A family of self-starting Extended Backward Differentiation Formulas (BDFs) with two superfuture points for stiff initial value problems

R.D. French S.N. Jator

Department of Mathematics and Statistics
Austin Peay State University

SIAM SEAS 2013 Annual Meeting

Outline

1. Motivation
   - Solving Stiff IVPs
   - The Extended BDF of Cash
   - The Modified Extended BDF of Cash
   - Inspiration for Hyper-Extended BDF
2. Derivation and Implementation
   - Derivation
   - Implementation
3. Results and Summary
   - Numerical Results
   - Summary

Outline

1. Motivation
   - Solving Stiff IVPs
   - The Extended BDF of Cash
   - The Modified Extended BDF of Cash
   - Inspiration for Hyper-Extended BDF
2. Derivation and Implementation
   - Derivation
   - Implementation
3. Results and Summary
   - Numerical Results
   - Summary

Outline

Formal Statement of Problem

- We are interested integrating "Stiff" IVPs
- These problems have the form

$$\frac{dy}{dx} = f(x, y), \;\; y(x_0) = y_0 \tag{1}$$

- They are defined on the finite interval $[a, b]$
- $y, f \in \mathbb{R}^m$
- $f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$

Formal Statement of Problem

- We are interested integrating “Stiff” IVPs
- These problems have the form

$$\frac{dy}{dx} = f(x, y),\ \ y(x_0) = y_0 \tag{1}$$

- They are defined on the finite interval $[a, b]$
- $y, f \in \mathbb{R}^m$
- $f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$

Formal Statement of Problem

- We are interested integrating “Stiff” IVPs
- These problems have the form

$$\frac{dy}{dx} = f(x, y), \;\; y(x_0) = y_0 \tag{1}$$

- They are defined on the finite interval $[a, b]$
- $y, f \in \mathbb{R}^m$
- $f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$

Formal Statement of Problem

- We are interested integrating "Stiff" IVPs
- These problems have the form

$$\frac{dy}{dx} = f(x, y), \;\; y(x_0) = y_0 \tag{1}$$

- They are defined on the finite interval $[a, b]$
- $y, f \in \mathbb{R}^m$
- $f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$

Formal Statement of Problem

- We are interested integrating "Stiff" IVPs
- These problems have the form

$$\frac{dy}{dx} = f(x, y), \;\; y(x_0) = y_0 \tag{1}$$

- They are defined on the finite interval $[a, b]$
- $y, f \in \mathbb{R}^m$
- $f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$

Definition of Stiffness

- No agreed definition of "Stiffness"
- We use the definition given by Ramos and Vigo-Aguiar
- They define (1) as stiff if its Jacobian has eigenvalues λ_i with $Re(\lambda_i) \leq 0$, in addition to eigenvalues of moderate size.

Definition of Stiffness

- No agreed definition of "Stiffness"
- We use the definition given by Ramos and Vigo-Aguiar
- They define (1) as stiff if its Jacobian has eigenvalues λ_i with $Re(\lambda_i) \leq 0$, in addition to eigenvalues of moderate size.

Definition of Stiffness

- No agreed definition of "Stiffness"
- We use the definition given by Ramos and Vigo-Aguiar
- They define (1) as stiff if its Jacobian has eigenvalues λ_i with $Re(\lambda_i) \leq 0$, in addition to eigenvalues of moderate size.

BDFs solve Stiff problems well

- Gear's Method (Conventional BDF) performed well on stiff IVPs
- Cash's Extended BDF adds a "Superfuture" point
  - Colocation point beyond x_{n+k}
  - Enhanced stability properties
- We investigate the effects of a second Superfuture point

BDFs solve Stiff problems well

- Gear's Method (Conventional BDF) performed well on stiff IVPs
- Cash's Extended BDF adds a "Superfuture" point
  - Colocation point beyond x_{n+k}
  - Enhanced stability properties
- We investigate the effects of a second Superfuture point

BDFs solve Stiff problems well

- Gear's Method (Conventional BDF) performed well on stiff IVPs
- Cash's Extended BDF adds a "Superfuture" point
  - Colocation point beyond x_{n+k}
  - Enhanced stability properties
- We investigate the effects of a second Superfuture point

BDFs solve Stiff problems well

- Gear's Method (Conventional BDF) performed well on stiff IVPs
- Cash's Extended BDF adds a "Superfuture" point
  - Colocation point beyond x_{n+k}
  - Enhanced stability properties
- We investigate the effects of a second Superfuture point

BDFs solve Stiff problems well

- Gear's Method (Conventional BDF) performed well on stiff IVPs
- Cash's Extended BDF adds a "Superfuture" point
  - Colocation point beyond x_{n+k}
  - Enhanced stability properties
- We investigate the effects of a second Superfuture point

The Conventional BDF

A-stable only for $k = 1$

- Also called "Gear's Method"
- Takes the form

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h\beta_k f_{n+k} \tag{2}$$

The Conventional BDF

A-stable only for $k = 1$

- Also called "Gear's Method"
- Takes the form

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h\beta_k f_{n+k} \tag{2}$$

The Extended BDF of Cash

A-stable for $k \leq 3$

- Cash introduced a colocation point at x_{n+k+1}
- This "Extended" BDF takes the form

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h\beta_k f_{n+k} + h\beta_{k+1} f_{n+k+1} \tag{3}$$

- *A*-stable for $k \leq 3$

The Extended BDF of Cash

A-stable for $k \leq 3$

- Cash introduced a colocation point at x_{n+k+1}
- This "Extended" BDF takes the form

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h\beta_k f_{n+k} + h\beta_{k+1} f_{n+k+1} \tag{3}$$

- *A*-stable for $k \leq 3$

The Extended BDF of Cash

A-stable for $k \leq 3$

- Cash introduced a colocation point at x_{n+k+1}
- This "Extended" BDF takes the form

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h\beta_k f_{n+k} + h\beta_{k+1} f_{n+k+1} \tag{3}$$

- A-stable for $k \leq 3$

The Extended BDF of Cash

Notes on Implementation

- Gear's Method is used to predict y^*_{n+k}
- Then the Extended BDF is solved as a corrector via Newton's Method

The Extended BDF of Cash

Notes on Implementation

- Gear's Method is used to predict y^*_{n+k}
- Then the Extended BDF is solved as a corrector via Newton's Method

The Modified Extended BDF of Cash

This time with 2 Superfuture Points

- Cash analyzed a BDF with two superfuture points of the form

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h\beta_k f_{n+k} + h\beta_{k+1} f_{n+k+1} + h\beta_{k+2} f_{n+k+2} \tag{4}$$

- Iterative implementation means 2 matrix factorizations per step
- Question: Possible to salvage method with different strategy?

The Modified Extended BDF of Cash

This time with 2 Superfuture Points

- Cash analyzed a BDF with two superfuture points of the form

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h\beta_k f_{n+k} + h\beta_{k+1} f_{n+k+1} + h\beta_{k+2} f_{n+k+2} \qquad (4)$$

- Iterative implementation means 2 matrix factorizations per step
- Question: Possible to salvage method with different strategy?

The Modified Extended BDF of Cash

This time with 2 Superfuture Points

- Cash analyzed a BDF with two superfuture points of the form

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h\beta_k f_{n+k} + h\beta_{k+1} f_{n+k+1} + h\beta_{k+2} f_{n+k+2} \tag{4}$$

- Iterative implementation means 2 matrix factorizations per step
- Question: Possible to salvage method with different strategy?

Repurposing the MEBDF

Using a Self-Starting Predictor/Corrector Strategy

- Replace iterative solving with Predictor/Corrector strategy
  - Use a $(k+2)$-step Adams-Bashforth method as an explicit predictor
- Rederive MEBDF as a continuous scheme (As discussed in Jator, Swindell, and French)
  - This will allow the method to be self-starting method
- Together, those should overcome the problems encountered by Cash

Repurposing the MEBDF

Using a Self-Starting Predictor/Corrector Strategy

- Replace iterative solving with Predictor/Corrector strategy
  - Use a $(k+2)$-step Adams-Bashforth method as an explicit predictor
- Rederive MEBDF as a continuous scheme (As discussed in Jator, Swindell, and French)
  - This will allow the method to be self-starting method
- Together, those should overcome the problems encountered by Cash

Repurposing the MEBDF

Using a Self-Starting Predictor/Corrector Strategy

- Replace iterative solving with Predictor/Corrector strategy
  - Use a $(k+2)$-step Adams-Bashforth method as an explicit predictor
- Rederive MEBDF as a continuous scheme (As discussed in Jator, Swindell, and French)
  - This will allow the method to be self-starting method
- Together, those should overcome the problems encountered by Cash

Repurposing the MEBDF

Using a Self-Starting Predictor/Corrector Strategy

- Replace iterative solving with Predictor/Corrector strategy
  - Use a $(k+2)$-step Adams-Bashforth method as an explicit predictor
- Rederive MEBDF as a continuous scheme (As discussed in Jator, Swindell, and French)
  - This will allow the method to be self-starting method
- Together, those should overcome the problems encountered by Cash

Repurposing the MEBDF

Using a Self-Starting Predictor/Corrector Strategy

- Replace iterative solving with Predictor/Corrector strategy
  - Use a $(k+2)$-step Adams-Bashforth method as an explicit predictor
- Rederive MEBDF as a continuous scheme (As discussed in Jator, Swindell, and French)
  - This will allow the method to be self-starting method
- Together, those should overcome the problems encountered by Cash

Deriving the Continuous Scheme

- Start with an approximation $U(x) = a_0 + a_1 x + \cdots + a_{k+2}x^{k+2}$ and $V = (y_0, y_1, \ldots, y_{k-1}, f_k, f_{k+1}, f_{k+2})$
- Want to solve for a_i
- Form a matrix M from coefficients of U and U' evaluated at x_i
- Compute $M^{-1} \cdot V$ to find values of a_i
- Discrete methods can be obtained by evaluating $U(x)$ at appropriate points
- Proof of Algorithm given in Jator

Deriving the Continuous Scheme

- Start with an approximation $U(x) = a_0 + a_1 x + \cdots + a_{k+2} x^{k+2}$ and $V = (y_0, y_1, \ldots, y_{k-1}, f_k, f_{k+1}, f_{k+2})$
- Want to solve for a_i
- Form a matrix M from coefficients of U and U' evaluated at x_i
- Compute $M^{-1} \cdot V$ to find values of a_i
- Discrete methods can be obtained by evaluating $U(x)$ at appropriate points
- Proof of Algorithm given in Jator

Deriving the Continuous Scheme

- Start with an approximation $U(x) = a_0 + a_1 x + \cdots + a_{k+2} x^{k+2}$ and $V = (y_0, y_1, \ldots, y_{k-1}, f_k, f_{k+1}, f_{k+2})$
- Want to solve for a_i
- Form a matrix M from coefficients of U and U' evaluated at x_i
- Compute $M^{-1} \cdot V$ to find values of a_i
- Discrete methods can be obtained by evaluating $U(x)$ at appropriate points
- Proof of Algorithm given in Jator

Deriving the Continuous Scheme

- Start with an approximation $U(x) = a_0 + a_1 x + \cdots + a_{k+2} x^{k+2}$ and $V = (y_0, y_1, \ldots, y_{k-1}, f_k, f_{k+1}, f_{k+2})$
- Want to solve for a_i
- Form a matrix M from coefficients of U and U' evaluated at x_i
- Compute $M^{-1} \cdot V$ to find values of a_i
- Discrete methods can be obtained by evaluating $U(x)$ at appropriate points
- Proof of Algorithm given in Jator

Deriving the Continuous Scheme

- Start with an approximation $U(x) = a_0 + a_1 x + \cdots + a_{k+2} x^{k+2}$ and $V = (y_0, y_1, \ldots, y_{k-1}, f_k, f_{k+1}, f_{k+2})$
- Want to solve for a_i
- Form a matrix M from coefficients of U and U' evaluated at x_i
- Compute $M^{-1} \cdot V$ to find values of a_i
- Discrete methods can be obtained by evaluating $U(x)$ at appropriate points
- Proof of Algorithm given in Jator

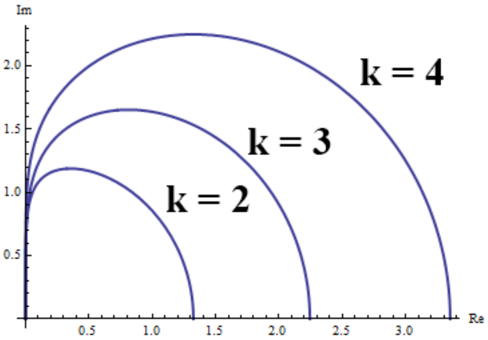
Austin Peay State University

Deriving the Continuous Scheme

- Start with an approximation $U(x) = a_0 + a_1x + \cdots + a_{k+2}x^{k+2}$ and $V = (y_0, y_1, \ldots, y_{k-1}, f_k, f_{k+1}, f_{k+2})$
- Want to solve for a_i
- Form a matrix M from coefficients of U and U' evaluated at x_i
- Compute $M^{-1} \cdot V$ to find values of a_i
- Discrete methods can be obtained by evaluating $U(x)$ at appropriate points
- Proof of Algorithm given in Jator

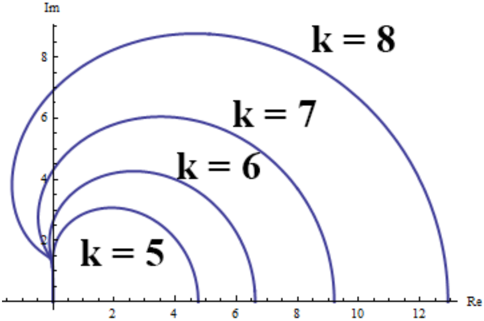
Notes on Stability

A-stable for $k = 2, 3, 4$



Notes on Stability

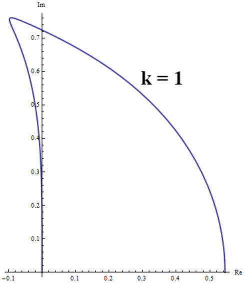
$A(\alpha)$-stable for $k = 5, 6, 7, 8$



Nearly A-stable for $k = 5$ ($\alpha > 89.9°$)

Notes on stability

Poor Stability for $k = 1$



This was unanticipated

Solving the Initial Block

- Discrete methods obtained from $U(x)$ can be solved as a system
- This will yield the first $k+2$ solution points $y_1, y_2, \ldots, y_{k+2}$
- The remainder of the interval can be solved with Predictor/Corrector

Solving the Initial Block

- Discrete methods obtained from $U(x)$ can be solved as a system
- This will yield the first $k+2$ solution points $y_1, y_2, \ldots, y_{k+2}$
- The remainder of the interval can be solved with Predictor/Corrector

Solving the Initial Block

- Discrete methods obtained from $U(x)$ can be solved as a system
- This will yield the first $k+2$ solution points $y_1, y_2, \ldots, y_{k+2}$
- The remainder of the interval can be solved with Predictor/Corrector

Employing the Predictor/Corrector Strategy

- Because our method is implicit, we need to predict y_n^*,y_{n+1}^*, and y_{n+2}^* in order to find y_n
- We use $(k+2)$-step Adams-Bashforth to predict y_n^*
  - Then again to predict y_{n+1}^*
  - Then again to predict y_{n+2}^*
- Once these values exist, we may compute y_n via our method

Employing the Predictor/Corrector Strategy

- Because our method is implicit, we need to predict y_n^*,y_{n+1}^*, and y_{n+2}^* in order to find y_n
- We use $(k+2)$-step Adams-Bashforth to predict y_n^*
  - Then again to predict y_{n+1}^*
  - Then again to predict y_{n+2}^*
- Once these values exist, we may compute y_n via our method

Employing the Predictor/Corrector Strategy

- Because our method is implicit, we need to predict y_n^*,y_{n+1}^*, and y_{n+2}^* in order to find y_n
- We use $(k+2)$-step Adams-Bashforth to predict y_n^*
  - Then again to predict y_{n+1}^*
  - Then again to predict y_{n+2}^*
- Once these values exist, we may compute y_n via our method

Employing the Predictor/Corrector Strategy

- Because our method is implicit, we need to predict y_n^*,y_{n+1}^*, and y_{n+2}^* in order to find y_n
- We use $(k+2)$-step Adams-Bashforth to predict y_n^*
  - Then again to predict y_{n+1}^*
  - Then again to predict y_{n+2}^*
- Once these values exist, we may compute y_n via our method

Employing the Predictor/Corrector Strategy

- Because our method is implicit, we need to predict y_n^*,y_{n+1}^*, and y_{n+2}^* in order to find y_n
- We use $(k+2)$-step Adams-Bashforth to predict y_n^*
  - Then again to predict y_{n+1}^*
  - Then again to predict y_{n+2}^*
- Once these values exist, we may compute y_n via our method

Problem 1

A Chemistry Problem Suggested by Robertson

- We compare our performance to Cash's EBDF on the following problem

$$\begin{array}{ll} f_1 = -0.04y_1 + 10000y_2y_3, & y_1(0) = 1 \\ f_2 = 0.04y_1 - 10000y_2y_3 - 3 * 10^7 y_2^2, & y_2(0) = 0 \\ f_3 = 3 * 10^7 y_2^2, & y_3(0) = 0 \end{array}$$

- Solved on $[0, 40]$ using $h = 0.0002$
- Global error for 3-step EBDF (Cash) was $4 * 10^{-17}$
- Global error for 2-step Hyper EBDF was $5.5 * 10^{-17}$

Problem 1

A Chemistry Problem Suggested by Robertson

- We compare our performance to Cash's EBDF on the following problem

$$\begin{aligned} f_1 &= -0.04y_1 + 10000y_2y_3, & y_1(0) &= 1 \\ f_2 &= 0.04y_1 - 10000y_2y_3 - 3*10^7y_2^2, & y_2(0) &= 0 \\ f_3 &= 3*10^7y_2^2, & y_3(0) &= 0 \end{aligned}$$

- Solved on $[0, 40]$ using $h = 0.0002$
- Global error for 3-step EBDF (Cash) was $4 * 10^{-17}$
- Global error for 2-step Hyper EBDF was $5.5 * 10^{-17}$

Problem 1

A Chemistry Problem Suggested by Robertson

- We compare our performance to Cash's EBDF on the following problem

$$\begin{array}{ll} f_1 = -0.04y_1 + 10000y_2y_3, & y_1(0) = 1 \\ f_2 = 0.04y_1 - 10000y_2y_3 - 3*10^7y_2^2, & y_2(0) = 0 \\ f_3 = 3*10^7y_2^2, & y_3(0) = 0 \end{array}$$

- Solved on $[0, 40]$ using $h = 0.0002$
- Global error for 3-step EBDF (Cash) was $4*10^{-17}$
- Global error for 2-step Hyper EBDF was $5.5*10^{-17}$

Problem 1

A Chemistry Problem Suggested by Robertson

- We compare our performance to Cash's EBDF on the following problem

$$\begin{array}{ll} f_1 = -0.04y_1 + 10000y_2y_3, & y_1(0) = 1 \\ f_2 = 0.04y_1 - 10000y_2y_3 - 3 * 10^7 y_2^2, & y_2(0) = 0 \\ f_3 = 3 * 10^7 y_2^2, & y_3(0) = 0 \end{array}$$

- Solved on $[0, 40]$ using $h = 0.0002$
- Global error for 3-step EBDF (Cash) was $4 * 10^{-17}$
- Global error for 2-step Hyper EBDF was $5.5 * 10^{-17}$

Problem 2

A Chemistry Problem Suggested by Gear

- We also looked at another problem presented by Cash in his paper:

$$\begin{aligned} f_1 &= -0.013y_1 - 1000y_1y_3, & y_1(0) = 1 \\ f_2 &= -2500y_2y_3, & y_2(0) = 1 \\ f_3 &= -0.013y_1 - 1000y_1y_3 - 2500y_2y_3, & y_3(0) = 0 \end{aligned}$$

- Solved on $[0, 50]$ with $h = 0.0001$
- Global error for 3-step EBDF (Cash) was $4 * 10^{-16}$
- Global error for 2-step Hyper EBDF was $2.2 * 10^{-16}$

Problem 2

A Chemistry Problem Suggested by Gear

- We also looked at another problem presented by Cash in his paper:

$$\begin{aligned} f_1 &= -0.013y_1 - 1000y_1y_3, & y_1(0) = 1 \\ f_2 &= -2500y_2y_3, & y_2(0) = 1 \\ f_3 &= -0.013y_1 - 1000y_1y_3 - 2500y_2y_3, & y_3(0) = 0 \end{aligned}$$

- Solved on $[0, 50]$ with $h = 0.0001$
- Global error for 3-step EBDF (Cash) was $4 * 10^{-16}$
- Global error for 2-step Hyper EBDF was $2.2 * 10^{-16}$

Problem 2

A Chemistry Problem Suggested by Gear

- We also looked at another problem presented by Cash in his paper:

$$\begin{aligned} f_1 &= -0.013y_1 - 1000y_1y_3, & y_1(0) &= 1 \\ f_2 &= -2500y_2y_3, & y_2(0) &= 1 \\ f_3 &= -0.013y_1 - 1000y_1y_3 - 2500y_2y_3, & y_3(0) &= 0 \end{aligned}$$

- Solved on $[0, 50]$ with $h = 0.0001$
- Global error for 3-step EBDF (Cash) was $4 * 10^{-16}$
- Global error for 2-step Hyper EBDF was $2.2 * 10^{-16}$

Problem 2

A Chemistry Problem Suggested by Gear

- We also looked at another problem presented by Cash in his paper:

$$\begin{array}{ll} f_1 = -0.013y_1 - 1000y_1y_3, & y_1(0) = 1 \\ f_2 = -2500y_2y_3, & y_2(0) = 1 \\ f_3 = -0.013y_1 - 1000y_1y_3 - 2500y_2y_3, & y_3(0) = 0 \end{array}$$

- Solved on $[0, 50]$ with $h = 0.0001$
- Global error for 3-step EBDF (Cash) was $4 * 10^{-16}$
- Global error for 2-step Hyper EBDF was $2.2 * 10^{-16}$

Problem 3

A Problem with a Known Solution Suggested by Cash

- Lastly, we compare performance on a problem with a known analytic solution:

$$
\begin{aligned}
f_1 &= -1y_1 - 15y_2 + 15e^{-t}, \quad & y_1(0) = 1 \\
f_1 &= 15y_1 - 1y_2 - 15e^{-t}, \quad & y_1(0) = 1
\end{aligned}
$$

- This problem illustrates comparable endpoint error behavior

	HEBDF Error	EBDF Error
t = 20	$y_1 \rightarrow 2.86 * 10^-11$	$y_1 \rightarrow 3.60 * 10^-11$
	$y_2 \rightarrow 2.77 * 10^-12$	$y_2 \rightarrow 1.08 * 10^-11$
t = 40	$y_1 \rightarrow 9.11 * 10^-21$	$y_1 \rightarrow 2.55 * 10^-20$
	$y_2 \rightarrow 8.59 * 10^-20$	$y_2 \rightarrow 8.81 * 10^-20$
t = 80	$y_1 \rightarrow 1.30 * 10^-37$	$y_1 \rightarrow 1.13 * 10^-37$
	$y_2 \rightarrow 2.37 * 10^-37$	$y_2 \rightarrow 3.12 * 10^-37$

Problem 3

A Problem with a Known Solution Suggested by Cash

- Lastly, we compare performance on a problem with a known analytic solution:

$$f_1 = -1y_1 - 15y_2 + 15e^{-t}, \quad y_1(0) = 1$$
$$f_1 = 15y_1 - 1y_2 - 15e^{-t}, \quad y_1(0) = 1$$

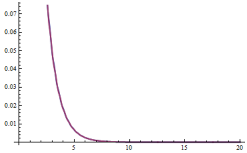
- This problem illustrates comparable endpoint error behavior

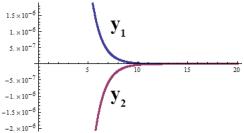
	HEBDF Error	EBDF Error
t = 20	$y_1 \rightarrow 2.86 * 10^-11$ $y_2 \rightarrow 2.77 * 10^-12$	$y_1 \rightarrow 3.60 * 10^-11$ $y_2 \rightarrow 1.08 * 10^-11$
t = 40	$y_1 \rightarrow 9.11 * 10^-21$ $y_2 \rightarrow 8.59 * 10^-20$	$y_1 \rightarrow 2.55 * 10^-20$ $y_2 \rightarrow 8.81 * 10^-20$
t = 80	$y_1 \rightarrow 1.30 * 10^-37$ $y_2 \rightarrow 2.37 * 10^-37$	$y_1 \rightarrow 1.13 * 10^-37$ $y_2 \rightarrow 3.12 * 10^-37$

Problem 3

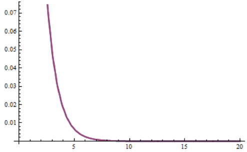
A Problem with a Known Solution Suggested by Cash

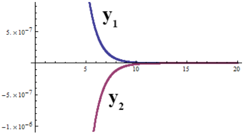
- We graph the solution and error given by applying Cash's method:





- We also graph the solution and error given by applying our Hyper EBDF:

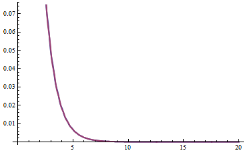


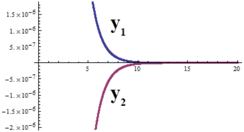


Problem 3

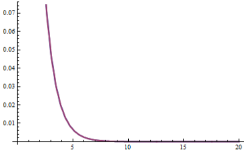
A Problem with a Known Solution Suggested by Cash

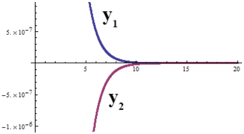
- We graph the solution and error given by applying Cash's method:





- We also graph the solution and error given by applying our Hyper EBDF:





Self-Starting Hyper Extended BDF

Summary

- Comparable to Cash's Extended BDF for methods of the same order.
- Whereas Extended BDF is A-stable for $k \leq 3$, the Hyper EBDF is A-stable for $k \leq 4$.
- The method is viable if implemented in a self-starting predictor/corrector fashion.
  - An initial block is generated from a continuous scheme
  - That block is solved to recover $y_1, y_2, \ldots, y_{k+2}$
  - Remainder of interval is solved as Predictor/Corrector using a $(k+2)$-step Adams-Bashforth method together with our own
- Unexpected stability properties for $k = 1$

Self-Starting Hyper Extended BDF

Summary

- Comparable to Cash's Extended BDF for methods of the same order.
- Whereas Extended BDF is A-stable for $k \leq 3$, the Hyper EBDF is A-stable for $k \leq 4$.
- The method is viable if implemented in a self-starting predictor/corrector fashion.
    - An initial block is generated from a continuous scheme
    - That block is solved to recover $y_1, y_2, \ldots, y_{k+2}$
    - Remainder of interval is solved as Predictor/Corrector using a $(k+2)$-step Adams-Bashforth method together with our own
- Unexpected stability properties for $k = 1$

Self-Starting Hyper Extended BDF

Summary

- Comparable to Cash's Extended BDF for methods of the same order.
- Whereas Extended BDF is A-stable for $k \leq 3$, the Hyper EBDF is A-stable for $k \leq 4$.
- The method is viable if implemented in a self-starting predictor/corrector fashion.
    - An initial block is generated from a continuous scheme
    - That block is solved to recover $y_1, y_2, \ldots, y_{k+2}$
    - Remainder of interval is solved as Predictor/Corrector using a $(k+2)$-step Adams-Bashforth method together with our own
- Unexpected stability properties for $k = 1$

Self-Starting Hyper Extended BDF

Summary

- Comparable to Cash's Extended BDF for methods of the same order.
- Whereas Extended BDF is A-stable for $k \leq 3$, the Hyper EBDF is A-stable for $k \leq 4$.
- The method is viable if implemented in a self-starting predictor/corrector fashion.
  - An initial block is generated from a continuous scheme
  - That block is solved to recover $y_1, y_2, \ldots, y_{k+2}$
  - Remainder of interval is solved as Predictor/Corrector using a $(k+2)$-step Adams-Bashforth method together with our own
- Unexpected stability properties for $k = 1$

Austin Peay State University

Self-Starting Hyper Extended BDF

Summary

- Comparable to Cash's Extended BDF for methods of the same order.
- Whereas Extended BDF is A-stable for $k \leq 3$, the Hyper EBDF is A-stable for $k \leq 4$.
- The method is viable if implemented in a self-starting predictor/corrector fashion.
    - An initial block is generated from a continuous scheme
    - That block is solved to recover $y_1, y_2, \ldots, y_{k+2}$
    - Remainder of interval is solved as Predictor/Corrector using a $(k+2)$-step Adams-Bashforth method together with our own
- Unexpected stability properties for $k = 1$

Self-Starting Hyper Extended BDF

Summary

- Comparable to Cash's Extended BDF for methods of the same order.
- Whereas Extended BDF is A-stable for $k \leq 3$, the Hyper EBDF is A-stable for $k \leq 4$.
- The method is viable if implemented in a self-starting predictor/corrector fashion.
  - An initial block is generated from a continuous scheme
  - That block is solved to recover $y_1, y_2, \ldots, y_{k+2}$
  - Remainder of interval is solved as Predictor/Corrector using a $(k+2)$-step Adams-Bashforth method together with our own
- Unexpected stability properties for $k = 1$

Self-Starting Hyper Extended BDF

Summary

- Comparable to Cash's Extended BDF for methods of the same order.
- Whereas Extended BDF is A-stable for $k \leq 3$, the Hyper EBDF is A-stable for $k \leq 4$.
- The method is viable if implemented in a self-starting predictor/corrector fashion.
  - An initial block is generated from a continuous scheme
  - That block is solved to recover $y_1, y_2, \ldots, y_{k+2}$
  - Remainder of interval is solved as Predictor/Corrector using a $(k+2)$-step Adams-Bashforth method together with our own
- Unexpected stability properties for $k = 1$

Self-Starting Hyper Extended BDF

Future Work

- Resolve or explain stability region issues for $k = 1$
- Apply a mesh-refinement algorithm for variable stepsize
- Implement via Newton's method to compare more closely with Cash's implementation
- Apply the problem to PDEs via Method of Lines

Self-Starting Hyper Extended BDF

Future Work

- Resolve or explain stability region issues for $k = 1$
- Apply a mesh-refinement algorithm for variable stepsize
- Implement via Newton's method to compare more closely with Cash's implementation
- Apply the problem to PDEs via Method of Lines

Self-Starting Hyper Extended BDF

Future Work

- Resolve or explain stability region issues for $k = 1$
- Apply a mesh-refinement algorithm for variable stepsize
- Implement via Newton's method to compare more closely with Cash's implementation
- Apply the problem to PDEs via Method of Lines

Self-Starting Hyper Extended BDF

Future Work

- Resolve or explain stability region issues for $k = 1$
- Apply a mesh-refinement algorithm for variable stepsize
- Implement via Newton's method to compare more closely with Cash's implementation
- Apply the problem to PDEs via Method of Lines

References

- C.W. Gear Algorithm 407, Difsub for the solution of ordinary differential equations *Comm. ACM*, 14:185-190, 1971
- J.R. Cash On the Integration of Stiff Systems of O.D.E.s Using Extended Backward Differentiation Formulae *Numerische Mathematik*, 34:235-246, 1980.
- J.R. Cash Modified extended backward differentiation formulae for the numerical solution of stiff initial value problems in ODEs and DAEs *Journal of Computational and Applied Mathematics*, 125:117-130, 2000.
- S.N. Jator, S. Swindell, and R.D. French Trigonometrically fitted block Numerov type method for $y'' = f(x, y, y')$ *Numerical Algorithms* 38, 2012
- H. Ramos, J. Vigo-Aguiar A fourth-order Runge-Kutta method based on BDF-type Chebyshev approximations *Journak of Computational and Applied Mathematics* 204:124-136, 2007

References

- C.W. Gear Algorithm 407, Difsub for the solution of ordinary differential equations *Comm. ACM*, 14:185-190, 1971
- J.R. Cash On the Integration of Stiff Systems of O.D.E.s Using Extended Backward Differentiation Formulae *Numerische Mathematik*, 34:235-246, 1980.
- J.R. Cash Modified extended backward differentiation formulae for the numerical solution of stiff initial value problems in ODEs and DAEs *Journal of Computational and Applied Mathematics*, 125:117-130, 2000.
- S.N. Jator, S. Swindell, and R.D. French Trigonometrically fitted block Numerov type method for $y'' = f(x, y, y')$ *Numerical Algorithms* 38, 2012
- H. Ramos, J. Vigo-Aguiar A fourth-order Runge-Kutta method based on BDF-type Chebyshev approximations *Journak of Computational and Applied Mathematics* 204:124-136, 2007

References

- C.W. Gear Algorithm 407, Difsub for the solution of ordinary differential equations *Comm. ACM*, 14:185-190, 1971
- J.R. Cash On the Integration of Stiff Systems of O.D.E.s Using Extended Backward Differentiation Formulae *Numerische Mathematik*, 34:235-246, 1980.
- J.R. Cash Modified extended backward differentiation formulae for the numerical solution of stiff initial value problems in ODEs and DAEs *Journal of Computational and Applied Mathematics*, 125:117-130, 2000.
- S.N. Jator, S. Swindell, and R.D. French Trigonometrically fitted block Numerov type method for $y'' = f(x, y, y')$ *Numerical Algorithms* 38, 2012
- H. Ramos, J. Vigo-Aguiar A fourth-order Runge-Kutta method based on BDF-type Chebyshev approximations *Journak of Computational and Applied Mathematics* 204:124-136, 2007

References

- C.W. Gear Algorithm 407, Difsub for the solution of ordinary differential equations *Comm. ACM*, 14:185-190, 1971
- J.R. Cash On the Integration of Stiff Systems of O.D.E.s Using Extended Backward Differentiation Formulae *Numerische Mathematik*, 34:235-246, 1980.
- J.R. Cash Modified extended backward differentiation formulae for the numerical solution of stiff initial value problems in ODEs and DAEs *Journal of Computational and Applied Mathematics*, 125:117-130, 2000.
- S.N. Jator, S. Swindell, and R.D. French Trigonometrically fitted block Numerov type method for $y'' = f(x, y, y')$ *Numerical Algorithms* 38, 2012
- H. Ramos, J. Vigo-Aguiar A fourth-order Runge-Kutta method based on BDF-type Chebyshev approximations *Journak of Computational and Applied Mathematics* 204:124-136, 2007

References

- C.W. Gear Algorithm 407, Difsub for the solution of ordinary differential equations *Comm. ACM*, 14:185-190, 1971
- J.R. Cash On the Integration of Stiff Systems of O.D.E.s Using Extended Backward Differentiation Formulae *Numerische Mathematik*, 34:235-246, 1980.
- J.R. Cash Modified extended backward differentiation formulae for the numerical solution of stiff initial value problems in ODEs and DAEs *Journal of Computational and Applied Mathematics*, 125:117-130, 2000.
- S.N. Jator, S. Swindell, and R.D. French Trigonometrically fitted block Numerov type method for $y'' = f(x, y, y')$ *Numerical Algorithms* 38, 2012
- H. Ramos, J. Vigo-Aguiar A fourth-order Runge-Kutta method based on BDF-type Chebyshev approximations *Journak of Computational and Applied Mathematics* 204:124-136, 2007

Hyper-Extended BDF

A Self-Starting, Predictor/Corrector BDF with 2 Superfuture Points

Any Questions?