. In tabel 4.1 is een overzicht gegeven van $-{}^{10}\log \varepsilon$,

$$|\varepsilon| = \max_k |u_k[1] - \tilde{u}_k[1]|.$$

Bovendien zijn ter vergelijking enkele resultaten met andere exponentieel aangepaste methoden opgenomen, en wel met de formules $F^{(1)}$ en $F^{(2)}$ met één Newton iteratie van Liniger en Willoughby [2] en een derde-orde gegeneraliseerde Runge-Kutta methode van Lawson [3].

Tabel 4.1

Resultaten met exponentieel aangepaste methoden

methode	aantal f.e. per stap	staplengte						
		1	.5	.2	.1	.05	.02	.01
efrk r=1, l=1	2	.1	.6	1.1	1.4	1.7	2.2	2.5
r=2, l=1	3	.6	1.3	2.2	2.9	3.5	4.4	5.0
r=3, l=1	4	1.2	2.2	3.5	4.5	5.4	6.7	7.3
r=4, l=1	5	1.9	3.1	4.1	5.5	6.8	8.0	9.3
r=5, l=1	6	-.2	1.1	3.2	4.1	5.7	7.5	8.0
$F^{(1)}$	2				1.5	1.8	2.2	2.5
$F^{(2)}$	3				9.5	10.1	10.9	10.1
Lawson	4				5.3	6.2	7.4	8.3

Bij de methode van Liniger en Willoughby is een Jacobiaan evaluatie voor een evaluatie geteld.

In tabel 4.2 geven we de resultaten met de Runge-Kutta formules van Zonneveld [11] en Baanstra [12] van orde p, p = 2, 3, 4, 5. Deze methoden werken met variabele staplengte.

Tabel 4.2

Resultaten met Runge-Kutta formules van Zonneveld en Baanstra

orde	totaal aantal f.e.	$-^{10}\log \varepsilon$	totaal aantal f.e.	$-^{10}\log \varepsilon$
2	3232	2.0	24064	4.8
3	4072	2.1	6044	5.0
4	4880	2.5	5290	5.4
5	3424	2.8	3834	5.7

Uit bovenstaande tabellen blijkt, dat de exponentieel aangepaste methoden vele malen sneller zijn dan de gewone Runge-Kutta methoden. De eenvoud van het stelsel laat de voordelen van exponentiële aanpassing echter wel duidelijker uitkomen.

De onnauwkeurige resultaten bij de zes-punts exponential fitted runge kutta wordt veroorzaakt door interne instabiliteit. Uit (2.12) volgt namelijk

$$M(z) \simeq \frac{5}{6} \tau^5 . 10^{13}.$$

De door interne instabiliteit veroorzaakte fout is van de grootte

$$M(z)\ 10^{-12}/u[1] \simeq \tau^5.$$

Uit paragraaf 2.7 blijkt tevens, dat de interne instabiliteit bij $\theta_{n-1} = 1$ niet toeneemt met grotere waarden van 1, bij $\theta_{n-1} = \frac{3}{4}$ echter wel.
Dit verschijnsel wordt duidelijk geïllustreerd in tabel 4.3; hierbij is weer met constante staplengte geïntegreerd van 0 tot 1.

Tabel 4.3

Interne instabiliteit van exponential fitted runge kutta met r = 3

Overzicht van $-^{10}\log \varepsilon$

l	$\theta_{n-1} = 1$		$\theta_{n-1} = \frac{3}{4}$	
	stap=1	stap=.1	stap=1	stap=.1
1	1.2	4.5	1.2	4.5
2	1.2	4.5	.2	4.5
3	1.2	4.5	- .2	4.1
4	1.2	4.6	-3.9	2.9
5	1.2	4.6	-6.0	1.9
6	1.2	4.5	-8.7	-8.7

Deze instabiliteit treedt niet op indien tol een voldoende kleine waarde heeft.

Voor tol = 1/60 (en $\theta_{n-1} = \frac{3}{4}$) leidde dit tot de in tabel 4.4 opgenomen resultaten. De staplengte wordt hierbij bepaald door formule (2.13).

Tabel 4.4

Interne instabiliteit, tol = 1/60, r = 3 en $\theta_{n-1} = \frac{3}{4}$

l	staplengte	$-^{10}\log \varepsilon$
4	.121	2.6
5	.058	3.2
6	.034	4.1

Voor de volledigheid geven we de aanroep van
exponential fitted runge kutta:

```
k:= 0;
exponential fitted runge kutta (t, 1, 1, 2, u, 1000, 3.14, 0,
                                derivative, k, step, r, l, beta,
                                thirdorder, tol, output);
```

Het array beta[0:r] had hierbij de waarden $[1,\ldots,\frac{1}{r!}]$, tol bezat de waarde 10^{10} (behalve in het geval van tabel 4.4) en thirdorder was false (behalve waar $\theta_{n-1} = \frac{3}{4}$).

4.2. Een derde orde differentiaal-vergelijking

Beschouw het beginwaarde-probleem

$$(4.2)\qquad \begin{cases} \dddot{u} + (1-2r_o\cos\phi)\ddot{u} + r_o(r_o-2\cos\phi)\dot{u} + r_o^2 u = 0, \\ u(0) = 1,\ \dot{u}(0) = 0,\ \ddot{u}(0) = 0, \end{cases}$$

waarin r_o en ϕ gegeven parameters zijn.

Dit probleem kan geschreven worden in de equivalente vorm

$$(4.2')\qquad \begin{cases} \dot{\vec{u}} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -r_o^2 & -r_o(r_o-2\cos\phi) & 2r_o\cos\phi-1 \end{pmatrix} \vec{u}, \\ \vec{u}(0) = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \end{cases}$$

waarbij u, $\dot{u}$ en $\ddot{u}$ de componenten van $\vec{u}$ zijn.

De eigenwaarden van de Jacobiaan van (4.2') zijn

$$-1,\ r_o e^{i\phi} \quad \text{en} \quad r_o e^{-i\phi},$$

zodat de drie-cluster methode voor grote waarden van r_o en $\pi/2 \leq \phi \leq \pi$ een geschikte integratie-methode is voor probleem (4.2').
In tabel 4.5 geven we de resultaten van de integratie over het interval [0,1] met exponential fitted runge kutta. Getabelleerd is $-{}^{10}\log \max_k |u_k[1] - \tilde{u}_k[1]|$, als beginstuk voor het stabiliteitspolynoom is genomen $1 + z + \ldots + \frac{z^r}{r!}$.

Tabel 4.5

drie-cluster-methode, toegepast op probleem (4.3')
met r = 1000 en $\phi = 2\pi/3$. Overzicht van $-^{10}\log \varepsilon$

r \ stap	1	.5	.2	.1	.05	.025	.01
1	.4	.9	1.4	1.7	2.0	2.4	2.8
2	.9	1.6	2.6	3.2	3.8	4.5	5.4
3	1.5	2.5	3.9	4.8	5.8	6.7	8.0
4	2.2	3.5	5.2	6.5	7.7	9.0	10.7
5	.6	-1.2	3.5	6.0	8.0	9.9	11.3

De aanroep van de procedure exponential fitted runge kutta luidde:

```
k:= 0;
exponential fitted runge kutta (t, 1, 1, 3, u, 1000, 2.0943951, 0,
                                derivative, k, step, r, 2, beta, r≥3,
                                10^10, output);
```

De drie-cluster-methode met l = 2 is zeer gevoelig voor storingen in de eigenwaarde met groot negatief reëel deel, zoals uit formule (2.6) volgt.
Een aanroep van exponential fitted runge kutta, waarbij phi de waarde 2.0944 had, leidde tot een fout in u[1] op t = 1 van 10^{+20} (r=5, step=.1). Deze instabiliteit treedt niet op wanneer diameter de juiste waarde heeft, in dit geval 2 * 1000 * dphi = 10^{-2}. De maximale door stabiliteit toegelaten stap is dan .0268 en $\max_k |u_k[1] - \tilde{u}_k[1]|$ is $3.9 * 10^{-10}$.

4.3. De vergelijking $\dot{u} = -e^t U + e^t \ln t + 1/t$

Beschouw het beginwaarde-probleem

$$(4.3)\qquad \begin{cases} \dfrac{dU}{dt} = -e^t U + e^t \ln t + \dfrac{1}{t}, & t \geq 10^{-2}, \\ u(10^{-2}) = \ln(10^{-2}). \end{cases}$$

Het is duidelijk, dat dit probleem als oplossing heeft

$$U(t) = \ln(t).$$

De eigenwaarde van de Jacobiaan van het beschouwde probleem, -exp(t), is in dit geval een functie van t; voor t > 3 wordt de differentiaal-vergelijking in toenemende mate stijf. Het ligt daarom voor de hand een exponentiële methode te gebruiken. Wij hebben exponential fitted runge kutta op probleem (3.3) toegepast.
Doordat de differentiaal-vergelijking niet lineair is, gaat de stabiliteits conditie (2.5) een belangrijke rol spelen. We eisen namelijk dat de eigenwaarden aan het begin en aan het eind van een stap ter lengte τ binnen het stabiliteitsgebied ligt. Na enig rekenen geeft dit de voorwaarde

$$\tau \leq e^{-\frac{r}{1+r}t} \beta_r^{-\frac{1}{1+r}},$$

dus (daar sigma = e^t),

$$\text{diameter} = 2(\text{sigma}^l / \beta_r)^{\frac{1}{l+r}}.$$

In tabel 4.6 is een overzicht van de effectieve staplengte bij t = 4,6 (dit is staplengte / aantal functie evaluaties per stap) gegeven, voor enige waarden van r en l.
Uit deze tabel blijkt dat voor grote waarden van t een lage orde formule met een hoge orde exponentiële aanpassing het meest efficiënt is. In de beginfase, waar de vergelijking niet stijf is, is omwille van de nauwkeurigheid een hoge orde formule geschikter.
Wij hebben daarom probleem (4.3) op het interval [.01,8] opgelost met verschillende formules:

op [.01,2] kozen we r = 4, l = 1, op [2,4] r = 3, l = 2,

op [4,6] r = 2, l = 3 en op [6,8] r = 1, l = 4.

Tabel 4.6

effectieve staplengte bij probleem (4.3)

	r \ l	1	2	3	4	5
t = 4	1	.068	.088	.092	.090	.085
	2	.023	.034	.040	.044	.046
	3	.012	.018	.023	.026	.028
	4	.008	.012	.014	.017	.019
t = 6	1	.025	.045	.056	.060	.061
	2	.008	.015	.021	.025	.029
	3	.004	.008	.011	.014	.016
	4	.003	.005	.007	.009	.011

In tabel 4.7 zijn de resultaten opgenomen van de volgende aanroepen van de procedure exponential fitted runge kutta:

```
r:= 4; l:= 1; te:= 2; k:= 0;
again;  output;
exponential fitted runge kutta (t, te, 1, 1, u, sigma, 3.14, diameter,
                                derivative, k, t, r, l, beta, r≥3, 10^-4,
                                output);
r:= r-1; l:= l+1; te:= te+2;
if te < 8 then goto again;
```

De waarden van sigma en diameter werden in de procedure output berekend; hiermee bereiken we dat deze expressies precies één maal per stap worden geëvalueerd. Het array beta[0:r] was gevuld met de waarden [1,...,1/r!].

Tabel 4.7

twee-cluster-methode toegepast op probleem (4.3)

k	t_k	τ_k	$\varepsilon_k = \lvert\tilde{u}_k - u_k\rvert$	sigma
5	.320	.320	$2.5\ 10^{-2}$	1.38
10	2.431	.333	$2.6\ 10^{-4}$	11.4
15	3.648	.160	$7.0\ 10^{-4}$	38.4
20	4.443	.194	$6.4\ 10^{-4}$	85
25	5.288	.139	$5.9\ 10^{-4}$	198
30	5.913	.087	$4.3\ 10^{-4}$	370
35	7.107	.241	$3.8\ 10^{-5}$	1220
39	8.000		$2.5\ 10^{-4}$	2981

Exponential fitted runge kutta is met bovenstaande aanroep compatibel met de drie-cluster-methode met variabele staplengte [7].
De volgende strategie, die gebaseerd is op de overweging dat de afbreek fout van de orde van grootte $h^{r+1}/(r+1)!$ zal zijn, levert aanzienlijk betere resultaten op:

S: zoek voor iedere nieuwe integratiestap de maximale staplengte h en de bijbehorende waarde van r (bij l = 5-r), zodat aan de voorwaarden

$$\frac{1}{(r+1)!}\, h^{r+1} \leq \text{tol}$$

en

$$h \leq h_{stab}$$

voldaan is, en pas de procedure met deze waarde van h en r toe.

De met deze strategie verkregen resultaten zijn in tabel 4.8 opgenomen.

Tabel 4.8

twee-cluster-methode met strategie S

tol	r	k	t_k	$\varepsilon_k = \lvert\tilde{u}_k - u_k\rvert$	$\lvert\varepsilon\rvert_\infty$
1	4	1	.024	$1.7\ 10^{-2}$	$5.7\ 10^{-2}$
	3	2	.057	$3.4\ 10^{-2}$	
	2	6	1.659	$1.4\ 10^{-2}$	
	1	24	8.000	$8.7\ 10^{-6}$	
10^{-2}	4	13	2.321	$1.4\ 10^{-3}$	$6.1\ 10^{-3}$
	3	14	2.677	$4.7\ 10^{-4}$	
	2	16	3.407	$3.3\ 10^{-4}$	
	1	31	8.000	$8.0\ 10^{-5}$	
10^{-4}	4	29	2.709	$6.0\ 10^{-4}$	$1.1\ 10^{-3}$
	3	32	3.434	$6.7\ 10^{-4}$	
	2	52	6.467	$5.9\ 10^{-4}$	
	1	68	8.000	$2.0\ 10^{-6}$	

Referenties

[1] D.A. Pope — An exponential method of numerical integration of ordinary differential equations, Communications of the ACM, Numerical analysis, vol. 6, no. 8, (1963).

[2] W. Liniger en R. Willoughby — Efficient integration of stiff systems of ordinary differential equations, IBM Research Report RC 1970, (1967).

[3] J.D. Lawson — Generalized Runge-Kutta processes for stable systems with large Lipschitz constants, SIAM J. Numer. Anal., vol. 4, no. 3, (1967).

[4] M.E. Fowler en R.M. Warten — A numerical integration technique for ordinary differential-equations with widely separated eigenvalues, IBM Journal, 537-543, (1967).

[5] P.J. van der Houwen — One step methods for linear initial value problems I. (TW 119) M.C.

[6] P.J. van der Houwen — One step methods for linear initial value problems II. (TW 122) M.C.

[7] P.J. van der Houwen — One step methods for linear initial value problems III, Numerical examples (TW 130) M.C.

[8] P.J. van der Houwen — Stabilized Runge-Kutta methods with limited storage requirements (TW 124) M.C.

[9] P.J. van der Houwen — A survey of stabilized Runge-Kutta formulae (MC-25 informatica symposium).

[10] T.J. Dekker — ALGOL 60-procedures in numerical algebra 1 (MC tracts 22).

[11] J.A. Zonneveld — Automatic numerical integration. (MC tracts 8).

[12] W.E. Baanstra — Methoden voor het oplossen van beginwaarde problemen (scriptie).