module 08
solving odes

ahmad f. taha

ce 2989 — numerical methods in civil & env. engineering

email: ahmad.taha@vanderbilt.edu

webpage: http://lab.vanderbilt.edu/taha

April 15, 2025

module motivation and outline

- we've reached the penultimate chapter of this course: solving ordinary differential equations (odes)
- whatever we've learned before will be useful in solving odes or initial value problem (ivp) of the form

$$\text{ivp}: \quad \dot{y} = \frac{dy(t)}{dt} = y'(t) = f(t, y), \quad t \in [a, b], \quad y(a) = y_a \text{ initial conditions}$$

- **note 1:** here we focus on the case when $y(t) \in \mathbb{R}$ and not $y(t) \in \mathbb{R}^n$
- **note 2:** we can write any higher order ode/ivp with higher order derivatives as the ivp above. how?
- hence, it's important to study methods to solve the above ivp
- this ivp models 1000s of dynamic systems present in cee applications
- energy, traffic, hydraulics, hydrology, water quality, air quality, power flow, epidemics, political opinions, misinformation spread, etc...
- a lot of odes can be solved analytically for a unique solution, but this isn't how these odes are typically solved
- computers and ode solvers such as `ode45`, `ode15`, `ode233` use numerical methods that we will learn about in this module

ivp/ode theory 101

- before we actually get to the numerical methods, it's important to understand whether an ode/ivp has a solution
- because if you simply go and test a numerical method, that solution might be trash like elon musk's ideas
- this topic, existence and uniqueness of solutions to odes, is indeed rich
- to do so, we need to talk about uncle rudolf lipschitz

uncle lipschitz

a function $f(t, y)$ is called lipschitz continuous in y in a set $y \in \mathcal{Y}$ if there's a constant K such that for all y_1 and $y_2 \in \mathcal{Y}$ the following property holds:

$$|f(t, y_1) - f(t, y_2)| \leq K|y_1 - y_2|$$

where $K > 0$ is called the lipschitz constant of $f(\cdot)$

existence + uniqueness of ode solutions

if $f(t, y)$ is continuous in its domain and it is lipschitz with constant K, then then the IVP $y' = f(t, y)$ has a unique solution

how to verify lipschitzness?

- so for any $y' = f(t, y)$ and a given initial condition $y(0) = y(a)$, to verify uniqueness, you gotta find that constant K
- **example 1:** does the ivp $y'(t) = 3ty(t)$ in the region $y \in [-3, 4]$ and $t \in [1, 2]$ have a unique solution?
- applying lipschitz definition, we get

$$|f(t, y_1) - f(t, y_2)| = 3t|y_1 - y_2| \leq 6|y_1 - y_2|$$

  hence the ivp is lipschitz and a unique solution exists
- **example 2:** $y'(t) = y^2(t)$ for $y \in [-2, 2]$
- **example 3:** system of ODEs $\dot{y}(t) = \begin{bmatrix} \dot{y}_1(t) \\ \dot{y}_2(t) \end{bmatrix} = \begin{bmatrix} ay_1(t) + by_2(t) \\ 1 - \cos(cy_1(t)) \end{bmatrix}$
- the topic of computing lipschitz constants is so deep and still interesting[1]

[1] Sebastian, Taha, and Hoang. "Nonlinear dynamic systems parameterization using interval-based global optimization: Computing lipschitz constants and beyond." *IEEE Transactions on Automatic Control* 67.8 (2021): 3836-3850.

alternate definitions

- the previous lipschitz definition is hard to work with. why?
- alternatively, another way to compute K is via this result

alternate result

if a function $f(t, y)$ satisfies $\dfrac{\partial f(t, y)}{\partial y} \leq K$ for a $K > 0$ then $f(t, y)$ satisfies the above lipschitz continuity definition

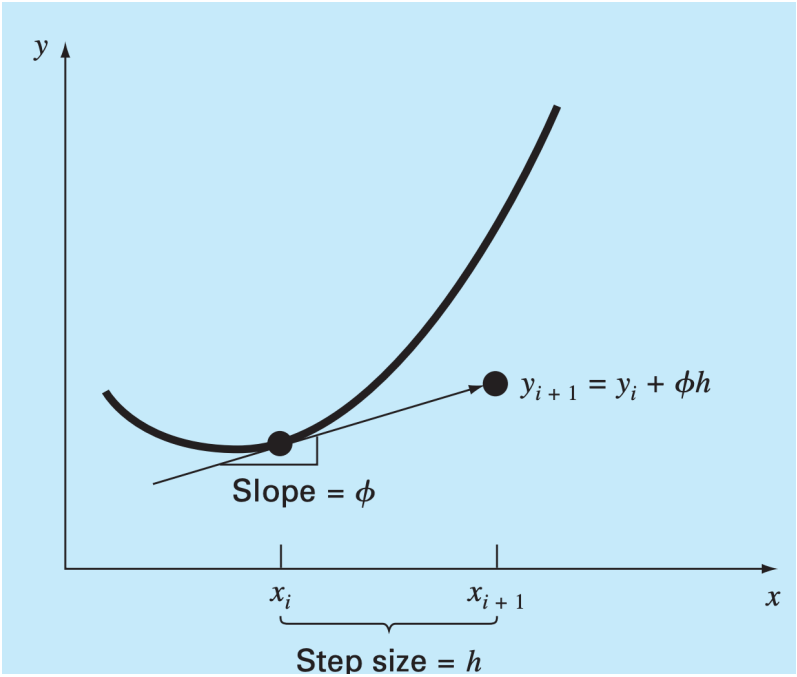
- **example 4:** use the above result to show that this ivp $y'(t) = 1 + t \sin(ty)$ for $t \in [0, 2]$
- **example 5:** $y'(t) = e^{t - y(t)}$ for $t \in [0, 1]$ and $y(0) = 1$
- next we will learn the first class of methods to numerically solve odes: the one-step methods

one-step methods: an intro

- to solve the ode/ivp $\dot{y}(t) = f(t, y)$, we will first resort to one-step methods that can all be formulated as

$$y_{t_{i+1}} = y_{t_i} + h\phi_{t_i} \quad \text{or} \quad y_{t+1} = y_t + h\phi_t$$

- where h is the *sampling time* or *step size* and ϕ is the direction
- different methods propose different ϕ's with different properties and error bounds and we will study these in this module



euler's method, again

$$\text{ivp}: \quad \dot{y} = \frac{dy(t)}{dt} = y'(t) = f(t, y), \quad t \in [a, b], \quad y(a) = y_a \text{ initial conditions}$$

- we have to talk about forward euler and forward approximations
- to solve the ivp via euler's method, we divide the interval into n segments with step size $h = (b - a)/n$
- then we can write $t_i = a + ih$ for $i = 0, 1, \ldots, n$ and obtain the taylor series approximation for $y'(t) = f(t, y(t))$ as

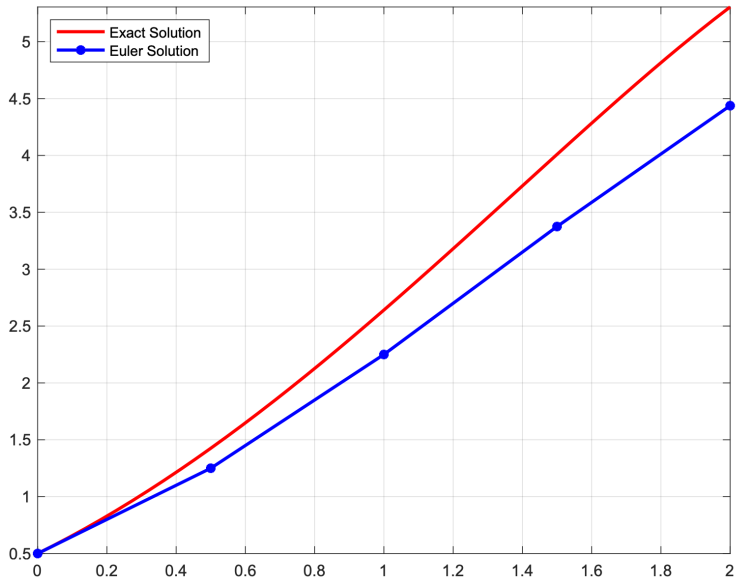
$$y(t_{i+1}) = y(t_i) + (t_{i+1} - t_i)y'(t_i) + \frac{(t_{i+1} - t_i)^2}{2}y''(\xi_i) = y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(\xi_i), \quad t_i \leq x_i \leq t_i$$

- which gives us the simple **forward euler method** to solve any ode/ivp:

$$t_{i+1} = t_i + h, \quad y_{i+1} = y_i + hf(t_i, y_i), \quad i = 0, 1, \ldots, n - 1$$

example on euler

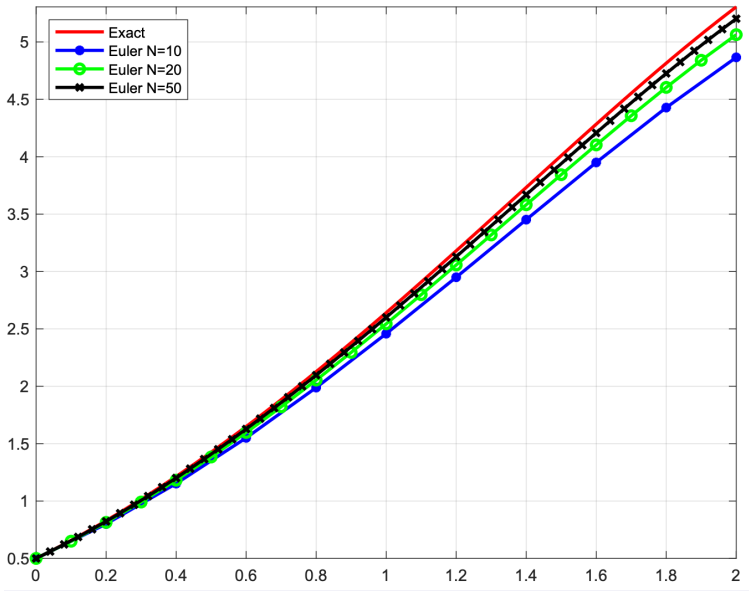
- solve this ivp $y'(t) = y(t) - t^2 + 1$ for $h = 0.5, y(0) = 0.5$ and for $t \in [0, 2]$
- first determine the segments/intervals $t = \{0, 0.5, 1, 1.5, 2\}$
- then compute $y_1 = y_0 + hf(t_0, y_0) = 0.5 + 0.5(0.5 - 0^2 + 1) = 1.25$
- similary, $y_2 = y_1 + hf(t_1, y_1) = 1.25 + 0.5(1.25 - 0.5^2 + 1) = 2.25$
- actual solution for the ivp is $y(t) = (t + 1)^2 - 0.5e^t$
- results can be plotted as follows



- clearly this method is trash for this $h = 0.5$...why?

pick a smaller h

- if we picked a smaller h in the previous problem, error gets substantially smaller
- for $h = 1/10, 1/20, 1/50$ we get



error bounds for euler

for the ivp discussed earlier, and for a lipschitz $f(t, y)$ defined over $t \in [a, b]$ with constant K, assume that $|y''(t)| \leq C$ then the error from euler's method can be bounded as follows

$$|y(t_i) - y_i| \leq \frac{Ch}{2K}(-1 + (1 + Kh)^i) \leq \frac{Ch}{2K}(-1 + e^{-K(b-a)})$$

more on errors in forward euler

$$|e(t_i)| = |y(t_i) - y_i| \leq \frac{Ch}{2K}(e^{K(t_i-a)} - 1))$$

- notice that error in this method is order of h, not good at all
- how is this result derived? why does it make sense?
- you can also see that the error grows substantially as t_i increases with the constants being the same
- this error by the way, is different from the truncation error in the taylor series approximation
- we call $e(t_i)$ as the **global truncation error** (gte) because it accumulates the errors as t increases
- this suggests that there is *local truncation error* (lte) which is computed from the first order taylor series approximation

$$y(t_{i+1}) = y(t_i) + hy'(t_i) + \underbrace{\frac{h^2}{2}y''(\xi_i)}_{\text{lte of order } h^2}, \quad t_i \leq x_i \leq t_{i+1}$$

- clearly, the gte is greater than the lte which actually makes sense

clearly, we can improve the error

- how could we reduce the order of the error in both lte and gte?
- higher order taylor series approximation can help with that:

$$y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(t_i) + \frac{h^3}{3!}y'''(t_i) + \frac{h^n}{n!}y^{(n)}(t_i) + \frac{h^{n+1}}{(n+1)!}y^{(n+1)}(x_i), \quad t_i \leq x_i \leq t_{i+1}$$

- notice that $y'(t_i) = f(t, y_i), y''(t_i) = f'(t, y_i), \ldots$ which can all (i.e., the derivatives) be computed analytically
- this method works then by computing $t_{i+1} = t_i + h$ then finding $y_{i+1} = y_i + hf(t_i, y_i) + h^2 f'(t_i, y_i) + h^3/3! f''(t_i, y_i) + \ldots$
- if using taylor's approximation with order n, we get lte order of h^{n+1}
- but this method requires evaluations of higher order derivatives
- in general, this method can be written as $y_{i+1} = y_i + h\phi_o(t_i, y_i)$ where

$$\phi_o(t_i, y_i) = f(t_i, y_i) + \frac{h}{2}f'(t_i, y_i) + \ldots + \frac{h^{o-1}}{o!}f^{(o-1)}(t_i, y_i)$$

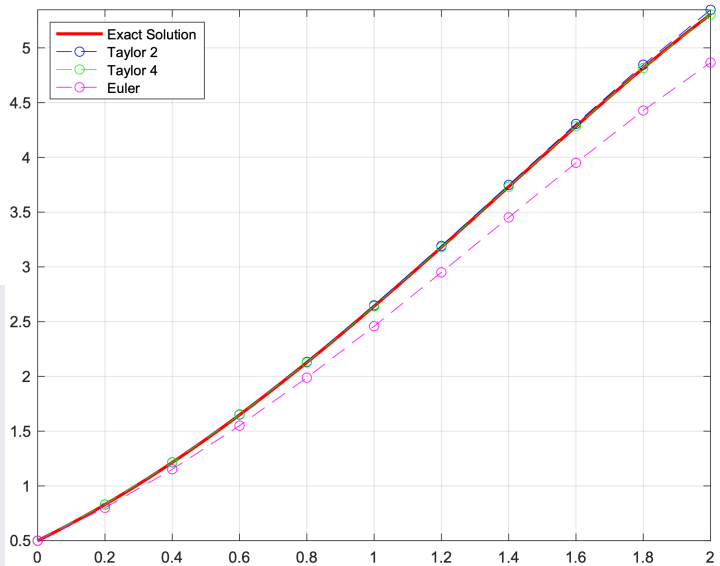
example on higher order taylor for solving odes

- solve this ivp $y'(t) = y(t) - t^2 + 1, y(0) = 0.5$ and for $t \in [0, 2]$ using 2nd order taylor and using $n = 10$ segments
- the iteration can be written as follows where $h = 2/10 = 0.2$

$$y_{i+1} = y_i + hf(t_i, y_i) + \frac{h^2}{2!} f'(t_i, y_i) = y_i + h(y_i - t_i^2 + 1) + \frac{h^2}{2}(y_i - t_i^2 + 1 - 2t_i)$$

- similarly, we can apply or compute a 4th order taylor approximation
- results can be tabulated as follows:

	Exact	Euler		Taylor(2)		Taylor(4)	
t_i	y(t_i)	y_i	|e_i|	y_i	|e_i|	y_i	|e_i|
0.0	0.50000	0.50000	0.00000	0.50000	0.00000	0.50000	0.00000
0.2	0.82930	0.80000	0.02930	0.83000	0.00070	0.82930	0.00000
0.4	1.21409	1.15200	0.06209	1.21580	0.00171	1.21409	0.00000
0.6	1.64894	1.55040	0.09854	1.65208	0.00314	1.64895	0.00001
0.8	2.12723	1.98848	0.13875	2.13233	0.00510	2.12724	0.00001
1.0	2.64086	2.45818	0.18268	2.64865	0.00779	2.64087	0.00002
1.2	3.17994	2.94981	0.23013	3.19135	0.01141	3.17996	0.00002
1.4	3.73240	3.45177	0.28063	3.74864	0.01624	3.73243	0.00003
1.6	4.28348	3.95013	0.33336	4.30615	0.02266	4.28353	0.00004
1.8	4.81518	4.42815	0.38702	4.84630	0.03112	4.81524	0.00006
2.0	5.30547	4.86578	0.43969	5.34768	0.04221	5.30556	0.00008



computing intermediate values

- from the previous example, 4th order taylor series was nearly perfect
- what if we wanna evaluate the ode/ivp solution at a specific point $t = 1.52$ for example?
- well, remember that taylor series-based ivp solution yielded a literal function evaluation at the specific discrete data points y_i
- in the previous example, we have access to $y(t_i = 1.4) = 3.732$ and $y(t_j = 1.6) = 4.283$ so we can use $n = 1$ (two data points) lagrange interpolation $p_1(t)$ to construct a linear line between these two points:

$$p_1(t = 1.52) = y(1.4)\frac{1.52 - 1.6}{1.4 - 1.6} + y(1.6)\frac{1.527 - 1.4}{1.6 - 1.4} = 4.0823$$

- the actual solution evaluated at $t = 1.52$ is $y(1.52) = 4.0835$
- the error here is not so bad but it is > 30 times worse than the error at 1.4 or 1.6
- how can you improve this result? by now we should all know
- any other (and better) interpolation technique: hermites, splines, etc..
- cubic hermite poly can be computed as such via the divided diff tableau:

$$h_3(t) = 3.732 + 2.772(t - 1.4) - 0.0846(t - 1.4)^2 - 0.376(t - 1.4)^2(t - 1.6)$$

yielding $h_3(t = 1.52) = 0.00005$, similar to the errors for 1.4 and 1.6

rk methods

$$\text{ivp}: \quad \dot{y} = \frac{dy(t)}{dt} = y'(t) = f(t,y), \quad t \in [a,b], \quad y(a) = y_a \text{ initial conditions}$$

- runge-kutta (rk) methods achieve the accuracy of a taylor series approach without needing higher order derivatives
- let's see how this actually works for the second order rk method, that uses second order taylor series approximation
- taking the derivative of the ivp above yields:

$$y''(t) = f_t(t,y) + f_y(t,y)y'(t) = f_t(t,y) + f_y(t,y)f(t,y)$$

  where f_t (f_y) is the time (or y) derivative of $f(t,y)$
- recall that the second order taylor series can be written as follows

$$y(t+h) = y(t) + hy'(t) + \frac{h^2}{2}y''(t) + O(h^3) = y(t) + hf(t,y) + \frac{h^2}{2}\left(f_t(t,y) + f_y(t,y)f(t,y)\right) + O(h^3)$$

- we will see how this can be used without knowing the derivative of $f(t,y)$

rk method derivation

$$y(t+h) = y(t) + hf(t,y) + \frac{h^2}{2}\left(f_t(t,y) + f_y(t,y)f(t,y)\right) + O(h^3) \qquad (*)$$

- i dont know if you remember but first order taylor series in two variables can be written as follows

$$f(t+ah, y+bh) = f(t,y) + ahf_t(t,y) + bhf_y(t,y) + O(h^2)$$

and for $a = 1, b = f$ and multiplying both sides by $h/2$, we get:

$$\frac{h}{2}f(t+h, y+hf) = \frac{h}{2}f(t,y) + \frac{h^2}{2}\left(f_t(t,y) + f(t,y)f_y(t,y)\right) + \frac{h}{2}O(h^2)$$

- plugging this into $(*)$, we get

$$y(t+h) = y(t) + \frac{h}{2}f(t,y) + \frac{h}{2}f(t+h, y+hf(t,y)) + O(h^3)$$

yielding this lovely rk formula/approximation

$$\boxed{y(t+h) \approx y(t) + \frac{h}{2}f(t,y) + \frac{h}{2}f(t+h, y+hf(t,y))}$$

with error order of h^3 and DOES NOT require any derivative evaluation

rk summarized

- we can rewrite rk method as follows:

$$\textbf{modified euler}:\ y_{i+1} = y_i + \frac{1}{2}\left(\underbrace{hf(t_i, y_i)}_{k_1} + \underbrace{hf(t_i + h, y_i + hf(t_i, y_i))}_{k_2 = hf(t_i + h, y_i + k_1)}\right),\ i = 0, 1, \ldots, n-1$$

- the previous procedure basically boils down to matching these two equations:

$$y(t+h) = y(t) + hf(t,y) + \frac{h^2}{2}\left(f_t(t,y) + f_y(t,y)f(t,y)\right) + O(h^3)$$

$$y(t+h) = y(t) + a_1 f(t+\alpha, y+\beta) = y(t) + a_1 f(t,y) + a_1\alpha f_t(t,y) + a_1\beta f_y(t,y) + \text{hot}$$

- this yields $a_1 = h$, $\alpha = h/2$, and $\beta = h/2 f(t,y)$
- this matching produces another second order method, different from the euler (which still matches these two equations)
- this other second order method can be written as:

$$\textbf{midpoint rk method}:\ y_{i+1} = y_i + hf\left(t_i + \frac{h}{2}, y_i + \frac{h}{2}f(t_i, y_i)\right),\ i = 0, 1, \ldots, n-1$$

- can be written as $y_{i+1} = y_i + k_2$ where $k_2 = hf\left(t_i + \frac{h}{2}, y_i + \frac{1}{2}k_1\right)$ with $k_1 = hf(t_i, y_i)$
- midpoint and modified euler produce lte of h^2 and gte of h^3

examples

- given this ivp: $y'(t) = f(t, y) = y - t^2 + 1,\ t \in [0, 2],\ y(0) = 0.5,$ $h = 0.2,\ n = 10$
- let's see how modified euler and the midpoint methods work here
- **for modified euler**, we have $k_1 = hf(t_0, y_0), k_2 = hf(t_0 + h, y_0 + k_1)$, so

$$y_1 = y_0 + 1/2(k_1 + k_2) = 0.5 + 0.5(0.3 + 0.352) = 0.826$$

- then for $i = 2,\ y_2 = y_1 + 1/2(k_1 + k_2) = \ldots = 1.206$
- simiarly, **for the midpoint rk method** we obtain $k_1 = hf(t_0, y_0) = 0.3$ and $k_2 = hf\left(t_0 + \frac{h}{2}, y_0 + \frac{1}{2}k_1\right) = 0.328$ hence $y_1 = y_0 + k_2 = 0.828$
- then for $i = 2,\ y_2 = y_1 + k_2 = \ldots = 1.2113$...results tabulated:

t_i	y(t_i)	1.Eul	1.Error	2.ModEul	2. Error	3.Midpt	3.Error
0.0	0.50000	0.50000	0.00000	0.50000	0.00000	0.50000	0.00000
0.2	0.82930	0.80000	0.02930	0.82600	0.00330	0.82800	0.00130
0.4	1.21409	1.15200	0.06209	1.20692	0.00717	1.21136	0.00273
0.6	1.64894	1.55040	0.09854	1.63724	0.01170	1.64466	0.00428
0.8	2.12723	1.98848	0.13875	2.11024	0.01699	2.12128	0.00595
1.0	2.64086	2.45818	0.18268	2.61769	0.02317	2.63317	0.00769
1.2	3.17994	2.94981	0.23013	3.14958	0.03036	3.17046	0.00948
1.4	3.73240	3.45177	0.28063	3.69369	0.03871	3.72117	0.01123
1.6	4.28348	3.95013	0.33336	4.23510	0.04839	4.27062	0.01286
1.8	4.81518	4.42815	0.38702	4.75562	0.05956	4.80096	0.01422
2.0	5.30547	4.86578	0.43969	5.23305	0.07242	5.29037	0.01510

- conclusions?

rk-4 + rk-5 (butcher's) methods

- in practice, the default ivp/ode solver is the rk-4 (runge-kutta method of order 4 or *four stages*)
- which can be written as follows:

$$\boxed{\textbf{rk-4 method}:\ y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)}$$

$$k_1 = hf(t_i, y_i),\ k_2 = hf\left(t_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1\right),\ k_3 = hf\left(t_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2\right)$$

and $k_4 = hf\left(t_i + h, y_i + k_3\right)$

- this method yields lte of h^5 and gte of h^4 and requires 4 function evaluations at each step i
- applying this for the previous example yields

$$k_1 = 0.3,\ k_2 = 0.328,\ k_3 = 0.3308,\ k_4 = 0.358,\ y_1 = y_0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) = 0.829$$

- butcher's or rk-5 method is given as follows:

$$\boxed{\textbf{rk-5 method}:\ y_{i+1} = y_i + \frac{h}{90}(7k_1 + 32k_3 + 12k_4 + 32 + k_5 + 7k_6)}$$

- frankly there isn't a point of using anything more than rk-4 method

discussions

- obviously there's no free lunch: as you go higher in the order of the method, you will require more function evaluations
- this results in higher computational cost because advanced/complex functions take longer to evaluate (in higher dimensions too)
- for basic euler, we only needed one evaluation, for modified euler/midpoint methods, we required two and for rk-4 we required four evaluations
- what happens if you can only evaluate the function only k times?
- you'd have to then change the sampling time/step-size so that you only are using k evaluations
- for the previous example, we'd need h to be 0.05 for euler, $h = 0.1$ for for modified euler, and $h = 0.2$ for rk-4 if only $k = 40$ evaluations
- remarkably, even when rk-4 uses far fewer evaluations, it is still superior:

t_i	Exact	Euler h = 0.05	ModEul h = 0.1	RK4 h = 0.2
0.0	0.5000000	0.5000000	0.5000000	0.5000000
0.2	0.8292986	0.8214716	0.8284350	0.8292933
0.4	1.2140877	1.1973995	1.2122113	1.2140762
0.6	1.6489406	1.6222790	1.6458789	1.6489220
0.8	2.1272295	2.0894190	2.1227827	2.1272027
1.0	2.6408591	2.5906863	2.6347973	2.6408227
1.2	3.1799415	3.1161950	3.1720009	3.1798942
1.4	3.7324000	3.6539290	3.7222788	3.7323401
1.6	4.2834838	4.1892822	4.2708390	4.2834095
1.8	4.8151763	4.7045011	4.7996197	4.8150857
2.0	5.3054720	5.1780062	5.2865672	5.3053630

why not use past data?

- the methods covered previously revolve around single-step methods
- that is, given time t_i and evaluations at time t_i $(f(t_i, y_i), f(t_i + \alpha, y_i + \beta))$, we computed y_{i+1}
- mth order multi-step methods utilize historic data: $y_{i-m}, y_{i-m+1}, \ldots, y_i$ to compute y_{i+1}
- because we already know past information, we can utilize it
- caveat here is that you need to know the initial m conditions for the ivp/ode whereas for the classic ivp/ode, you only needed a single initial condition $y(t_0) = y(0)$
- in general, these m-step multi-step methods can be written as

$$\boxed{\textbf{m-step method: } y_{i+1} = \boldsymbol{a}^\top \boldsymbol{y}_{i|m} + h\boldsymbol{b}^\top \boldsymbol{f}_{i|m+1}}$$

where

- $\boldsymbol{a} = \begin{bmatrix} a_{m-1} & a_{m-2} & \ldots & a_0 \end{bmatrix}^\top$, $\boldsymbol{y}_{i|m} = \begin{bmatrix} y_i & y_{i-1} & \ldots & y_{i-m+1} \end{bmatrix}^\top$
- $\boldsymbol{b} = \begin{bmatrix} b_m & b_{m-1} & \ldots & b_0 \end{bmatrix}$, $\boldsymbol{f}_{i|m} = \begin{bmatrix} f_{i+1} & f_i & \ldots & f_{i-m+1} \end{bmatrix}^\top$
- $i = m-1, m, m+1, \ldots, N-1$ and $f_i = f(t_i, y_i)$
- vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ are constant vectors defining the title of the m-step method as we will discuss soon
- if $b_m = 0$, method is called *explicit*; otherwise *implicit*
- to perform the m-step method, you need m initial conditions $y_0, \ldots, y_{m-1}$

examples on the m-step method

- consider $m = 4$, with m initial conditions $y_{0,1,2,3}$ and hence $i = 3, 4, \ldots, n-1$
- various studies have derived explicit forms for the m-step methods
- for $m = 4$, we have two versions: explicit and implicit
  - for the explicit 4-step method, we have the adams-bashforth method:

$$y_{i+1} = y_i + \frac{h}{24}\left(55f(t_i, y_i) - 59f(t_{i-1}, y_{i-1}) + 37f(t_{i-2}, y_{i-2}) - 9f(t_{i-3}, y_{i-3})\right)$$

  - for the implicit 4-step method, we have the adams-moulton method:

$$y_{i+1} = y_i + \frac{h}{24}\left(9f(t_{i+1}, y_{i+1}) + 19f(t_i, y_i) - 5f(t_{i-1}, y_{i-1}) + f(t_{i-2}, y_{i-2})\right)$$

  - what are vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ for these two methods?
- the word *implicit* here means that that you have to solve for the unknown rather than just obtaining it via the forward equations
- we will next see how some of these multi-step methods are actually derived

deriving multi-step methods

$$\text{ivp}: \quad \dot{y} = \frac{dy(t)}{dt} = y'(t) = f(t, y), \quad t \in [a, b], \quad y(a) = y_a \text{ initial conditions}$$

- we can integrate the above ivp between t_i and t_{i+1} to obtain

$$y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} f(t, y(t))dt \approx y(t_i) + \int_{t_i}^{t_{i+1}} p_n(t)dt$$

  where $p_n(t)$ is an nth degree interpolation of the function $f(t, y(t))$ which can be obtained via any of the methods we learned previously
- note that we couldn't have integrated $f(t, y(t))$ because we do not know $y(t)$ so that integral is not possible to be computed
- whereas $p_n(t)$ is just an nth degree polynomial that resembles $f(t, y(t))$
- starting with the linear lagrange poly $p_1(t)$ between t_i and t_{i+1}, and defining $f_j = f(t_j, y_j)$, we can now write

adam-bashforth 2-step explicit method $y_{t_{i+1}} := y_{i+1} = y_i + h\left(\frac{3}{2}f_i - \frac{1}{2}f_{i-1}\right)$

what if we use quadratic lagrange polys?

- so let's try to derive a three-step, multi-step method $n = 2$
- for quad lagr. polys, we have 3 points: $\{t_{i-1,i,i+1}, y_{i-1,i,i+1}\}$
- basically you have to integrate

$$\int_{t_i}^{t_{i+1}} (L_{2,0}(t)f_{i-1} + L_{2,1}(t)f_i + L_{2,2}(t)f_{i+1})\,dt = h\left(\frac{1}{12}f_{i-1} - \frac{2}{3}f_i + \frac{5}{12}f_{i+1}\right)$$

- can we verify the above computation?
- this results in the **adam-moulton two-step implicit method**

adam-moulton 2-step implicit method $y_{i+1} = y_i + h\left(\frac{1}{12}f_{i-1} - \frac{2}{3}f_i + \frac{5}{12}f_{i+1}\right)$

- comparative results for solving this ivp : $y'(t) = f(t,y) = y - t^2 + 1$, $t \in [0,2]$, $y(0) = 0.5$, $h = 0.2$, $n = 10$

Matlab Outputs

t_i	y(t_i)	AD2	Error	AD3	Error	AD4	Error
0.00	0.50000	0.50000	0.00000	0.50000	0.00000	0.50000	0.00000
0.20	0.82930	0.82930	0.00000	0.82930	0.00000	0.82930	0.00000
0.40	1.21409	1.21609	0.00200	1.21409	0.00000	1.21409	0.00000
0.60	1.64894	1.65398	0.00504	1.64934	0.00040	1.64894	0.00000
0.80	2.12723	2.13657	0.00934	2.12827	0.00104	2.12731	0.00008
1.00	2.64086	2.65614	0.01529	2.64280	0.00194	2.64108	0.00022
1.20	3.17994	3.20333	0.02339	3.18311	0.00316	3.18035	0.00041
1.40	3.73240	3.76672	0.03432	3.73724	0.00484	3.73306	0.00066
1.60	4.28348	4.33240	0.04891	4.29059	0.00710	4.28449	0.00101
1.80	4.81518	4.88344	0.06827	4.82531	0.01013	4.81666	0.00148
2.00	5.30547	5.39924	0.09377	5.31962	0.01415	5.30758	0.00211

>>

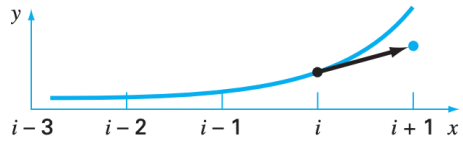
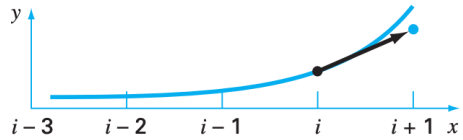
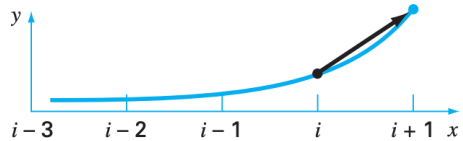
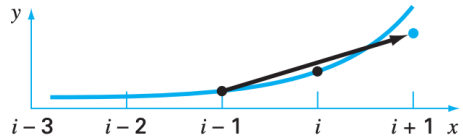
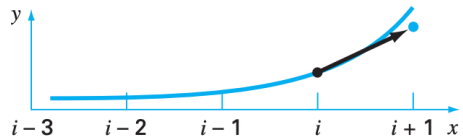
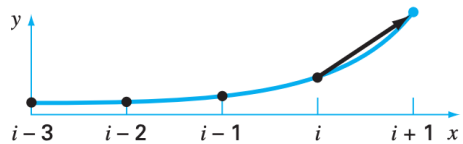
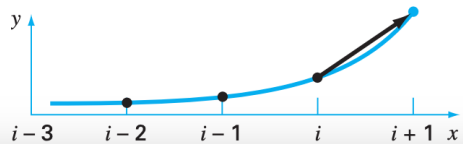
notes and discussions

- how do you find initial conditions if only $y(a) = y(t_0) = y(0)$ is given?
- well you can always run the classic rk-4 method and compute $y_{1,2,3,\ldots}$ from y_0 (so apply a single-step method like rk-4 or even modified euler)
- now that you have access to the initial conditions, you can simulate the explicit adam-bashforth method of any step order m
- what about implicit methods? well for some ivps/odes, applying them is so difficult and actually cumbersome, especially if $f(t, y)$ is nonlinear
- **example:** $f(t, y) = y' = e^t$ then via adam-moulton (am) implicit 2-step method we get

$$y_{i+1} = y_i + h(\gamma e^{y_{i+1}} + \eta e^{y_i} + \rho e^{y_i - 1})$$

  for some constants γ, η, ρ
- so while we have $y_{0,1,2}$, we still need to solve for y_3 through solving the above nonlinear equation (remember module 3?)
- so this becomes cumbersome and computationally intractable
- implicit methods can perform well though so we should not discount them: they are more numerically stable + larger h can be used
- the final class of methods is based on predictor-corrector methods:
  - use rk-4 or modified euler to compute m initial conditions
  - compute $\tilde{y}_{i+1}$ via the explicit ab method
  - use this value $\tilde{y}_{i+1}$ in the rhs of am method and the lhs will be y_{i+1}

summary of the methods

Method	Formulation	Graphic Interpretation	Errors
Euler (First-order RK)	$y_{i+1} = y_i + hk_1$ $k_1 = f(x_i, y_i)$		Local error $\simeq O(h^2)$ Global error $\simeq O(h)$
Ralston's second-order RK	$y_{i+1} = y_i + h(\frac{1}{3}k_1 + \frac{2}{3}k_2)$ $k_1 = f(x_i, y_i)$ $k_2 = f(x_i + \frac{3}{4}h, y_i + \frac{3}{4}hk_1)$		Local error $\simeq O(h^3)$ Global error $\simeq O(h^2)$
Classic fourth-order RK	$y_{i+1} = y_i + h(\frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4)$ $k_1 = f(x_i, y_i)$ $k_2 = f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_1)$ $k_3 = f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_2)$ $k_4 = f(xi + h, y_i + hk_3)$		Local error $\simeq O(h^5)$ Global error $\simeq O(h^4)$
Non-self-starting Heun	Predictor: (midpoint method) $y^0_{i+1} = y^m_{i-1} + 2hf(x_i, y^m_i)$		Predictor modifier: $E_p \simeq \frac{4}{5}(y^m_{i,u} + y^0_{i,u})$
	Corrector: (trapezoidal rule) $y^j_{i+1} = y^m_i + h\dfrac{f(x_i, y^m_i) + f(x_{i+1}, y^{j-1}_{i+1})}{2}$		Corrector modifier: $E_c \simeq -\dfrac{y^m_{i+1,u} - y^0_{i+1,u}}{5}$
Fourth-order Adams	Predictor: (fourth Adams-Bashforth) $y^0_{i+1} = y^m_i + h(\frac{55}{24}f^m_i - \frac{59}{24}f^m_{i-1} + \frac{37}{24}f^m_{i-2} - \frac{9}{24}f^m_{i-3})$		Predictor modifier: $E_p \simeq \frac{251}{270}(y^m_{i,u} - y^0_{i,u})$
	Corrector: (fourth Adams-Moulton) $y^j_{i+1} = y^m_i + h(\frac{9}{24}f^{j-1}_{i+1} + \frac{19}{24}f^m_i - \frac{5}{24}f^m_{i-1} + \frac{1}{24}f^m_{i-2})$		Corrector modifier: $E_c \simeq -\frac{19}{270}(y^m_{i+1,u} - y^0_{i+1,u})$

ode models in infrastructure

- as you will see in the homework, and as we discussed in the classroom numerous, dynamic models in infrastructure are in reality complex
- and a function of many variables, not just one $y(t)$
- the majority of advanced models in infrastructure can be written as

$$\textbf{ivp/ode model: } \dot{\boldsymbol{y}}(t) = \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u}, t), \quad \boldsymbol{y}(t_0) = \boldsymbol{y}_0, \quad \boldsymbol{u}(t) \text{ is a given input}$$

  where $\boldsymbol{y}(t) \in \mathbb{R}^{n_y}$, $\boldsymbol{u}(t) \in \mathbb{R}^{n_u}$, $\boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u}) : \mathbb{R}^{n_y} \times \mathbb{R}^{n_u} \times \mathbb{R}_+ \to \mathbb{R}^{n_y}$
- **examples** in traffic networks, power grids, water quality, stormwater systems, flood control dynamics, global positioning system, etc...
- we will now focus on solving system of odes, interlinked via the vector-valued mapping $\boldsymbol{f}(\cdot)$
- we will specifically focus on numerical methods to solve this $\dot{\boldsymbol{y}}(t) = \boldsymbol{f}(\boldsymbol{y}, t)$ with $\boldsymbol{y}(t_0) = \boldsymbol{y}_0$ as the initial conditions
- this will allow us to actually predict the behavior of the system for any time-instant t in the future and hence predict deviations from nominal system state

analytical solution for linear odes/ivps

- consider now a special case of the nonlinear ode:

$$\dot{\boldsymbol{y}}(t) = \boldsymbol{A}\boldsymbol{y}(t) + \boldsymbol{B}\boldsymbol{u}(t)$$

- we can solve this numerically or analytically
- analytically, and using the Leibniz integral rule

Theorem—Let $f(x,t)$ be a function such that both $f(x,t)$ and its partial derivative $f_x(x,t)$ are continuous in t and x in some region of the xt-plane, including $a(x) \leq t \leq b(x)$, $x_0 \leq x \leq x_1$. Also suppose that the functions $a(x)$ and $b(x)$ are both continuous and both have continuous derivatives for $x_0 \leq x \leq x_1$. Then, for $x_0 \leq x \leq x_1$,

$$\frac{d}{dx}\left(\int_{a(x)}^{b(x)} f(x,t)\,dt\right) = f\big(x,b(x)\big)\cdot\frac{d}{dx}b(x) - f\big(x,a(x)\big)\cdot\frac{d}{dx}a(x) + \int_{a(x)}^{b(x)}\frac{\partial}{\partial x}f(x,t)\,dt.$$

the solution can be written as follows:

$$\textbf{linear-ivp-solution: } \boldsymbol{y}(t) = e^{A(t-t_0)}\boldsymbol{y}(t_0) + \int_{t_0}^{t} e^{\boldsymbol{A}(t-\tau)}\boldsymbol{B}\boldsymbol{u}(\tau)d\tau$$

- this analytical solution is unfortunately cumbersome because it involves computing a matrix exponential but also integrating a complicated term
- **example:** solve the following coupled ivps:

$$\dot{y}_1(t) = -y_1(t) + 2y_2(t) + u_1(t), \quad \dot{y}_2(t) = y_2(t) + 0.1u_1 + u_2(t)$$

analytical solution for nonlinear odes/ivp

- to solve the nonlinear ivp $\dot{\boldsymbol{y}}(t) = \boldsymbol{f}(\boldsymbol{y}, t)$, we can integrate the above vectorized

$$\boldsymbol{y}(t) = \boldsymbol{y}(t_0) + \int_{t_0}^{t} f(\boldsymbol{y}(\tau), \tau) d\tau = \begin{bmatrix} y_1(t_0) \\ y_2(t_0) \\ \vdots \\ y_n(t_0) \end{bmatrix} + \begin{bmatrix} \int_{t_0}^{t} f_1(\boldsymbol{y}(\tau), \tau) d\tau \\ \int_{t_0}^{t} f_2(\boldsymbol{y}(\tau), \tau) d\tau \\ \vdots \\ \int_{t_0}^{t} f_m(\boldsymbol{y}(\tau), \tau) d\tau \end{bmatrix}$$

- you can obtain discretization of this model through numerical methods to individually integrate via methods of integration that we learned about before
- alternatively, we can apply the same methods we learned about before, while noting that the derivations are now for vectors and not scalars
- for example, forward euler can now be written as

$$t_{i+1} = t_i + h, \quad \boldsymbol{y}_{i+1} = \boldsymbol{y}_i + h\boldsymbol{f}(\boldsymbol{y}_i, t_i), \quad i = 0, 1, \ldots, n-1$$

- apologies for the confusion in the notation

runge-kutta method for a system of odes

- as mentioned before, the default ivp/ode solver is the rk-4 (runge-kutta method of order 4 or *four stages*) even for a system of odes
- which can be written as follows:

$$\boxed{\textbf{rk-4 method: } \boldsymbol{y}_{i+1} = \boldsymbol{y}_i + \frac{1}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4)}$$

$$\boldsymbol{k}_1 = h\boldsymbol{f}(t_i, \boldsymbol{y}_i), \boldsymbol{k}_2 = h\boldsymbol{f}\left(t_i + \frac{1}{2}h, \boldsymbol{y}_i + \frac{1}{2}\boldsymbol{k}_1\right), k_3 = h\boldsymbol{f}\left(t_i + \frac{1}{2}h, \boldsymbol{y}_i + \frac{1}{2}\boldsymbol{k}_2\right)$$

and $\boldsymbol{k}_4 = h\boldsymbol{f}\left(t_i + h, \boldsymbol{y}_i + \boldsymbol{k}_3\right)$

- **example:** solve the following system of equations

$$\dot{\boldsymbol{y}}(t) = \boldsymbol{f}(\boldsymbol{y}, t) = \begin{bmatrix} y_1(t) + 4x_2(t) - e^t \\ x_1(t) + y_2(t) + 2e^t \end{bmatrix}, \quad t \in [0, 1], \quad \boldsymbol{y}(0) = \begin{bmatrix} 4 \\ \frac{5}{4} \end{bmatrix}, \quad h = 0.1$$

- **solution:**

more complicated nonlinear odes

- how do you solve higher order, nonlinear coupled odes?
- let's see via these examples
- **example 1:**

$$\sin(t)y'''(t)+\cos(ty(t))+\sin(t^2+y''(t))+(y')^3 = \log(t), \quad t \in [2,3], \quad y(2) = 7, y'(2) = 3,$$

- **example 2:**

$$x''(t)+te^{y(t)}+y'(t) = x'(t)-x(t), \quad y(t)y''(t)-\cos(x(t)y'(t))+\sin(tx'(t)y(t))-x(t) = 0$$

- with initial conditions and time-span given as follows:

$$t \in [1,2], \quad x(1) = 1, x'(1) = 2, y(1) = 3, y'(1) = 4, \quad h = 0.1$$

- **solutions:**