Runge-Kutta Methods

Runge-Kutta Methods

- Consider the typical IVP that you want to solve:

$$\left\{\begin{array}{rcl}\dot{\mathbf{y}} & = & \mathbf{f}(t,\mathbf{y}) \\ \mathbf{y}(0) & = & \mathbf{c}\end{array}\right. \qquad t \in [0,b]$$

- The Runge-Kutta integration process is the sum of two tasks:
  - Task 1: compute the s stage values (the time consuming part):

$$\mathbf{Y}_i = \mathbf{y}_{n-1} + h\sum_{j=1}^{s} a_{ij}\mathbf{f}(t_{n-1}+c_j h, \mathbf{Y}_j), \qquad 1 \leq i \leq s$$

  - Task 2: compute the solution at t_n (this is trivial…):

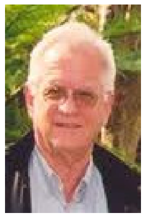
$$\mathbf{y}_n = \mathbf{y}_{n-1} + h\sum_{i=1}^{s} b_i\mathbf{f}(t_{n-1}+c_i h, \mathbf{Y}_i)$$

  - Note that these two tasks are carried out at each integration time step t_1, t_2, etc.

Runge-Kutta (RK) Methods

- Three sets of parameters together define a RK method: a_{ij}, b_i, and c_i.
- The coefficients defining a RK method are given to you and typically grouped together in what's called Butcher's Tableau

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array} = \begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b}^{\mathrm{T}} \end{array}$$


Professor John Butcher,
New Zealand, awesome guy

- **A**, **b**, and **c** are defined to represent the corresponding blocks of Butcher's Tableau (see above)
- All properties of a RK scheme (stability, accuracy order, convergence order, etc.) are completely defined by the entries in **A**, **b**, and **c**
  - Nomenclature: number of stages s is defined by the number of rows in **A**

Example: Classical Fourth Order RK Method

$$\mathbf{Y}_i = \mathbf{y}_{n-1} + h\sum_{j=1}^{s} a_{ij}\mathbf{f}(t_{n-1}+c_j h, \mathbf{Y}_j), \qquad 1 \le i \le s$$

$$\mathbf{y}_n = \mathbf{y}_{n-1} + h\sum_{i=1}^{s} b_i\mathbf{f}(t_{n-1}+c_i h, \mathbf{Y}_i)$$

$$Y_1 = y_{n-1}$$

$$Y_2 = y_{n-1} + \tfrac{h}{2} f(t_{n-1}, Y_1)$$

$$Y_3 = y_{n-1} + \tfrac{h}{2} f(t_{n-1} + \tfrac{h}{2}, Y_2)$$

$$Y_4 = y_{n-1} + h f(t_{n-1} + \tfrac{h}{2}, Y_3)$$

$$y_n = y_{n-1} + \tfrac{h}{6}\left(f(t_{n-1}, Y_1) + 2f(t_{n-1} + \tfrac{h}{2}, Y_2) + 2f(t_{n-1} + \tfrac{h}{2}, Y_3) + f(t_n, Y_4)\right)$$

- The Butcher Tableau representation looks like this:

0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
	1/6	1/3	1/3	1/6

Choosing A, b, and c for an Explicit RK

- Purpose of this and next slide: point out how challenging it is to generate a good RK method
- Recall that it boils down to choosing the coefficients in **A**, **b**, and **c**
- It has been proved that given a number of stages "s" that you accept to have in an explicit RK method, a limit on the order of the method "p" ensues:

s	1	2	3	4	5	6	7	8	9	10
p	1	2	3	4	4	5	6	6	7	7

Choosing A, b, and c for RK

- Example:
  - *Necessary* conditions for an explicit method to have order 5
  - Notation used: **C**=diag($c_1,\ldots,c_s$) and **1**=$(1,1,\ldots,1)^T$

$$\mathbf{b}^T\mathbf{C}^4\mathbf{1} = \frac{1}{5} \qquad \mathbf{b}^T\mathbf{A}\mathbf{C}^3\mathbf{1} = \frac{1}{20} \qquad \mathbf{b}^T\mathbf{C}\mathbf{A}^2\mathbf{C}\mathbf{1} = \frac{1}{30}$$

$$\mathbf{b}^T\mathbf{A}^4\mathbf{1} = \frac{1}{120} \qquad \mathbf{b}^T\mathbf{C}^2\mathbf{A}\mathbf{C}\mathbf{1} = \frac{1}{10} \qquad \mathbf{b}^T\mathbf{A}\mathbf{C}\mathbf{A}\mathbf{C}\mathbf{1} = \frac{1}{40}$$

$$\mathbf{b}^T\mathbf{A}^2\mathbf{C}^2\mathbf{1} = \frac{1}{60} \qquad \mathbf{b}^T\mathbf{C}\mathbf{A}\mathbf{C}^2\mathbf{1} = \frac{1}{15} \qquad \sum_{i,j,k} b_i\, a_{ij}\, c_j\, a_{ik}\, c_k = \frac{1}{20}$$

- The number of *necessary* and *sufficient* conditions to **guarantee** a certain order for an RK method is as follows:

Order p	1	2	3	4	5	6	7	8	9	10
no. of conditions	1	2	4	8	17	37	85	200	486	1205

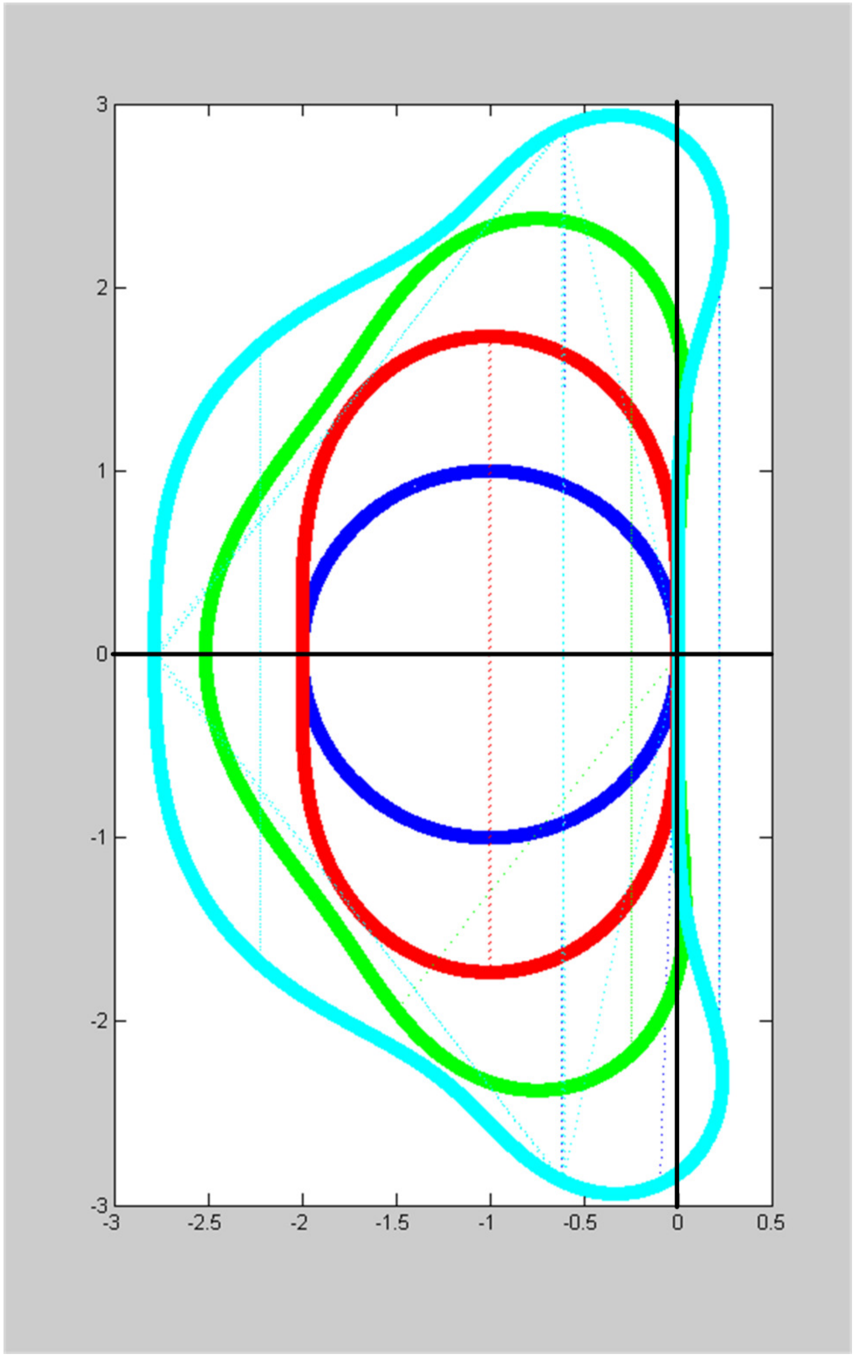
- Conclusion: Building a high-order RK is tricky…

Absolute Stability Regions

- Plots report absolute stability regions for explicit RK methods with s stages and of order p=s, for s=1,2,3,4
  - Blue: s=1
  - Red: s=2
  - Green: s=3
  - Cyan: s=4
- Methods are stable inside the curves
- Absolute stability region given by

$$|1 + h\lambda + \frac{(h\lambda)^2}{2!} + \cdots + \frac{(h\lambda)^p}{p!}| \leq 1$$

$$p = 1, \ldots, 4$$



Absolute Stability Regions [Cntd.]

- MATLAB script to generate the fourth order abs-stability region (cyan):

```
th=0:0.001:2*pi;
a=zeros(4,length(th));
for k=1:length(th)
    c=[1./24. 1./6. 0.5 1 1-exp(i*th(k))];
    a(:,k)=roots(c);
end

hold on
plot(a(1,:), 'co:')
plot(a(2,:), 'co:')
plot(a(3,:), 'co:')
plot(a(4,:), 'co:')
hold off
```

Exercise

- Generate the Convergence Plot of the fourth order RK provided a couple of slides ago for the following IVP:

$$\text{IVP:} \quad \begin{cases} \dot{x} = x - y \\ \dot{y} = 4x - 3y \\ x(0) = y(0) = 1 \end{cases} \qquad t \in [0, 4]$$

- Note that the exact solution of this IVP is:

$$x(t) = (t+1)e^{-t}$$

$$y(t) = (2t+1)e^{-t}$$

RK Method, A Different Possibility to Advance the Numerical Solution

- Recall that in stage "i" of the s stage approach, we generated a value Y_i . We call this approach "y-flavored":
  - First, for each of the s stages,

$$\mathbf{Y}_i = \mathbf{y}_{n-1} + h \sum_{j=1}^{s} a_{ij} \mathbf{f}(t_{n-1} + c_j h, \mathbf{Y}_j), \qquad 1 \leq i \leq s$$

  - Next, a combination of these stage values leads to the solution at t_n:

$$\mathbf{y}_n = \mathbf{y}_{n-1} + h \sum_{i=1}^{s} b_i \mathbf{f}(t_{n-1} + c_i h, \mathbf{Y}_i)$$

- A different approach can be followed, this is "f-flavored"
  - It approximates derivatives at each stage rather than values y
  - See next slide…

RK Method, A Different Possibility to Advance the Numerical Solution

- At each of the s stages of the RK method, you need to figure out F_i:

$$\mathbf{F}_i = f\left(t_{n-1} + c_i h,\ \mathbf{y}_{n-1} + h\sum_{j=1}^{s} a_{ij}\mathbf{F}_j\right), \qquad 1 \le i \le s$$

- Once the stage values are available, the solution is computed as

$$\mathbf{y}_n = \mathbf{y}_{n-1} + h\sum_{i=1}^{s} b_i \mathbf{F}_i$$

- Personally, I find the f-flavor better than the y-flavor implementation

RK Method, A Different Possibility to Advance the Numerical Solution

- Exercise: show that the f-flavor is easily obtained from the y-flavor by using an appropriate notation.

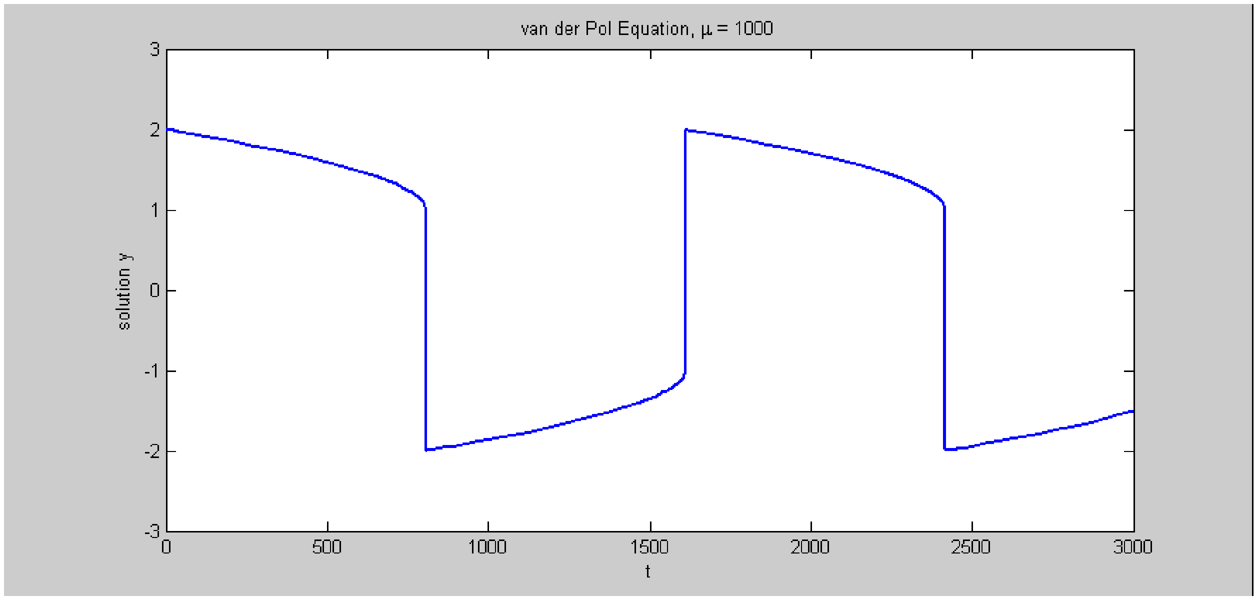
Exercises

- Note that Forward Euler, Backward Euler, and Trapezoidal Formula can all be considered as belonging to the RK family
  - Provide the Butcher Tableau representation for Forward Euler
  - Provide the Butcher Tableau representation for Backward Euler
  - Provide the Butcher Tableau representation for the Trapezoidal Formula

Integration Error Control

- The problem: imagine a dynamic system that varies rapidly every once in a while, but the remaining time is very tame
  - Example: solution of the van der Pole IVP

$$\text{IVP:} \quad \begin{cases} \frac{d^2y}{dt^2} + \mu(y^2-1)\frac{dy}{dt} + y = 0 \\ y(0) = 2 \quad \& \quad \dot{y}(0) = 0 \end{cases}$$



```
tspan = [0, 3000];
y0 = [2; 0];
Mu = 1000;
ode = @(t,y) vanderpoldemo(t,y,Mu);
[t,y] = ode15s(ode, tspan, y0);

plot(t,y(:,1))
title('van der Pol Equation, \mu = 1000')
axis([0 3000 -3 3])
xlabel('t')
ylabel('solution y')
```

Integration Error Control

- If you don’t adjust the integration step-size h you are forced to work during the entire simulation with a very conservative value of h
  - Basically, you have to work with that value of h that can negotiate the high transients
  - This would be for almost the entire simulation a waste of resources

- Basic Idea:
  - When you have high transients, reduce h to make sure you are ok
  - When the dynamics is tame, increase the value of h and sail quickly through these intervals

- On what should you base the selection of the step size h?
  - On the value of local error
  - It would be good to be able to use the actual error, but that’s impossible to do

Integration Error Control: The Details

- In the end, we need a mechanism that tries to guarantee that the local error at each time step stays below a user-prescribed threshold value

- Computing the threshold value
  - Draws on two values specified by the user: absolute tolerance ATOL and relative tolerance RTOL (think of these as allowances)
  - If dealing with an m-dimensional problem, threshold value ξ_i for component "i" of solution y is computed as

$$\xi_i = ATOL_i + \max(\mathbf{y}[i]_{n-1}, \mathbf{y}[i]_n) \cdot RTOL_i$$

- The key observation: the entire error control effort concentrates on keeping an *approximation* of the local error at t_n smaller than ξ

$$|\, \mathbf{l}[i]_n \,| \leq \xi_i$$

Integration Error Control: The Details

- What's left at this point is to somehow provide an approximation of the local error **l**[i]$_n$ at time step t$_n$

- To get **l**[i]$_n$, you produce a *second* approximation of the solution at t$_n$, and you pretend that that second solution is the actual solution(kind of funny). Then you can get an approximation of the local error:

$$|\,\mathbf{y}[i]_n - \hat{\mathbf{y}}[i]_n\,| \le \xi_i$$

- Here we had:
  - $\mathbf{y}[i]_n$ – the i^{th} component of the solution approximation $\mathbf{y}_n$ at t_n.
  - $\hat{\mathbf{y}}[i]_n$ – the i^{th} component of the solution approximation $\hat{\mathbf{y}}_n$ at t_n. This is the second approximation, of higher order, considered to be the 'reference' solution used in computing the local error.

Integration Error Control: The Details

- A measure of the acceptability "a" of the solution given the user prescribed tolerance is obtained as

$$a = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\frac{\mathbf{y}[i]_n - \hat{\mathbf{y}}[i]_n}{\xi_i}\right)^2}$$

- Note that asymptotically, since the method we use is assumed to be order p, we have for v that (K is an unknown constant):

$$a \approx K \cdot h^{p+1}$$

- Note that any reading $a \leq 1$ indicates an acceptable situation
- Otherwise, if $a > 1$, it's an indication that the quality of the solution does not meet the user prescribed tolerance
  - If this is the case, the step size should be decrased, y_n is rejected and it's to be computed again…

Integration Error Control: The Details

- Summary of possible scenarios
  - Step-size is too small, you are being way more accurate than the user needs

    $$a \ll 1$$

  - Step-size is exactly where you want it to be, acceptability is on the margin

    $$a \approx 1 \quad \text{but} \quad a \leq 1$$

  - Step-size is too large, you are to aggressive and this leads to local errors that are exceeding the user specified tolerance

    $$a > 1$$

Integration Error Control: The Details

- Finally, how do you choose the optimal step-size h_{opt}?
  - You want to be in the sweet spot, acceptability is 1.0
  - The step-size is chosen to meet this requirement:

$$\left.\begin{aligned} a &\approx K \cdot h^{p+1} \\ 1 &\approx K \cdot h_{opt}^{p+1} \end{aligned}\right\} \Rightarrow h_{opt} = h \cdot \left(\frac{1}{a}\right)^{\frac{1}{p+1}}$$

  - Because there was some hand waving involved and these arguments are in general true only asymptotically, one usually uses a safety factor s=0.9 to play it conservatively. Then the new step size is chosen as

$$h_{opt} = s \cdot h \cdot \left(\frac{1}{a}\right)^{\frac{1}{p+1}}$$

Integration Error Control: The "Embedded Method"

- How do you usually get the second approximate solution?
- The idea is to use the same stage values you produce to generate the first solution
- In other words, use the same **A** and **c**, but change only **b**
- When using Butcher's Tableau, this is captured by adding a new row for the new values of $\hat{\mathbf{b}}$:

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b} \end{array} \oplus \begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \hat{\mathbf{b}} \end{array} = \begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b} \\ & \hat{\mathbf{b}} \end{array}$$

Original Method:
Produces num solution

Embedded Method:
Produces second num solution
(used in local error control)

Typical notation used
for Butcher's Tableau

Example 1: RK Embedded Methods

- The Fehlberg 4(5) pair
  - Empty cells have a zero in them

0						
1/4	1/4					
3/8	3/32	9/32				
12/13	1932/2197	-7200/2197	7296/2197			
1	439/216	-8	3680/513	-845/4104		
1/2	-8/27	2	-3544/2565	1859/4104	-11/40	
	25/216	0	1408/2565	2197/4104	-1/5	0
	16/135	0	6656/12825	28561/56430	-9/50	2/55

Example 2:
RK Embedded Methods

- The Dormand-Prince 4(5) pair
  - Empty cells have a zero in them
  - This is what's used in MATLAB as the default for the ODE45 solver

0							
1/5	1/5						
3/10	3/40	9/40					
4/5	44/45	-56/15	32/9				
8/9	19372/6561	-25360/2187	64448/6561	-212/729			
1	9017/3168	-355/33	46732/5247	49/176	-5103/18656		
1	35/384	0	500/1113	125/192	-2187/6784	11/84	
	5179/57600	0	7571/16695	393/640	-92097/339200	187/2100	1/40
	35/384	0	500/1113	125/192	-2187/6784	11/84	0

Explicit vs. Implicit RK

- One can immediately figure out whether a RK method is explicit or implicit by simply inspecting Butcher's Tableau

- If the **A** matrix has nonzero entries on the diagonal or in the upper triangular side, the method is implicit

- Implicit RK methods belong to several subfamilies
  - Gauss methods
    - They are maximum order methods: for s stages, you get order 2s (as good as it gets)
  - Radau methods
    - Attain order 2s-1 for s stages
  - Lobatto methods
    - Attain order 2s-2 for stages

Examples, Implicit RK Methods

- Members of the Gauss subfamily

1/2	1/2
	1

Implicit Midpoint
s=1, p=2

$\frac{3-\sqrt{3}}{6}$	1/4	$\frac{3-2\sqrt{3}}{12}$
$\frac{3+\sqrt{3}}{6}$	$\frac{3+2\sqrt{3}}{12}$	1/4
	1/2	1/2

No name, s=2, p=4

- Members of the Radau subfamily

1	1
	1

Backward Euler
s=1, p=1

1/3	5/12	-1/12
1	3/4	1/4
	3/4	1/4

No name, s=2, p=3

- Members of the Lobatto subfamily

0	0	0
1	1/2	1/2
	1/2	1/2

Trapezoidal Method
s=2, p=2

0	0	0	0
1/2	5/24	1/3	0
1	1/6	2/3	1/6
	1/6	2/3	1/6

No name, s=3, p=4

Implicit RK Methods: Implementation Issues

- Implicit RK methods are notoriously hard to implement

- Suppose you have an IVP where the dimension of the unknown function is m:

$$\mathbf{y}(t) \in \mathbb{R}^m$$

- Then, the dimension of the nonlinear system that you have to solve at each time step is of an s-stage implicit RK method is s*m

- This is a serious drawback
  - A lot of research goes into parallelizing this process: rather than solving one nonlinear system of dimension s*m, the idea is to solve s systems of dimension m
  - This is still not that impressive, to be compared to the effort in multistep methods (to be covered shortly…)

Exercise

- Consider the van der Pol IVP, which is to be solved using the order 3 Radau formula
- Write down the nonlinear system of equations that one has to solve when advancing the simulation by one time step h
  - Use the F-flavor representation of the RK method

Diagonal Implicit RK Methods (DIRK Methods)

- One immediate way to decouple the large nonlinear system and have s systems of dimension m is to use diagonal implicit RK methods
  - Called DIRK methods
  - If *all* the diagonal entries in the A matrix are the same, then the method is called SDIRK (singly diagonal implicit RK) method
  - Note that for SDIRK, each of the s decoupled nonlinear systems have the same iteration matrix (Jacobian is the same)

- Example, SDIRK methods
  - Backward Euler
  - Also the following two look good…

$$\gamma = \frac{3+\sqrt{3}}{6} \qquad \begin{array}{c|cc} \gamma & \gamma & 0 \\ 1-\gamma & 1-2\gamma & \gamma \\ \hline & 1/2 & 1/2 \end{array}$$

s=2, p=3

$$\gamma = \frac{2-\sqrt{2}}{2} \qquad \begin{array}{c|cc} \gamma & \gamma & 0 \\ 1 & 1-\gamma & \gamma \\ \hline & 1-\gamma & \gamma \end{array}$$

s=2, p=2

D2

RK and Stiff Decay

- Stiff Decay is also called in the literature L-stability
- There is a theorem that provides sufficient conditions for stiff decay of a RK method
- Specifically, the following are sufficient conditions for stiff decay
  - A matrix is nonsingular, and
  - The last row of the **A** matrix is identical to **b**^{T}
- Example, SDIRK with stiff decay:

$$\gamma = \frac{2-\sqrt{2}}{2}$$

γ	γ	0
1	$1-\gamma$	γ
	$1-\gamma$	γ

s=2, p=2

$$\left.\begin{array}{rl} \text{Last row of } \mathbf{A}: & [1-\gamma \quad \gamma] \\ \text{Vector } \mathbf{b}^T: & [1-\gamma \quad \gamma] \end{array}\right\} \Rightarrow \quad \text{L-stability}$$

Slide 29

D2 Verify that this is indeed a sufficient condition
Dan Negrut, 11/7/2009

RK Methods – Final Thoughts

- Explicit RK relatively straight forward to implement
- Implicit RK are challenging to implement due to the large nonlinear system that ensues discretization
- This family of methods is well understood
  - Reliable
  - On the expensive side in terms of computational effort (for each time step, you have to do multiple function evaluations)

- Things of interest that we didn't cover
  - Estimation of global error
  - Stiffness detection
  - Sensitivity to data perturbations (sensitivity analysis)
  - Symplectic methods for Hamiltonian systems

Exercises

- Problem 4.8 – tricky at times
- Problem 4.12 – deals with step-size control for a sun-earth problem
- Example 4.6: use MATLAB to generate an approximate solution of the IVP therein. The solution is y(t)=sin(t). If the approximate MATLAB solution doesn't look good, try to tinker with MATLAB or implement your own numerical scheme to solve the problem

New Topic:
Linear Multistep Methods

Multistep vs. RK Methods

- Fewer function evaluations per time step

- Simpler, more streamlined method design
  - Recall the table with number of conditions that the RK method coefficients had to satisfy to be guaranteed a certain order for the RK method

- Error estimation and order control are much simpler
  - In fact, order control (the ability to change the order of the method on the fly) is something that is not typically done for RK
  - Order control is very common for Multistep Methods

- On the negative side
  - There is high overhead when changing the integration step-size
  - Loses some of the flexibility of one RK methods (there you had many parameters to adjust, not that much the case for Multistep methods)
  - More simpleton in nature than their sophisticated RK cousins

Review of Framework

- Interested in finding a function **y**(t) over an interval [0,b]
- This m-dimensional function y(t) must satisfy the following IVP:

$$\begin{cases} \dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y}) \\ \mathbf{y}(0) = \mathbf{c} \end{cases} \qquad t \in [0, b]$$

- We assume that **f** is bounded and smooth, so that **y** exists, is unique, and smooth

- Given to you:
  - The constants **c** and **b**
  - The function **f**(t,**y**).

Multistep Methods - Nomenclature

- Notation used:
  - $\mathbf{y}_l$ represents an approximation at time t_l of the actual solution $\mathbf{y}(t_l)$
  - f_l represents the value of the function f evaluated at t_l and y_l
- We work with *multistep* methods. We'll use k to represent the number of steps in a particular Multistep method
- The general form of a Multistep method (M-method) is as follows

$$\sum_{j=0}^{k} \alpha_j \mathbf{y}_{n-j} = h \sum_{j=0}^{k} \beta_j \mathbf{f}_{n-j}$$

- α_j and β_j are coefficients specific to each M method

Examples - Multistep Methods

- General Form:

$$\sum_{j=0}^{k} \alpha_j \mathbf{y}_{n-j} = h \sum_{j=0}^{k} \beta_j \mathbf{f}_{n-j}$$

- BDF method

$$y_n - \frac{4}{3} y_{n-1} + \frac{1}{3} y_{n-2} = \frac{2}{3} h f(t_n, y_n)$$

- Adams-Bashforth method

$$y_n - y_{n-1} = \frac{h}{12}(23 f_{n-1} - 16 f_{n-2} + 5 f_{n-3})$$

- Adams-Moulton method

$$y_n - y_{n-1} = \frac{h}{12}(5 f_n + 8 f_{n-1} - f_{n-2})$$

M Methods: Further Remarks

- To eliminate arbitrary scaling, it is assumed that

$$\alpha_0 = 1$$

- To truly talk about a k-step method, it is also assumed that

$$|\alpha_k| + |\beta_k| \neq 0$$

- Note that if β_j=0 the method is explicit. Otherwise, it is implicit

- Finally, note that the step size over the last k integration step is assumed constant
  - This is going to give some headaches later on when you actually want to change the step size on the fly to control error

Quick One Slide Review: Local Truncation Error, Forward Euler

- Consider how the solution is obtained:

$$\frac{y_n - y_{n-1}}{h} - f(t_{n-1}, y_{n-1}) = 0$$

- Note that in general, if you stick the actual solution in the equation above it is not going to be satisfied:

$$\frac{y(t_n) - y(t_{n-1})}{h} - f(t_{n-1}, y(t_{n-1})) \neq 0$$

- By <u>definition</u>, the quantity above is called the truncation error and is denoted by

$$\mathcal{N}(y, t, h) = \frac{y(t_n) - y(t-h)}{h} - f(t-h, y(t-h))$$

- Note that this depends on the function (y), the point where you care to evaluate the truncation error (t_n), and the step size used (h)

The Local Truncation Error: Multistep Methods

- Consider the linear operator (assume y is scalar function, for simplicity of notation)

$$\mathcal{L}(y,t,h) = \sum_{j=0}^{k} [\alpha_j y(t-jh) - \beta_j \dot{y}(t-jh)]$$

- Equivalently, since y is the exact solution of the IVP,

$$\mathcal{L}(y,t,h) = \sum_{j=0}^{k} [\alpha_j \, y(t-jh) - \beta_j \, f(t-jh, y(t-jh))]$$

- Then it follows that

$$\mathcal{N}(\mathbf{y},t,h) = \frac{\mathcal{L}(\mathbf{y},t,h)}{h}$$

- Or, in other words, the local truncation error is

$$d_n = h^{-1}\mathcal{L}(y,t_n,h)$$

M Methods: Order Conditions

- Recall that by definition a method is accurate of order p if

$$d_n = \mathcal{O}(h^p)$$

- To assess the order of d_n, carry out a Taylor expansion of $y(t-jh)$ and $\dot{y}(t-jh)$
  - This to be done for j=0,…,k, then collect terms to obtain the following representation of the linear operator

$$\mathcal{L}(y,t,h) = C_0 y(t) + C_1 h\dot{y}(t) + \ldots + C_q h^q y^{(q)}(t) + \ldots$$

- Then, we get the following
  - The M method is accurate of order p if and only if

$$C_0 = C_1 \ldots = C_p = 0, \quad C_{p+1} \neq 0$$

  - The local truncation error d_n is expressed as

$$d_n = C_{p+1} h^p y^{(p+1)}(t_n) + \mathcal{O}(h^{p+1})$$

M Methods: Order Conditions

- From the Taylor series expansions, one can obtain in a straightforward fashion that

$$C_0 = \sum_{j=0}^{k} \alpha_j$$

$$C_i = (-1)^j \left[\tfrac{1}{i!} \sum_{j=1}^{k} j^i \alpha_j + \tfrac{1}{(i-1)!} \sum_{j=0}^{k} j^{i-1} \beta_j \right], \quad i = 1, 2, \ldots$$

- Nomenclature:
  - When the order is p, then C_{p+1} is called the error constant of the method
  - Obviously, one would like a method that has C_{p+1} as small as possible

Exercises

- Proof that the expression of C_i on the previous slide is correct
- Pose the Forward Euler method as a M method and verify its order conditions (should be order 1)
- Pose the Backward Euler method as a M method and verify its order conditions (should be order 1)
- Pose the Trapezoidal method as a M method and verify its order conditions (should be order 2)

Quick Review: Order "p" Convergence

- Theorem:

$$\text{Consistency} \quad + \quad \text{0-stability} \quad \Rightarrow \quad \text{Convergence}$$

- Some more specifics:
  - If the method is accurate of order p and 0-stable, then it is convergent of order p:

$$e_n = \mathcal{O}(h^p), \qquad n = 1, 2, ..., N$$

M Methods: Convergence Results

- We saw what it takes for a M method to have a certain accuracy order
- What's left is to prove 0-stability
- The concept of characteristic polynomial comes in handy:

$$\rho(\xi) = \sum_{j=0}^{k} \alpha_j \xi^{k-j}$$

- Note that for the k stage M method, the characteristic polynomial only depends on α_j

M Methods: The Root Condition

- We provide without proof the following condition for a M-method to be 0-stable (the “root condition”)
  - Let ξ_i be the k roots of the characteristic polynomial. That is,

$$\rho(\xi_i) = \sum_{j=0}^{k} \alpha_j \xi_i^{k-j} = 0$$

  - Then, the M-method is 0-stable if and only if
    - $|\xi_i| \leq 1$, for $i = 1, \ldots, k$
    - In case $|\xi_i| = 1$, then ξ_i is a simple root (has multiplicity one)

M Methods: Convergence Criterion

- An M-method is convergent to order p if the following conditions hold:
  - The root condition holds
  - The method is accurate to order p
  - The initial values required by the k-step method are accurate to order p

- Exercise:
  - Identify the convergence order of the Forward Euler, Backward Euler, and Trapezoidal Methods

M Methods: Exercise, Root Condition

- Consider the following M-method:

$$y_n = -4y_{n-1} + 5y_{n-2} + h(4f_{n-1} + 2f_{n-2})$$

- What is the accuracy order of the method?

- Does the method satisfy the root condition?

- Use the M-method above to find the solution of the simple IVP

$$\text{IVP:} \quad \begin{cases} \dot{y} = 0 \\ y(0) = 0 \end{cases} \qquad t \in [0, 10]$$

  - For the M-method, take $y_0 = 0 \quad \& \quad y_1 = \epsilon.$

The Root Condition: Further Comments

- Exercise: Generate the convergence plot for Milne’s method…

$$y_n = y_{n-2} + \frac{1}{3}h(f_n + 4f_{n-1} + f_{n-2})$$

- … in conjunction with the following IVP:

$$\text{IVP:} \quad \begin{cases} \dot{y} = -10y \\ y(0) = 1 \end{cases} \qquad t \in [0, 10]$$

- Compute the starting points using the exact solution of the above IVP

Short Side Trip: Difference Equations

- Difference equations, the framework
  - Someone gives you k initial values $x_0, \ldots, x_{k-1}$
  - You find the next value x_k by solving a "difference equation":

$$a_0 x_n + a_1 x_{n-1} + \ldots + a_k x_{n-k} = 0$$

- It's obvious that the value of x_n is uniquely defined once you have the first k values
- How can we compute this unique value x_n yet not explicitly reference the first k values?
- Trick used: assume the following expression for x_n: $x_n = \xi^n$

- This choice of the expression of x_n leads to the following equation that must be satisfied by ξ (typically called Characteristic Equation)

Characteristic Equations: $a_0 \xi^k + a_1 \xi^{k-1} + \ldots + a_k = 0$

Short Side Trip: Difference Equations [Cntd.]

- Characteristic Equation (CE):
  - Has degree k
  - Has k roots (might be distinct or multiple roots amongst them): $\xi_1, \xi_2, \ldots, \xi_k$
  - Exercise: show that the value of x_n can be expressed as (assume no multiple roots)

$$x_n = c_1\xi_1^n + c_2\xi_2^n + \ldots + c_k\xi_k^n = \sum_{i=1}^{k} c_i\xi_i^n$$

- Expression of x_n gets slightly more complicated for multiple roots:
  - Double root (say $\xi_1=\xi_2$):

$$x_n = (c_{11} + c_2 n)\xi_1^n + \sum_{i=3}^{k} c_i\xi_i^n$$

  - Triple root (say $\xi_1 = \xi_2 = \xi_3$):

$$x_n = [c_{11} + c_2 n + c_3 n(n-1)(n-2)]\xi_1^n + \sum_{i=4}^{k} c_i\xi_i^n$$

NOTE: This Difference Equations theory relevant when looking into absolute stability

Absolute Stability [quick review]

- The process used to find out the region of absolute stability
  - We started with the test problem

$$\begin{cases} \dot{y} &= \lambda y \\ y(0) &= 1 \end{cases}$$

  - We required that for the test problem, the numerical approximation should behave like the exact solution. That is, we required that

$$|y_n| \leq |y_{n-1}|$$

  - Used the discretization scheme to express how y_n is related to y_{n-1} and impose the condition above
  - This leads to a condition that the step size should satisfy in relation to the parameter λ
  - Example: for Forward Euler, we obtained that for absolute stability that

$$|1 + h\lambda| < 1$$

Region of Absolute Stability

- Apply the methodology on previous slide for the test problem when used in conjunction with a multistep scheme

$$\sum_{j=0}^{k} \alpha_j \mathbf{y}_{n-j} = h \sum_{j=0}^{k} \beta_j \mathbf{f}_{n-j}$$

- This leads to

$$\sum_{j=0}^{k} \alpha_j y_{n-j} = h\lambda \sum_{j=0}^{k} \beta_j y_{n-j}$$

- Recall that we had the expression for x_n Re

$$y_n = c_1\xi_1^n + c_2\xi_2^n + \ldots + c_k\xi_k^n = \sum_{i=1}^{k} c_i\xi_i^n$$

- For us to hope that $y_n \rightarrow 0$, we need $|\xi_i| \leq 1$ for $\forall$ i $\geq$ k

Region of Absolute Stability [Cntd.]

- Drop the subscript i for convenience. The conclusion is that any root of the Characteristic Equation; i.e. any ξ that satisfies…

$$\sum_{j=0}^{k} \alpha_j \xi^{n-j} = h\lambda \sum_{j=0}^{k} \beta_j \xi^{n-j}$$

- … must also satisfy $|\xi| \leq 1$

- Note that if the above condition holds, then we will get to the desired condition that y_n is monotonically decreasing in absolute value:

$$|\xi| = \frac{|\xi^n|}{|\xi^{n-1}|} = \frac{|y_n|}{|y_{n-1}|} \leq 1 \qquad \Rightarrow \qquad |y_n| \leq |y_{n-1}|$$

Region of Absolute Stability [Cntd.]

- So in the end, it boils down to this simple sufficient condition: if hλ is such that the roots of the CE all have the norm less than or equal to 1, then hλ belongs to the stability region
  - Recall that the CE assumes the form

$$\sum_{j=0}^{k} \alpha_j \xi^{n-j} = h\lambda \sum_{j=0}^{k} \beta_j \xi^{n-j}$$

- How would you find the boundaries of the stability region?
  - This is precisely that situation where $|\xi|=1$, or in other words, where $\xi=e^{i\theta}$
  - So the boundary is given by those values of hλ for which $\xi=e^{i\theta}$
  - Yet note that from the CE, one has that for $\theta\in[0,2\pi)$,

$$h\lambda = \frac{\sum_{j=0}^{k} \alpha_j \xi^{n-j}}{\sum_{j=0}^{k} \beta_j \xi^{n-j}} = \frac{\sum_{j=0}^{k} \alpha_j e^{i\theta(n-j)}}{\sum_{j=0}^{k} \beta_j e^{i\theta(n-j)}}$$

Exercise

- Plot the region of absolute stability for Milne’s method

Absolute Stability: Closing Comments

- It is relatively straight forward to show that no explicit M method can be A-stable

- Lindquist's Barrier (1962, not simple to prove)
  - You cannot construct an A-stable M method that has order higher than 2
  - Note that there is no such barrier for RK methods

- The second order A-stable implicit M method with smallest error constant ($C_3=1/12$) is the trapezoidal integration method
  - The problem with the trapezoidal formula is that it does not have stiff decay (it is A-stable but not L-stable)

How Did People Get M-Methods?

- One early approach (about 1880): integrate the ordinary differential equation, and approximate the function f using a polynomial

$$\dot{y} = f(t, y) \qquad \Rightarrow \qquad y(t_n) = y(t_{n-1}) + \int_{t_{n-1}}^{t_n} f(t, y(t))dt$$

- Based on previous values $f(t_{n-1},y_{n-1}),\ldots, f(t_{n-k},y_{n-k})$, one can fit a k-1 degree polynomial in the variable t to approximate the unknown function f(t,y)

- Once the polynomial is available, simply plug it back in the integral above and evaluate it to get y_n (an approximation of $y(t_n)$)

- NOTE: this approach leads to a family of explicit integration formulas called Adams-Bashforth Multistep methods (AB-M methods)

$$y_n = y_{n-1} + \sum_{j=1}^{k} \beta_j f_{n-j}$$

Exercise

- Derive the AB-M method for k=1, k=2, and k=3

- Plot the absolute stability region for the AB-M methods above

AB-M Method, Closing

- Table below provides convergence order p, the number of steps k of the M method, the coefficients β_{n-j}, and the value of the leading coefficient of the error term C_{p+1}

p	k	$j\rightarrow$	1	2	3	4	5	6	C_{p+1}
1	1	β_{n-j}	1						1/2
2	2	$2\beta_{n-j}$	3	-1					5/12
3	3	$12\beta_{n-j}$	23	-16	5				3/8
4	4	$24\beta_{n-j}$	55	-59	37	-9			251/720
5	5	$720\beta_{n-j}$	1901	-2774	2616	-1274	251		95/288
6	6	$1440\beta_{n-j}$	4277	-7923	9982	-7298	2877	-475	19087/60480

- Example: based on the above table, the third order AB-M formula is

$$y_n = y_{n-1} + \frac{h}{12}(23f_{n-1} - 16f_{n-2} + 5f_{n-3})$$

Starting a M Method

- Implementation question: How do you actually start a M method?
  - In general, you need information for the first k steps to start a M method

- If you work with a scheme of order p, you don't want to have in your first k values $y_0, \ldots, y_{k-1}$ error that is larger than $O(h^p)$

- Most common approach is to use for the first k-1 steps a RK method of order p.

- A second approach starts using a method of order 1 with smaller step, than increases to order 2 when you have enough history, then increase to order 3, etc.

- NOTE: for the previous exercise, you have the exact solution so you can use it to generate the first k steps

Exercise

- Generate the Convergence Plot of the AB-M method for k=3 and k=4 for the following IVP:

$$\text{IVP:} \quad \begin{cases} \dot{x} = x - y \\ \dot{y} = 4x - 3y \\ x(0) = y(0) = 1 \end{cases} \qquad t \in [0, 4]$$

- Indicate whether your results come in line with the expected convergence behavior
- Note that the exact solution of this IVP is:

$$x(t) = (t+1)e^{-t}$$

$$y(t) = (2t+1)e^{-t}$$

Exercise

- Prove that the AB-M method with k=3 is convergent with order 3

Exercise

- Plot the absolute stability regions for the AB-M formulas up to order 6
- Comment on the size of the absolute convergence regions

The AM-M Method

- The AB-M method is known for small absolute stability methods

- Idea that partially addressed the issue:
  - Rather than only using the previous values $f(t_{n-1},y_{n-1}),\dots, f(t_{n-k},y_{n-k})$, one should include the extra point $f(t_n,y_n)$ to fit a k degree polynomial in the variable t to approximate the unknown function f(t,y)

- The side-effect of this approach:
  - The resulting scheme is implicit: you use $f(t_n,y_n)$ in the process of finding y_n
  - The resulting scheme will assume the following form:

$$y_n = y_{n-1} + \sum_{j=0}^{k} \beta_j f_{n-j}$$

- This family of formulas is called Adams-Moulton Multistep (AM-M) methods

Exercise

- Derive the AM-M method for k=2 and then k=3

- Plot the absolute stability region for the AM-M methods above n2

AM-M Method, Closing

- Table below provides convergence order p, the number of steps k of the M method, the coefficients β_{n-j}, and the value of the leading coefficient of the error term C_{p+1}

p	k	j→	0	1	2	3	4	5	C_{p+1}
1	1	β_{n-j}	1						-1/2
2	1	$2\beta_{n-j}$	1	1					-1/12
3	2	$12\beta_{n-j}$	5	8	-1				-1/24
4	3	$24\beta_{n-j}$	9	19	-5	1			-19/720
5	4	$720\beta_{n-j}$	251	646	-264	106	-19		-3/160
6	5	$1440\beta_{n-j}$	475	1427	-798	482	-173	27	-863/60480

- Example: based on the above table, the third order AM-M formula (k=2) is

$$y_n = y_{n-1} + \frac{h}{12}(5f_n + 8f_{n-1} - f_{n-2})$$

Slide 66

n2 understand well why there are two orders for k=1 (backward Euler and trapezoidal)
negrut, 11/22/2009

Exercise

- Prove that the AM-M method with k=3 is convergent with order 4

Exercise

- Generate the Convergence Plot of the AB-M method for k=2 and k=3 for the following IVP:

$$\text{IVP:} \quad \begin{cases} \dot{x} = x - y \\ \dot{y} = 4x - 3y \\ x(0) = y(0) = 1 \end{cases} \qquad t \in [0, 4]$$

- Indicate whether your results come in line with the expected convergence behavior
- Note that the exact solution of this IVP is:

$$x(t) = (t+1)e^{-t}$$

$$y(t) = (2t+1)e^{-t}$$

- NOTE: use the analytical solution to generate the first k steps of the integration formula

Exercise

- Plot the absolute stability regions for the AM-M formulas up to order 6
- Comment on the size of the absolute convergence regions

Implicit AM-M: Solving the Nonlinear System

- Since the AM-M method is implicit it will require at each time step the solution of a system of equations
  - If **f** is nonlinear in y this system of equations will be nonlinear
    - This is almost always the case
- Approaches used to solve this nonlinear system:
  - Functional iteration
  - Predictor Corrector schemes
  - Modified Newton iteration
- Focus on first two, defer discussion of last for a couple of slides

M Methods: Functional Iteration

- Idea similar to the one introduced for the RK method
- Iterative process carried out as follows:

$$y_n^{(\nu+1)} = h\beta_0 f(t_n, y_n^{(\nu)}) + K, \qquad \nu = 0, 1, \ldots$$

  - Notation: K represents a constant pre-computed based on past information
    - It does not change during the iterative process

$$K = -\sum_{j=1}^{k} \alpha_j y_{n-j} + h \sum_{j=1}^{k} \beta_j f_{n-j}$$

  - As a starting point, for ν=0, typically one takes this value to by y_{n-1}
    - This will be revisited shortly, when discussing predictor-corrector schemes
  - Stopping criteria identical to and discussed in relation to modified Newton iteration

n3

M Methods: Functional Iteration

- This represents a fixed point iteration
- Fixed point iteration converges to the fixed point provided it is a contraction, which is the case if the following condition holds

$$||h\beta_0 \frac{\partial f}{\partial y}|| \leq r < 1$$

- NOTE: this condition basically limits the Functional Iteration approach to nonstiff problems

Slide 72

n3 See if the notation spells out what || . || stands for. Should be max of a function
negrut, 11/22/2009

M Methods: The Predictor-Corrector Approach

- The predictor corrector formula is very similar to the Functional Iteration approach
- There are two differences:
  - The starting point is chosen in a more intelligent way
  - The number of iterations is predefined
    - This is unlike the Functional Iteration approach, where convergence is monitored and it is not clear how many iterations ν will be necessary for convergence

The Predictor-Corrector Approach: Choosing the Starting Point

- The key question is how should one choose $y_n^{(0)}$
- An explicit method is used to this end
- This step is called prediction ("**P**"), and the explicit M method used to obtain $y_n^{(0)}$ is called "predictor"
- Most of the time, the predictor is an AB-M method:

$$\mathrm{P}: \qquad y_n^{(0)} + \hat{\alpha}_1 y_{n-1} + \ldots + \hat{\alpha}_k y_{n-k} = h(\hat{\beta}_1 f_{n-1} + \ldots + \hat{\beta}_k f_{n-k})$$

- The predicted value for y is immediately used to evaluate ("**E**") the value of the function f:

$$\mathrm{E}: \qquad f_n^0 = f(t_n y_n^{(0)})$$

The Predictor-Corrector Approach: Carrying out Corrections

- The second distinctive attribute of a Predictor-Corrector integration formula is that a predefined number ν of corrections of are carried out
  - In other words, ν_{end} is predetermined, and the final value for y_n is

$$y_n = y_n^{(\nu_{end})}$$

- The corrector ("**C**") formula is usually chosen to be the AM-M method
- Starting with ν=0, the correction step assumes then the expression

$$\mathrm{C}: \qquad y_n^{(\nu+1)} + \alpha_1 y_{n-1} + \ldots + \alpha_k y_{n-k} = h(\beta_0 f_n^{(\nu)} + \beta_1 f_{n-1} + \ldots + \beta_k f_{n-k})$$

- Typically, the C step is followed by an E step to obtain a new expression for f that goes hand in hand with the newly corrected; i.e., improved, value of y:

$$\mathrm{E}: \qquad f_n^{(\nu+1)} = f(t_n y_n^{(\nu)})$$

The Predictor-Corrector Approach: Carrying out Corrections

- The predictor-corrector integration method process just described is called PECE
  - It predicts (P), evaluates (E), corrects (C), and finally evaluates again (E)
  - Note that strictly speaking, the last (E) could be regarded as superfluous since it's not used for computation of y_n anymore
  - Last E is essential though since it's used in the computation of y_{n+1} and it improves the stability properties of the integration method

- Note that approach described (PECE), corresponds to choosing $\nu_{\text{end}}=1$

- For larger values of ν_{end} the "EC" part in PECE is executed ν_{end} times
  - The nomenclature used for these methods is $\text{P(EC)}^{\nu}\text{E}$
  - Example: $\text{P(EC)}^3\text{E}$ refers to the following predictor-corrector integration formula:

$$P \rightarrow \underbrace{E \rightarrow C}_{1^{st}} \rightarrow \underbrace{E \rightarrow C}_{2^{nd}} \rightarrow \underbrace{E \rightarrow C}_{3^{rd}} \rightarrow E$$

Example: PECE Method

- The following example combines a two step AB-M method, with the second-order one step AM-M method (the trapezoidal formula)

- Given y_{n-1}, f_{n-1}, f_{n-2}:

$$\text{P:} \qquad y_n^{(0)} = y_{n-1} + \frac{h}{2}(3f_{n-1} - f_{n-2})$$

$$\text{E:} \qquad f_n^{(0)} = f(t_n, y_n^{(0)})$$

$$\text{C:} \qquad y_n = y_{n-1} + \frac{h}{2}(f_n^{(0)} + f_{n-1})$$

$$\text{E:} \qquad f_n = f(t_n, y_n)$$

- It can be shown that the local truncation error for this method is

$$d_n = -\frac{h^2}{12}\dddot{y}(t_n) + O(h^3)$$