110.302 Lecture 32:

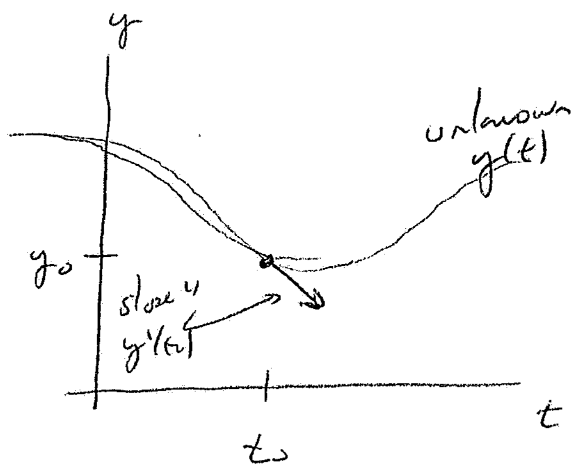
Q1: In the absence of analytic tools, how can one "see" a soln to an IVP?

Q2: How do computers "solve" ODEs?

Model:

$$\frac{dy}{dt} = f(t,y), \quad y(t_0) = y_0$$

where both f, f_y are cont "around" $(t_0, y_0) \in \mathbb{R}^2$.



- Near (t_0, y_0) solns are unique.
- Slope field element at (t_0, y_0) gives us $y'(t_0)$ for solution $y(t)$ that satisfies $y(t_0) = y_0$.
- eqn of line tangent to $y(t)$ at (t_0, y_0) is

ℓ: $y - y_0 = y'(t_0)(t - t_0)$ or ℓ: $y = y_0 + y'(t_0)(t - t_0)$.

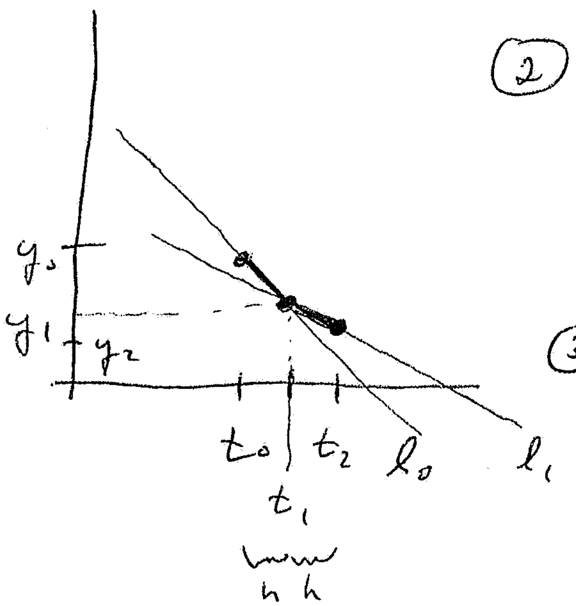
Like in Calc I, if we keep t close to t_0, then this tangent line is a "good" approx to the actual solution $y(t)$.

Idea: ① Follow tangent line for a short time:

Let $t_1 = t_0 + h$ for small h.

Then $y_1 = y_0 + y'(t_0)(t_1 - t_0)$. (call this line l_0)

$= y_0 + f(t_0, y_0)\, h$



② Call the short line segment on l_0 from (t_0, y_0) to (t_1, y_1) your approximated soln.

③ Take (t_1, y_1) as your new starting pt and repeat ① & ② to generate (t_2, y_2):

$$t_2 = t_1 + h$$

$$y_2 = y_1 + \underbrace{f(t_1, y_1)}_{y'(t_1)} h$$

for the different solution passing thru (t_1, y_1)

Notes ⓐ Doing this for a small range of times will <u>Not</u> produce a solution, but will produce something "near" the actual solution passing thru (t_0, y_0)

ⓑ The amount one is "off" one time is called error, and is measurable.

ⓒ This kind of computation is tailor made for computers

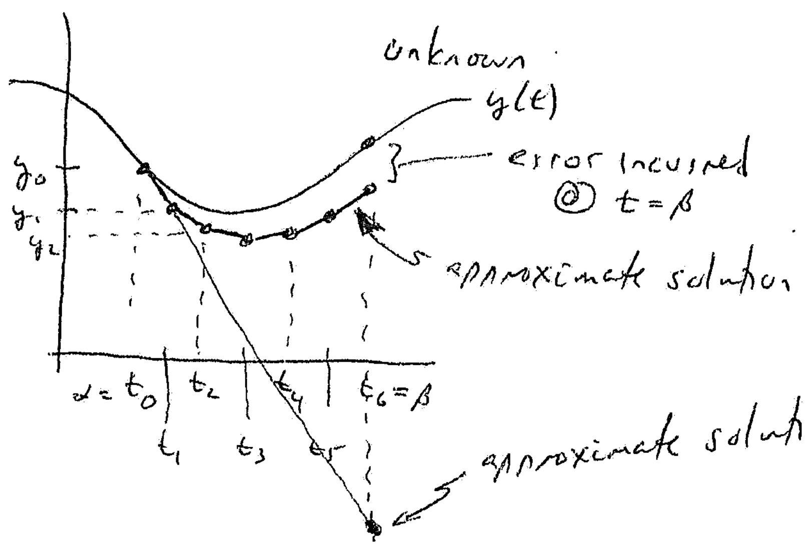
Euler Method

Given $y' = f(t,y)$, $y(t_0) = y_0$, with unique solutions, one can approx. $y(t)$ on $[\alpha,\beta]$, where $t_0 = \alpha$ by

(I) Partition $[\alpha,\beta]$ via step size h:

$$\alpha = t_0, t_1, \dots, t_{n-1}, t_n = \beta, \qquad t_i - t_{i-1} = h$$

(II) Evaluate $y_{i+1} = y_i + h \cdot f(t_i, y_i)$



(III) Connect pts (t_i, y_i) by straight lines.

Q: What are the issues of the Euler Method?

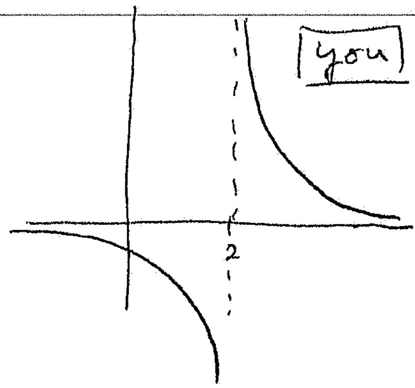
① If step size is too large, approximate solution is far away from real one.

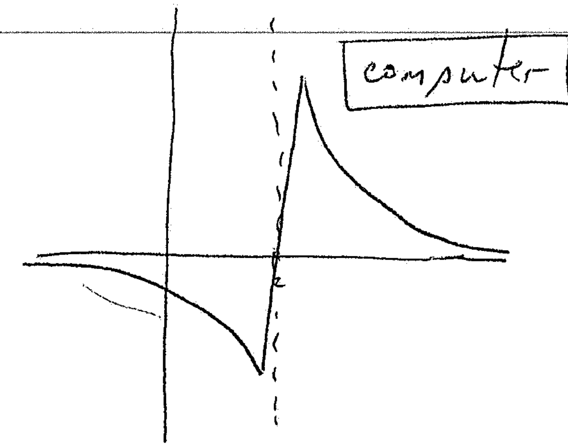
② If step size is too small, computation is expensive

③ Computers cannot think (yet)....

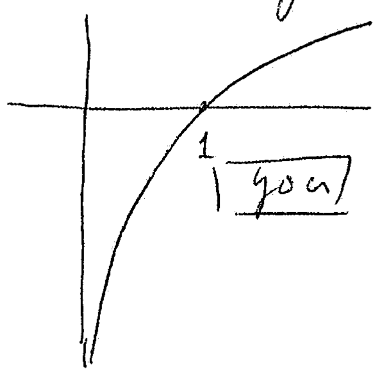
- Computers cannot "see" asymptotes:

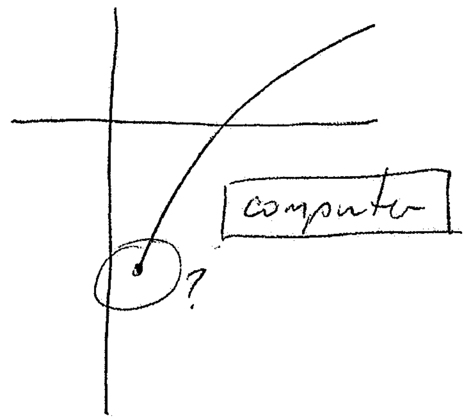
ex. Graph $f(x) = \frac{1}{x-2}$



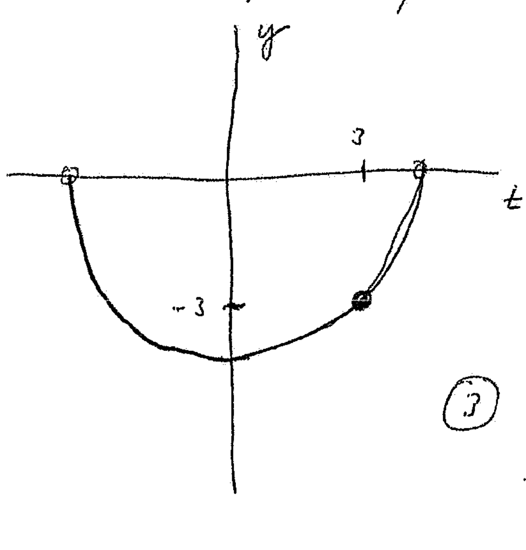


ex. Graph $y = \ln x$





ex. ~~Draw phase portrait~~ graph solution of $y' = \frac{-t}{y}$. $y(3) = -3$.



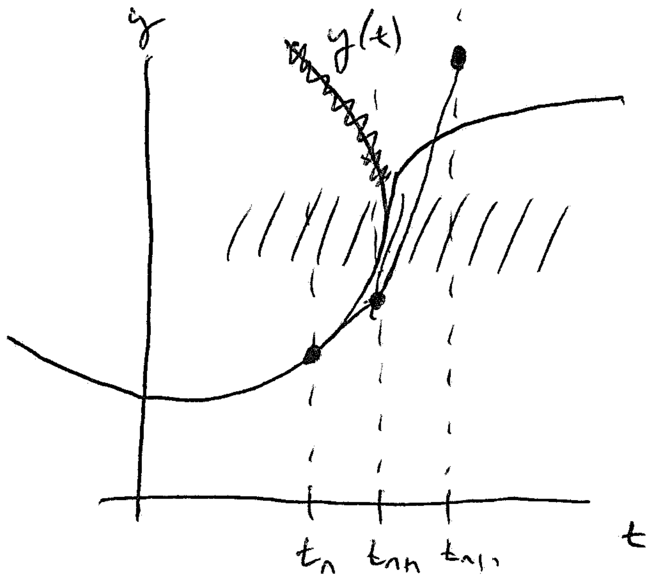
① what is y-intercept?

② Find t where $y(t) = 1$ (there are 2 of them).

③ See JOde phase portraits of this solution for different solution methods.

④ Near where the vector field is very steep, (almost vertical), strange things can happen

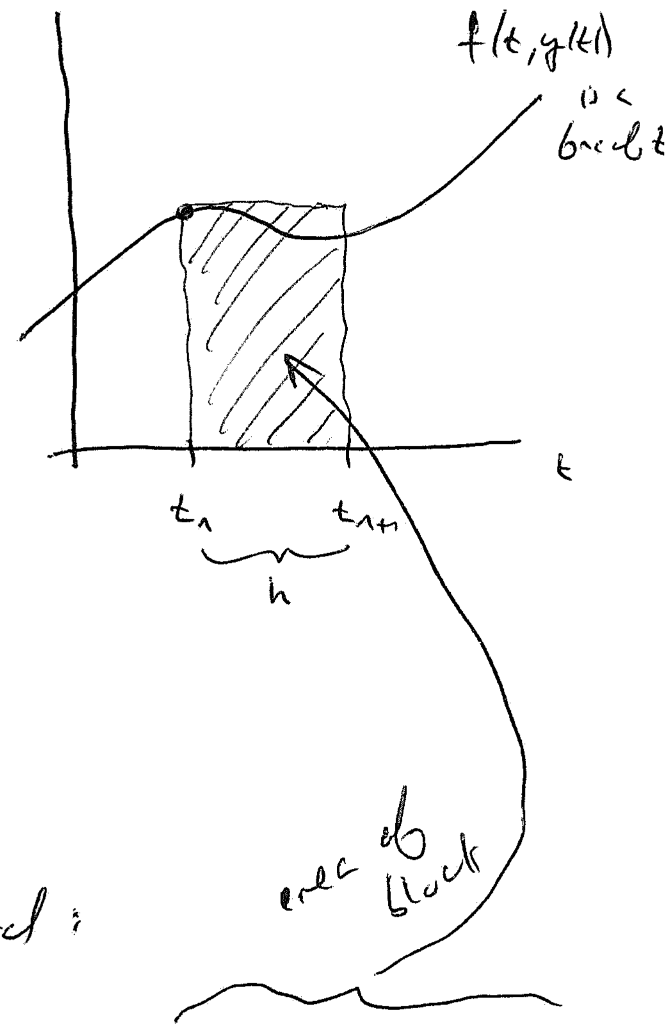
This is because the step size horizontally may be small, but vertically may be great!



See JODE.

How to generalize the Euler Method

Euler Method $y_{n+1} = y_n + h \cdot f(t_n, y_n)$



Integrating $f(t, y(t))$ over the interval $[t_n, t_{n+1}]$ would $y(t_{n+1})$ precisely, since $y' = f(t, y(t))$, and

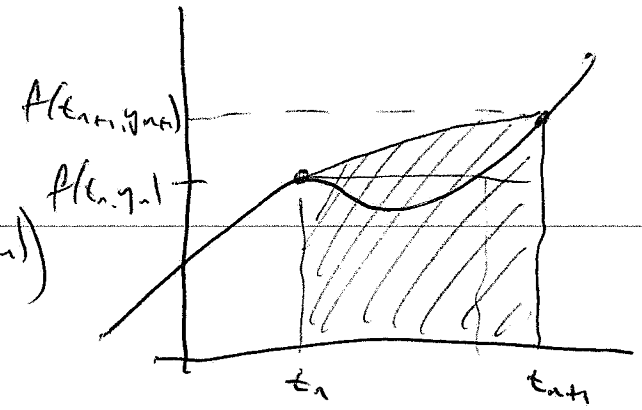
$$y(t) = y(t_n) + \int_{t_n}^{t} f(s, y(s))\,ds$$

By FTC.

Instead, we estimate the integral:

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(s, y(s))\,ds \approx y(t_n) + f(t_n, y_n) \cdot \underbrace{(t_{n+1} - t_n)}_{h}.$$

Improving the Euler Method involves only finding better estimates of the integral..



Trapezoid Rule.

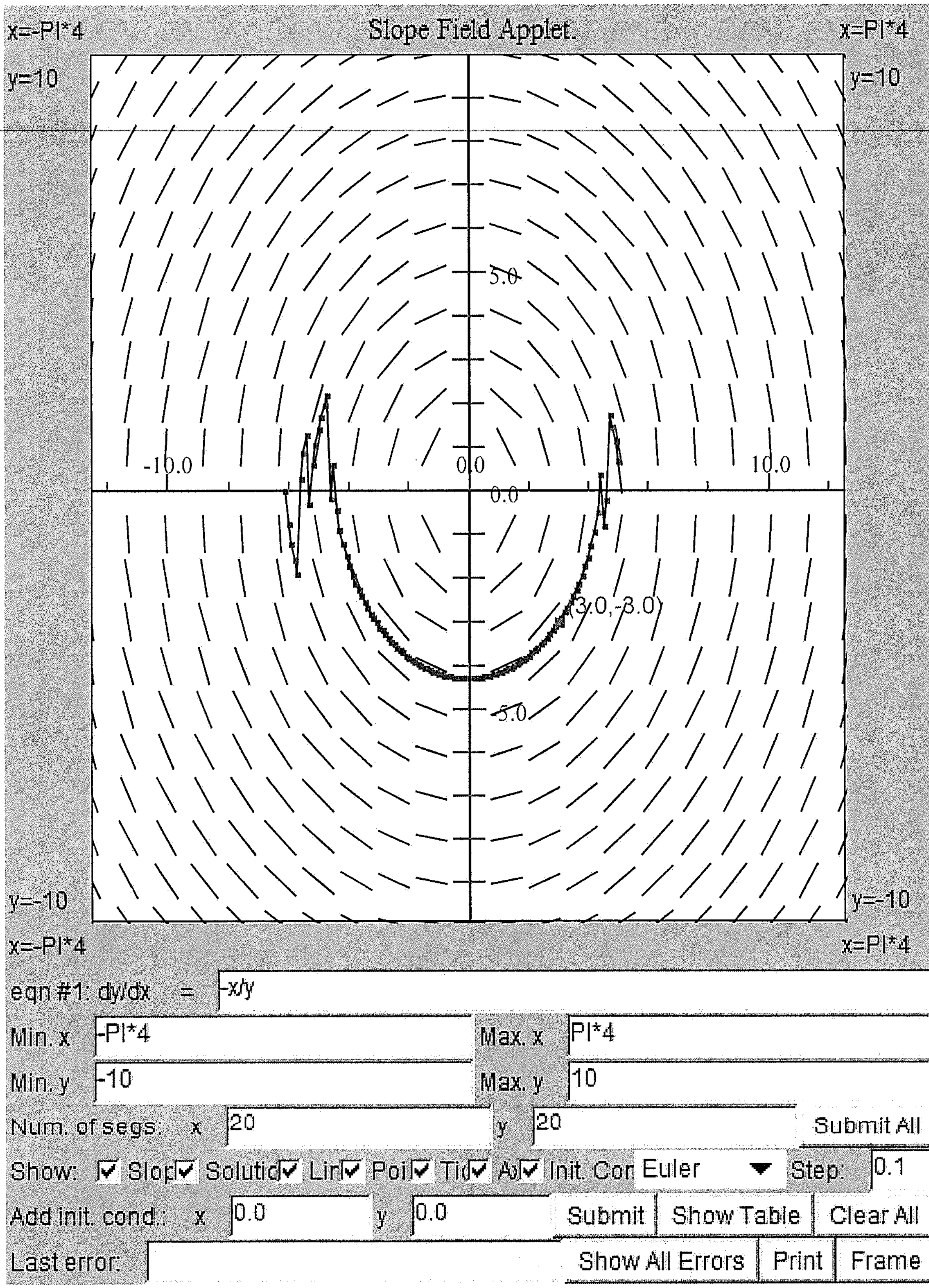
$$y_{n+1} = y_n + h\left(\frac{f(t_n,y_n)+f(t_{n+1},y_{n+1})}{2}\right)$$

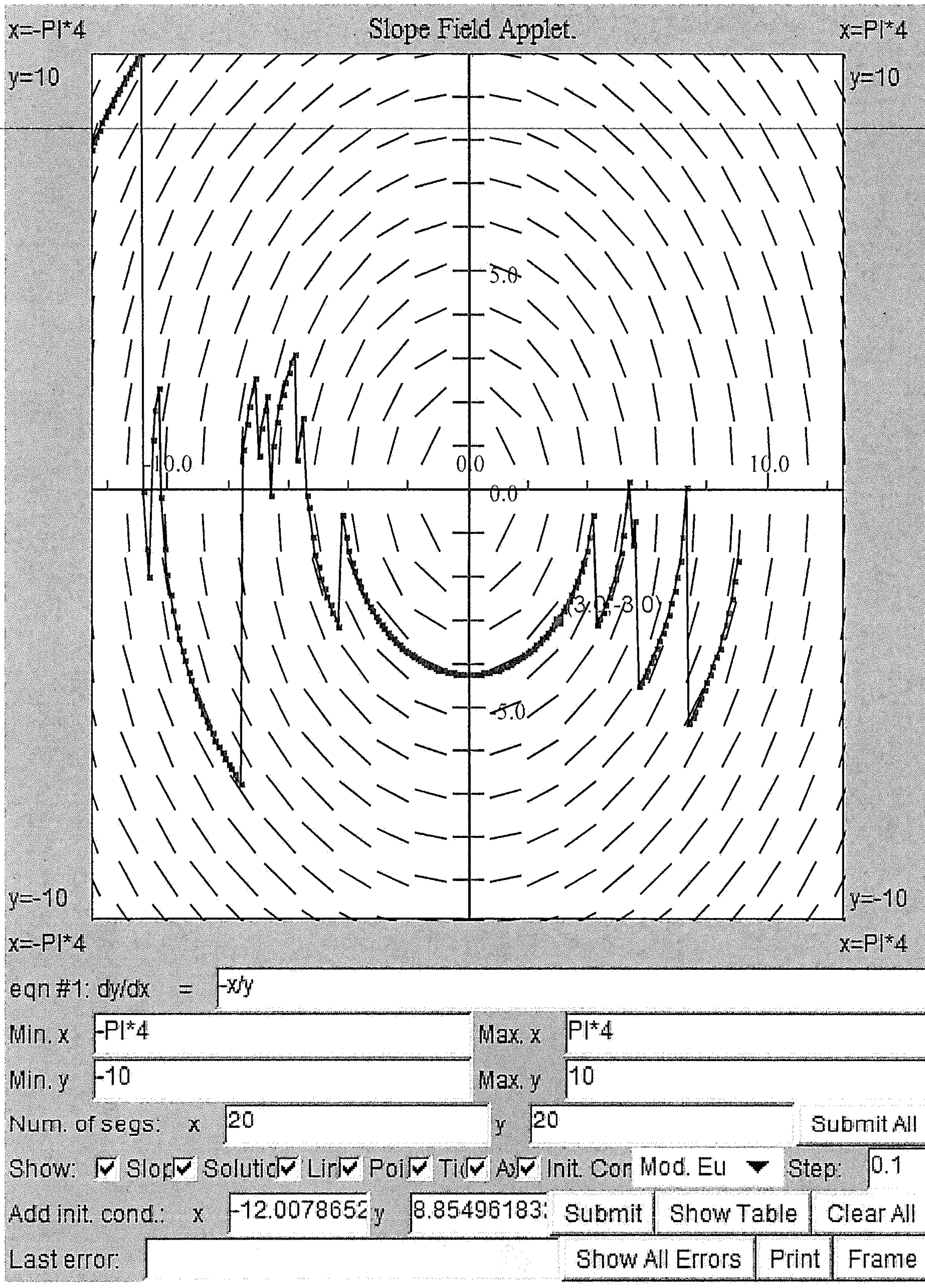
weird.

Instead, estimate y_{n+1} on RHS via standard Euler formula

$$y_{n+1} = y_n + \tfrac{1}{2}h\Big(f(t_n,y_n) + f(t_{n+1},\ \underbrace{y_n + h f(t_n,y_n)}_{\text{Euler formula estimate for } y_{n+1}})\Big)$$

The Runge-Kutta methods are simply better, more computationally expensive versions of this estimation process.


Slope Field Applet.
x=-PI*4
x=PI*4
y=10
y=10
5.0
-10.0
0.0
10.0
0.0
(3.0,-3.0)
-5.0
y=-10
y=-10
x=-PI*4
x=PI*4
eqn #1: dy/dx =
-x/y
Min. x
-PI*4
Max. x
PI*4
Min. y
-10
Max. y
10
Num. of segs: x
20
y
20
Submit All
Show:
Slop
Solutic
Lir
Poi
Ti
Init. Con
Euler
Step:
0.1
Add init. cond.: x
0.0
y
0.0
Submit
Show Table
Clear All
Last error:
Show All Errors
Print
Frame


x=-PI*4
Slope Field Applet.
x=PI*4
y=10
y=10
5.0
-10.0
0.0
10.0
0.0
(3.0,-3.0)
-5.0
y=-10
y=-10
x=-PI*4
x=PI*4
eqn #1: dy/dx = -x/y
Min. x -PI*4
Max. x PI*4
Min. y -10
Max. y 10
Num. of segs: x 20 y 20
Submit All
Show: Slop Solutic Lir Poi Ti A Init. Con
Mod. Eu
Step: 0.1
Add init. cond.: x -12.0078652 y 8.85496183
Submit
Show Table
Clear All
Last error:
Show All Errors
Print
Frame