Math 2250-004 Week 4 notes

We will not necessarily finish the material from a given day's notes on that day. We may also add or subtract some material as the week progresses, but these notes represent an in-depth outline of what we plan to cover. These notes include material from 2.3-2.6, with an introduction to 3.1-3.2.

Mon Jan 29

2.3 Improved acceleration models: linear and quadratic drag forces.

Announcements: none! §2.3 today, so you can finish HW & Lab.

Warm-up Exercise: Draw the phase diagram for our favorite 1st order

a) differential equation, written today for a function v(t) : "alpha minus rho times v"

'til 10:48

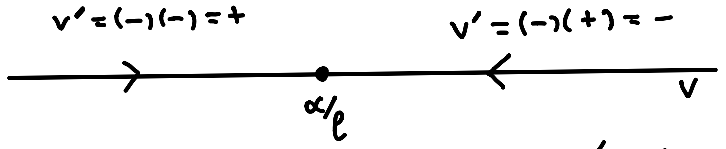
$$v' = \alpha - \rho\cdot v \qquad \alpha, \rho \text{ constants, } \underline{\underline{\rho>0}}.$$

Indicate stability for the one equilibrium soltn

b) Solve for v(t): $\begin{cases} v' = \alpha - \rho v \\ v(0) = v_0 \end{cases}$ $(\rho > 0)$

a). equil soltns to autonomous 1st order DE ARE constant soltns.

in this case set $v'(t) \equiv 0 = \alpha - \rho v \Rightarrow \boxed{v = \frac{\alpha}{\rho}}$



$$v' = \alpha - \rho v = -\rho\left(v - \frac{\alpha}{\rho}\right)$$
$\rho > 0.$

b) for v(t):

$$v' + \rho v = \alpha$$
$$e^{\rho t}\left[v' + \rho v\right] = \alpha e^{\rho t}$$
$$\frac{d}{dt}\left(e^{\rho t} v\right) = \alpha e^{\rho t}$$
$$e^{\rho t} v = \int \alpha e^{\rho t}\, dt = \frac{\alpha}{\rho} e^{\rho t} + C$$
$$\Rightarrow \boxed{v = \frac{\alpha}{\rho} + C e^{-\rho t}}$$
$$\lim_{t\to\infty} v(t) = \alpha/\rho \ ✓$$

$$\boxed{\begin{array}{l} \underline{\text{IVP}}: \\ v(0) = v_0 = \frac{\alpha}{\rho} + C \\ \Rightarrow C = v_0 - \frac{\alpha}{\rho} \\ v(t) = \frac{\alpha}{\rho} + \left(v_0 - \frac{\alpha}{\rho}\right)e^{-\rho t} \end{array}}$$

2.3 Improved acceleration models: velocity-dependent drag

For 1-dimensional particle motion, with

$$\text{position } x(t) \ (\text{or } y(t))\ , \quad \bullet$$
$$\text{velocity } x'(t) = v(t)\ , \text{ and} \quad \bullet$$
$$\text{acceleration } x''(t) = v'(t) = a(t) \quad \bullet$$

We have Newton's 2^{nd} law

$$m\, v'(t) = F$$

where F is the net force, possible a sum consisting of several terms.

- By now we're very familiar with constant force $F = m\,\alpha$, where α is a constant:

$$v'(t) = \alpha \qquad \bullet$$
$$v(t) = \alpha\, t + v_0 \qquad \bullet$$
$$x(t) = \frac{1}{2}\alpha\, t^2 + v_0\, t + x_0\,. \qquad \bullet$$

Examples we've seen already in this course

- $\alpha = -g$ near the surface of the earth, if up is the positive direction, or $\alpha = g$ if down is the positive direction. •
- α arising from a charged particle moving through a constant electric field (lab problem) •
- boats or cars moving with constant acceleration or deceleration (homework). •

New today !!! Combine a constant background force with a velocity-dependent drag force, at the same time. The text calls this a "resistance" force:

$$m\, v'(t) = m\,\alpha + F_R$$

Empirically/mathematically the resistance forces F_R depend on velocity, in such a way that their magnitude is

$$|F_R| \approx k\,|v|^p \ ,\ 1 \le p \le 2\,.$$

- $p = 1$ (linear model, drag proportional to velocity):

$$m\,v'(t) = m\,\alpha \underline{-k\,v}$$

This linear model makes sense for "slow" velocities, as a linearization of the frictional force function, assuming that the force function is differentiable with respect to velocity...recall Taylor series for how the velocity resistance force might depend on velocity:

- $$F_R(v) = \overset{\overset{0}{\shortparallel}}{F_R(0)} + F_R'(0)\,v + \boxed{\cancel{\frac{1}{2!}F_R''(0)v^2 + \dots}}$$

$\boxed{F_R(0) = 0}$ and for small enough v the higher order terms might be negligable compared to the linear term, so

$$F_R(v) \approx F_R'(0)\,v \approx -k\,v\ . \qquad , k>0$$

We write $-k\,v$ with $k > 0$, since the frictional force opposes the direction of motion, so sign opposite of the velocity's.

http://en.wikipedia.org/wiki/Drag_(physics)#Very_low_Reynolds_numbers:_Stokes.27_drag

Exercise 1: Let's rewrite the linear drag model

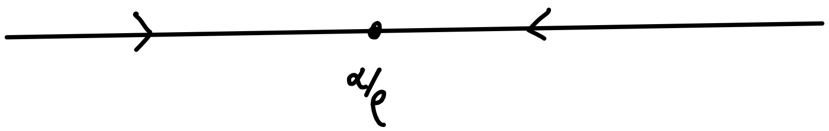
$$m\,v'(t) = m\,\alpha - k\,v$$

as

$$\boxed{v'(t) = \alpha - \rho\,v}$$

where the $\boxed{\rho = \frac{k}{m}}$ Now construct the phase diagram for v. (Hint: there is one critical value for v.)

we drew phase diagram in warm-up.



The value of the constant velocity solution is called the $\boxed{\overset{\alpha/\rho}{\shortparallel}\ \textit{terminal velocity,}}$ which makes good sense when you think about the underlying physics and phase diagram.

- $p = 2$, for the power in the resistance force. This can be an appropriate model for velocities which are not "near" zero....described in terms of "Reynolds number" Accounting for the fact that the resistance opposes direction of motion we get

$$m\,v'(t) = m\,\alpha\ \overbrace{-k\,v^2}^{F_R} \quad \text{if } v > 0,$$ F_R should be negative

$$m\,v'(t) = m\,\alpha + k\,v^2 \quad \text{if } v < 0.,$$ F_R should be positive.

Do you understandt the sign of the drag terms in these two cases?

http://en.wikipedia.org/wiki/Drag_(physics)#Drag_at_high_velocity

Once again letting $\rho = \frac{k}{m}$ we can rewrite the DE's as

$$v'(t) = \alpha - \rho\,v^2 \quad \text{if } v > 0$$
$$v'(t) = \alpha + \rho\,v^2 \quad \text{if } v < 0.$$

Exercise 2) Consider the case in which $\alpha = -g$, so we are considering vertical motion, with up being the positive direction.

$$v'(t) = -g - \rho\,v^2 \quad \text{if } v > 0$$
$$v'(t) = -g + \rho\,v^2 \quad \text{if } v < 0.$$

Draw the phase diagrams. Note that each diagram contains a half line of v-values. Make conclusions about velocity behavior in case $v_0 > 0$ and $v_0 \le 0$. Is there a terminal velocity?

How would you set up and get started on finding the solutions to these two differential equations? A couple of your homework problems are related to this quadratic drag model.

Exercise 3a Returning to the linear drag model and with. Solve the IVP

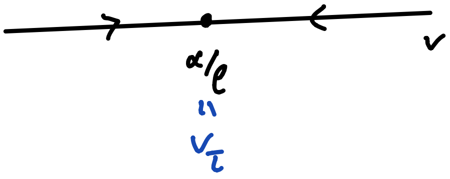
$$v'(t) = \alpha - \rho v$$
$$v(0) = v_0$$

and verify that your solutions are consistent with the phase diagram analysis two pages back. (This is, once again, our friend the first order constant coefficient linear differential equation.)

$$v(t) = \frac{\alpha}{\rho} + \left(v_0 - \frac{\alpha}{\rho}\right) e^{-\rho t} \qquad v(0) = \frac{\alpha}{\rho} + \left(v_0 - \frac{\alpha}{\rho}\right) = v_0 \ \checkmark$$

$$\lim_{t \to \infty} v(t) = \alpha/\rho$$

$$= v_\tau$$



v_τ = terminal velocity

3b integrate the velocity function above to find a formula for the position function $y(t)$. Write $y(0) = y_0$.

$$v(t) = \frac{\alpha}{\rho} + \left(v_0 - \frac{\alpha}{\rho}\right) e^{-\rho t} = v_\tau + (v_0 - v_\tau) e^{-\rho t}$$

$$\Rightarrow \quad y(t) = \int v(t)\,dt = \frac{\alpha}{\rho} t + \left(v_0 - \frac{\alpha}{\rho}\right) \frac{1}{-\rho} e^{-\rho t} + C$$

$$y(t) = v_\tau t + \frac{(v_0 - v_\tau)}{-\rho} e^{-\rho t} + C$$

$$y(0) = y_0 = 0 + \frac{v_0 - v_\tau}{-\rho} + C$$

$$y_0 + \frac{v_0 - v_\tau}{\rho} = C$$

$$y(t) = y_0 + v_\tau t + \frac{v_0 - v_\tau}{\rho}\left(1 - e^{-\rho t}\right)$$

$$C$$

Comparison of Calc 1 constant acceleration vs. linear drag acceleration model:

We consider the bow and deadbolt example from the text, page 102-104. It's shot vertically into the air (watch out below!), with an initial velocity of 49 $\frac{m}{s}$. (That initial velocity is chosen because its numerical value is 5 times the numerical value of $g = 9.8\ \frac{m}{s^2}$, which simplifies some of the computations.) In the no-drag case, this could just be the vertical component of a deadbolt shot at an angle. With drag, one would need to study a more complicated system of coupled differential equations for the horizontal and vertical motions, if you didn't shoot the bolt straight up. So we're shooting it straight up.

No drag:

$$v'(t) = -g \approx -9.8\ \frac{m}{s^2} \qquad \bullet$$

$$v(t) = -g\,t + v_0 = -g\,t + 5\,g = g\cdot(-t+5)\quad \frac{m}{s} \qquad \bullet$$

$$x(t) = -\frac{1}{2}g\,t^2 + v_0\,t + x_0 = -\frac{1}{2}g\,t^2 + 5\,g\,t = g\,t\left(-\frac{1}{2}t + 5\right)\quad m \qquad \bullet$$

So our deadbolt goes up for 5 seconds, then drops for 5 seconds until it hits the ground. Its maximum height is given by

$v(5) = 0$ $\qquad$ $x(0) = 0 = x(10)$

$$x(5) = \frac{g\cdot 5\cdot 5}{2} = 122.5\ m \qquad \bullet$$

Linear drag: The same deadbolt, with the same initial velocity with numerical value $5\,g = 49\ \frac{m}{s}$. We're told that our deadbolt has a measured terminal velocity of $v_\tau = -245\ \frac{m}{s}$ which is the numerical value of $-25\,g$. The initial value problem for velocity is

$$\left\{\begin{array}{l} v'(t) = -g - \rho\,v \\ v(0) = v_0 = \cancel{25\,g = 245.}\ 5g = 49\ m/s \end{array}\right.$$

So, in these letters the terminal velocity is (easily recoverable by setting $v'(t) = 0$) and is given by

$$\bullet \quad v_\tau = -\frac{g}{\rho} = -25\,g \Rightarrow \rho = .04\,. \quad \bullet$$

So, from our earlier work: Substituting $\alpha = -g$ into the formulas for terminal velocity, velocity, and height:

$$v(t) = v_\tau + \left(v_0 - v_\tau\right) e^{-\rho\,t} = -245 + 294\,e^{-.04\,t}\,. \qquad 294 = 49 + 245$$

$$y(t) = -245\,t + \frac{294}{.04}\left(1 - e^{-.04\,t}\right).$$

The maximum height occurs when $v(t) = 0$,

$$-245 + 294\,\mathrm{e}^{-.04\,t} = 0$$

which yields $t = 4.56$ sec: vs $\underline{5\text{ sec}}$

$$> \ -\frac{\ln\left(\frac{245.}{294.}\right)}{.04};$$

$$4.558038920 \qquad \textbf{(1)}$$

And the maximum height is 108.3 m:

$$> \ y := t \rightarrow -245.\cdot t + \frac{294}{.04}\left(1 - \mathrm{e}^{-.04\cdot t}\right);$$

$$y(4.558038920);$$

$$y := t \rightarrow (-1)\cdot 245.\,t + \frac{294\left(1 - \mathrm{e}^{(-1)\cdot 0.04\,t}\right)}{0.04}$$

$$108.280465 \qquad \text{vs. } 122.5 \qquad \textbf{(2)}$$

So the drag caused the deadbolt to stop going up sooner ($t = 4.56$ vs. $t = 5$ sec) and to not get as high (108.3 *vs* 122.5 *m*). This makes sense. It's also interesting what happens on the way down - the drag makes the descent longer than the ascent: 4.85 seconds on the descent, vs. 4.56 on the ascent.

$$> \ solve(y(t) = 0, t);$$

$$9.410949931,\ 0. \qquad \textbf{(3)}$$

$$> \ 9.411 - 4.558;$$

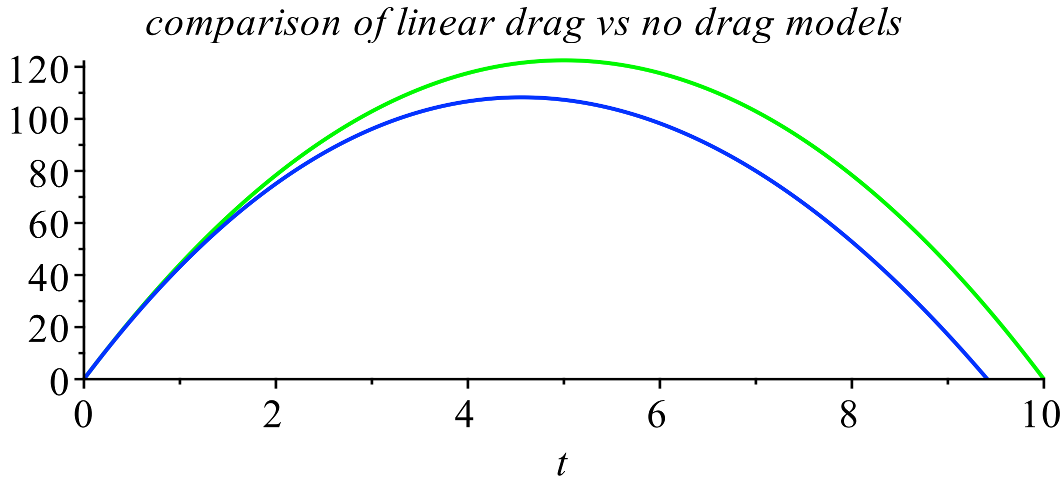
$$4.853 \qquad \textbf{(4)}$$

IMPORTANT to note that we needed to use a "solve" command (or something sophisticated like Newton's method) to find when the deadbolt landed. You cannot isolate the t algebraically when trying to solve $y(t) = 0$ for t. This situation will also happen in some of the lab and/or homework problems this week.

$$-245.\cdot t + \frac{294}{.04}\left(1 - \mathrm{e}^{-.04\cdot t}\right) = 0$$

picture:

```
> z := t→49 t − 4.9·t^2 :
  with(plots) :
  plot1 := plot(z(t), t = 0 ..10, color = green) :
  plot2 := plot(y(t), t = 0 ..9.4110, color = blue) :
  display({plot1, plot2}, title = `comparison of linear drag vs no drag models`);
```



```
>
```

Tues Jan 30

2.4 Euler's method for solving first order differential equations

Announcements:
- last page of HW due tomorrow is corrected HW score] uncorrected HW score
- 2.4-2.6 HW is not supposed to be bold & underline (HW due tomorrow is through §2.3)
- On Thursday lab, bring at least a calculator, or smartphone @ Wolfram alpha.
  best: bring laptop with access to Matlab.

Warm-up Exercise:

Consider the IVP
$$\begin{cases} y'(x) = f(x,y) \\ y(1) = 2 \end{cases}$$
rate of change function
///

If all you know is the value of the slope function $f(x,y)$ at the initial point $(1,2)$, say
$$y'(1) = f(1,2) = 3,$$
what's your best approximation for $y(1.1)$? for $y(.9)$?

Ans ≈ 2.3 $\qquad \approx 1.7$

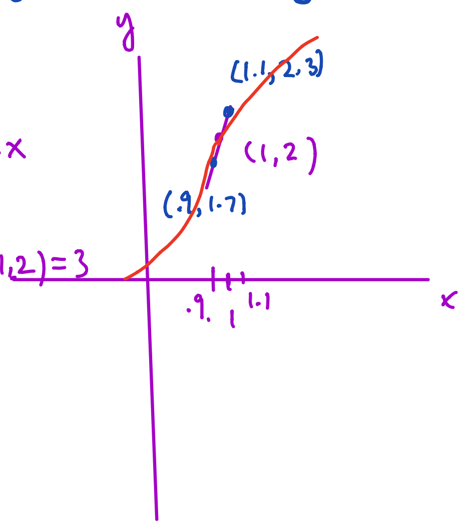
'til 10:48

$$\begin{aligned} y(1.1) &\approx y(1) + dy \\ &= y(1) + y'(1)\Delta x \\ &= 2 + 3(.1) \\ &= 2.3. \end{aligned}$$

$\Delta y \approx y'(x)\Delta x$

$\Delta x = .1$

$y'(1) = f(1,2) = 3$



$$\begin{aligned} y(.9) &\approx y(1) + dy \\ &= 2 + y'(1)\Delta x \\ &= 2 + 3(-.1) \\ &= 1.7 \end{aligned}$$

$y' = f(x,y)$

@ (x_0, y_0)

$\Delta y \approx f(x_0, y_0)\,\Delta x$

2.4 Euler's method.

In these notes we will study numerical methods for approximating solutions to first order differential equations. Later in the course we will see how higher order differential equations can be converted into first order systems of differential equations. It turns out that there is a natural way to generalize what we do now in the context of a single first order differential equations, to systems of first order differential equations. So understanding this material will be an important step in understanding numerical solutions to higher order differential equations and to systems of differential equations later on. I wrote these notes using the software package Maple. Your lab questions on Thursday will let you do analogous computations in Matlab. Today we'll focus on Euler's method. Tomorrow we'll study "improved Euler" (section 2.5) and Runge Kutta (section 2.6).

Euler's Method:

The most basic method of approximating solutions to differential equations is called Euler's method, after the 1700's mathematician who first formulated it. Consider the initial value problem

$$\text{IVP} \begin{cases} \dfrac{dy}{dx} = f(x, y) \\ y(x_0) = y_0. \end{cases}$$

We make repeated use of the familiar (tangent line) approximation

$$y(x + \Delta x) \approx y(x) + y'(x)\Delta x$$

where we substitute in the slope function $f(x, y)$, for $y'(x)$.

As we iterate this procedure we and the text will write "h" fixed step size, $\Delta x := h$. As we increment the inputs x and outputs y we are approximating the graph of the solution to the IVP. We begin at the initial point (x_0, y_0). Then the next horizontal value on the discrete graph will be

$$x_1 = x_0 + h = x_0 + \Delta x$$

And the next vertical value y_1 (approximating the exact value $y(x_1)$) is

$$y_1 = y_0 + f(x_0, y_0)h. \qquad = y_0 + y'(x_0)\Delta x$$

In general, if we've approximated (x_j, y_j) we set

$$x_{j+1} = x_j + h$$
$$y_{j+1} = y_j + f(x_j, y_j)h.$$

We could also approximate in the $-x$ direction from the initial point (x_0, y_0), by defining e.g.

$$x_{-1} = x_0 - h$$
$$y_{-1} = y_0 - hf(x_0, y_0)$$

and more generally via

$$x_{-j-1} = x_{-j} - h$$
$$y_{-j-1} = y_{-j} - hf(x_{-j}, y_{-j})$$

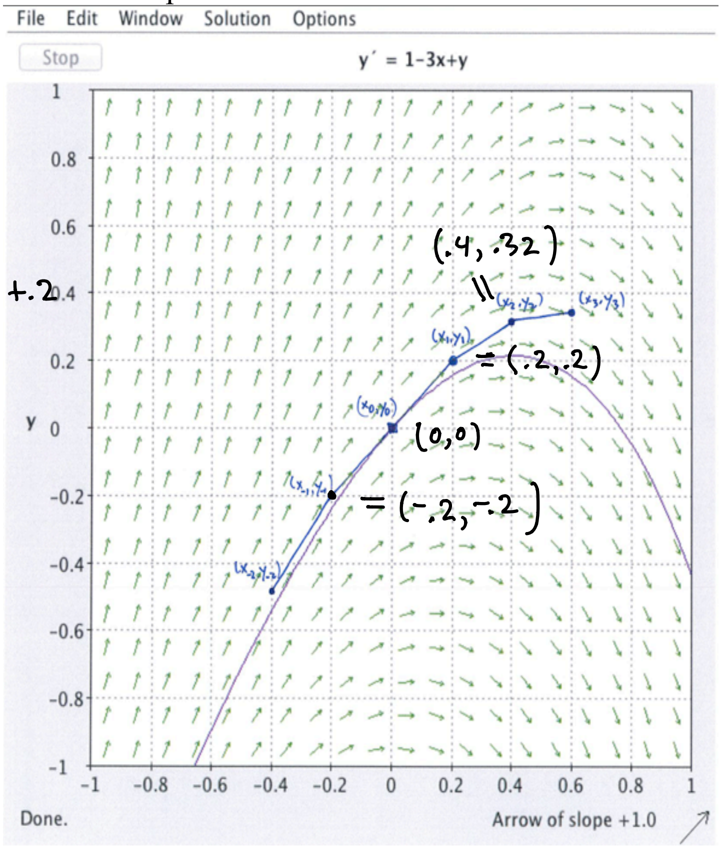
...etc.

Below is a graphical representation of the Euler method, illustrated for the IVP

$$\begin{cases} y'(x) = 1 - 3x + y = f(x,y) \\ y(0) = 0 \end{cases}$$

with step size $h = 0.2$.

Exercise 1 Find the numerical values for several of the labeled points.



$x_0 = 0 \quad y_0 = 0 \quad f(x_0,y_0) = f(0,0) = 1$

$x_1 = .2 \quad \Delta y \approx f(0,0)\cdot h = 1(.2) = .2$

$y_1 = y_0 + \Delta y$

$= 0 + .2 = .2$

$x_1 = .2, \quad y_1 = .2 \quad f(.2,.2) = 1 - .6 + .2$

$= .6$

$\Delta y \approx (.6)(.2)$

$= .12$

$y_2 = y_1 + \Delta y = .2 + .12 = .32$

$x_2 = .4, \quad y_2 = .32$

$x_{-1} = -.2, \quad$ from $(x_0, y_0) = (0,0)$.

$f(0,0) = 1$

$\Delta y = f(0,0)(\Delta x)$

$= 1(-.2)$

$= -.2$

$y_{-1} = y_0 + \Delta y$

$= 0 - .2 = -.2$

$x_{-1} = -.2, \quad y_{-1} = -.2$

As a running example in the remainder of these notes we will use one of our favorite IVP's from the time of Calculus, namely the initial value problem for $y(x)$,

$$y'(x) = y \; = f(x,y)$$
$$y(0) = 1.$$

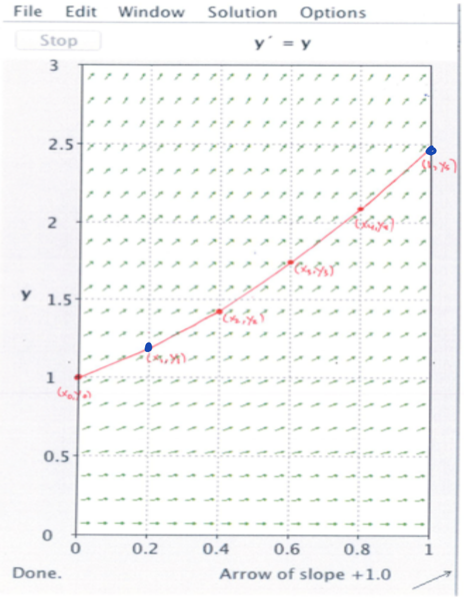
We know that $\underline{y = e^x}$ is the solution, so we'll be able to compare approximate and exact solution values.

Exercise 2 Work out by hand the approximate solution to the IVP above on the interval $[0, 1]$, with $n = 5$ subdivisions and $h = 0.2$, using Euler's method. This table might help organize the arithmetic. In this differential equation the slope function is $f(x, y) = y$ so is especially easy to compute.

step i	x_i	y_i	slope $f(x_i, y_i) = y_i$	Δx	$\Delta y = f(x_i, y_i)\Delta x$	x_{i+1}	$y_i + \Delta y$
0	0	1	1	0.2	0.2 $= 1(.2)$	0.2	1.2 $= 1 + (.2)$
1	0.2	1.2	1.2	0.2	$(1.2)(.2) = .24$	$.4$	$1.2 + .24 = 1.44$ | $1.2 + (1.2)(.2) = (1.2)(1+.2) = (1.2)^2$
2	$.4$	1.44	$1.44 = (1.2)^2$	0.2	$.288$ or $(1.2)^2(.2)$	$.6$	$1.44 + .288 = 1.728$ | $(1.2)^2 + (.2)(1.2)^2 = (1.2)^2(1+.2) = (1.2)^3$
3	$.6$	$(1.2)^3$	$(1.2)^3$	$.2$	$(1.2)^3(.2)$	$.8$	$(1.2)^3 + .2(1.2)^3 = (1.2)^4$
4	$.8$	$(1.2)^4$	$(1.2)^4$	$.2$	$(1.2)^4(.2)$	1	$(1.2)^4 + (1.2)^4(.2) = (1.2)^5 = 2.488$
5	1	$2.488 = (1.2)^5$					

Euler Table

Your work should be consistent with the picture below.



Here is the automated computation for the previous exercise, and a graph comparing the approximate solution points to the actual solution graph:

Initialize:

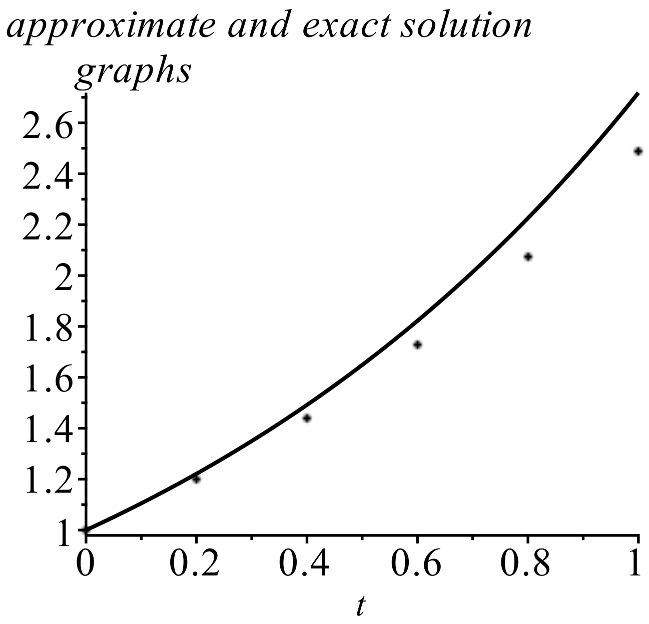
```
> restart : #clear all memory, if you wish
> Digits := 6 : #use floating point arithmetic with 6 digits.  could use more if desired
> unassign(`x`, `y`); # in case you used the letters elsewhere
> f := (x, y)→y; # slope field function for the DE  y'(x)=y
                        # change for different DE's!!!
                              f := (x, y)→y                                  (5)
> x[0] := 0; y[0] := 1; #initial point
  h := 0.2; n := 5;    #step size and number of steps
                                 x_0 := 0
                                 y_0 := 1
                                 h := 0.2
                                 n := 5                                      (6)
> exactsol := x→e^x; #exact solution for the IVP  y'=y, y(0)=1
                       #change (or omit) for different IVP's!!!
                              exactsol := x→e^x                              (7)
>
```

Euler Loop

```
> for i from 0 to n do  #this is an iteration loop, with index "i" running from 0 to n
     print(i, x[i], y[i], exactsol(x[i]));
         #print iteration step, current x,y, values, and exact solution value
   { k := f(x[i], y[i]); #current slope function value
   { x[i + 1] := x[i] + h;
   { y[i + 1] := y[i] + h·k;
  end do: #how to end a do loop in Maple
                                0, 0, 1, 1
                            1, 0.2, 1.2, 1.22140
                            2, 0.4, 1.44, 1.49182
                            3, 0.6, 1.728, 1.82212
                            4, 0.8, 2.0736, 2.22554
                            5, 1.0, 2.48832, 2.71828                          (8)
```

Plot results:

```
> with(plots) :
  Eulerapprox := pointplot({seq([x[i], y[i]], i = 0 ..n)}) : #approximate soln points
   exactsolgraph := plot(exactsol(t), t = 0 ..1, `color` = `black`) :  #exact soln graph
        #used t because x has been defined to be something else
   display({Eulerapprox, exactsolgraph}, title = `approximate and exact solution graphs`);
```



Exercise 3 Why are our approximations too small in this case, compared to the exact solution?

It should be that as your step size $h = \Delta x$ gets smaller, your approximations to the actual solution get better. This is true if your computer can do exact math (which it can't), but in practice you don't want to make the computer do too many computations because of problems with round-off error and computation time, so for example, choosing $h = .0000001$ would not be practical. But, trying $h = 0.01$ in our previous initial value problem should be instructive.

If we change the n-value to 100 and keep the other data the same we can rerun our experiment, copying, pasting, modifying previous work.

Initialize

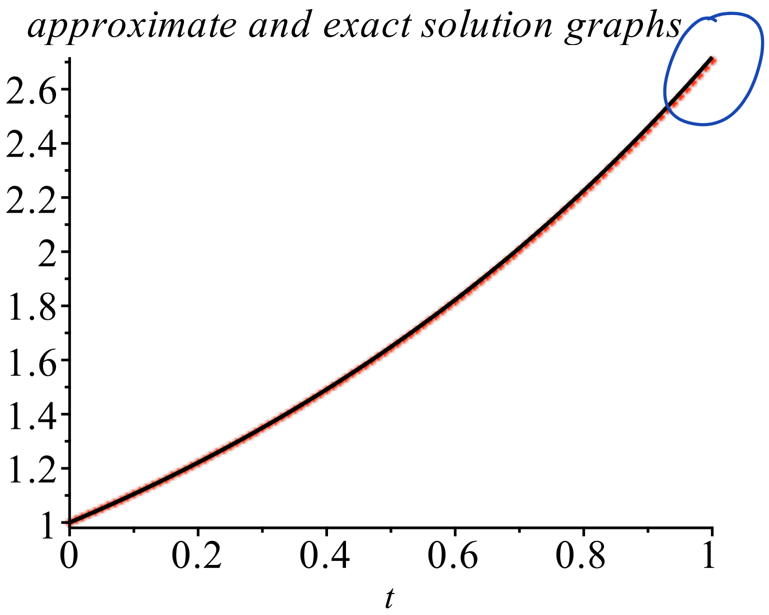
```
> restart : #clear all memory, if you wish
> unassign(`x`, `y`); # in case you used the letters elsewhere
> Digits := 6 :
> f := (x, y) → y : # slope field function for the DE  y'(x)=y
                    # change for different DE's!!!
> x[0] := 0 : y[0] := 1 : #initial point
  h := 0.01 : n := 100 :  #step size and number of steps
> exactsol := x → e^x : #exact solution for the IVP  y'=y, y(0)=1
                        #change (or omit) for different IVP's!!!
>
```

Euler Loop (modified using an "if then" conditional clause to only print every 10 steps)

```
> for i from 0 to n do #this is an iteration loop, with index "i" running from 0 to n
      if frac(i/10) = 0
          then print(i, x[i], y[i], exactsol(x[i]));
      end if: #only print every tenth time
     k := f(x[i], y[i]); #current slope function value
     x[i + 1] := x[i] + h;
     y[i + 1] := y[i] + h·k;
  end do: #how to end a for loop in Maple
                               0, 0, 1, 1
                       10, 0.10, 1.10463, 1.10517
                       20, 0.20, 1.22020, 1.22140
                       30, 0.30, 1.34784, 1.34986
                       40, 0.40, 1.48887, 1.49182
                       50, 0.50, 1.64463, 1.64872
                       60, 0.60, 1.81670, 1.82212
                       70, 0.70, 2.00676, 2.01375
                       80, 0.80, 2.21672, 2.22554
                       90, 0.90, 2.44864, 2.45960
                      100, 1.00, 2.70483, 2.71828                 (9)
```

plot results:

```
> with(plots) :
  Eulerapprox := pointplot({seq([x[i], y[i]], i = 0 ..n)}, color = red) :
   exactsolgraph := plot(exactsol(t), t = 0 ..1, `color` = `black`) :
       #used t because x was already used has been defined to be something else
   display({Eulerapprox, exactsolgraph}, title = `approximate and exact solution graphs`);
```



Exercise 4: For this very special initial value problem

$$y'(x) = y$$
$$y(0) = 1$$

which has $y(x) = e^x$ as the solution, set up Euler on the x-interval $[0, 1]$, with n subdivisions, and step size $h = \frac{1}{n}$. Write down the resulting Euler estimate for $\exp(1) = e$. What is the limit of this estimate as $n \to \infty$? You learned this special limit in Calculus!

step i	x_i	y_i	$f(x_i, y_i)$	Δx	x_{i+1}	y_{i+1}
⋮						
⋮						
n						

=

Wed Jan 31
2.5-2.6 Improved Euler and Runge Kutta.

Announcements:

- Bring laptop with access to Matlab tomorrow, to lab, if you have it. Access to a calculator or Wolfram alpha will be needed.
- quiz at end of class
- Handout with example run in Matlab – bring to lab tomorrow (these files are on CANVAS) (and we'll demo it today)

Warm-up Exercise: Solve for velocity $v(t)$

$$\begin{cases} v' = 6 - 3v \\ v(0) = 4 \end{cases}$$

is your answer consistent with the phase diagram?

$$v' + 3v = 6$$

$$e^{3t}(v' + 3v) = 6e^{3t}$$

$$\int \frac{d}{dt}\left(e^{3t}v\right) = \int 6e^{3t}\,dt$$

$$e^{3t}v = 2e^{3t} + C$$

$\div e^{3t}$

$$v = 2 + Ce^{-3t}$$

$$v(0) = 4 = 2 + C \Rightarrow C = 2$$

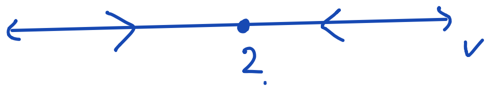
$$\boxed{v(t) = 2 + 2e^{-3t}}$$

$$\lim_{t \to \infty} v(t) = 2$$

$6 - 3v = v'$

$-3(v-2) = v'$

$v' = (-)(-) = (+)$ $\quad$ $v' = (-)(+) = (-)$



asymptotically stable (terminal velocity)

consistent

2.5-2.6 Improved Euler and Runge Kutta

In more complicated differential equations it is a very serious issue to find relatively efficient ways of approximating solutions. An entire field of mathematics, ``numerical analysis" deals with such issues for a variety of mathematical problems. Our text explores improvements to Euler in sections 2.5 and 2.6, in particular it discusses improved Euler, and Runge Kutta. Runge Kutta-type codes are actually used in commerical numerical packages, e.g. in Wofram, Matlab, Maple etc.

Let's summarize some highlights from 2.5-2.6.

Suppose we already knew the solution $y(x)$ to the initial value problem

$$y'(x) = f(x, y)$$
$$y(x_0) = y_0.$$

If we integrate the DE from x to $x + \overset{,,\Delta x}{h}$ and apply the Fundamental Theorem of Calculus, we get

$$\int_x^{x+h} y'(t)\,dt = y(x+h) - y(x) = \int_x^{x+h} f(t, y(t))\,dt\text{, i.e.}$$

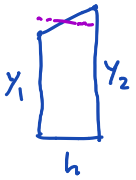
$$y(x+h) = y(x) + \underbrace{\int_x^{x+h} f(t, y(t))\,dt}\,.$$

One problem with Euler is that we approximate this integral above by $h \cdot f(x, y(x))$, i.e. we use the value at the left-hand endpoint as our approximation of the integrand, on the entire interval from x to $x + h$. This causes errors that are larger than they need to be, and these errors accumulate as we move from subinterval to subinterval and as our approximate solution diverges from the actual solution. The improvements to Euler depend on better approximations to the integral. These are subtle, because we don't yet have an approximation for $y(t)$ when t is greater than x, so also not for the integrand $f(t, y(t))$ on the interval $[x, x + h]$.

Improved Euler uses an approximation to the Trapezoid Rule to compute the integral in the formula

$$y(x+h) = y(x) + \int_x^{x+h} f(t, y(t))\, \mathrm{d}t\,.$$

Recall, the trapezoid rule to approximate the integral



area $h\left(\frac{y_1+y_2}{2}\right)$

$$\int_x^{x+h} f(t, y(t))\, \mathrm{d}t$$

would be

$$\frac{1}{2} h \cdot \left(f(x, y(x)) + f(x+h, y(x+h))\right).$$

Since we don't know $y(x+h)$ we approximate its value with unimproved Euler, and substitute into the formula above. This leads to the improved Euler "pseudocode" for how to increment approximate solutions.

Improved Euler pseudocode:

```
$k_1 = f(x_j, y_j)$                    #current slope
$k_2 = f(x_j + h, y_j + h\,k_1)$       #use unimproved Euler guess for $y(x_j + h)$ to estimate slope function when $x = x_j + h$
$k = \frac{1}{2}(k_1 + k_2)$           #average of two slopes
$x_{j+1} = x_j + h$                    #increment $x$
$y_{j+1} = y_j + h\,k$                 #increment $y$
```

Exercise 1 Contrast Euler and Improved Euler for our IVP

$$y'(x) = y$$
$$y(0) = 1$$

$= f(x,y)$

actual sol'n $y = e^x$

to estimate $y(1) \approx$ e with one step of size $h = 1$. Your computations should mirror the picture below. Note that with improved Euler we actually get a better estimate for e with ONE step of size 1 than we did with 5 steps of size $h = 0.2$ using unimproved Euler on Tuesday. (It's still not a very good estimate.)

$x_0 = 0\,,\ y_0 = 1$

$k_1 = f(0,1) = 1$

$k_2 = f(0+1, 1+1\cdot 1) = f(1,2) = 2$

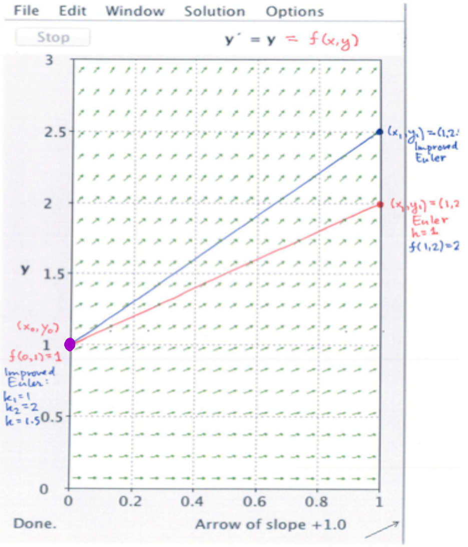
$k = \frac{1}{2}(k_1 + k_2) = \frac{1}{2}(1+2) = 1.5$

$x_1 = 0 + 1 = 1$

$y_1 = 1 + 1\cdot(1.5) = \boxed{2.5}$

$e = 2.718\ldots$

better then just Euler with 5 steps yesterday.



In the same vein as ``improved Euler" we can use the Simpson approximation for the integral for Δy instead of the Trapezoid rule, and this leads to the Runge-Kutta method. You may or may not have talked about Simpson's Parabolic Rule for approximating definite integrals in Calculus. It is based on a parabolic approximation to the integrand , whereas the Trapezoid rule is based on an approximationg of the graph with trapezoids, on small subintervals.

<u>Simpson's Rule:</u>

Consider two numbers $x_0 < x_1$, with interval width $h = x_1 - x_0$ and interval midpoint $\underline{x} = \frac{1}{2}\left(x_0 + x_1\right)$.
If you fit a parabola $p(x)$ to the three points

$$\left(x_0, g\left(x_0\right)\right),\ \left(\underline{x}, g(\underline{x})\right),\ \left(x_1, g\left(x_1\right)\right)$$

then

$$\int_{x_0}^{x_1} p(t)\mathrm{d}t = \frac{h}{6}\left(g\left(x_0\right) + 4\cdot g(\underline{x}) + g\left(x_1\right)\right).$$

(You will check this fact in your homework this week!) This formula is the basis for Simpson's rule, which you can also review in your Calculus text or at Wikipedia. (Wikipedia also has a good entry on Runge-Kutta.) Applying the Simpson's rule approximation for our DE, and if we already knew the solution function $y(x)$, we would have

$$y(x+h) = y(x) + \int_x^{x+h} f(t, y(t))\,dt$$

$$\approx y(x) + \frac{h}{6}\cdot\left(f(x, y(x)) + 4f\left(x + \frac{h}{2}, y\left(x + \frac{h}{2}\right)\right) + f(x+h, y(x+h))\right) .$$

However, we have the same issue as in Trapezoid - that we've only approximated up to $y(x)$ so far. Here's the magic pseudo-code for how Runge-Kutta takes care of this. (See also section 2.6 to the text.)

<u>Runge-Kutta pseudocode</u>:

```
$k_1 = f\left(x_j, y_j\right)$   # left endpoint slope
$k_2 = f\left(x_j + \frac{h}{2}, y_j + \frac{h}{2}k_1\right)$ # first midpoint slope estimate, using $k_1$ to increment $y_j$
$k_3 = f\left(x_j + \frac{h}{2}, y_j + \frac{h}{2}k_2\right)$  # second midpoint slope estimate using $k_2$ to increment $y_j$
$k_4 = f\left(x_j + h, y_j + h\,k_3\right)$        # right hand slope estimate, using $k_3$ to increment $y_j$
$k = \frac{1}{6}\left(k_1 + 2\,k_2 + 2\,k_3 + k_4\right)$ #weighted average of all four slope estimates, consistent with Simpson's rule
$x_{j+1} = x_j + h$      # increment $x$
$y_{j+1} = y_j + h\,k$     # increment $y$
```

Exercise 2 Contrast Euler and Improved Euler to Runge-Kutta for our IVP

$$y'(x) = y = f(x,y)$$
$$y(0) = 1$$

to estimate $y(1) \approx$ e with one step of size $h = 1$. Your computations should mirror the picture below. Note that with Runge Kutta you actually get a better estimate for e with ONE step of size $h = 1$ than you did with 100 steps of size $h = 0.01$ using unimproved Euler!

Runge-Kutta pseudocode:

```
$k_1 = f(x_j, y_j)$    # left endpoint slope
$k_2 = f\left(x_j + \frac{h}{2}, y_j + \frac{h}{2} k_1\right)$    # first midpoint slope estimate, using $k_1$ to increment $y_j$
$k_3 = f\left(x_j + \frac{h}{2}, y_j + \frac{h}{2} k_2\right)$    # second midpoint slope estimate using $k_2$ to increment $y_j$
$k_4 = f(x_j + h, y_j + h\,k_3)$    # right hand slope estimate, using $k_3$ to increment $y_j$
$k = \frac{1}{6}(k_1 + 2\,k_2 + 2\,k_3 + k_4)$    #weighted average of all four slope estimates, consistent with Simpson's rule
$x_{j+1} = x_j + h$    # increment $x$
$y_{j+1} = y_j + h\,k$    # increment $y$
```

$x_0 = 0, \quad y_0 = 1$

$k_1 = f(0,1) = 1$

$k_2 = f(.5, 1 + (.5)(1)) = f(.5, 1.5) = 1.5$

$k_3 = f(.5, 1 + (.5)(1.5)) = f(.5, 1.75) = 1.75$

$k_4 = f(1, 1 + 1(1.75)) = f(1, 2.75) = 2.75$

$k = \frac{1}{6}(1 + \underbrace{2(1.5)}_{3} + \underbrace{2(1.75)}_{3.5} + 1(2.75))$

$= \frac{10.25}{6} = 1.7083.$

$$\begin{array}{r} 1 \\ 3 \\ 3.5 \\ 2.75 \\ \hline 10.25 \end{array}$$

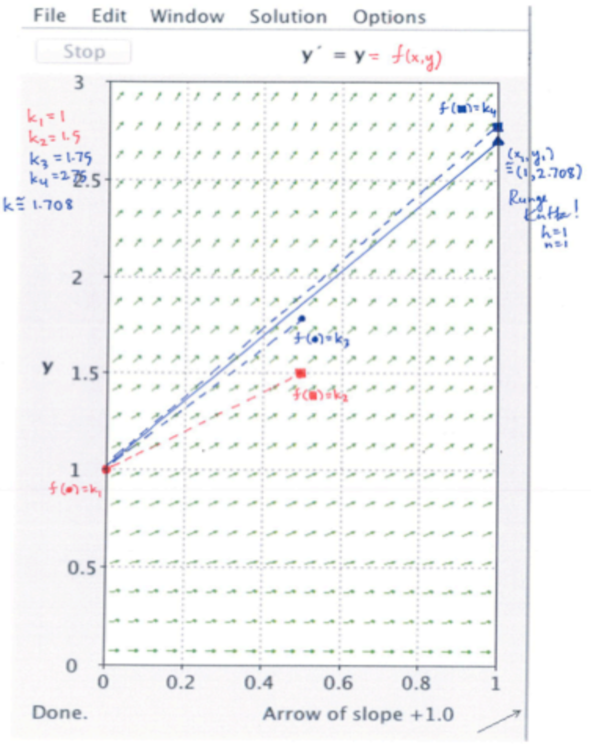
$x_1 = 1$

$y_1 = 1 + 1(1.7083)$

$= 2.7083$

$e \approx 2.71828\ldots$

better than 100 Euler steps with $h = .01$



Numerical experiments

Estimate e *by estimating* $y(1)$ for the solution to the IVP

$$y'(x) = y$$
$$y(0) = 1.$$

Apply Runge-Kutta with n = 10, 20, 40 ... subintervals, successively doubling the number of subintervals until you obtain the target number below - rounded to 9 decimal digits - twice in succession.

```
> evalf(e);
                              2.718281828459045                              (10)
```

It worked with $n = 40$:

Initialize

```
> restart : # clear any memory from earlier work
> Digits := 15 :  # we need lots of digits for "famous numbers"
>
> unassign(`x`, `y`) :  # in case you used the letters elsewhere, and came back to this piece of code
> f := (x, y) → y :  # slope field function for our DE i.e. for y'(x)=f(x,y)
> x[0] := 0 : y[0] := 1 :  #initial point
   h := 0.025 : n := 40 :    #step size and number of steps - first attempt for "famous number e"
```

Runge Kutta loop. WARNING: ONLY RUN THIS CODE AFTER INITIALIZING

```
> for i from 0 to n do  #this is an iteration loop, with index "i" running from 0 to n
     if (frac(i/10) = 0)
          then print(i, x[i], y[i]); #print iteration step, current x,y, values, and exact solution value
     end if:
     k1 := f(x[i], y[i]); #current slope function value
     k2 := f(x[i] + h/2, y[i] + h/2·k1);
       # first estimate for slope at right endpoint of midpoint of subinterval
     k3 := f(x[i] + h/2, y[i] + h/2·k2); #second estimate for midpoint slope
     k4 := f(x[i] + h, y[i] + h·k3);  #estimate for right-hand slope
     k := (k1 + 2· k2 + 2·k3 + k4)/6;  # Runge Kutta estimate for rate of change of y
     x[i + 1] := x[i] + h;
     y[i + 1] := y[i] + h·k;
  end do:  #how to end a for loop in Maple
                                    0, 0, 1
                          10, 0.250, 1.28402541566434
                          20, 0.500, 1.64872126807197
                          30, 0.750, 2.11700001155073
                          40, 1.000, 2.71828181979283                        (11)
```

For other problems you may wish to use or modify the following Euler and improved Euler loops.

Euler loop: WARNING: ONLY RUN THIS CODE AFTER INITIALIZING

```
> for i from 0 to n do  #this is an iteration loop, with index "i" running from 0 to n
    print(i, x[i], y[i]);
        #print iteration step, current x,y, values, and exact solution value
    k := f(x[i], y[i]);  #current slope function value
    x[i + 1] := x[i] + h;
    y[i + 1] := y[i] + h·k;
  end do:  #how to end a for loop in Maple
>
```

Improved Euler loop: WARNING: ONLY RUN THIS CODE AFTER INITIALIZING

```
> for i from 0 to n do  #this is an iteration loop, with index "i" running from 0 to n
    print(i, x[i], y[i]);  #print iteration step, current x,y, values, and exact solution value
    k1 := f(x[i], y[i]);  #current slope function value
    k2 := f(x[i] + h, y[i] + h·k1); #estimate for slope at right endpoint of subinterval
    k := (k1 + k2)/2;  # improved Euler estimate
    x[i + 1] := x[i] + h;
    y[i + 1] := y[i] + h·k;
  end do:  #how to end a do loop in Maple
>
```

Before running, please put Eulers.m, Runge_Kutta.m, Improved_Eulers.m, SlopeField.m, example.m in the same folder. To see the result, input **example** in the command windows

example.m

```
% Clear the command window screen, all the variables and ...
    close all windows
clc, clear, close all

% Input your function f(x,y) (as a string), where f(x,y) = ...
    dy/dx. In this example, f(x,y) = sin(x*y)
f = '1-3.*x+y';

% Draw the slope field in the region xmin≤x≤xmax, ...
    ymin≤y≤ymax for dy/dx = f(x,y)
xmin = -1;
xmax = 1;
ymin = -1;
ymax = 1;

SlopeField(xmin,xmax,ymin,ymax,f)

hold on

% Use different numerical methods to solve the IVP on the ...
    interval [x0,b] with n subintervals :
% dy/dx = f(x,y)
% y(x0) = y0
x0 = 0;
b = 1;
y0 = 0;
n = 10;

% Use Euler Method to Solve
[x y]=Eulers(x0,y0,b,n,f);
plot(x,y,'--','LineWidth',2,'Color','r');

% Use Improved Euler Method to solve
[x y]=Improved_Eulers(x0,y0,b,n,f);
plot(x,y,'-','LineWidth',2,'Color',[0.9100    0.4100    ...
    0.1700]);

% Use Runge Kutta method to solve
[x y]=Runge_Kutta(x0,y0,b,n,f);
plot(x,y,':','LineWidth',2,'Color','b');

xlabel('x');ylabel('y')
legend('Slope Field','Euler','Improved Euler','Runge Kutta')
title(['dy/dx = ' f])
```

.* to multiply scalars   ' ' quotes for string

$(f(x,y) = 1-3x+y)$

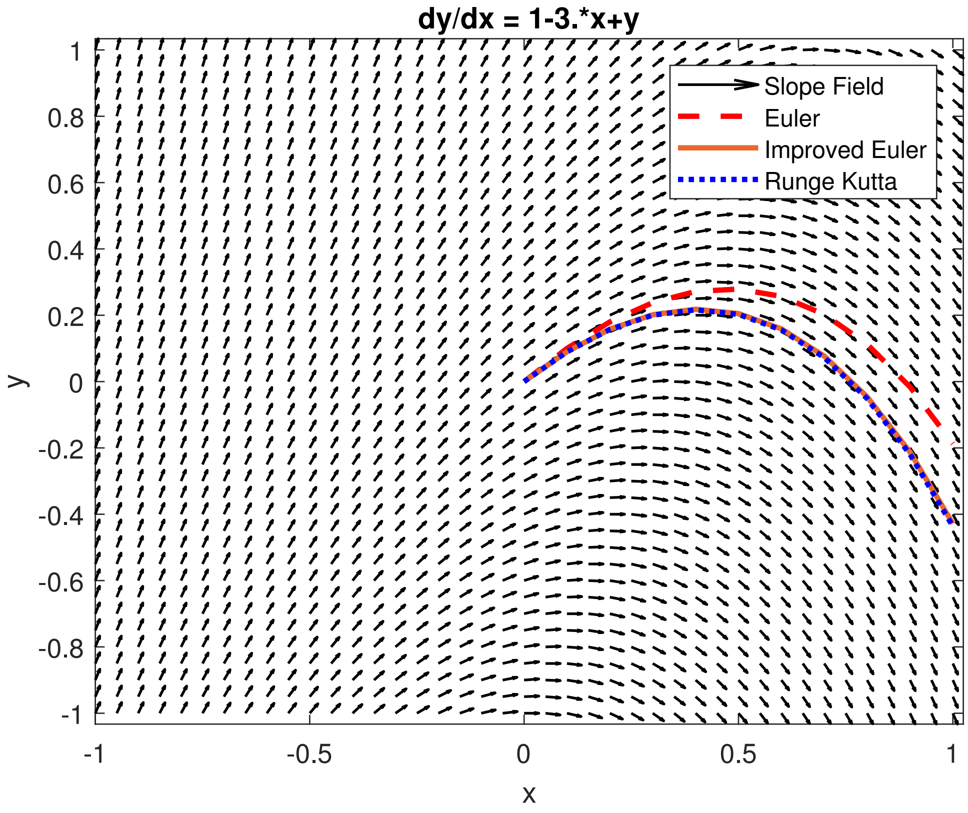
← to make a display with lots of plots

google is your friend.

right-hand endpoint

The code gives you

$y' = 1 - 3x + y$



The functions used in example.m

SlopeField.m

```
function []= SlopeField(xmin,xmax,ymin,ymax,f)
% the function f(x,y) must be entered as a string. For example if
% f(x,y)=y*cos(x), then you need to enter 'y*cos(x)' for f

f = str2func(['@(x,y) ',f]); % converts the string input into a ...
     function handle

stepsize=abs(xmin-xmax)/40;
[X,Y] = meshgrid(xmin:stepsize:xmax, ymin:stepsize:ymax);
dY=f(X,Y);
dX=ones(size(dY));
L=sqrt(1+dY.^2);
quiver(X,Y,dX./L,dY./L,'Color','k','LineWidth',1,'AutoScaleFactor',0.5);
axis tight
```

```
end
```

Eulers.m

```matlab
function [x,y] = Eulers(x_0,y_0,b,n,f)
% Eulers method with n intervals to solve the initial value ...
    problem dy/dx=f(x,y) with
% initial conditions y(x_0)=y_0 on the interval [x_0,b]. The ...
    function
% f(x,y) must be entered as a string.

% For example, if you wish to solve dy/dx=y*cos(x) with y(0)=1 ...
    on the
% interval [0,15] with 50 intervals you would enter
% Eulers(0,1,15,50,'y.*cos(x)')

f = str2func(['@(x,y) ',f]); %converts the string input into a ...
    function handle

h=(b-x_0)/n;
x=zeros(n+1,1); y= zeros(n+1,1); %Pre-allocoting vectors for speed
x(1)=x_0; y(1)=y_0;

for i=1:n
    x(i+1)=x_0+i*h;
    y(i+1)=y(i)+h*f(x(i),y(i));
end
end
```

Improved_Eulers.m

```
function [x,y] = Improved_Eulers(x_0,y_0,b,n,f)
% Improved Eulers  method with n intervals to solve the initial ...
    value problem dy/dx=f(x,y) with
% initial conditions y(x_0)=y_0 on the interval [x_0,b]. The ...
    function
% f(x,y) must be entered as a string.

% For example, if you wish to solve dy/dx=y*cos(x) with y(0)=1 ...
    on the
% interval [0,15] with 50 intervals you would enter
% Improved_Eulers(0,1,15,50,'y.*cos(x)')

f = str2func(['@(x,y) ',f]); % converts the string input into a ...
    function handle

h=(b-x_0)/n;
x=zeros(n+1,1); y= zeros(n+1,1); % Pre-allocoting vectors for speed
x(1)=x_0; y(1)=y_0;
for i=1:n
    x(i+1)=x_0+i*h;
    k1=f(x(i),y(i));
    u=y(i)+h*k1;
    k2=f(x(i+1),u);
    y(i+1)=y(i)+h*(k1+k2)/2;
end
end
```

Runge_Kutta.m

```matlab
function [x,y] = Runge_Kutta(x_0,y_0,b,n,f)
% Runge Kutta method with n intervals to solve the initial value ...
    problem dy/dx=f(x,y) with
% initial conditions y(x_0)=y_0 on the interval [x_0,b]. The ...
    function
% f(x,y) must be entered as a string.

% For example, if you wish to solve dy/dx=y*cos(x) with y(0)=1 ...
    on the
% interval [0,15] with 50 intervals you would enter
% Runge_Kutta(0,1,15,50,'y.*cos(x)')

f = str2func(['@(x,y) ',f]); % converts the string input into a ...
    function handle

h=(b-x_0)/n;
x=zeros(n+1,1); y= zeros(n+1,1); % Pre-allocoting vectors for speed
x(1)=x_0; y(1)=y_0;
for i=1:n
    x(i+1)=x_0+i*h;
    k1=f(x(i),y(i));
    k2=f(x(i)+h/2,y(i)+h/2*k1);
    k3=f(x(i)+h/2,y(i)+h/2*k2);
    k4=f(x(i+1),y(i)+h*k3);
    y(i+1)=y(i)+h*(k1+2*k2+2*k3+k4)/6;
end
end
```

Math 2250-004
Friday Feb 2
3.1-3.2 systems of linear (algebraic) equations

Announcements: • Chapters 3-4 is about linear algebra.

relates to HW 8 simpson's rule

example.

Warm-up Exercise: We're trying to find a parabola,

$\longrightarrow y = ax^2 + bx + c = p(x)$

passing through the three points

$(x,y) = (-1,0), (0,2), (1,2)$

Find the coefficients a, b, c to make this so!

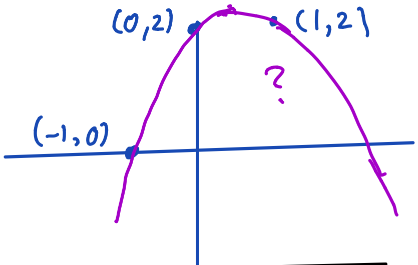
For example,

$@x=-1, y=0$, so $0 = a(-1)^2 + b(-1) + c$

$0 = a - b + c$

get two more eqtns, then find a,b,c.

'til 10:48



$@(-1,0): \quad 0 = a - b + c$

$@(0,2): \quad 2 = 0 + 0 + c \Rightarrow \boxed{c=2}$

$@(1,2): \quad 2 = a + b + c$

$E_1 \quad 0 = a - b + 2$

$E_3 \quad 2 = a + b + 2$

$E_1 + E_3: \quad 2 = 2a + 4$

$-2 = 2a \Rightarrow \boxed{a=-1}$

$E_1 - E_3: \quad -2 = -2b \Rightarrow \boxed{b=1}$

$a = -1$
$b = 1$
$c = 2$

$p(x) = -x^2 + x + 2$

$p(-1) = 0 = -1 - 1 + 2$ ✓

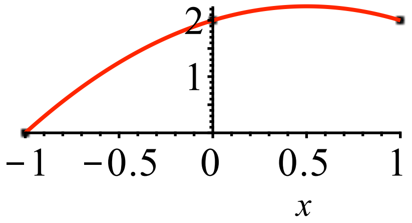
$p(0) = 2 = 0 + 0 + 2$ ✓

$p(1) = 2 = -1 + 1 + 2$ ✓

<u>Exercise 1:</u> (Relates to this week's homework, and introduces systems of linear algebraic equations:

Find the quadratic function $p(x) = a\,x^2 + b\,x + c$ so that the graph $y = p(x)$ interpolates (i.e. passes through) the three points $(-1, 0)$, $(0, 2)$, $(1, 2)$:

we just did this.



quadratic fit to three points (-1,0),(0,2),(1,2)

In Chapters 3-4 we temporarily leave differential equations in order to study basic concepts in linear algebra. Almost all of you have studied linear systems of equations and matrices before, and that's where Chapter 3 starts. Linear algebra is foundational for many different disciplines, and in this course we'll use the key ideas when we return to higher order linear differential equations and to systems of differential equations. As it turns out, there's an example of solving simultaneous linear equations in this week's homework. It's related to Simpson's rule for numerical integration, which is itself related to the Runge-Kutta algorithm for finding numerical solutions to differential equations.

In 3.1-3.2 our goal is to understand systematic ways to solve systems of (simultaneous) linear equations. Although we used *a*, *b*, *c* for the unknowns in the previous problem, this is not our standard way of labeling.

- We'll often call the unknowns $x_1, x_2, \ldots x_n$, or write them as elements in a vector

$$\underline{\boldsymbol{x}} = \left[x_1, x_2, \ldots x_n\right].$$

- Then the general linear system (LS) of *m* equations in the *n* unknowns can be written as

(LS)

$$\begin{array}{l} a_{11}\, x_1 + a_{12}\, x_2 + \ldots + a_{1\,n}\, x_n = b_1 \\ a_{21}\, x_1 + a_{22}\, x_2 + \ldots + a_{2\,n}\, x_n = b_2 \\ \quad \vdots \qquad\qquad\qquad\qquad\quad \vdots \quad \vdots \\ a_{ml}\, x_1 + a_{m2}\, x_2 + \ldots + a_{m\,n}\, x_n = b_m \end{array}$$

m equations in n unknowns $x_1, x_2 \ldots x_n$.

a_{ij}, b_j are numbers

where the coefficients $a_{i\,j}$ and the right-side number b_j are known. The goal is to find values for the vector $\underline{\boldsymbol{x}}$ so that all equations are true. (Thus this is often called finding "simultaneous" solutions to the linear system, because all equations will be true at once, for the given vector $\underline{\boldsymbol{x}}$.

Notice that we use two subscripts for the coefficients $a_{i\,j}$ and that the first one indicates which equation it appears in, and the second one indicates which variable its multiplying; in the corresponding *coefficient matrix A* , this numbering corresponds to the row and column of $a_{i\,j}$:

row → ← column.

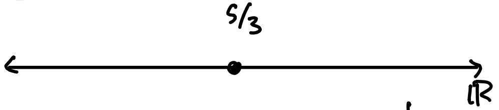
$$A := \begin{bmatrix} a_{11} & a_{12} & a_{13} & \ldots & a_{1\,n} \\ a_{21} & a_{22} & \boxed{a_{23}} & \ldots & a_{2\,n} \\ \vdots & & & & \vdots \\ a_{ml} & a_{m2} & a_{m3} & \ldots & a_{m\,n} \end{bmatrix}$$

a_{23} is coeff in 2nd row, 3rd column.
it came from coef. of x_3 in 2nd eqtn.

Let's start small, where geometric reasoning will help us understand what's going on:
Exercise 2: Describe the solution set of each single equation below; describe and sketch its geometric realization in the indicated Euclidean spaces $\mathbb{R}^n$.

2a) $3\,x = 5$, for $x \in \mathbb{R}$.

$x = 5/3$



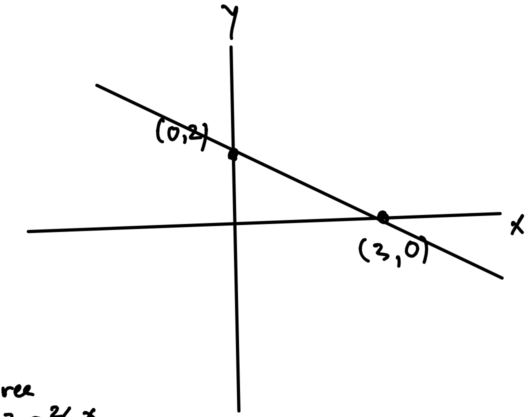
soltn set was a single point in ℝ

2b) $2\,x + 3\,y = 6$, for $[x, y] \in \mathbb{R}^2$.

$3y = 6 - 2x$

$y = 2 - 2/3\,x$, x "free", $(x \in \mathbb{R})$

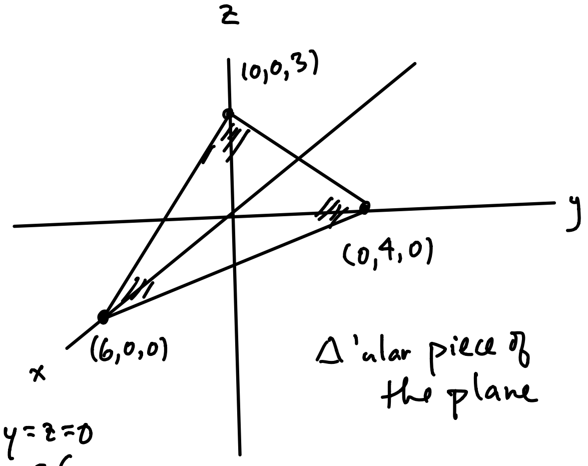
line with y-intercept 2
slope $-2/3$



x free
$y = 2 - 2/3\,x$

2c) $2\,x + 3\,y + 4\,z = 12$, for $[x, y, z] \in \mathbb{R}^3$.

plane in $\mathbb{R}^3$
x free
y free
$z = 3 - \frac{1}{2}x - \frac{3}{4}y$



Δ'ular piece of
the plane

$4z = 12 - 2x - 3y$
$z = 3 - \frac{1}{2}x - 3/4\,y$

@ $y = z = 0$
$x = 6$
@ $x = z = 0$
$y = 4$
@ $x = y = 0$
$z = 3$

2 linear equations in 2 unknowns:

$$a_{11}\,x + a_{12}\,y = b_1$$
$$a_{21}\,x + a_{22}\,y = b_2$$

goal: find all $[x, y]$ making both of these equations true. So geometrically you can interpret this problem as looking for the intersection of two lines.

Exercise 3: Consider the system of two equations E_1, E_2:

$$E_1 \qquad 5\,x + 3\,y = 1$$
$$E_2 \qquad x - 2\,y = 8$$

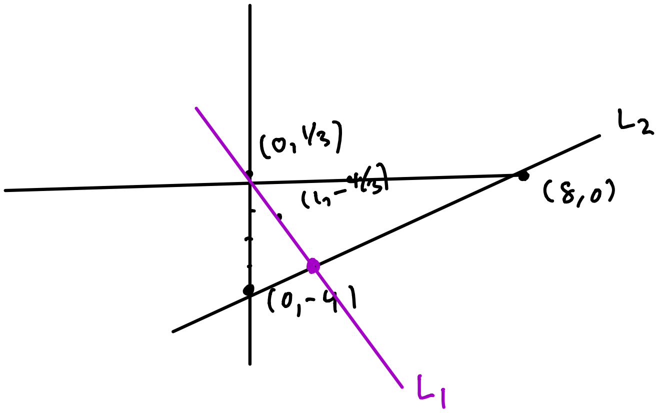
@ $x=1$

$5+3y=1$
$3y=-4$
$y=-4/3$

3a) Sketch the solution set in $\mathbb{R}^2$, as the point of intersection between two lines.

step 1

$E_2 \to E_1$ $\quad 1 \cdot x - 2y = 8$
$E_1 \to E_2$ $\quad 5x + 3y = 1$



3b) Use the following three "elementary equation operations" to systematically reduce the system E_1, E_2 to an equivalent system (i.e. one that has the same solution set), but of the form

$$1\,x + 0\,y = c_1$$
$$0\,x + 1\,y = c_2$$

(so that the solution is $x = c_1, y = c_2$). Make sketches of the intersecting lines, at each stage.

The three types of elementary equation operation are below. Can you explain why the solution set to the modified system is the same as the solution set before you make the modification?

- interchange the order of the equations
- multiply one of the equations by a non-zero constant
- replace an equation with its sum with a multiple of a different equation.

$$\begin{aligned} x - 2y &= 8\\ 5x + 3y &= 1 \end{aligned}$$

$$-5(\,x - 2y = 8\,)$$
$$-5x + 10y = -40$$

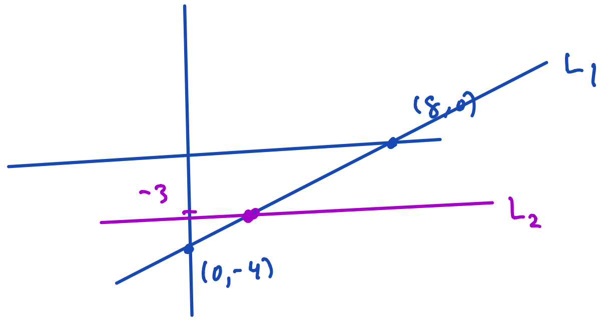
~~$E_1 \quad 5x + 3y = 1$~~

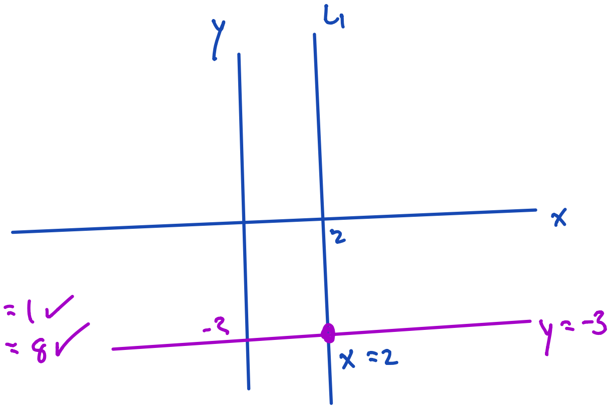
~~$E_2 \quad x - 2y = 8$~~

$$\begin{array}{rl} & x - 2y = 8\\ -5E_1+E_2 \to E_2 & 0 + 13y = -39 \end{array}$$

$$\begin{array}{rl} & x - 2y = 8\\ E_2/13 & y = -3 \end{array}$$

$$\begin{array}{rl} 2E_2+E_1\to E_1 & x = 2\\ & y = -3 \end{array}$$





$$(x,y) = (2,-3)$$

$$\begin{array}{ll} 5x+3y = 1 & 5\cdot 2 - 9 = 1 \checkmark\\ x - 2y = 8 & 2 + 6 = 8 \checkmark \end{array}$$

synthetically

$$\begin{array}{rcc|c} & 5 & 3 & 1\\ & 1 & -2 & 8\\ \hline R_2\to R_1 & 1 & -2 & 8\\ R_1\to R_2 & 5 & 3 & 1\\ \hline & 1 & -2 & 8\\ -5R_1+R_2 & 0 & 13 & -39\\ \hline & 1 & -2 & 8\\ \frac{R_2}{13}\to R_2 & 0 & 1 & -3\\ \hline 2R_2+R_1\to R_1 & 1 & 0 & 2\\ & 0 & 1 & -3 \end{array} \longrightarrow \begin{array}{l} 1\cdot x = 2\\ 1\cdot y = -3 \end{array} \qquad \begin{bmatrix} x\\ y \end{bmatrix} = \begin{bmatrix} 2\\ -3 \end{bmatrix}$$

$$2\,[\begin{array}{cc|c} 0 & 1 & -3 \end{array}]$$
$$\begin{array}{cc|c} 0 & 2 & -6 \end{array}$$

3c) Look at your work in 3b. Notice that you could have save a lot of writing by doing this computation "synthetically", i.e. by just keeping track of the coefficients and right-side values. Using R_1, R_2 as symbols for the rows, your work might look like the computation below. Notice that when you operate synthetically the "elementary equation operations" correspond to "elementary row operations":

- interchange two rows
- multiply a row by a non-zero number
- replace a row by its sum with a multiple of another row.

$$\begin{array}{r|rr|r}
 & 5 & 3 & 1 \\
 & 1 & -2 & 8 \\
\hline
R_2 & 1 & -2 & 8 \\
R_1 & 5 & 3 & 1 \\
\hline
 & 1 & -2 & 8 \\
-5R_1+R_2 & 0 & 13 & -39 \\
\hline
 & 1 & -2 & 8 \\
 & 0 & 1 & -3 \\
\hline
2R_2+R_1 & 1 & 0 & 2 \\
 & 0 & 1 & -3
\end{array}
\longrightarrow
\begin{array}{l} x=2 \\ y=-3 \end{array} \; !$$

3d) What are the possible geometric solution sets to 1, 2, 3, 4 or any number of linear equations in two unknowns? mystery

Solutions to linear equations in 3 unknowns:

What is the geometric question you're answering?

Exercise 4) Consider the system

$$\begin{aligned} x + 2\,y + z &= 4 \\ 3\,x + 8\,y + 7\,z &= 20 \\ 2\,x + 7\,y + 9\,z &= 23\,. \end{aligned}$$

Use elementary equation operations (or if you prefer, elementary row operations in the synthetic version) to find the solution set to this system. There's a systematic way to do this, which we'll talk about. It's called Gaussian elimination.

Hint: The solution set is a single point, $[x, y, z] = [5, -2, 3]$.

$$\left[\begin{array}{ccc|c} \textcircled{1} & 2 & 1 & 4 \\ 3 & 8 & 7 & 20 \\ 2 & 7 & 9 & 23 \end{array}\right]$$

$$\begin{array}{r} \\ -3R_1 + R_2 \to R_2 \\ -2R_1 + R_3 \to R_3 \end{array} \left[\begin{array}{c|cc|c} 1 & 2 & 1 & 4 \\ 0 & \textcircled{2} & 4 & 8 \\ 0 & 3 & 7 & 15 \end{array}\right]$$

$$\begin{array}{r} \\ \frac{R_2}{2} \to R_2 \\ \\ \end{array} \left[\begin{array}{ccc|c} 1 & 2 & 1 & 4 \\ 0 & 1 & 2 & 4 \\ 0 & 3 & 7 & 15 \end{array}\right]$$

$$\begin{array}{r} \\ \\ -3R_2 + R_3 \to R_3 \end{array} \left[\begin{array}{ccc|c} 1 & 2 & 1 & 4 \\ 0 & 1 & 2 & 4 \\ 0 & 0 & \textcircled{1} & 3 \end{array}\right]$$

$$\begin{array}{r} -R_3 + R_1 \to R_1 \\ -2R_3 + R_2 \\ \\ \end{array} \left[\begin{array}{ccc|c} 1 & 2 & 0 & 1 \\ 0 & \textcircled{1} & 0 & -2 \\ 0 & 0 & 1 & 3 \end{array}\right]$$

$$\begin{array}{r} -2R_2 + R_1 \to R_1 \\ \\ \\ \end{array} \left[\begin{array}{ccc|c} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & 3 \end{array}\right] \qquad \begin{aligned} x &= 5 \\ y &= -2 \\ z &= 3 \end{aligned}$$

Exercise 5 There are other possibilities. In the two systems below we kept all of the coeffients the same as in Exercise 4, except for a_{33} , and we changed the right side in the third equation, for 4a. Work out what happens in each case.

5a)

$$\begin{gathered} x + 2\,y + z = 4 \\ 3\,x + 8\,y + 7\,z = 20 \\ 2\,x + 7\,y + 8\,z = 20\,. \end{gathered}$$

5b)

$$\begin{gathered} x + 2\,y + z = 4 \\ 3\,x + 8\,y + 7\,z = 20 \\ 2\,x + 7\,y + 8\,z = 23\,. \end{gathered}$$

5c) What are the possible solution sets (and geometric configurations) for 1, 2, 3, 4,... equations in 3 unknowns?