Next topic:

Numerical Methods for Ordinary Differential Equations (ODE's)

SEC 5.7 OF EG

References:

- E+G Chapter 5, especially: 5.1 – 5.3.3
  5.4 up to p. 152
  5.5
  5.7
- AMATH 301 Notes, N. Kutz. Section 5
- E+G Chapter 4 (nullclines)
- E+G Lab Manual sec. 12–13.

Computational methods for $\dot{x} = f(x)$. Sec. 5.7 of E+G.

• EULER METHOD: SIMPLEST

$$\frac{dx}{dt} = f(x)$$

$$= \lim_{h \to 0} \frac{x(t+h) - x(t)}{h} = f(x(t))$$

or, $x(t+h) = x(t) + h \cdot f(x(t))$

Iterate: where $x(0) = x_0$

$x(h) = x_1$

$x(2h) = x_2 \ldots$

$$\boxed{x_{n+1} = x_n + h \cdot f(x_n)}$$ Euler Method

• Ex// $f(x) = x$

euler_illustrate.m

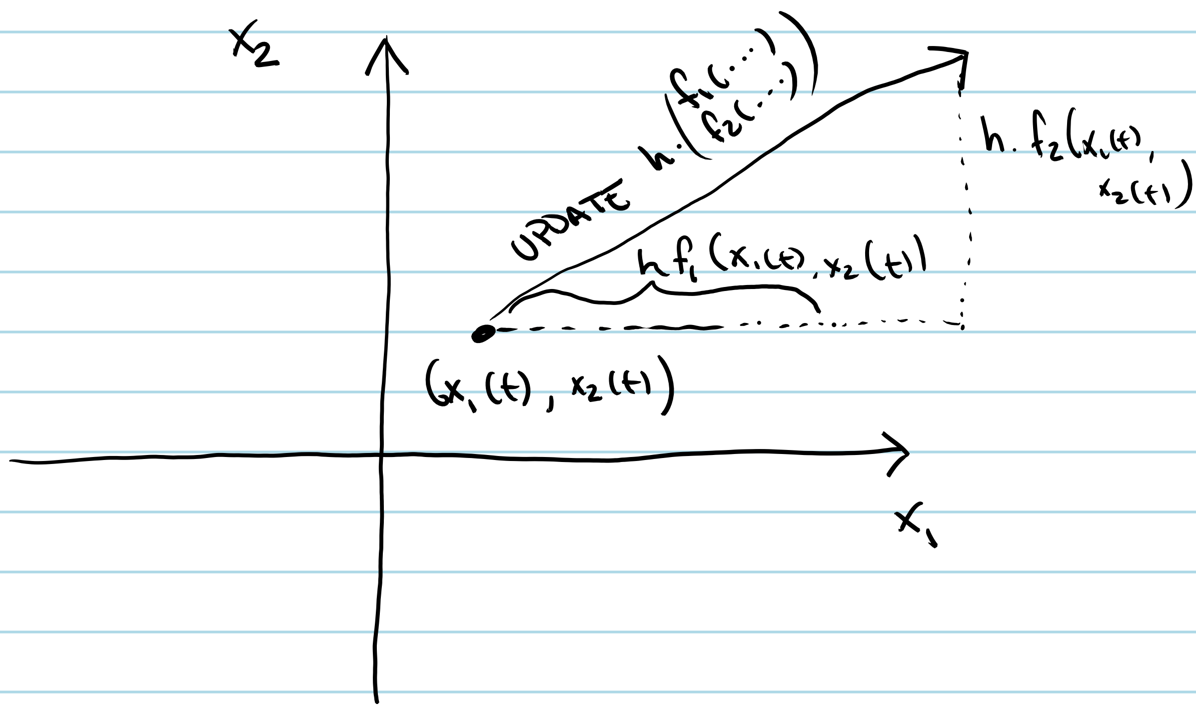
[try timestep $h = 0.1$, $h = 0.01$

zooming, still see substantial error

can skip some of the following (2011), depending on previous lec

Note... Euler's Method is key for understanding why "direction field plots" give TRAJECTORIES that solve

$$\frac{dx}{dt} = f(t, x)$$



1. Say I'm at $x_1(t)$, $x_2(t)$. Want to ADVANCE trajectory to timestep $t+h$

2. According to Euler Method:

$$x_1(t+h) = x_1(t) + \underbrace{h \cdot f_1\left(t, \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}\right)}_{\text{step in HORIZONTAL DIRECTION of size } h \cdot f_1(\ldots)}$$

$$x_2(t+h) = x_2(t) + h \cdot f_2\left(t, \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}\right)$$

3. Put together, get step in direction of the "quiver" arrows in direction_field_plotter.m

Implement this – in MATLAB CODE

direction-field-plotter-and-euler-method-demo.m

· See comments in that code.

—·—

Try with different timesteps h.

Use... my-odefun.m , which gives

$$\begin{cases} \dfrac{dx_1}{dt} = x_2 \\ \dfrac{dx_2}{dt} = -x_1 \end{cases}$$

$h = 0.01$
$T_{max} = 5$ → see solution trajectory

$h = 0.1$
$T_{max} = 5$ → again, but "jerkier", more approximate.

$h = 0.01$ → solution "spirals out."
$T_{max} = 50$

→ Hmm, was that supposed to happen?

Let's see. Define

$$r^2 = x_1^2 + x_2^2$$

$$\frac{d}{dt} r^2 = 2x_1 \frac{dx_1}{dt} + 2x_2 \frac{dx_2}{dt} = 2x_1x_2 - 2x_2x_1 = \underline{\underline{0}}$$

So "RADIUS" __NOT__ __supposed__ to change. Apparently our method is __APPROXIMATE__ indeed.

Note, get closer answer if take smaller h.

BUT, small $h \to \frac{T_{max}}{h}$ timesteps, MANY TIMESTEPS $\to$ TAKES __TOO LONG__ TO RUN...

__Is__ there a SMARTER way to do a timestep-update of a given, __FIXED__ size h? That's our NEXT TOPIC!

Sec. 5 of Amath 301 Notes, by Nathan Kutz (see website).
reference:

ANALYSIS AND DESIGN OF ALGORITHMS FOR ORDINARY DIFF. EQNS:

$\dot{x} = f(t, x)$; timestep h.

Propose:

$$\tilde{x}(t+h) = x(t) + h\left[A f(t, x(t)) + B f\left(t + Ph,\ x(t) + Qh\, f(t, x(t))\right)\right] \quad (1)$$

A, B, P, Q constants.

Euler method: $A = 1$, $B = 0$.

NOTATION
$f_t(t, x(t)) = \frac{\partial}{\partial t} f(t, x(t))$.
etc.

Taylor-expand final term in (1):

$$f\left(t + Ph,\ x(t) + Qh\, f(t, x(t))\right) =$$

$$f(t, x(t)) + Ph\, f_t(t, x(t)) + Qh\, f(t, x(t))\, f_x(t, x(t)) + \underbrace{O(h^2)}_{\text{size ``higher''}} \quad (2)$$

Plug (2) $\rightarrow$ (1):

$$\tilde{x}(t+h) = x(t) + h(A+B) f(t, x(t)) + h^2\left(BP\, f_t(t, x(t)) + BQ\, f(t, x(t))\, f_x(t, x(t)) + O(h^3)\right) \quad (3)$$

Direct Taylor expansion: THIS IS ACCURATE TO $O(h^3)$ BY DEFINITION.

$$x(t+h) = x(t) + h\frac{dx}{dt} + \frac{h^2}{2}\frac{d^2x}{dt^2} + O(h^3) \quad (4).$$

$$\frac{dx}{dt} = f(t, x(t))$$

$$\frac{d^2x}{dt^2} = f_t(t, x(t)) + f_x(t, x(t))\,\frac{dx}{dt}$$

$$= f_t(t, x(t)) + f_x(t, x(t))\, f(t, x(t))$$

So, (4) gives: THIS IS THE GROUND-TRUTH WE SEEK TO MATCH VIA PROPOSAL (1)

$$x(t+h) = x(t) + h\, f(t, x(t)) + \frac{h^2}{2}\Big(f_t(t, x(t)) + f_x(t, x(t))\, f(t, x(t)) \Big) \tag{5}$$

Compare (5) and (3) $\rightarrow$ SEEK TO MAKE PROPOSAL $\tilde{x}(t+h)$ EQUAL TO (5) UP TO $O(h^3)$.

$$(*)\quad \begin{cases} A + B = 1 \\ BP = 1/2 \\ BQ = 1/2 \end{cases}$$

is accurate to $O(h^3)$

$(*)$ is 3 eqns in 4 unknowns. Some freedom ...

1. Additionally set $A = 0$

$$x(t+h) = x(t) + h \cdot f\left(t + \frac{h}{2},\; x(t) + \frac{h}{2} f(t, x(t)) \right) + O(h^3)$$

Modified Euler–Cauchy Method

2) Additionally set $A = \frac{1}{2}$

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

Heun's Method

— · —

CODE: illustrate_heun.m ← Exists?

— · —

ORDER OF ACCURACY OF NUMERICAL METHODS FOR ORDINARY DIFFERENTIAL EQUATIONS:

We approximate $x(t+h)$ by $x_{approx}(t+h) = x(t) + \underbrace{h \cdot [\cdots]}_{\text{update from numerical method}}$

If $|x_{approx}(t+h) - x(t+h)| = O(h^{d+1})$, say numerical method has ORDER d.

Reason: to solve for $x_{approx}(t)$, over $t \in [0, T_{max}]$, need $\frac{T_{max}}{h}$ timesteps.

Say error at each timestep is $b \cdot h^{d+1}$.

Total error $= \frac{T_{max}}{h} \, b \cdot h^{d+1} = \underline{\underline{c\, h^d}}$

ESB note: rel. b/w stability and accuracy. Accuracy calcs assumed we were always evaluating at correct base point. True for one step. But errors can accumulate from timestep to timestep, as evaluate at increasingly wrong base point. This gives instability -- in which case even "accurate" methods give exponentially wrong answers!

__Note!__ Accuracy is not __only__ feature of an algorithm to check. Also: __STABILITY__. Places upper bounds on h.

__Ex!__ Euler method for $\dot{x} = -ax$ (*)

$$x(t+h) = x(t) - ahx(t) = x(t)(1-ah)$$

$$x(n \cdot h) = x(0)(1-ah)^n$$

$$|x(nh)| = |x(0)| \underbrace{|1-ah|}_{k}{}^n$$

__Note:__ if $ah > 2$, $k > 1$, and $|x(nh)| \to \infty$.

Wrong behavior of (*) !!!

Therefore, have RESTRICTION, $h < 2/a$.

" For stability, need h less than FASTEST TIMESCALE IN SYSTEM "

- Consider system with multiple timescales.

Illustrative example:
$$\dot{x}_1 = -ax_1 + g_1(x,y)$$
$$\dot{x}_2 = (-x_2 + g_2(x,y))\varepsilon$$

$a >> 1.$

Need h small so system stable.
But, x_2 changes on timescale of $\frac{1}{\varepsilon}$.

$\therefore T_{max} = \alpha/\varepsilon$

$h < 2/a$

Need $\frac{T_{max}}{h} = \frac{\alpha}{2} \cdot \frac{a}{\varepsilon}$ timesteps

big

small

→ IMPOSSIBLE!

- IMPLICIT METHODS are specialized to these multiple-timescale problems. p.180 E+C.

Def System is stiff if has multiple timescales.

See codes:

euler_illustrate.m

heun_illustrate.m

See: With FIXED timestep h, get more-accurate (much!) with 2ND order heun method ...

TAKE-HOME on NUMERICAL METHODS:

- Do not believe numerical results until have checked with different timesteps h and different methods
- 'Analytical' / theoretical results key for checking answers that are computed ...

Given an odefun on file, MATLAB automatically and rapidly implements these numerical methods.

See Sec. 13 of Lab Manual

- ode45 4TH order Runge-Kutta method (as in E+C sec. 5.7, with "adaptive" choice of stepsize h)

- ode15 for STIFF problems

SYNTAX: >> help ode45 ← Type this into MATLAB for more info + examples.

STEP 1) Define odefun! E.g. my_odefun.m (As before).

important:
use [0:dtmax:Tmax]
to control max stepsize

2) [tlist, state_matrix] = ode45(@my_odefun, [0,Tmax], initialstate)

specify as Rows
$[x_1(0), x_2(0)]$

col. list of times →

$$\begin{bmatrix} t_0; \\ t_1; \\ t_2; \\ \vdots \\ T_{max} \end{bmatrix} \qquad \begin{bmatrix} x_1(t_0), x_2(t_0); \\ x_1(t_1), x_2(t_1); \\ x_1(t_2), x_2(t_2); \\ \vdots \quad \vdots \\ x_1(T_{max}), x_2(T_{max}) \end{bmatrix}$$

← MATRIX OF state vector at these times. Each row → one timestep

Please see

direction-field-plotter-and-euler-method-and-ode45-demo.m

where this is implemented.

Note:

1) MATLAB chooses h automatically ("adaptively")

2) High accuracy achieved: r^2 radius preserved: with h that is not too tiny

3) Many more OPTIONS exist: see
>> help ode45 again.

A final note... you can augment your odefun files to include parameters as additional inputs. See >>help ode45 for more on this.

example-odefun(t, x, p)

vector of parameters here.