The PVODE and IDA Algorithms

Alan C. Hindmarsh
Lawrence Livermore National Laboratory

UCRL-ID-141558

December 2000

The PVODE and IDA Algorithms*

Alan C. Hindmarsh

December 15, 2000

Preface

In October-November 2000, I gave a series of talks, describing in some detail the algorithms in two general-purpose solvers—

- the PVODE solver for systems of ordinary differential equations (ODEs), and
- the IDA solver for systems of differential-algebraic equations (DAEs).

The material was organized into three parts:

- Part A: Overview
- Part B: The PVODE Algorithm
- Part C: The IDA Algorithm

This document consists of the viewgraphs for the corresponding three talks.

Except for the correction of some minor errors, the talk viewgraphs are given here exactly as presented. Each of the three sets of pages is numbered independently, with page numbers starting at 1.

Preceding the viewgraphs, on the next four pages is a brief outline of each of the talks, and a list of references, some of which are cited in the viewgraphs.

Alan Hindmarsh
Center for Applied Scientific Computing

*This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

The PVODE and IDA Algorithms - Outline
==========

Part A. Overview
---

1. ODE dy/dt = f(t,y) vs DAE F(t,y,dy/dt) = 0; DAE index

2. BDF Methods
   a) Basic formulas
   b) Symbolic derivation: hD = log E
   c) Fixed-step vs fully variable-step vs fixed-leading-coefficient
   d) Example: 3 forms of BDF2

3. Stiffness and absolute stability
   a) stiffness; example
   b) absolute stability; local linear approx.; dy/dt = lambda y
   c) abs. stability regions; A-stability, etc.
   d) BDF absolute stability regions

4. Norms - weighted RMS via tolerances

5. Errors
   a) Local Error vs Local Truncation Error vs Global Error
   b) LTE = C h^{q+1} y^{(q+1)} + h.o.t.
   c) Error estimation:
      prediction = order-q explicit analog; correction;
      E(h) = est. local error = C' (predictor - corrector) + h.o.t.
   c) Error control:
      Set h' via ||E(h')|| = 1, E(h') = (h'/h)^{q+1} E(h)

6. Solving the implicit system
     G(y_n) = 0, G(y) = { y - gamma f(t_n,y) - a_n   [PVODE]
                        { F(t_n, y, (y - a_n)/gamma) [IDA]
     Newton: M (delta y) = - G(y),
             M approximates G'(y) = { I - gamma J [PVODE]
                                     { F_y + c F_y' [IDA]
     Newton-direct: dense or band treatment of M
     Newton-Krylov: GMRES with preconditioning
     Relaxation w.r.t. gamma

Part B. The PVODE Algorithm
---

1. Initial step size
   Estimate second derivative; do order 1 error control

2. Storing the history - Nordsieck array = scaled derivatives
   Scaling; prediction via Pascal Triangle;
   Correcting the history array

3. Interpolating to output times

4. Newton iteration algorithm
   a) Modified vs Inexact Newton
   b) J/P update strategies; Jacobian-saving
   c) Convergence test; rate estimate

5. Local error test

6. Step and order selection
   a) Order selection: E(h,k) = est. local error at order k;
      set h' = h'(k) via ||E(h',k)|| = tolerance; find max h'(k).
   b) Heuristics

7. Adjustments on change of step or order

Part C. The IDA Algorithm
---

1. Initial condition calculation

2. Initial step size
   Do "order 0" error control

3. Storing the history - modified divided differences
   Predicting; Correcting the history array

4. Interpolating to output times

5. Newton iteration algorithm
   a) Inexact Newton
   b) J/P update strategies
   c) Convergence test; rate estimate

6. Local error test - on truncation and interpolation errors

7. Order/step selection
   a) Order selection - Taylor series terms
   b) Step selection - via LTE
   c) Heuristics

8. Inequality constraints

References:
---

[1] K. E. Brenan, S. L. Campbell, and L. R. Petzold, Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations, SIAM, Philadelphia, 1996, Section 5.2 ("Algorithms and Strategies in DASSL").

[2] P. N. Brown, "Fixed Leading Coefficient BDF Formulas," hand-written manuscript, October 1987.

[3] P. N. Brown, G. D. Byrne, and A. C. Hindmarsh, "VODE, A Variable-Coefficient ODE Solver," SIAM J. Sci. Stat. Comput., 10 (1989), pp. 1038-1051.

[4] P. N. Brown, A. C. Hindmarsh, and L. R. Petzold, "Using Krylov Methods in the Solution of Large-Scale Differential-Algebraic Systems," SIAM J. Sci. Comput. 15 (1994), pp. 1467-1488.

[5] P. N. Brown, A. C. Hindmarsh, and L. R. Petzold, "Consistent Initial Condition Calculation for Differential-Algebraic Systems," SIAM J. Sci. Comp. 19 (1998), pp. 1495-1512.

[6] G. D. Byrne and A. C. Hindmarsh, "User Documentation for PVODE, An ODE Solver for Parallel Computers," LLNL Report UCRL-ID-130884, May 1998.

[7] G. D. Byrne and A. C. Hindmarsh, "PVODE, An ODE Solver for Parallel Computers," Int. J. High Perf. Comput. Applic., 13 (1999), pp. 354-365.

[8] G. D. Byrne, "Pragmatic Experiments with Krylov Methods in the Stiff ODE Setting," Computational Ordinary Differential Equations, J. Cash and I. Gladwell, eds., Oxford Univ. Press, Oxford, 1992, pp. 323-356.

[9] A. C. Hindmarsh and A. G. Taylor, "User Documentation for IDA, a Differential-Algebraic Equation Solver for Sequential and Parallel Computers," LLNL Report UCRL-MA-136910, December 1999.

[10] K. R. Jackson and R. Sacks-Davis, "An Alternative Implementation of Variable Step-Size Multistep Formulas for Stiff ODEs," ACM Trans. Math. Softw. 6 (1980), pp. 295-318.

The PVODE and IDA Algorithms

Alan C. Hindmarsh

Part A: Overview

1. IVPs – ODEs vs DAEs
2. Backward Differentiation Formulas
3. Stiffness and absolute stability
4. Norms - weighted RMS via tolerances
5. Errors - local error; error control
6. Solving the implicit system - Newton variants

1. Initial Value Problems

ODE systems:

$$\dot{y} \equiv dy/dt = f(t, y) \ , \quad y(t_0) \text{ given}, \quad y \in R^N$$

DAE systems:

$$F(t, y, \dot{y}) = 0 \ , \quad y(t_0) \text{ and } \dot{y}(t_0) \text{ given}$$

Index of DAE systems:

$F(t, y, \dot{y}) = 0$ is index-0 if $\partial F/\partial \dot{y}$ is nonsingular (can solve for $\dot{y}$, in principle).

$F(t, y, \dot{y}) = 0$ is index-1 if $F = \begin{pmatrix} f(t, u, v, \dot{u}) \\ g(t, u, v) \end{pmatrix}$ with $\partial f/\partial \dot{u}$ and $\partial g/\partial v$ both square and nonsingular (can solve $g = 0$ for v, substitute, solve $f = 0$ for $\dot{u}$). (There are more general forms of index-1 systems.)

2. Backward Differentiation Formulas

PVODE and IDA use BDF methods. With PVODE, user can select Adams methods for nonstiff problems.

Basic formulas. Discrete values are t_n and $y_n \approx y(t_n)$. Step size is $h = t_n - t_{n-1}$.

Backward Euler (BE): $y_n = y_{n-1} + h\dot{y}_n$

Fixed-step BDF of order q (BDFq):

$$y_n = \sum_{i=1}^{q} \alpha_i y_{n-i} + h\beta_0 \dot{y}_n \quad \text{(BDF1 = BE)}$$

Coefficients α_i, β_0 depend only on order q.

One of many Linear Multistep Formulas (Methods).

"BDF" because it differentiates y with backward values: $\dot{y}_n =$ (linear combination of $y_n, y_{n-1}, y_{n-2}, \ldots$)

Interpretation for $\dot{y} = f$: Solve for y_n in

$$y_n = \sum_{i=1}^{q} \alpha_i y_{n-i} + h\beta_0 f(t_n,\ y_n)\ .$$

Interpretation for $F(t, y, \dot{y}) = 0$: Solve for y_n in

$$F\left(t_n, y_n, \frac{y_n - \Sigma_{i=1}^{q} \alpha_i y_{n-i}}{h\beta_0}\right) = 0\ .$$

Symbolic operator derivation.
For any sequence $\{x_n\}$, define operators

$$Ex_n = x_{n+1} \ \text{(increment)}$$
$$\Delta x_n = x_{n+1} - x_n \ \text{(forward difference)}$$
$$\nabla x_n = x_n - x_{n-1} \ \text{(backward difference)}$$

Also define $1 =$ identity operator. Note inverse and other relations, and develop an algebra of operators:

$$E^{-1}x_n = x_{n-1} \ , \quad \Delta = E - 1 \ , \quad \nabla = 1 - E^{-1} \ .$$

A sequence may or may not be the set of discrete values of a smooth function of t. When it is, we can define a differentiation operator, $Df_n = \dot{f}(t_n)$.
Infinite Taylor series for f:

$$\begin{aligned} Ef_n &= f_{n+1} = f(t_n + h) = f_n + h\dot{f}_n + \frac{h^2}{2}\ddot{f} + \dots \\ &= f_n + hDf_n + \frac{h^2D^2}{2}f_n + \dots \\ &= (1 + hD + \frac{h^2D^2}{2} + \dots)f_n \ , \qquad \text{or } E = e^{hD} \ . \end{aligned}$$

Now from $1 - \nabla = E^{-1}$, get $E = (1 - \nabla)^{-1}$,

$$hD = \log(E) = \log[1/(1 - \nabla)] = \nabla + \frac{1}{2}\nabla^2 + \frac{1}{3}\nabla^3 + \dots$$

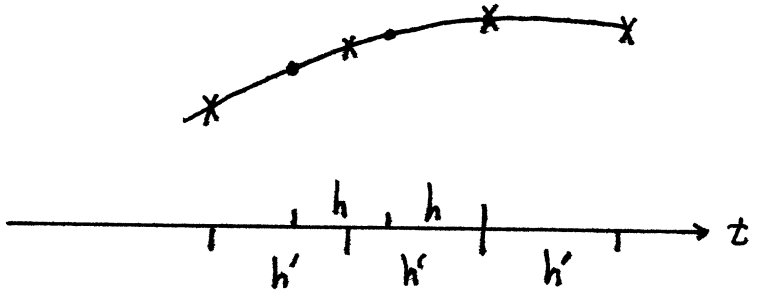
Applied to $y(t)$:

$$h\dot{y}_n = \nabla y_n + \frac{1}{2}\nabla^2 y_n + \frac{1}{3}\nabla^3 y_n + \dots$$

For BDFq, truncate this series at ∇^q term.

How to vary the stepsizes? Three ways:

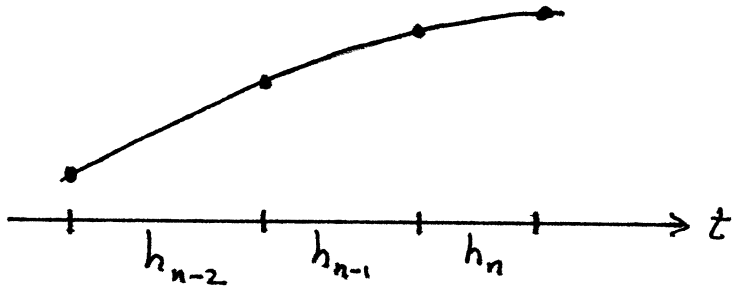
(1) Use fixed-step BDF but interpolate at a spacing equal to the new stepsize h'.



[This is the method in LSODE and its variants.]
Behavior can be unstable if step changes are frequent.

(2) Use fully variable-step form of BDF:

$$y_n = \sum_{i=1}^{q} \alpha_{n,i} y_{n-i} + h_n \beta_{n,0} \dot{y}_n \ .$$



Coefficients $\alpha_{n,i}, \beta_{n,0}$ depend on stepsizes $h_n = t_n - t_{n-1}$, $h_{n-1}, \ldots, h_{n-q+1}$. If $h = max\{h_n, \ldots, h_{n-q+1}\}$, then the coefficients are uniquely determined by:

$$(1/\beta_{n,0})[y(t_n) - \sum_{i=1}^{q} \alpha_{n,i} y(t_{n-i})] = h_n \dot{y}(t_n) + O(h^{q+1}) \ .$$

[This is the method in EPISODE and its variants.]
Disadvantage: variation of $h_n \beta_{n,0}$ makes Newton matrix hard to keep current.

(3) Use "fixed-leading coefficient" form of BDF.
A compromise. The choice in PVODE and IDA.
Formula has fixed-step value of $\beta_{n,0} = \beta_0 = 1/(\Sigma_{j=1}^{q} j^{-1})$ but at the cost of an extra term $\beta_{n,1}\dot{y}_{n-1}$:

$$y_n = \sum_{i=1}^{q} \alpha_{n,i} y_{n-i} + h_n \beta_0 \dot{y}_n + h_n \beta_{n,1} \dot{y}_{n-1} \ .$$

Coefficient $\beta_{n,1}$ vanishes when stepsizes are equal.

Coefficients are defined via interpolating polynomials:
Given $y_{n-1}, \ldots, y_{n-q}, \ \dot{y}_{n-1}$, define the predictor polynomial $\omega^p(t)$, of degree $\leq q$, by

$$\omega^p(t_{n-j}) = y_{n-j} \ \ (j = 1 \ldots q) \ , \ \ \dot{\omega}^p(t_{n-1}) = \dot{y}_{n-1} \ .$$

Corrector polynomial $\omega^c(t)$ is defined in terms of $\omega^p(t)$ and the unknown value y_n by:

$$\omega^c(t_n) = y_n \ , \ \ \omega^c(t_n - jh_n) = \omega^p(t_n - jh_n) \ \ (j = 1..q) \ .$$

Then the BDF is the equation $\dot{y}_n = \dot{\omega}^c(t_n)$.

More constructively, get predicted values $y_{n(0)} = \omega^p(t_n)$ and $\dot{y}_{n(0)} = \dot{\omega}^p(t_n)$, and then build $\omega^c(t)$ as

$$\omega^c(t) = \omega^p(t) + c(t)(y_n - y_{n(0)}) \ \text{ where}$$

$$c(t_n - h_n) = \ldots = c(t_n - qh_n) = 0 \ , \ \ c(t_n) = 1 \ .$$

Then the BDF is

$$h_n \dot{y}_n = h_n \dot{y}_{n(0)} + h_n \dot{c}(t_n)(y_n - y_{n(0)}) \ ,$$

where $h_n \dot{c}(t_n) = 1/\beta_0$.

Example: BDF2

(1) Fixed-step:

$$h\dot{y}_n = \nabla y_n + \frac{1}{2}\nabla^2 y_n = y_n - y_{n-1} + (y_n - 2y_{n-1} + y_{n-2})/2$$

$$y_n = \frac{4}{3}y_{n-1} - \frac{1}{3}y_{n-2} + \frac{2}{3}h\dot{y}_n$$

(2) Variable-step:
With $\rho \equiv h_n/h_{n-1}$,

$$y_n = \frac{1}{2\rho+1}\left[(\rho+1)^2 y_{n-1} - \rho^2 y_{n-2} + (\rho+1)h_n\dot{y}_n\right]$$

(3) Fixed-leading-coefficient:
With $\rho \equiv h_n/h_{n-1}$,

$$y_n = \left(1 + \frac{\rho^2}{3}\right) y_{n-1} - \frac{\rho^2}{3}y_{n-2} + \frac{2}{3}h_n\dot{y}_n - \left(\frac{\rho-1}{3}\right) h_n\dot{y}_{n-1}$$

3. Stiffness and Absolute Stability

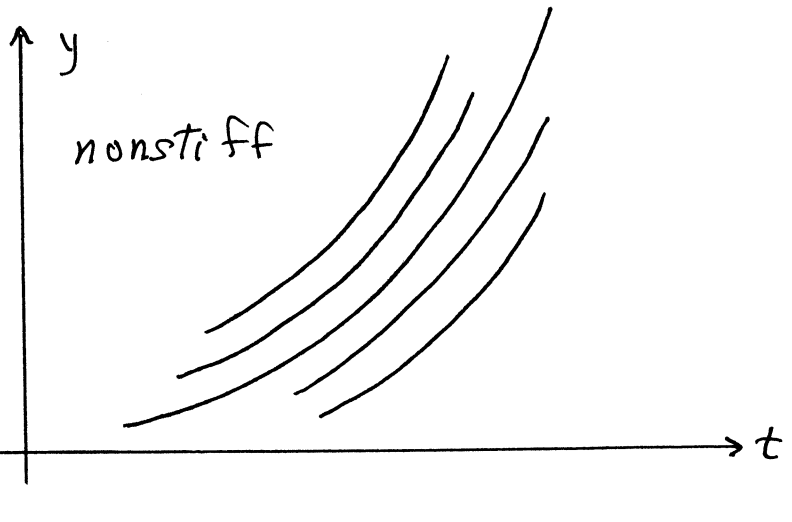
Stiffness.
Characterized by presence of a strongly damped mode.
In a linear approximation, there is a mode $e^{\lambda t}$ with $Re(\lambda) << 0$, such that the corresponding time constant $\tau = -1/Re(\lambda)$ is $<<$ the t scale of interest in solution.
Problem is not (yet) stiff if $y(t)$ is changing on scale of τ.

Simple example:

$$\dot{y} = 2t + 10^6(t^2 - y) \ , \quad t \in [0,1] \ , \quad y(0) \text{ given} \ .$$

Here $y = t^2 + y_0 e^{-t/\tau}$, $\tau = 10^{-6}$. Beyond $t = 10\tau$ (say), solution time scale is ~ 1. But damped mode with time constant τ is still present, so problem is stiff there. Stepsizes $\sim \tau$ if the wrong method is used.

Graphically: Among all the particular solutions, a few are smooth (in quasi-equilibrium), but most are strongly damped toward a smooth solution. In contrast, the solutions for a nonstiff problem are more nearly parallel.



Absolute Stability (for ODEs).
Look at how a perturbation in initial conditions generates a perturbation in the solution: $y(t) \to y(t) + w(t)$.
For $\dot{y} = f(t, y)$, get

$$\dot{w} = f(t, y + w) - f(t, y) \approx J(t)w(t) \quad , \quad \text{where}$$

$$J(t) = (\partial f / \partial y)(t, y(t)) \ .$$

In a local linear approximation, look at simply $\dot{w} = Jw$ with J constant.

Diagonalize J. Some nonsingular matrix transforms J into a diagonal matrix with eigenvalues λ.

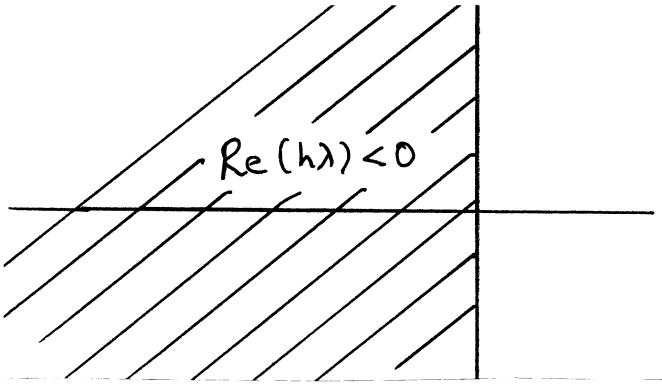
The true solution of $\dot{w} = Jw$ is a linear combination of exponentials $e^{\lambda t}$.

Using linearity of the BDF, the BDF solution is the same linear combination of the BDF solutions of $\dot{y} = \lambda y$.

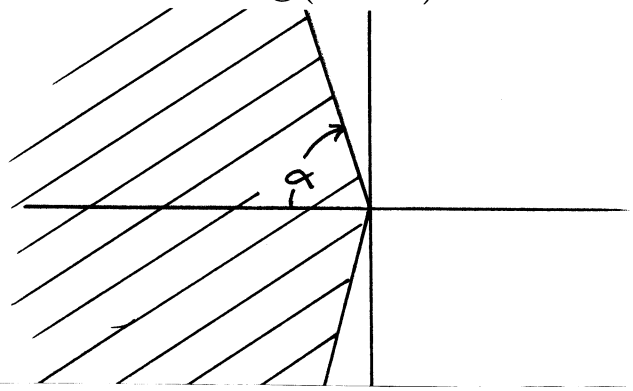
Define the Absolute Stability Region of any ODE method as the set of $h\lambda$ for which the method is stable, i.e. damped, for $\dot{y} = \lambda y$ at stepsize h.

So you can judge the stability of any given method on a given class of problems simply from the Absolute Stability Region of the method combined with the spectral properties of the problems.

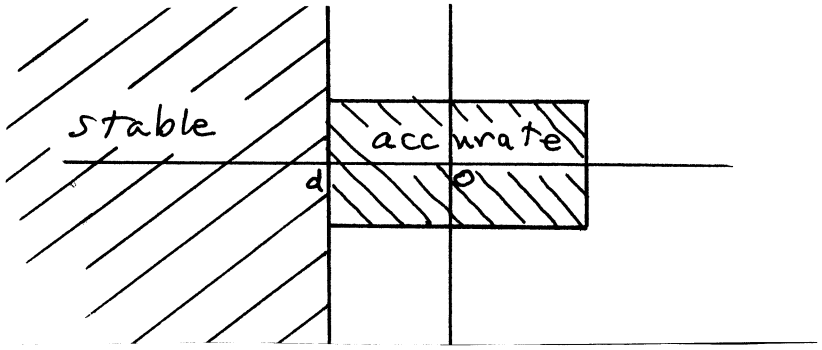
Method is "A-Stable" if A.S.R. includes the left half-plane $\{Re(z) < 0\}$; i.e. whenever the ODE $\dot{y} = \lambda y$ is stable, so is its numerical solution.
True for BDF1 = BE and BDF2. Dahlquist proved there is no A-stable Linear Multistep Method of order ≥ 3.



All the BDFs up to order 6 are "A(α)-Stable": A.S.R. includes a sector $-\alpha < \arg(-h\lambda) < \alpha$ for some $\alpha < \pi/2$.



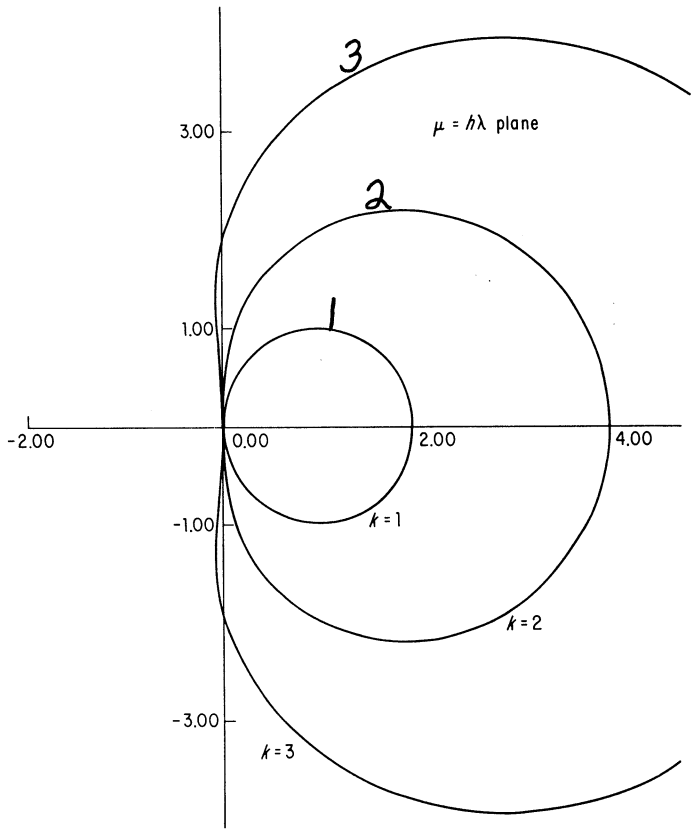
All the BDFs up to order 6 are "Stiffly Stable": A.S.R. includes a half-plane left of some $d < 0$, and method is accurate in an adjoining rectangle centered at the origin.

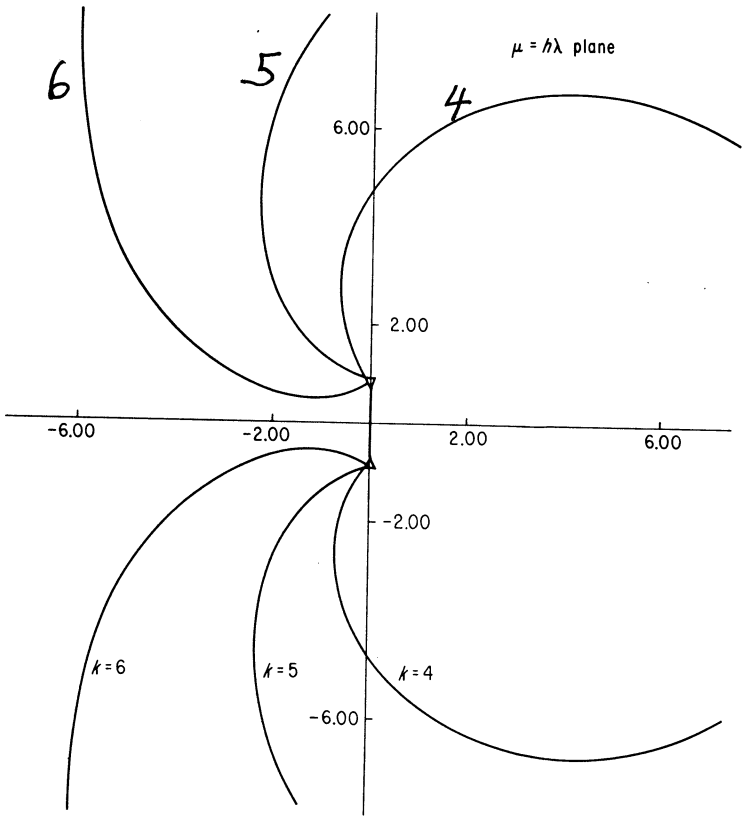


BDF Absolute Stability Regions:

Method is stable *outside* the closed curve shown.

BDF1 = BE: $\{|h\lambda - 1| > 1\}$ (A-stable)
BDF2 is also A-stable





Order 6 is close to failing to be A(α)-Stable or stiffly stable, so is excluded in the BDF solvers.

4. Norms

The user must supply tolerances,

rtol = relative tolerance (scalar)

atol = absolute tolerance (scalar or vector)

These define weights for the solution vector y:

$$w^i = \text{rtol}\ |y^i| + \text{atol}^i \quad (i = 1 \ldots N)$$

To control errors in y^i relative to w^i, use a weighted norm.

To eliminate bias when expanding problem size (as in mesh refinement), we use a root-mean-square norm.

So all error/convergence tests use a Weighted RMS norm

$$\|v\|_{WRMS} = \left[\frac{1}{N}\sum_1^N (v^i/w^i)^2\right]^{1/2}$$

on any error vector v.

Roughly (ignoring effects of RMS), $\|v\| < 1$ means that

$$|v^i| < w^i \quad \text{for all } i$$

and roughly that

$$\text{either } |v^i|/|y^i| < \text{rtol} \quad \text{or} \quad |v^i| < \text{atol}^i$$

So a unit vector in this norm is really a small vector.

5. Errors

Local errors.

Local Error = error committed on one step, taken with exact solution values for past values.

Local Truncation Error = residual of the Linear Multistep Formula when an exact solution is inserted. (Depends on how one normalizes the LMF.)

LE and LTE are different, but close. Both are $O(h^{q+1})$ for a method of order q. With suitable normalization, LE = LTE $+O(h^{q+2})$.

Global Error = error in y_n (after n steps) from exact initial value y_0, reflecting cumulative effect of local errors. Convergence theorems show Global Error $= O(h^q)$, under suitable conditions.

For any LMF, LTE $= C_n h^{q+1} y^{(q+1)}(t_n) + O(h^{q+2})$, where C_n is a computable function of q and (in variable-step cases) of the past q stepsizes h_j.

Error estimation.

Predict y_n as $y_{n(0)} = \omega^p(t_n)$, the explicit analog of BDF:

$$y_{n(0)} = \sum_{i=1}^{q} \alpha_{n,i}^p y_{n-i} + h_n \beta_{n,1}^p \dot{y}_{n-1}$$

Similar asymptotic analysis of $y_n - y_{n(0)}$ gives

$$y_n - y_{n(0)} = \bar{C}_n h^{q+1} y^{(q+1)}(t_n) + O(h^{q+2})$$

for another known constant $\bar{C}_n$.

So within $O(h^{q+2})$,

$$\begin{aligned} (\text{LTE in } y_n) &\approx C_n h^{q+1} y^{(q+1)}(t_n) \\ &\approx \left(\frac{C_n}{\bar{C}_n}\right)(y_n - y_{n(0)}) \ . \end{aligned}$$

So we define the Estimated Local Truncation Error as ELTE $= C'_n(y_n - y_{n(0)})$, with $C'_n = C_n/\bar{C}_n$.

Error control.

Given $E(h) =$ ELTE for the tentative step taken at order q and stepsize h, we accept the step if $\|E(h)\| < 1$, and redo the step otherwise.

In either case, we want a new stepsize h' for which the current or next step will succeed.

Asymptotic formulas imply $E(h') \approx (h'/h)^{q+1} E(h)$, ignoring variation in C_n.

With this approximation, the value of h' that makes $\|E(h')\| = 1$ is given by

$$|h'/h|^{q+1} \|E(h)\| = 1$$

or

$$h' = h/\|E(h)\|^{1/(q+1)}$$

We insert a heuristic factor < 1 into this formula to compensate for estimation errors etc.

Order selection.

Under certain conditions, we choose a new order q' as well as a new step size. This is based on similar ideas, is done quite differently for PVODE and IDA.

6. Solving the Implicit System

Nonlinear system.

At every step, a nonlinear system $G(y_n) = 0$ must be solved for the new vector y_n.

Write the BDF as $y_n = a_n + \gamma \dot{y}_n$, $\gamma \equiv h\beta_0$. Then

For PVODE: $G(y) = y - \gamma f(t_n, y) - a_n$.

For IDA: $G(y) = F(t_n, y, (y - a_n)/\gamma)$.

Newton iteration.

The initial guess is the predictor value $y_{n(0)}$.

The iteration is $M[y_{n(m+1)} - y_{n(m)}] = -G(y_{n(m)})$
with Newton matrix $M \approx G'(y)$ at some nearby value y.

For PVODE: $G'(y) = I - \gamma J$, $J \equiv \partial f/\partial y$.

For IDA: $G'(y) = \partial F/\partial y + \alpha \partial F/\partial \dot{y}$, $\alpha \equiv 1/\gamma$.

Newton-Direct solution:
Dense or band LU factorization of M; backsolve $Mx = b$.

Newton-Krylov solution:
Solve $Mx = b$ by preconditioned GMRES.
Precondition on left or right in PVODE, left only in IDA.

Linear system relaxation (Newton-direct case).

Newton matrix used is either

$$\bar{M} = I - \bar{\gamma} J \quad \text{or} \quad \bar{M} = \partial F/\partial y + (1/\bar{\gamma})\partial F/\partial \dot{y}$$

with $\bar{\gamma}$ = the value of $h\beta_0$ when M was last evaluated. Even if the rest of $\bar{M}$ has not changed, $\bar{\gamma} \neq \gamma$ can degrade convergence.

We help by doing relaxation: use $\Delta y = -c\bar{M}^{-1}G$ instead of $\Delta y = -\bar{M}^{-1}G$, with scalar c. What c?

Consider a linear ODE system, $\dot{y} = Jy$ ($F = \dot{y} - Jy$) with J constant. Then either

$$G' = M = I - \gamma J \ , \quad \bar{M} = I - \bar{\gamma} J \ \text{ or}$$
$$G' = M = \gamma^{-1} I - J \ , \quad \bar{M} = \bar{\gamma}^{-1} I - J \ .$$

In either case, the error at each iteration is reduced by the error matrix $E = I - c\bar{M}^{-1}M$, and convergence rate is the spectral radius $\rho(E)$.

Observe that for λ = an eigenvalue of J, the corresponding eigenvalue of E is either

$$\epsilon = 1 - c\left(\frac{1-\gamma\lambda}{1-\bar{\gamma}\lambda}\right) \ , \quad \text{or} \ \ \epsilon = 1 - c\left(\frac{\gamma^{-1}-\lambda}{\bar{\gamma}^{-1}-\lambda}\right) \ .$$

We expect $Re(\lambda) < 0$ but otherwise $Sp(J)$ is unknown. Choosing c to minimize $\max\{\ |\epsilon| \ : \ Re(\lambda) < 0\}$ leads to

$$c = \frac{2}{1+\gamma/\bar{\gamma}} \ \text{(PVODE)}, \ \ \text{or} \ \ c = \frac{2}{1+\bar{\gamma}/\gamma} \ \text{(IDA)} \ .$$

Shorthand notation for the step algorithm:

P = prediction

E = evaluation of f or F

C = correction

Each step is: P (E C)^m.

Stopping test.

Basis for error control is valid only if y_n is an accurate solution of the implicit equation. So stopping test must (try to) insure that iteration error in final $y_{n(m)}$ has a relatively small effect on the local error test quantity.
I.e. tolerances for Newton stopping and for local error are strongly related.

Test is on $\|y_{n(m+1)} - y_{n(m)}\|$, not on $\|G(y_{n(m+1)})\|$.
(No final E.) Reasons:
(a) G magnifies the error in stiff ODE case;
(b) $\|G\|$ is nonsense in DAE case; weights not valid.

The PVODE and IDA Algorithms

Alan C. Hindmarsh

Part B: PVODE

1. Initial step size
2. Storing the history - Nordsieck array
3. Interpolating to output times
4. Newton iteration algorithm
5. Local error test
6. Step and order selection
7. Adjustments on change of step or order
8. Example PVODE run

0. Introduction

PVODE solves the initial value problem

$$\dot{y} = f(t, y) \ , \quad y \in R^N \ , \quad y(t_0) \text{ given } .$$

Methods available are:

variable-order (1-12) variable-step Adams-Moulton,

var.-order (1-5) var.-step (fixed-leading-coeff.) BDF

For stiff problems, use BDF.

Discrete values are t_n and $y_n \approx y(t_n)$, with step sizes $h_n = t_n - t_{n-1}$.

In terms of predicted values $y_{n(0)}$ and $\dot{y}_{n(0)}$, the BDF of order q has the form

$$y_n - y_{n(0)} = h_n \beta_0 (\dot{y}_n - \dot{y}_{n(0)}) \ ,$$

where $\beta_0 = 1/(\Sigma_{j=1}^{q} 1/j)$.

Desirable absolute stability properties of BDF methods for stiff problems comes from single (implicit) $\dot{y}$ term in linear multistep formula $y_n = \Sigma_{j=1}^{q} \alpha_j y_{n-j} + h\beta_0 \dot{y}_n$

Where the algorithm is method-specific, only the BDF methods are covered here.

1. The Initial Step Size

First step will be with BDF1 = Backward Euler.
For this step, LTE= $\frac{1}{2}h^2\ddot{y}(t_0) + O(h^3)$.
We want an h that roughly solves $\|\text{LTE}\|_{WRMS} = 1$, or

$$|h|^2\|\ddot{y}(t_0)\|/2 = 1 \ .$$

Must estimate $\ddot{y}_0 = \ddot{y}(t_0)$, knowing $y_0,\ \dot{y}_0 = f(t_0, y_0)$.

For a given guess $\bar{h}$, estimate

$$\ddot{y}_0 \approx [f(t_0 + \bar{h},\ y_0 + \bar{h}\dot{y}_0) - \dot{y}_0]/\bar{h} \ .$$

Now iterate.

To get started, use bounds for $|h|$ based on initial time t_0 and first requested output time t_{out}.

Lower bound is

$$h_L = 100 \cdot (\text{unit roundoff}) \cdot \max\{|t_0|, |t_{out}|\} \ .$$

Upper bound is $h_U = 0.1|t_{out} - t_0|$, possibly adjusted downward to ensure that

$$h_U|\dot{y}_0^i| \leq 0.1|y_0^i| + \text{atol}^i \ \ (\text{all } i) \ .$$

Start iteration with $\bar{h} = \sqrt{h_L h_U}\ sign(t_{out} - t_0)$.
Stop when $1/2 < h/\bar{h} < 2$ (good enough, since the error control in first step may reset h).

2. The Nordsieck History Array

Recall: Data $\{\dot{y}_n, y_n, y_{n-1}, \ldots, y_{n-q+1}\}$ defines a unique polynomial $\omega_n^p(t)$ of degree $\leq q$ [earlier denoted $\omega^p(t)$].

The Nordsieck array is defined as the $N \times (q+1)$ array of scaled derivatives of $\omega = \omega_n^p$ at t_n:

$$z_n \equiv [y_n,\ h\dot{y}_n,\ \frac{h^2}{2!}\ddot{y}_n,\ \ \ldots,\ \frac{h^q}{q!}y_n^{(q)}]\ ,\quad y_n^{(k)} \equiv \omega^{(k)}(t_n)\ ,$$

where $h =$ current (tentative) step size.

Prediction of z_n from z_{n-1}:

$$z_{n(0)} = z_{n-1}A_q\ ,\quad A_q = \text{order } (q+1) \text{ Pascal Triangle.}$$

$$A_5 = \begin{pmatrix} 1 & & & & & \\ 1 & 1 & & & & \\ 1 & 2 & 1 & & & \\ 1 & 3 & 3 & 1 & & \\ 1 & 4 & 6 & 4 & 1 & \\ 1 & 5 & 10 & 10 & 5 & 1 \end{pmatrix}\ .$$

Done in place with repeated additions, not multiplies:

$$\text{for } k = 1...q\ \{\text{for } j = q...k\ \{z^{j-1} \leftarrow z^{j-1} + z^j\ \}\ \}\ ,$$

where $z^j =$ column j of z_{n-1} $(j = 0...q)$.

Correcting the history array.

The array $z_{n(0)}$ represents predictor polynomial $\omega_{n-1}^p(t)$ via scaled derivatives at t_n with step size $h = h_n$.

After computing and accepting y_n, in order to start the next step, we must represent $\omega_n^p(t)$, which interpolates the data set $\dot{y}_n, y_n, \ldots, y_{n+1-q}$. We do this by looking at $\Delta(t) \equiv \omega_n^p(t) - \omega_{n-1}^p(t)$, then adjusting the columns of $z_{n(0)}$ by the scaled derivatives $(h^j/j!)\Delta^{(j)}(t_n)$.

Recall that the BDF is

$$h(\dot{y}_n - \dot{y}_{n(0)}) = -\alpha_0(y_n - y_{n(0)}) \ ,$$

where $-\alpha_0 = \Sigma_{j=1}^q 1/j$. (The reason for this notation is that the BDF is often written in the alternate form

$$\sum_0^q \alpha_j y_{n-j} + h \sum_0^1 \beta_j \dot{y}_{n-j} = 0 \ , \quad \beta_0 = 1 \ .)$$

From the various interpolation conditions,

$$\Delta(t_{n-1}) = \Delta(t_{n-2}) = \ldots \Delta(t_{n+1-q}) = 0 \ ,$$

$$\Delta(t_n) = y_n - y_{n(0)} \equiv \Delta_n \ ,$$

$$\dot{\Delta}(t_n) = \dot{y}_n - \dot{y}_{n(0)} = (-\alpha_0/h)\Delta_n \ .$$

From its zeros and values at t_n, we can write

$$\Delta(t) = \Pi_{j=1}^{q-1} \left(\frac{t - t_{n-j}}{t_n - t_{n-j}} \right) [\Delta_n + (t - t_n)\mu\Delta_n]$$

for some scalar μ. We have

$$\begin{aligned}\dot{\Delta}(t) &= \left(\sum_{j=1}^{q-1}\frac{1}{t-t_{n-j}}\right)\left(\Pi_{j=1}^{q-1}\frac{t-t_{n-j}}{t_n-t_{n-j}}\right)[1+(t-t_n)\mu]\Delta_n \\ &+ \left(\Pi_{j=1}^{q-1}\frac{t-t_{n-j}}{t_n-t_{n-j}}\right)\mu\Delta_n\end{aligned}$$

$$h\dot{\Delta}(t_n) = \left(\sum_{j=1}^{q-1}\frac{h}{t_n-t_{n-j}}\right)\Delta_n + h\mu\Delta_n = (\sigma+h\mu)\Delta_n \ .$$

So $(\sigma+h\mu)\Delta_n = -\alpha_0\Delta_n$ implies $\mu = -(\sigma+\alpha_0)/h$.

Define dimensionless quantities

$$\xi_j \equiv (t_n - t_{n-j})/h = (h_n + h_{n-1} + \ldots + h_{n-j+1})/h$$

and

$$\xi^* \equiv 1/(h\mu) = 1/(-\alpha_0-\sigma) = 1/\left(\sum_1^q 1/j - \sum_1^{q-1} 1/\xi_j\right) \ .$$

At the same time, it helps to change from t to the dimensionless variable

$$x = \frac{t-t_n}{h} \ .$$

Then

$$\frac{t-t_{n-j}}{t_n-t_{n-j}} = \frac{t_n-t_{n-j}+hx}{h\xi_j} = 1 + x/\xi_j \ ,$$

and $1+(t-t_n)\mu = 1+x/\xi^*$.

We get

$$\Delta(t) = \Lambda(x)\Delta_n \ ,$$

where $\Lambda(x)$ is the scalar polynomial

$$\Lambda(x) = \left[\Pi_{j=1}^{q-1}(1 + x/\xi_j)\right](1 + x/\xi^*) \ .$$

Now the scaled derivatives of $\Delta(t)$ at t_n are just the Taylor coefficients of $\Lambda(x)\Delta_n$ at $x = 0$:

$$\frac{h^j \Delta^{(j)}(t_n)}{j!} = \frac{\Lambda^{(j)}(0)}{j!}\Delta_n = \ell_j \Delta_n \ ,$$

$$\Lambda(x) = \sum_0^q \ell_j x^j \ .$$

We can find the coefficients ℓ_j of this polynomial easily. Denoting $z_n = [z_n^0, z_n^1, \ldots, z_n^q]$, the correction to the history array is simply

$$z_n^j = z_{n(0)}^j + \ell_j \Delta_n \ \ (j = 0, \ldots, q) \ \ \text{or}$$

$$z_n = z_{n(0)} + \Delta_n \, \ell \ \ , \quad \ell \equiv (\ell_0, ..., \ell_q) \ .$$

Note that

$$\ell_1 = \sum_{j=1}^{q-1} 1/\xi_j + 1/\xi^* = \sum_{j=1}^{q} 1/j = 1/\beta_0 \ .$$

3. Interpolating to Output Times

Suppose the user requests the solution at $t = t_{out}$, which was just reached: $t_{n-1} < t_{out} \leq t_n$ or $t_{n-1} > t_{out} \geq t_n$. As the computed approximation to $y(t_{out})$, we return to the user

$$y_{out} \equiv \omega(t_{out})$$

using $\omega = \omega_n^p$.

As given by Taylor series based at t_n,

$$y_{out} = y_n + (t_{out} - t_n)\dot{y}_n + \ldots + \frac{(t_{out} - t_n)^q}{q!} y_n^{(q+1)} \ .$$

Given the Nordsieck array z_n, with columns z^j,

$$y_{out} = \sum_0^q \left(\frac{t_{out} - t_n}{h} \right)^j \ z^j \ .$$

We also provide, on request, solution derivatives

$$y_{out}^{(k)} = \omega^{(k)}(t_{out}) = \sum_{j=k}^{q} \left(\begin{array}{c} j \\ k \end{array} \right) \left(\frac{(t_{out} - t_n)^{j-k}}{h^j} \right) \ z^j$$

for $1 \leq k \leq q$.

In particular, this gives $\dot{y}(t_{out})$ cheaply and accurately. Evaluating $f(t_{out}, y_{out})$ is probably more expensive, and stiffness makes it less accurate.

4. The Newton Iteration Algorithm

Recall we are solving $G(y_n) = 0$ for y_n, where $G(y) = y - \gamma f(t_n, y) - a_n$ and $\gamma \equiv h\beta_0 = h/\ell_1$.

Newton iteration is $M\Delta y = -G(y_{n(m)})$, where M is some approximation to $G'(y) = I - \gamma J$ and $J \equiv \partial f/\partial y$.

PVODE (together with its serial twin CVODE) does two flavors of Newton iteration:

(a) Modified Newton in direct cases:
M fixed (usually out of date);
linear residual ≈ 0 via LU method;
relaxation w.r.t. $\bar{\gamma} \neq \gamma$ in M.

(b) Inexact Newton in Krylov case:
M current, using matrix-free product Jv;
precondition (left or right) with $P \approx M$;
linear residual nonzero but controlled.

In any case, the initial guess is

$$y_{n(0)} = \omega_{n-1}^p(t_n) = \text{column 0 of } z_{n(0)} \ ,$$

where ω_{n-1}^p represents the data at end of step $n-1$.

If a re-evaluation of M or P is done during step n, it is done at $(t_n, y_{n(0)})$, to maximize its effectiveness.

Jacobian/Preconditioner strategies.

The Newton matrix $M = I - \gamma J$ (direct cases) and preconditioner $P \approx M$ (Krylov case) are usually expensive. Balance between evaluation/preprocessing the matrix frequently (high cost) and infrequently (slow convergence).

We update M or P if:

* starting the problem ($n = 1$)

* > 20 steps have been taken since last update

* $|\gamma/\bar{\gamma} - 1| > .3$ where $\bar{\gamma} \equiv \gamma \big|_{\text{last update}}$

* just had a non-fatal convergence failure on this step

* just had an error test failure on this step

On an update of M or P, we may re-evaluate J (direct cases) or instruct user to re-evaluate Jacobian data in P (Krylov case). Or we may use a saved copy of J for M, or instruct user to use saved Jacobian data to form P.
On an update of M or P, we also re-evaluate if:

* starting the problem

* > 50 steps have been taken since the last evaluation

* just had conv. failure with J old and $|\gamma/\bar{\gamma} - 1| < .2$

* just had a convergence failure forcing h reduction

Convergence test.

The final computed value y_n will have to satisfy a local error test $\|y_n - y_{n(0)}\| \leq \epsilon_{LE}$. We want to insure that the iteration error $y_n - y_{n(m)}$ is small relative to ϵ_{LE}:

$$\|\text{iteration error in } y_{n(m)}\| < 0.1\epsilon_{LE} \ .$$

For this, estimate linear convergence rate constant R: We initialize R to 1; reset $R = 1$ when M or P is updated. After computing a correction $\delta_m = y_{n(m)} - y_{n(m-1)}$, we update R if $m > 1$ as

$$R \leftarrow \max\{0.3R, \|\delta_m\|/\|\delta_{m-1}\|\} \ .$$

Now

$$\begin{aligned} \|\text{error in } y_{n(m)}\| &= \|y_n - y_{n(m)}\| \approx \|y_{n(m+1)} - y_{n(m)}\| \\ &\approx R\|y_{n(m)} - y_{n(m-1)}\| = R\|\delta_m\| \ . \end{aligned}$$

So the convergence test is

$$R\|\delta_m\| < 0.1\epsilon_{LE} \ .$$

If convergence is superlinear (possible in Newton-Krylov case), then the error in $y_{n(m)}$ is even smaller.

We allow at most 3 Newton iterations, and declare the iteration diverged if any $\|\delta_m\|/\|\delta_{m-1}\| > 2$.

If convergence fails with J or P current, cut $h \leftarrow h/4$.

5. Local Error Test

For BDFq, at current step size $h = h_n$,

$$\text{Local Truncation Error} = Ch^{q+1}y^{(q+1)}(t_n) + O(h^{q+2})$$

for some C, where the ratios h_j/h are assumed bounded (above and away from zero). To express C, recall

$$\xi_j = (t_n - t_{n-j})/h \ , \quad \alpha_0 = -\sum_{j=1}^{q} 1/j \ ,$$

and also define

$$\hat{\alpha}_{n,0} = -\sum_{j=1}^{q} 1/\xi_j \ .$$

Then

$$C = \frac{(\Pi_1^q \xi_j)(\alpha_0 + 1 - \hat{\alpha}_{n,0})}{\alpha_0 (q+1)!} \ .$$

There is a similar asympotic formula for the LTE of the predictor formula for $y_{n(0)}$, with coefficient

$$C^p = \frac{(\Pi_1^q \xi_j)}{(q+1)!} \ .$$

In terms of $\Delta_n \equiv y_n - y_{n(0)}$, the LTE for corrector is

$$\text{LTE } = C'(y_n - y_{n(0)}) + O(h^{q+2}) \ , \text{ where}$$

$$C' = \frac{C}{C^p + q\alpha_0 C} = \frac{(\alpha_0 + 1 - \hat{\alpha}_{n,0})}{\alpha_0[1 + q(\alpha_0 + 1 - \hat{\alpha}_{n,0})]} \ .$$

For derivation, see Jackson & Sacks-Davis [10].

The local error test is:

$$\|\text{estimated LTE}\| \le 1 \ , \quad \text{or}$$

$$\|\Delta_n\| = \|y_n - y_{n(0)}\| \le 1/|C'| \equiv \epsilon_{LE} \ .$$

If test passes, go on to correction $z_{n(0)} \to z_n$

If test fails,

(a) restore z_{n-1} from $z_{n(0)}$ by repeated subtractions

(b) reset step size to $h' =$ solution of

$$(h'/h)^{q+1}\|\Delta_n\| = \epsilon_{LE}/6$$

($1/6 =$ safety factor, to account for deviations from asymptotic behavior)

(c) rescale z_{n-1}: $z^j_{n-1} \leftarrow (h'/h)^j z^j_{n-1}$

(d) retry the step (predict, correct, etc.)

If error test fails repeatedly (7 times) or $|h|$ reaches a user-supplied minimum, give up.

After 3 error test failures, we force an order reduction if $q > 1$, or restart from scratch if $q = 1$.

The ratio h'/h is limited (above) to .2 after 2 error test failures, and limited below to .1 after 3 failures.

6. Step and Order Selection

Basic idea: Pick the order such that the polynomial of that degree best fits the discrete data on the given t mesh.

First rule: Keep the current q and h if the current step had either a convergence failure or an error test failure.

Step selection at current order.

Since the error test $\|\Delta_n\| \leq \epsilon_{LE}$ passed,

$$(h'/h)^{q+1}\|\Delta_n\| = \epsilon_{LE}$$

may define a larger step size. With safety factor, use

$$h'/h = \left[\frac{\epsilon_{LE}}{6\|\Delta_n\|}\right]^{\frac{1}{q+1}} \equiv \eta_q$$

as the tentative step size ratio at order q.

Order selection timing.

We consider a change of order only after taking $q+1$ steps at order q: a heuristic with partial theoretical support. In that case, consider orders $q' = q-1$ (if $q > 1$) and $q' = q+1$ (if $q < 5$).

In a future version of PVODE: Following any step with $q \geq 3$, force a reduction to order $q-1$ if h is limited by the boundary of the Absolute Stability Region.

Consider order $q-1$.

The Local Truncation Error at order $q' = q-1$ is

$$\text{LTE}_{q-1} = C_{q-1} h^q y^{(q)}(t_n) + O(h^{q+1}) \ .$$

By a derivation similar to that for the order q error test,

$$C_{q-1} = \frac{(\Pi_1^{q-1} \xi_j)[(\alpha_0 + 1/q) + 1 - (\hat{\alpha}_{n,0} + 1/\xi_q)]}{(\alpha_0 + 1/q)\ q!} \ .$$

Here $h^q y^{(q)}(t_n)$ is easily estimated as $q! z_n^q$. So

$$\text{est.LTE}_{q-1} = C_{q-1} q! z_n^q \ .$$

We set a tentative new step size as before:

$$(h'/h)^q \|\text{est.LTE}_{q-1}\| = 1/6 \ , \quad \text{or}$$

$$h'/h = \left[\frac{1}{6\|\text{est.LTE}_{q-1}\|}\right]^{\frac{1}{q}} \equiv \eta_{q-1} \ .$$

Consider order $q+1$.

The Local Truncation Error at order $q' = q+1$ is

$$\text{LTE}_{q+1} = C_{q+1} h^{q+2} y^{(q+2)}(t_n) + O(h^{q+3}) \ ,$$

$$C_{q+1} = \frac{(\Pi_1^{q+1} \xi_j)[\alpha_0(q+1) + 1 - \hat{\alpha}_{n,0}(q+1)]}{\alpha_0(q+1)\ (q+2)!} \ .$$

We estimate $h^{q+2}y^{(q+2)}(t_n)$ using

$$\Delta_n = y_n - y_{n(0)} \approx \bar{C}_n h^{q+1} y^{(q+1)}(t_n) \quad \text{and}$$

$$\Delta_{n-1} = y_{n-1} - y_{n-1(0)} \approx \bar{C}_{n-1} h_{n-1}^{q+1} y^{(q+1)}(t_{n-1}) \ .$$

Even though the justification appears shaky, we take

$$h^{q+2}y^{(q+2)}(t_n) \approx \frac{1}{\bar{C}_n}\Delta_n - \frac{1}{\bar{C}_{n-1}}(h/h_{n-1})^{q+1}\Delta_{n-1} \ .$$

(We saved Δ_{n-1} and $\bar{C}_{n-1}$ at end of step $n-1$ if the waiting period was about to end.)
Insert into LTE equation to get est.LTE$_{q+1}$.
Use a smaller safety factor $= 1/10$ for tentative h':

$$h'/h = \left[\frac{1}{10\|\text{est.LTE}_{q+1}\|}\right]^{\frac{1}{q+2}} \equiv \eta_{q+1} \ .$$

Order selection.

We want the next step size to be as large as possible:

$$\eta = \max\{\eta_{q-1}, \eta_q, \eta_{q+1}\} \ .$$

Set $q' = q$ or $q \pm 1$ accordingly, and $h' = \eta h$.

Two final rules:
Don't bother changing q or h if $\eta < 1.5$.
Limit h'/h to 10^4 on step 1, and to 10 otherwise.

7. Adjustments on Change of Step or Order

Changes to h, q, and z_n decided on at end of step n are made at start of step $n+1$, in order not interfere with calculation of interpolated output values.

Actions when $q' \neq q$:

Adjust z_n to reflect new set of interpolated data.

Additional actions whenever $h' \neq h$:

Rescale the z_n^j by η^j $(j = 1..q)$, set $h = h'$, $q = q'$.

Adjustment for $q' = q - 1$.

Array z_n represents $\omega(t) = \omega_n^p(t)$, of degree q, which interpolates $\dot{y}_n,\ y_n,\ y_{n-1},\ \ldots,\ y_{n-q+1}$.

We need an array $\bar{z}_n$ representing $\bar{\omega}(t)$, of degree $q-1$, which interpolates $\dot{y}_n,\ y_n,\ y_{n-1},\ \ldots,\ y_{n-q+2}$.

As before, do this by determining the difference polynomial, $A(t) \equiv \omega(t) - \bar{\omega}(t)$ (of degree q). We have

$$A(t_n) = A(t_{n-1}) \ldots = A(t_{n-q+2}) = 0 = \dot{A}(t_n)$$

and the coefficient of t^q in $A(t)$ is the same as for $\omega(t)$, namely $y_n^{(q)}/q! = z_n^q/h^q$. This implies

$$A(t) = (t - t_n)^2 \left[\Pi_{j=1}^{q-2}(t - t_{n-j})\right] \frac{y_n^{(q)}}{q!} \ .$$

In terms of $x = (t - t_n)/h$ and $\xi_j = (t_n - t_{n-j})/h$,

$$\begin{aligned} A(t) &= (hx)^2 \left[\Pi_{j=1}^{q-2}(t_n - t_{n-j} + hx)\right] \frac{y_n^{(q)}}{q!} \\ &= x^2 \left[\Pi_{j=1}^{q-2}(\xi_j + x)\right] \frac{h^q y_n^{(q)}}{q!} = d(x) z_n^q \ . \end{aligned}$$

We compute the coefficients d_j of $d(x) = x^2 \Pi_1^{q-2}(\xi_j + x)$, then adjust z_n (in place) by

$$\bar{z}_n^j = z_n^j - d_j z_n^q \quad (j = 2, \ldots, q-1) \ .$$

Adjustment for $q' = q + 1$.

We need an array $\bar{z}_n$ representing $\bar{\omega}(t)$, of degree $q+1$, which interpolates $\dot{y}_n,\ y_n,\ y_{n-1},\ \ldots,\ y_{n-q}$.
Determine $A \equiv \bar{\omega} - \omega$ (of degree $q+1$). We have

$$A(t_n) = A(t_{n-1}) \ldots = A(t_{n-q+1}) = 0 = \dot{A}(t_n)$$

and $A(t_{n-q}) = y_{n-q} - \omega(t_{n-q}) \equiv A_{n-q}$. Therefore

$$A(t) = \left(\frac{t - t_n}{t_{n-q} - t_n}\right)^2 \left[\Pi_{j=1}^{q-1} \left(\frac{t - t_{n-j}}{t_{n-q} - t_{n-j}}\right)\right] A_{n-q} \ .$$

Setting $t = t_n + hx$, we have

$$\begin{aligned} A(t) &= \left(\frac{hx}{t_{n-q} - t_n}\right)^2 \left[\Pi_{j=1}^{q-1} \left(\frac{t_n - t_{n-j} + hx}{t_{n-q} - t_{n-j}}\right)\right] A_{n-q} \\ &= \left(\frac{x}{\xi_q}\right)^2 \left[\Pi_{j=1}^{q-1} \left(\frac{\xi_j + x}{\xi_j - \xi_q}\right)\right] A_{n-q} \ . \end{aligned}$$

Now recall the polynomial $\Delta(t) = \omega_n^p(t) - \omega_{n-1}^p(t)$ we constructed to correct $z_{n(0)}$ to z_n. We have

$$\Delta(t_{n-q}) = \omega_n^p(t_{n-q}) - y_{n-q} = -A_{n-q} \ .$$

From the construction $\Delta(t) = \Lambda(x)\Delta_n, \quad \Delta_n \equiv y_n - y_{n(0)}$. Evaluating at $t = t_{n-q}$, where $x = (t_{n-q} - t_n)/h = -\xi_q$,

$$\begin{aligned} A_{n-q} &= -\Delta(t_{n-q}) = -\Lambda(-\xi_q)\Delta_n \\ &= -(1 - \xi_q/\xi^*)\left[\Pi_{j=1}^{q-1}(1 - \xi_q/\xi_j)\right]\Delta_n \\ &= \xi_q(1/\xi^* - 1/\xi_q)\left[\frac{\Pi_{j=1}^{q-1}(\xi_j - \xi_q)}{\Pi_{j=1}^{q-1}\xi_j}\right]\Delta_n \ . \end{aligned}$$

Substituting, we get

$$\begin{aligned} A(t) &= x^2\Pi_{j=1}^{q-1}(\xi_j + x) \quad \left[\frac{1/\xi^* - 1/\xi_q}{\Pi_{j=1}^{q}\xi_j}\right] \quad \Delta_n \\ &= \qquad d(x) \qquad\qquad \bar{d} \qquad\qquad \Delta_n \ . \end{aligned}$$

We compute the coefficients d_j of the polynomial $d(x)$, compute the constant $\bar{d}$, and then adjust z_n by

$$\bar{z}_n^j = z_n^j + d_j\bar{d}\Delta_n \quad (j = 2, \ldots, q+1) \ .$$

Here $z_n^{q+1} = 0$, so this creates a new column $q+1$ for the history array.

Reference: Brown notes [2]

8. Example PVODE Run

Ozone model: two-species reaction-advection-diffusion in 2D with diurnal kinetics; time span = 24 hours.
(See PVODE User Document [6]).

Solution uses BDF + GMRES + left preconditioning;
P is block-diagonal with 2x2 blocks (no spatial coupling).

Rough history of order q and step size h:

t: 0 ..... 0.04s ..... 0.3s ..... (11-56s)... 12.4h ... 24h

q: 1 .... 2 ... 3 ... 4 ... 5 ... 4 ... 3 ... 2 .. (3-5) .. 5 5

h: 0.4ms ... 7ms ..... 0.03s ..... (2-131s) .... (6-14min)

Total time steps NST = 467

Total nonlinear iters. NNI = 586. Average = 1.255/step

Total linear iters. NLI = 588. Average = 1.003/Newton

Total P setups NSETUPS = 72. Average = 6.5 steps/setup

Total P evaluations NPE = 8. Average = 58.4 steps/eval.

Total error test failures = 23

Total Newton convergence failures = 0

Total Krylov convergence failures = 0

The PVODE and IDA Algorithms

Alan C. Hindmarsh

Part C: IDA

1. Initial condition calculation
2. Initial step size
3. Storing the history
4. Interpolating to output times
5. Newton iteration algorithm
6. Local error test
7. Step and order selection
8. Inequality constraints

0. Introduction

IDA[ACH,Taylor] evolved from
DASPK [Brown, ACH, Petzold], a variant of
DASSL [Petzold].

Change of notation: $F(t, y, y') = 0$, $y' \equiv dy/dt$, $y \in R^N$.

Discrete values are t_n and $y_n \approx y(t_n)$, with step sizes $h_n = t_n - t_{n-1}$.

In terms of predicted values $y_{n(0)}$ and $y'_{n(0)}$, the fixed-leading-coefficient BDF of order k has the form

$$\alpha_s(y_n - y_{n(0)}) + h_n(y'_n - y'_{n(0)}) = 0 \ ,$$

where

$$\alpha_s \equiv -\sum_{j=1}^{k} \frac{1}{j} \ .$$

Order k varies between 1 and 5.

1. Initial Condition Calculation

For the DAE system $F(t, y, y') = 0$, the user is to supply t_0 and initial condition vectors y_0 and y'_0. But it may be difficult or impossible to supply these consistent with the DAE system.

Problem 1. Suppose $F = 0$ corresponds to an ODE system with constraints, also called a "semi-explicit index-1" system:

$$u' = f(t, u, v) \ , \quad g(t, u, v) = 0 \quad (\partial g/\partial v \text{ nonsingular}) \ .$$

User may know $u_0 = u(t_0)$, but not have a consistent v_0.

Problem 2. For general F, suppose y'_0 is given but y_0 is unknown, and $\partial F/\partial y$ is nonsingular.

In both cases, IDA can help the user out.

Initialization Problem 1.

Allowing u' to be implicit, take the more general form:

$$F = \begin{pmatrix} f(t, u, v, u') \\ g(t, u, v) \end{pmatrix}$$

with $\partial f/\partial u'$ and $\partial g/\partial v$ both square and nonsingular. IDA actually accepts a more general class of problems: the differential components u and algebraic components v can be permuted, and the constraints may be implicit.

Because we also need u_0' to start the integration, define

$$x = \begin{pmatrix} u_0' \\ v_0 \end{pmatrix} \quad , \quad \phi(x) = \begin{pmatrix} f(t_0, u_0, v_0, u_0') \\ g(t_0, u_0, v_0) \end{pmatrix} .$$

Then we want to solve $\phi(x) = 0$ for x. The Jacobian,

$$\phi'(x) = \partial\phi/\partial x = \begin{pmatrix} f_{u'} & f_v \\ 0 & g_v \end{pmatrix} ,$$

is nonsingular (subscripts denote differentiation).

We want to do Newton iteration, but do not want to set up any new machinery, beyond what is involved for the integration of the DAE system. The integration involves solving Newton correction systems $J\Delta y = -F$, with

$$J = \partial F/\partial y + \alpha \partial F/\partial y' \ , \quad \alpha \equiv -\alpha_s/h \ .$$

The user is either supplying J or letting IDA generate J by difference quotients (direct cases), or is supplying and solving a preconditioner $P \approx J$ (Krylov case).

For the case of Problem 1,

$$J = \begin{pmatrix} f_u + \alpha f_{u'} & f_v \\ g_u & g_v \end{pmatrix} .$$

Now we play a trick. Artificially set h, and $\alpha = 1/h$, and approximate $\phi'(x)$ with a scaled form of J: Let

$$S \equiv \begin{pmatrix} hI_u & 0 \\ 0 & I_v \end{pmatrix} \quad , \quad \bar{J} \equiv JS = \begin{pmatrix} f_{u'} + hf_u & f_v \\ hg_u & g_v \end{pmatrix} .$$

Note that as $h \to 0$, $\bar{J} \to \phi'$.

Given that we can realize the operator J^{-1}, we can realize $\bar{J}^{-1} = S^{-1}J^{-1}$, an approximate inverse of ϕ'. We must pick h small enough to make $\bar{J} \approx \phi'$ but not so small that J is too badly conditioned.

The basic IC Calculation algorithm:

1. Pick an appropriate small h; set $\alpha = 1/h$.
2. Form J, or have user set up preconditioner P.
3. Set $r = -F(t_0, y_0, y_0')$ at current guess $x = \begin{pmatrix} u_0' \\ v_0 \end{pmatrix}$.
4. Solve $Jp = r$ by direct or precond. Krylov solve.
5. Set $\Delta x = S^{-1}p \;\; (= \bar{J}^{-1}r)$ and $x \leftarrow x + \Delta x$.
6. Loop to Step 3 until $\|p\|$ small.
7. If converging slowly, update J or P and continue from Step 3 with current x.
8. If diverging, reduce h and restart with original x.

Impact on user: must input integer vector ID identifying differential components u and algebraic components v.

Extends to index-0 case: solve $F(t_0, y_0, y_0') = 0$ for y_0' (no v or g). But the convergence test on $\|p\|$ must be rescaled to remove its artificial near-proportionality to h.

Initialization Problem 2.

Here we simply want to solve $\phi(y_0) \equiv F(t_0, y_0, y_0') = 0$.

The Jacobian $\phi' = F_y$ is simply the value of the system Jacobian J with $\alpha = 0$. So we use the same algorithm as in Problem 1, but pass $\alpha = 0$ at each point where J or P is evaluated and preprocessed (no h, hence no Step 8).

In both cases, the actual algorithm is more complicated. See [5]. The main complication is a linesearch backtracking algorithm to improve global convergence. It guarantees a reduction of $\|\phi\|^2$ at each Newton step, with relaxation: $x \leftarrow x + \lambda \Delta x$.

2. The Initial Step Size

PVODE uses an algorithm based on the Local Truncation Error of the order 1 method: estimate y_0'' and solve $h^2\|y_0''\|/2$ for h.

This is not feasible in IDA; $F = 0$ provides no way to get y'' analogous to PVODE's differencing of $y' = f$.

IDA resorts to using the LTE of the "order 0" method, $y_{n+1} = y_n$. For the order 0 step $y_1 = y_0$, the leading term of the LTE is hy_0', and we have y_0'. So as the tentative value of h take the solution of $\|hy_0'\| = 1$.

Adjustments to this choice:

add a safety factor;

restrict to a fraction of first output interval $|t_{out} - t_0|$;

attach the proper sign.

The result is:

$$h = sign(t_{out} - t_0) \min\left\{.001|t_{tout} - t_0|, \frac{1}{2\|y_0'\|}\right\} .$$

3. Storing the History

Classical divided differences of $\{y_n\}$:

$$[y_n] \equiv y_n \ , \quad [y_n,\ y_{n-1}] \equiv \frac{y_n - y_{n-1}}{t_n - t_{n-1}} \ ,$$

$$[y_n,\ y_{n-1},\ \ldots,\ y_{n-k}] \equiv \frac{[y_n,\ y_{n-1},\ \ldots,\ y_{n-k+1}] - [y_{n-1},\ y_{n-2},\ \ldots,\ y_{n-k}]}{t_n - t_{n-k}} \ .$$

For a smooth $y(t)$,

$$[y_n,\ y_{n-1},\ \ldots,\ y_{n-k}] = y^{(k)}(\xi)/k! \ , \quad \xi \in [t_{n-k}, t_n] \ .$$

Modified divided differences:

First define the t differences $(j = 1, \ldots, k)$

$$\psi_j(n) \equiv t_n - t_{n-j} = h_n + \ldots + h_{n+1-j}$$

$$[\psi_1(n) = h_n]$$

For $j = 0, \ldots, k$, the modified divided differences are:

$$\phi_0(n) \equiv y_n \ ,$$

$$\phi_1(n) \equiv \psi_1(n)[y_n,\ y_{n-1}] = y_n - y_{n-1} \ ,$$

$$\phi_j(n) \equiv \psi_1(n) \cdots \psi_j(n)[y_n,\ \ldots,\ y_{n-j}] \ .$$

Exception at $n = 0$:

$$\psi_1(0) \equiv h_1 \ , \quad \phi_1(0) \equiv h_1 y_0'$$

as if a BDF1 step were taken from y_{-1} to y_0 with $h_0 = h_1$.

We will also need some other scalars:

$$\begin{aligned}
\alpha_j(n) &\equiv h_n/\psi_j(n) && [\alpha_1(n) = 1] \\
\beta_j(n) &\equiv \frac{\psi_1(n)\cdots\psi_{j-1}(n)}{\psi_1(n-1)\cdots\psi_{j-1}(n-1)} && [\beta_1(n) = 1] \\
\gamma_j(n) &\equiv \sum_{\ell=1}^{j-1} 1/\psi_\ell(n) && [\gamma_1(n) = 0, \gamma_2(n) = 1/h_n]
\end{aligned}$$

Interpolating polynomial.
Given $y_n, y_{n-1}, \ldots, y_{n-k}$, there is an interpolating polynomial $\omega_n(t)$ of degree $\leq k$ that interpolates this data set. In terms of classical divided differences, this is

$$\begin{aligned}
\omega_n(t) &= y_n + (t-t_n)[y_n, y_{n-1}] + \\
&\quad (t-t_n)(t-t_{n-1})[y_n, y_{n-1}, y_{n-2}] + \ldots \\
&\quad (t-t_n)(t-t_{n-1})\cdots(t-t_{n-k+1})[y_n, \ldots, y_{n-k}]
\end{aligned}$$

For $j = 0, \ldots, k$, the j-th term above is

$$\begin{aligned}
\omega_{n,j}(t) &\equiv \left[\Pi_{\ell=0}^{j-1}(t-t_{n-\ell})\right][y_n, y_{n-1}, \ldots, y_{n-j}] \\
&= \left[\Pi_{\ell=0}^{j-1}(t-t_{n-\ell})\right]\phi_j(n)/\left[\Pi_{\ell=1}^{j}\psi_\ell(n)\right] \\
&= C_j(t)\phi_j(n)\ , \qquad C_j(t) \equiv \Pi_{\ell=0}^{j-1}\left(\frac{t-t_{n-\ell}}{\psi_{\ell+1}(n)}\right)
\end{aligned}$$

Prediction.
If we have only reached t_{n-1}, we use ω_{n-1} to predict $y_{n(0)} = \omega_{n-1}(t_n)$ and $y'_{n(0)} = \omega'_{n-1}(t_n)$.

In terms of the $\{\phi_j(n-1)\}$ this is given by:

$$\phi_j^*(n) = \beta_{j+1}(n)\phi_j(n-1) \quad (j = 0 \ldots k)$$

$$y_{n(0)} = \sum_0^k \phi_j^*(n) \ , \quad y'_{n(0)} = \sum_1^k \gamma_{j+1}(n)\phi_j^*(n) \ .$$

The ϕ_j^* are overwritten on the ϕ_j.

Correcting the history array.
Given y_n and $\Delta_n = y_n - y_{n(0)}$, the correction to the history is:

$$\phi_k(n) = \phi_k^*(n) + \Delta_n \ ,$$

$$\text{For } j = k-1, \ldots, 0 : \phi_j(n) = \phi_j^*(n) + \phi_{j+1}(n) \ .$$

Again, the ϕ_j are overwritten on the ϕ_j^*.

References: Brenan/Campbell/Petzold [1] (Sec. 5.2) and Shampine/Gordon book Computer Solution of ODEs (Chapter 5), with shift in j index by 1 in ϕ and ϕ^*.

4. Interpolating to Output Times

Suppose the user requests the solution at $t = t_{out}$, which was just reached: $t_{n-1} < t_{out} \le t_n$ or $t_{n-1} > t_{out} \ge t_n$. IDA returns to the user two vectors

$$y_{out} \equiv \omega(t_{out}) \ , \quad y'_{out} \equiv \omega'(t_{out}) \ ,$$

where $\omega(t) = \omega_n(t)$.

Use the $\phi_j(n)$ and $\psi_j(n)$ to compute

$$\begin{aligned} \omega(t_{out}) &= \sum_0^k C_j(t_{out})\phi_j(n) \ , \\ \omega'(t_{out}) &= \sum_1^k C'_j(t_{out})\phi_j(n) \end{aligned}$$

The interpolant is continuous, but not C^1.

Implementation:

$y \leftarrow y_n \ , \quad y' \leftarrow 0$

$C = 1 \ , \quad D = 0 \ , \quad \gamma = (t_{out} - t_n)/\psi_1$

For $j = 1, \ldots k$:

$$\begin{aligned} D &= D\gamma + C/\psi_j \ , \quad C = C\gamma \\ \gamma &= (t_{out} - t_n + \psi_j)/\psi_{j+1} \\ y &\leftarrow y + C\phi_j(n) \ , \quad y' \leftarrow y' + D\phi_j(n) \end{aligned}$$

5. The Newton Iteration Algorithm

Recall we are solving $G(y_n) = 0$ for y_n, where

$$G(y) = F(t_n,\ y,\ \alpha y + \beta)\ ,$$

with $\alpha \equiv -\alpha_s/h_n$ and $\beta \equiv y'_{n(0)} - \alpha y_{n(0)}$.

Newton iteration is $J\Delta y = -G(y_{n(m)})$, where J is some approximation to $G'(y) = \partial F/\partial y + \alpha \partial F/\partial y'$.

IDA does two flavors of Newton iteration:

(a) Modified Newton in direct cases:
J fixed (usually out of date);
linear residual ≈ 0 via LU method;
relaxation w.r.t. $\bar{\alpha} \neq \alpha$ in J.

(b) Inexact Newton in Krylov case:
J current, using matrix-free product Jv;
precondition on left* with $P \approx J$;
linear residual nonzero but controlled.

*Note: Left preconditioning is required to make norm ||linear residual|| meaningful; $\|J\Delta y + G\|$ is meaningless in general, since the weights are weights for y.

In any case, the initial guess is $y_{n(0)} = \omega_{n-1}(t_n)$, where ω_{n-1} represents the data at end of step $n-1$.

If a re-evaluation of J or P is done during step n, it is done at $(t_n, y_{n(0)}, y'_{n(0)})$, to maximize its effectiveness.

Jacobian/Preconditioner strategies.

The Newton matrix J (direct cases) and preconditioner $P \approx J$ (Krylov case) are usually expensive.
Balance between evaluation/preprocessing the matrix frequently (high cost) and infrequently (slow convergence).

We update J or P if:

* starting the problem ($n = 1$)

* $\alpha/\bar{\alpha} < 3/5$ or $\alpha/\bar{\alpha} > 5/3$, where $\bar{\alpha} \equiv \alpha\big|_{\text{last update}}$

* convergence failed non-fatally with old J or P

On an update of J or P, Jacobian data is always re-evaluated from scratch. Two-level update strategy of PVODE would require saving both $\partial F/\partial y$ and $\partial F/\partial y'$.

Convergence test.

We want to insure that the iteration error $y_n - y_{n(m)}$ is small relative to y itself, specifically:

$$\|\text{iteration error in } y_{n(m)}\| < 0.33 \ .$$

(Contrast PVODE's nonlinear test constant, which is proportional to the local error test constant.)

For this, estimate linear convergence rate constant R:
Corrections are $\delta_m = y_{n(m)} - y_{n(m-1)}$ for $m = 1, 2, \ldots$
If $m > 1$, we set

$$R = (\|\delta_m\|/\|\delta_1\|)^{\frac{1}{m-1}} \ .$$

We stop the Newton iteration if $R > 0.9$.

Now suppose R satisfies $\|\delta_{\ell+1}\| \le R\|\delta_\ell\|$ for all $\ell \ge m$. Then

$$y_n = y_{n(m)} + \delta_{m+1} + \delta_{m+2} + \ldots \qquad \text{implies}$$

$$\begin{aligned} \|y_n - y_{n(m)}\| \ &= \ \| \sum_{\ell=m+1}^{\infty} \delta_\ell \| \le \sum_{\ell=m+1}^{\infty} \|\delta_\ell\| \\ &\le \ \sum_{\ell=1}^{\infty} R^\ell \|\delta_m\| = \left(\frac{R}{1-R}\right) \|\delta_m\| \ . \end{aligned}$$

So when $m > 1$ and $R \le 0.9$, we set $S = R/(1-R)$.
Then for any m, the convergence test is

$$S\|\delta_m\| < 0.33 \ ,$$

but for $m = 1$ this uses an old value for S.

If convergence is superlinear (possible in Newton-Krylov case), then the error in $y_{n(m)}$ is even smaller.

We initialize S to 20, reset $S = 20$ each time J or P is updated, and reset $S = 100$ on a step with $\alpha \neq \bar{\alpha}$. This encourages recalculation of R on a major change in the Newton matrix.

If $m = 1$, we make an additional test, and stop the Newton iteration if $\|\delta_1\| < 10^{-4} \cdot 0.33$, because such a δ_1 is probably just noise and so not appropriate for use in R.

We allow at most 4 Newton iterations.

If convergence fails with J or P current, cut $h \leftarrow h/4$.

If convergence fails 10 times, give up.

Krylov convergence test.
We control the preconditioned linear residual to be small compared to the allowed error in the Newton iteration:

$$\|P^{-1}(Jx + G)\| < 0.05 \cdot 0.33 \ .$$

(Because the generic SPGMR solver uses L_2 norms of scaled vectors, it is given a tolerance of $\sqrt{N} \cdot 0.05 \cdot 0.33$ in order to obtain a WRMS-norm test.)

6. The Local Error Test

The BDF can be written in the alternate form, expressing $h_n y'_n$ as a series in the ϕ_j:

$$h_n y'_n = \sum_{j=1}^{k} \alpha_j(n)\phi_j(n) - [\alpha_s - \alpha^0(n)]\phi_{k+1}(n) \ ,$$

where

$$\alpha^0(n) \equiv - \sum_{j=1}^{k} \alpha_j(n) \ .$$

The true solution $y(t)$ (if sufficiently smooth) satisfies

$$h_n y'(t_n) = \sum_{j=1}^{\infty} \alpha_j(n)\phi_j(n) \ ,$$

in which the ϕ_j are evaluated on the solution $y(t)$.
In this series, $\phi_j(n)$ is $O(h^j)$ as the stepsizes all go to zero, bounded by h.

The Local Truncation Error of the BDF is the remainder in the formula when evaluated on a smooth solution.
So the LTE is

$$\text{LTE}_n = [\alpha_{k+1}(n) + \alpha_s - \alpha^0(n)]\phi_{k+1}(n) + O(h^{k+2}) \ .$$

This has been shown to be asymptotically correct for fixed stepsizes and for somewhat more general conditions.

Using the interpolatory conditions of the predictor and corrector polynomials, one can also show that

$$\phi_{k+1}(n) = \Delta_n \equiv y_n - y_{n(0)} \ .$$

Thus the estimated LTE is

$$\text{ELTE}_n = [\alpha_{k+1}(n) + \alpha_s - \alpha^0(n)]\Delta_n \ .$$

IDA requires that $\|\text{ELTE}_n\| \leq 1$. But it also requires that the interpolation error in $\omega_n(t)$ is bounded by 1 in WRMS-norm for t in the last step interval $[t_{n-1}, t_n]$. The principal term of this interpolation error can be bounded by

$$\alpha_{k+1}(n)\|\phi_{k+1}(n)\| \ .$$

So the Local Error Test in IDA is:

$$\max\{\alpha_{k+1}(n),\ |\alpha_{k+1}(n) + \alpha_s - \alpha^0(n)|\}\|\Delta_n\| \leq 1 \ .$$

[Note that this coefficient involves $h_n, h_{n-1}, \ldots, h_{n-k}$.
I.e. BDFk is treated as a $(k+1)$-step method.
BDF1 = Backward Euler is treated as a 2-step method.]

Optional altered Local Error Test: If y has algebraic components v that are coupled to the other components u, it is sometimes best to control the local error only in u. The ODE-based LTE theory is less likely to apply to v.

For this option, use must input bit vector ID identifying u vs v. IDA then forms a masked copy of the reciprocal weight vector w^{-1} for use in the Local Error Test.

7. Step and Order Selection

Special initial phase: For the first few steps, until

* the local error test fails, or
* the order is reduced, or
* $k = 5$ (the maximum order),

we raise the order by 1 and double the step size.

IDA's step/order selection uses Local Truncation Error estimates that apply in the fixed-step case, even though the last $k+1$ step sizes may not have been constant.

At fixed h, the leading term of the LTE at order k is

$$\frac{1}{k+1} h^{k+1} y_n^{(k+1)} \ .$$

To estimate this, we use another set of constants,

$$\sigma_j(n) \equiv \frac{h_n^j (j-1)!}{\psi_1(n) \cdots \psi_j(n)} \qquad [\sigma_1(n) = 1] \ ,$$

so that

$$\sigma_j(n)\phi_j(n) = h_n^j (j-1)! [y_n, \ldots, y_{n-j}] \approx h_n^j y_n^{(j)}/j \ .$$

Actual order choice is not based on maximizing h, but on requiring (roughly) that the $\|h^j y^{(j)}\|$ be monotonically decreasing for j near k. Estimate $h_n^j y_n^{(j)}$ as $j\sigma_j(n)\phi_j(n)$. This helps indirectly with BDF stability limit problem.

(a) Actions *before* passing the Local Error Test.

Set order k test quantities:

$$\begin{aligned} \mathrm{ELTE}_k &= E_k = \sigma_{k+1}(n)\|\phi_{k+1}(n)\| \\ &= \sigma_{k+1}(n)\|\Delta_n\| \\ \text{est.}\|h_n^{k+1}y_n^{(k+1)}\| &= T_k = (k+1)E_k \ . \end{aligned}$$

Set order $k-1$ test quantities (if $k > 1$):

$$\begin{aligned} \mathrm{ELTE}_{k-1} &= E_{k-1} = \sigma_k(n)\|\phi_k(n)\| \\ &= \sigma_k(n)\|\phi_k^*(n) + \Delta_n\| \\ \text{est.}\|h_n^k y_n^{(k)}\| &= T_{k-1} = kE_{k-1} \ . \end{aligned}$$

Set order $k-2$ test quantities (if $k > 2$):

$$\begin{aligned} \mathrm{ELTE}_{k-2} &= E_{k-2} = \sigma_{k-1}(n)\|\phi_{k-1}(n)\| \\ &= \sigma_{k-1}(n)\|\phi_{k-1}^*(n) + \phi_k^*(n) + \Delta_n\| \\ \text{est.}\|h_n^{k-1}y_n^{(k-1)}\| &= T_{k-2} = (k-1)E_{k-2} \ . \end{aligned}$$

Set the new order to:

$k' = k-1$ if $\quad k > 2$ & $\max(T_{k-1}, T_{k-2}) \le T_k$ or
$\qquad\qquad\qquad\quad k = 2$ & $T_1 \le T_2/2$

$k' = k$ otherwise .

Then do the Local Error Test.

Actions if Local Error Test fails:

- Restore $\{\phi_j\}_0^k$ and $\{\psi_j\}_1^k$ to step $n-1$ values
- Set $k = k'$
- Set $r = h'/h$ according to asymptotic h^{k+1} behavior of LTE, but use fixed-step LTE estimates, with safety factors:
$$r = 0.9/(2E_k)^{\frac{1}{k+1}} \ ,$$
adjusted so that $.25 \leq r \leq .9$
- On second failure, set $r = .25$
- On third and later failures, set $k = 1, r = .25$
- Set $h \leftarrow h' = rh$
- Retry step at order k, step size h
- Give up after 10 error test failures

(b) Actions *after* passing the Local Error Test.

No further action if: $k' = k-1$, or $k = 5$, or k was raised in previous step. If last $k+1$ steps were at constant order $k < 5$ and step size h, consider order $k+1$.

Set order $k+1$ test quantities:

$$\begin{aligned} \text{est.}\|h_n^{k+2} y_n^{(k+2)}\| &= T_{k+1} = \|\Delta_n - \Delta_{n-1}\| \\ \text{ELTE}_{k+1} &= E_{k+1} = T_{k+1}/(k+2) \ . \end{aligned}$$

Case $k = 1$: Set $k \leftarrow 2$ if $T_2 < T_1/2$.

Case $k > 1$:

Set $k \leftarrow k-1$ if $T_{k-1} \leq \min\{T_k, T_{k+1}\}$;

Else set $k \leftarrow k+1$ if $T_{k+1} < T_k$;

Else leave k unchanged $(T_{k-1} > T_k \leq T_{k+1})$.

In any case, set tentative step ratio

$$r = h'/h = 1/(2E_k)^{\frac{1}{k+1}}$$

using new k and corresponding estimated LTE E_k.

To increase h, we must be able to double it:

If $r \geq 2$, take $h' = 2h$;

If $r \leq 1$, adjust to make $.5 \leq r \leq .9$, set $h' = rh$;

If $1 < r < 2$, set $h' = h$.

8. Inequality Constraints

IDA user can impose constraints on y componentwise, by way of an input vector with 5 choices:

$$y^i > 0\,, \quad y^i \geq 0\,, \quad y^i \leq 0\,, \quad y^i < 0\,, \quad y^i \text{ unconstrained}\,.$$

Following the Newton iteration, if otherwise successful, we test resulting $y = y_{n(m)}$ for satisfaction of constraints. Test returns mask vector M ($M^i = 0$ if OK, 1 if not OK).

If there are any failures, compute a constraint violation vector V, such that $\bar{y}^i = y^i - V^i$ would pass the constraint test, but just barely. ($V^i = 0$ where $M^i = 0$.)

E.g., if constraint is $y^i \geq 0$, but $y^i < 0$, then $V^i = y^i$.

If constraint is $y^i > 0$, but $y^i \leq 0$, then $V^i = y^i - 0.2w^i$, where w is the vector of error weights,

$$w^i = \text{rtol}\,|y^i| + \text{atol}^i\,.$$

V is set similarly for constraints $y^i \leq 0$, $y^i < 0$.

Since the Newton convergence test is

$$\|\text{error in } y_{n(m)}\| \leq 0.33\,,$$

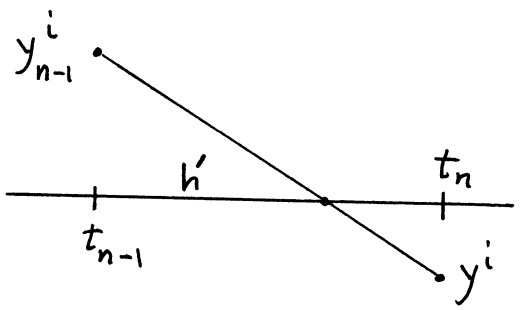
we accept $\bar{y} = y - V$ instead of y if $\|V\| \leq 0.33$.

Actions if $\|V\| > 0.33$:

We declare a convergence failure of the Newton iteration, and cut the step size. Rather than cut by an arbitrary factor like $1/4$, we estimate a new h', based on the failure:

If constraint is $y^i \geq 0$, but $y^i < 0$ (and $y^i_{n-1} \geq 0$), then a linear approximation of $y^i(t)$ crosses zero at $t_{n-1} + h'$, given by

$$h'/h = \frac{y^i_{n-1}}{y^i_{n-1} - y^i} \equiv r^i \ .$$



The same is true if the constraint is $y^i \leq 0$, but $y^i > 0$.

So a new step size valid for all components is given by

$$h'/h = r = \min\{r^i : M^i = 1\} \ .$$

To cover the strict inequality case, we apply a safety factor of 0.9 to r.

If $r = 0$, because some $y^i_{n-1} = 0$ but y^i has the wrong sign, we can only hope that some small value of h' will produce a valid y^i. Therefore, we restrict r to $r \geq 0.1$.

In any case, we set $h \leftarrow rh$, and try the step again. As before, give up after 10 such failures.

Inequality constraints in IC Calculation:

The linesearch process finds a corrected value of the vector x (= a mix of y and y' components) between current x and $x + \Delta x$.

Before setting λ by the unconstrained algorithm, we test the y components of $x + \Delta x$ for satisfaction of the constraints. If not satisfied, we compute λ_0 so that $x + \lambda_0 \Delta x$ will (just barely) satisfy the constraints.

Then restrict the subsequent λ choice to $0 < \lambda \leq \lambda_0$.